Learning to Ask: When LLM Agents Meet Unclear Instruction

Anonymous ACL submission

Abstract

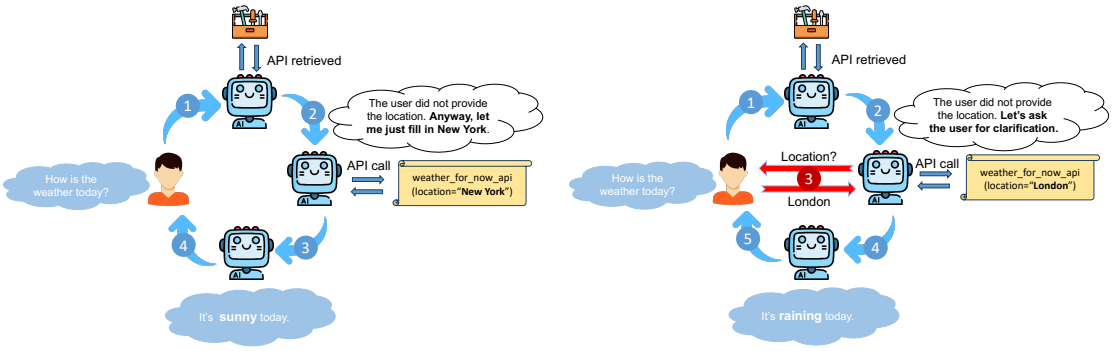
Equipped with the capability to call functions, modern LLM agents can leverage external tools for addressing a range of tasks unattainable through language skills alone. However, the effective execution of these tools relies heavily not just on the advanced capabilities of LLM agents but also on precise user instructions, which often cannot be ensured in the real world. To evaluate the performance of LLM agents tool-use under imperfect instructions, we meticulously examine the real-world instructions queried from users, analyze the error patterns, and build a challenging tool-use benchmark called Noisy ToolBench (NoisyToolBench). We find that due to the next-token prediction training objective, LLM agents tend to arbitrarily generate the missed argument, which may lead to hallucinations and risks. To address this issue, we propose a novel framework, Ask-when-Needed (AwN), which prompts LLM agents to ask questions to users whenever they encounter obstacles due to unclear instructions. Moreover, to reduce the manual labor involved in user-LLM interaction and assess LLM agents' performance in tool utilization from both accuracy and efficiency perspectives, we design an automated evaluation tool named ToolEvaluator. Our experiments demonstrate that the AwN significantly outperforms existing frameworks for tool learning in the NoisyToolBench. We will release all related code and datasets to support future research.

1 Introduction

LLMs have undergone remarkable development since OpenAI introduced ChatGPT-3.5 (Bang et al., 2023). This model demonstrates a significant advancement in solving multiple tasks, including code generation (Dong et al., 2023; Sakib et al., 2023; Feng et al., 2023), machine translation (Jiao et al., 2023; Peng et al., 2023), even game playing (Wu et al., 2024). However, despite their impressive capabilities, LLMs often struggle with complex computations and delivering accurate, timely information (Qu et al., 2024). Tool learning emerges as a promising solution to mitigate these limitations of LLMs by enabling dynamic interaction with external tools (Schick et al., 2024).

The incorporation of tool usage capabilities marks a pivotal step towards enhancing the intelligence of LLMs, pushing them closer to exhibiting human-like intelligence. The integration of tool usage allows LLMs to perform a broader array of complex and varied tasks. For example, LLMs can perform complex calculations using a calculator tool, access real-time weather updates through weather APIs, and execute programming code via interpreters (Qin et al., 2023a; Schick et al., 2024; Mialon et al., 2023; Yang et al., 2023a). Toolformer (Schick et al., 2024) is a pioneering work in empowering language models with self-learning capabilities for tool usage. Then, significant research efforts have been directed toward accessing a wider variety of tools or using multiple tools simultaneously to resolve a single query, such as Gorilla(Patil et al., 2023), RestGPT (Song et al., 2023) and ToolLLM (Qin et al., 2023b).

Despite the significant strides made, existing frameworks and benchmarks often operate under the assumption that user instructions are always explicit and unambiguous, a premise that diverges from real-world scenarios (Qin et al., 2023a; Song et al., 2023; Patil et al., 2023). Due to the feature of API calls, it requires precise user instructions since the arguments for the function call can hardly tolerate ambiguity. We find that due to the next-token prediction training objective, LLMs tend to arbitrarily generate the missed argument, which may lead to hallucinations and risks (as the example shown in Figure 1a). Furthermore, as the tasks assigned to LLMs grow in complexity, multiple and sequential API calls are needed to resolve a single task. This complexity amplifies the challenge, as any error in the sequence of API calls can culminate in an out-



(a) The execution process of previous frameworks.

(b) The execution process of our framework.

Figure 1: The motivating example of our Ask-when-Needed (AwN) framework.

come that strays from the user's original intention. Hence, LLMs tool-use under unclear instruction is an important but rarely investigated direction.

To address this oversight, we conduct a systematic analysis of actual user instructions, identifying and categorizing potential issues into several key areas. These include instructions lacking essential information, instructions with ambiguous references, instructions containing inaccuracies, and instructions that are unfeasible for LLMs to execute due to the limitations of the tools available. Building on this observation, we meticulously design a noisy instruction benchmark, named NoisyToolBench, which is pimarily used for assessing the capability of LLMs to detect ambiguities in user queries and to pose relevant questions for clarification accordingly. Specifically, NoisyToolBench includes a collection of provided APIs, ambiguous queries, anticipated questions for clarification, and the corresponding responses.

To improve the performance of LLMs tool-use under unclear instructions, we propose a novel framework called Ask-when-Needed (AwN). Our key insight is encouraging LLMs to proactively ask questions to seek clarifications from users when uncertainties arise during instruction execution. By facilitating dialogue throughout the process, our method aims to ensure the accurate invocation of functions (See Figure 1b)

To evaluate the performance of LLMs tool-use under unclear instruction, we design several innovative metrics from the accuracy and efficiency perspectives. For accuracy, we measure the LLMs' proficiency in asking appropriate clarifying questions, their ability to execute the correct function calls, and their success in delivering final responses that meet the users' needs. For efficiency, we calculate the average number of redundant asked questions and the average number of actions required to complete the instruction. An ideal LLM should achieve higher accuracy with fewer number of queries. Recognizing the labour-intensive nature of manually communicating with LLMs and verifying all execution results, we also innovatively design an automatic evaluation system, ToolEvaluator, to streamline the whole process. ToolEvaluator leverages the advanced problem-solving capabilities of GPT-4o to communicate with LLMs and automatically evaluate the performance of LLMs' tool-using under unclear instructions. Our experiments on 8 LLMs and 2 tool-using frameworks demonstrate that the AwN significantly outperforms existing baseline methods.

The contributions are summarized as follows:

- We conduct a systematic study on real-world user instruction for tool utilization and categorize the prevalent issues into four distinct categories.
- We create and release a novel benchmark, NoisyToolBench, which can be used to evaluate the performance of LLMs' tool-using under imperfect user instruction.
- We design five evaluation metrics from both accuracy and efficiency perspectives and introduce an automatic evaluation system, ToolEvaluator, that can proxy users to interact and assess LLMs.
- We introduce a novel framework, named AwN method, to prompt LLMs to actively ask questions to request clarifications from users when facing uncertainties. Experimental results show that AwN can significantly improve the LLMs' tool-using under unclear instructions.

2 Related Works

Tool Learning for LLMs. LLMs have recently made significant advancements, with ChatGPT being recognized as a major step towards achieving

AGI (Wu et al., 2023; Lund and Wang, 2023; Jiao et al., 2023). However, to progress further towards AGI, it is crucial for LLMs to master the utilization of tools. Toolformer is the first innovative AI model designed to use several specialized tools, such as a web browser, a code interpreter, and a language translator, within a single framework (Schick et al., 2023). The model's ability to seamlessly switch between these tools and apply them contextually represents a significant advancement in AI capabilities. Recent studies like RestGPT (Song et al., 2023) and ToolLLM (Qin et al., 2023b), have connected LLMs with real-life Application Programming Interfaces (APIs), allowing them to sequentially employ multiple external tools to solve user queries. The tool-augmented approach empowers LLMs to use various kinds of tools to do more sophisticated tasks, showcasing an enhanced level of capability compared to pure LLMs. Besides, API-Bank (Li et al., 2023), ToolAlpaca (Tang et al., 2023), ToolBench (Qin et al., 2023b), ToolQA (Zhuang et al., 2023) and RestBench (Song et al., 2023) are exemplary benchmarks to systematically evaluate the performance of tool-augmented LLMs performance in response to user's queries. However, current models often ignore the situations in which users might not give exact instructions, which can result in the tools not working properly. Thus, our study aims to tackle this specific challenge by developing a new benchmark specifically for ambiguous instructions.

Prompting LLMs for Decision Making. In certain situations, addressing user queries may require more than a single API call. This necessitates the effective division of the overarching task into smaller, more manageable components, which presents a significant challenge. Prior research has focused extensively on enhancing LLMs's ability to effectively plan and execute complex tasks. The 'Chain of Thought' prompting approach facilitates advanced reasoning by introducing intermediate steps in the reasoning process (Wei et al., 2022). The ReAct methodology improves the integration of reasoning and action, enabling LLMs to take informed actions based on environmental feedback (Yao et al., 2022). Meanwhile, Reflexion is designed to reduce errors in the reasoning process by revisiting and learning from previous mistakes (Shinn et al., 2023). DFSDT expands upon Reflexion, allowing LLMs to evaluate various options and choose the most viable path (Qin et al., 2023b). In our work, we innovatively involve users in the process of executing instructions. Our approach, referred to as AwN, motivates LLMs to consider the necessity of requesting further information from users during each tool invocation round. This strategy aims at clarifying users' ambiguous instructions to help execute the tasks in alignment with the users' intentions.

Learning to Ask. Since user queries may not always be clear, and the execution of LLMs may encounter uncertainties and ambiguities, learning to ask questions has emerged as a challenging yet crucial research area (Rao and Daumé III, 2018; Kuhn et al., 2022; Andukuri et al., 2024). For example, some researchers introduce a learning framework that empowers an embodied visual navigation agent to proactively seek assistance(Zhang et al., 2023). Recently, similar ideas have been adopted in the software engineering, leveraging a communicator to enhance the reliability and quality of generated code (Wu, 2023). Our work focuses on the tool-learning scenario, which is more sensitive to the user's unclear query. A concurrent study (Qian et al., 2024) also focuses on the reliability of tool-learning systems under unclear instruction. However, they did not systematically examine real-world user behavior, leading to the limited and biased nature of their dataset that doesn't accurately capture user errors. Additionally, Qian's methodology depends significantly on human manual interaction and assessment of LLM performances, which is time-consuming and hard to reproduce.

3 Noisy ToolBench

Several tool-learning benchmarks have been introduced to assess LLMs' ability in tool utilization. However, these benchmarks overlook the potential ambiguity in users' instruction, which might hinder LLMs from executing tasks as intended by the user. For instance, as depicted in Figure 1a, if a user inquires, "How is today's weather" without specifying the location, LLMs cannot accurately activate the APIs to fetch the correct weather information. This scenario underscores the critical role of interaction between users and LLMs in executing instructions accurately. However, previous tool-learning benchmarks only contain perfect user instruction in a one-query-one-execution manner.

To create a realistic benchmark for ambiguous instructions, the initial step involves a systematic examination of the common errors in user instructions that could complicate correct execution by

LLMs. Therefore, we first collect real-world user instructions that are problematic. Then, we classify these instructions into various categories based on their characteristics. Lastly, we manually create our dataset, ensuring it reflects the distribution of errors found in the real-world user instructions.

3.1 User Instruction Analysis

To analyze the issues in real-world user instruction, we recruit human annotators to write user queries according to the API provided. Firstly, we select 100 APIs from the ToolBench (Qin et al., 2023b), containing real-world RESTful APIs spanning 49 categories, ranging from sports to finance. Secondly, we hire 10 volunteers, who have a Bachelor's degree, are proficient in English, and have experience using LLMs. We provide them with the 100 APIs, and then ask them to write down an instruction to prompt LLMs to call each API, ending up with 1000 user queries. Finally, we manually identify the problematic user queries and categorized them as follows.

- **Instructions Missing Key Information (IMKI)**: These are user instructions that omit crucial details necessary for the successful execution of a function. An example of IMKI would be, "Set an alarm to wake me up" without providing a specific time. Asking for more information is needed when encountering this issue.
- **Instructions with Multiple References (IMR)**: These user instructions include elements that can be interpreted in several ways, potentially leading to confusion for LLMs in understanding the user's actual intent. For example, an IMR instance is "I want to know the director of the movie 'The Matrix'," where the ambiguity arises because there are multiple versions of 'The Matrix', each possibly having a different director. This issue is similar to IMKI but is more subtle and difficult to detect. Pointing out potential references and asking for clarification are needed when encountering this issue.
- **Instructions with Errors (IwE)**: This category consists of user instructions that contain the necessary information for executing a function, but the information is incorrect. An example of IWE is, "Please help me to log in to my Twitter. My user account is 'abcde@gmail.com' and the password is '123456'," where the user might have provided the wrong account details or password due to typographical errors. Asking for the correct information is needed when encountering this issue.
- **Instructions Beyond Tool Capabilities (IBTC)**: These are user instructions that request actions or answers beyond what LLMs can achieve with the available APIs. In such cases, the existing tool-augmented LLM frameworks might randomly choose an available API, leading to an incorrect function call. This scenario highlights the need for LLMs to recognize their limitations in tool usage. Telling the user that the query is beyond the capabilities and refusing to generate API calls are needed when encountering this issue.

Type of Issue	Ratio
Instructions Missing Key Information (IMKI)	56.0%
Instructions with Multiple References (IMR)	11.3%
Instructions with Errors (IwE)	17.3%
Instructions Beyond Tool Capabilities (IBTC)	15.3%

Table 1: Distribution of problematic instructions.

Table 1 shows the ratio of the four issues, where the most common issue in the instructions is "Instructions Missing Key Information", with a significant 56.0% of all errors. This issue is a clear indication that users often do not provide adequate information to effectively use the APIs. Additionally, issues such as "Instructions with Errors" and "Instructions Beyond Tool Capabilities" were identified at rates of 17.3% and 15.3%, respectively.

3.2 Data Construction

Our user instruction analysis reveals that there are four kinds of instruction issues that may lead to LLMs' tool utilization failures: Instructions Missing Key Information (IMKI), Instructions with Multiple References (IMR), Instructions with Errors (IwE), and Instructions Beyond Tool Capabilities (IBTC). So, we build our benchmark with the four issues by intentionally modifying the problem-free instructions from well-established datasets to problematic ones. We first select 200 data with problem-free instruction from ToolBench and then manually modify the user instructions to make them suffer from the four kinds of instruction issues. Then we annotate the expected questions that LLMs should ask when facing each imperfect user query, which will be used to measure whether LLMs can ask the right questions, as well as the answer to the question, which will be used to proxy the human responses. We conduct a two-round cross-verification to ensure the quality of the annotation. Each data is annotated and verified by different people and any disagreement data will be re-annotated

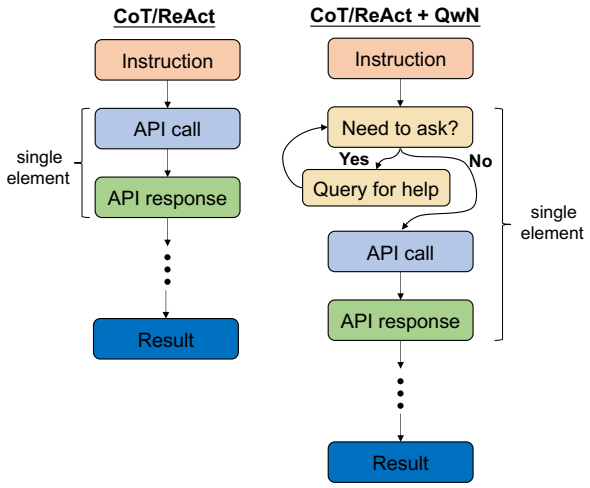


Figure 2: The comparison of our QwN prompting compared with original CoT/ReAct Prompting

until reach a consensus. Finally, each data entry in NoisyToolBench has the following five components: the imperfect user query, the available APIs, the questions that LLMs should ideally ask, the answers to these questions, and the expected function calls along with their respective arguments.

4 Ask-when-Needed Prompting

Previous approaches to tool-using often overlooked the importance of user engagement during the reasoning and planning stages. To address this oversight, we introduce a new prompting strategy named Ask-when-Needed (AwN). The key insight is prompting LLMs to detect the potential flaws in user instructions and proactively seek clarifications by asking questions before generating the API call.

AwN is built upon widely-used tool-using methods, such as CoT and ReAct. As in Figure 2, we introduce an additional step before the generation of API calls. This step involves presenting all available information to the LLMs, including the user query and API guideline, and prompting them to determine the adequacy and correctness of user instruction. If LLMs identify any missing argument needed for function execution based on the API's requirements, they are encouraged to ask questions to the user for this information. AwN prompts LLMs not to generate any API call until obtaining all the necessary information. In other words, only if no further information is needed, they can bypass the query step and directly initiate the API call. We also provide various kinds of specific instructions and demonstration examples for different kinds of instruction issues.

```
You are AutoGPT, tasked with processing user
requests through a variety of APIs you have
access to. Sometimes, the information provided
by users may be unclear, incomplete, or
incorrect.  Your main responsibility is to
determine if the user's instructions are
sufficiently clear and detailed for effective
use of the APIs. Here's your strategy:
1.  If user instructions are missing crucial
details for the APIs, pose a question to obtain
the necessary information.
2.  If the user's instructions appear to be
incorrect, delve deeper by asking questions to
clarify and rectify the details.
3.  If the user's request falls outside the
capabilities of your current APIs, notify them
that you're unable to meet the request by
stating: "Due to the limitation of the toolset,
I cannot solve the question".
...
```

5 Experiments

In this section, we evaluate the performance of our Ask-when-Needed (AwN) prompting technique on the NoisyToolBench dataset. We first introduce the evaluation metrics, where we specify the criteria used to assess the effectiveness of AwN. Then, we describe the evaluation pipeline, detailing the step-by-step process employed to measure AwN's performance. Lastly, we discuss the main experiments, presenting the results and findings from our comprehensive testing of the AwN technique.

5.1 Evaluation Metrics

We evaluate the performance of LLMs' tool-using under unclear instructions from two perspectives: accuracy and efficiency. The accuracy assessment aims to measure the LLMs' capability to make correct decisions during the instruction execution phase and to generate accurate final answers. In contrast, the efficiency assessment focuses on the number of redundant decisions made by the LLMs, considering that unnecessary communication could lead to a waste of processing time. Specifically, we design the following five metrics:

- **Accuracy 1 (A1).** A1 evaluates the capability of LLMs to ask the anticipated questions that pinpoint the ambiguous elements in user instructions. A1 is considered a success if the LLMs manage to ask the correct questions at any point. Conversely, it is deemed a failure if they do not.
- **Accuracy 2 (A2).** A2 assesses the ability of LLMs to use all available information to invoke the correct API calls. It is deemed a success if the LLMs call all the anticipated APIs with the correct arguments. If they fail to do so, it is considered a failure.

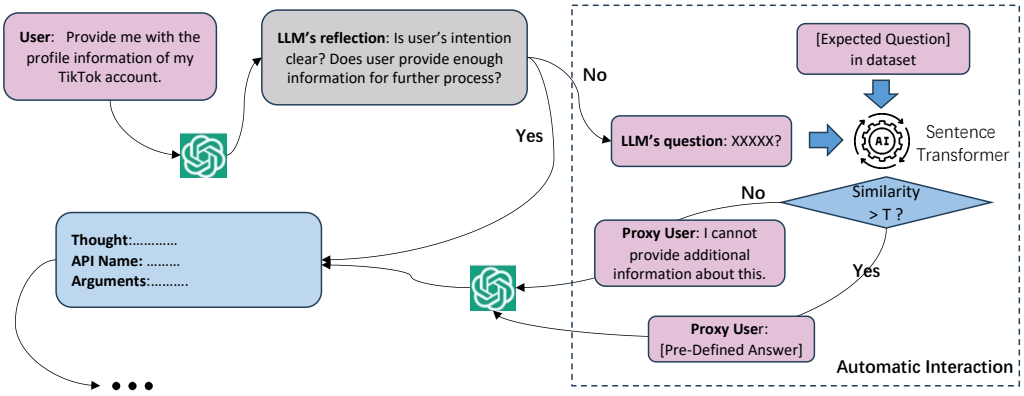


Figure 3: Illustration of the Auto-Interaction module.

- **Accuracy 3 (A3).** A3 measures the ability of LLMs to extract the anticipated information from previous API calls to fulfill the user's instructions. This is achieved and considered a success if the user's instructions are successfully executed. If not, it is regarded as a failure.
- **Average Redundant Asked questions (Re).** This metric evaluates the quantity of irrelevant or redundant questions asked by LLMs during the instruction process. Irrelevant questions are those that do not meet the initial expectations of the query, and redundant questions include those that are repetitive or have previously been asked. This metric is crucial for assessing the LLMs' ability to precisely identify the ambiguous aspects of user instructions and to formulate appropriate questions to clarify these uncertainties. The larger the value, the worse the performance.
- **Steps.** Steps quantifies the average number of actions required to complete an instruction, including inference generation, asking questions, and conducting API calls. A smaller number indicates fewer unnecessary steps in the instruction execution process, signifying a more efficient and direct approach to accomplishing the task.

5.2 Auto-Evaluation Pipeline

To assess how LLMs perform under unclear instructions, interacting with LLMs and making assessments are needed. Previous work employs individuals to interact with and evaluate LLMs throughout the entire evaluation process, which is inefficient and not reproducible. To address this, we design an automated evaluation method named ToolEvaluator to proxy this process. ToolEvaluator can automatically interact with LLMs and assess their performances.

Auto-Interaction. ToolEvaluator can proxy the user's communication with LLMs. When LLMs post a question, ToolEvaluator calculates the semantic similarity between the asked question and the expected question by the sentence-transformer (Reimers and Gurevych, 2019). If the similarity is higher than a threshold, ToolEvaluator replies with the predefined answer to the LLMs. Otherwise, this question is treated as an irrelevant question and ToolEvaluator replies with a standard reply of "Sorry, I cannot provide additional information about this.". This approach streamlines the evaluation process by reducing the need for human interaction with LLMs, as illustrated in Figure 3.

Auto-Assessment. ToolEvaluator can also automatically assess how well LLMs perform under ambiguous instructions according to the five metrics introduced above. A1 measures whether LLMs can ask the right question. ToolEvaluator calculates the semantic similarity between the LLMs-asked question and the expected question to asses A1. A2 measures whether LLMs can conduct correct API calls. Following the previous works (Yang et al., 2023b; Chiang and yi Lee, 2023; Wang et al., 2023; Yuan et al., 2023), ToolEvaluator adopts GPT-4o as a judge to identify whether the generated API calls are the same as the expected API calls. A3 measures whether LLMs can correctly generate the final answer. ToolEvaluator adopts GPT-4o as a judge to identify whether the final answer aligns with the user's intent. For measuring the efficiency, ToolEvaluator counts the number of generated irrelevant questions as Re and counts the total number of actions during the process as Steps.

5.3 The Effectiveness of ToolEvaluator

Since ToolEvaluator is an automatic evaluation method, the evaluation can be inaccurate due to the imperfect nature of AI techniques, such as sen-

Model	Framework	IMKI			IMR			IwE			IBTC
		A1(%)	A2(%)	A3(%)	A1(%)	A2(%)	A3(%)	A1(%)	A2(%)	A3(%)	A1(%)
gpt-3.5	CoT	0.74	0.36	0.22	0.20	0.24	0.12	0.5	0.24	0.16	0.38
	+ AwN	0.74	0.44	0.24	0.86	0.46	0.20	0.74	0.48	0.28	0.48
	DFSDT	0.64	0.16	0.12	0.60	0.18	0.16	0.48	0.14	0.14	0.46
	+ AwN	0.88	0.52	0.46	0.88	0.56	0.48	0.72	0.42	0.36	0.64
gpt-4	CoT	0.74	0.48	0.32	0.72	0.52	0.36	0.52	0.26	0.24	0.34
	+ AwN	0.94	0.62	0.50	0.76	0.44	0.38	0.48	0.34	0.34	0.94
	DFSDT	0.82	0.16	0.16	0.70	0.28	0.26	0.54	0.12	0.10	0.54
	+ AwN	0.80	0.56	0.48	0.80	0.50	0.44	0.52	0.38	0.36	0.94
gpt-4o	CoT	0.52	0.48	0.34	0.18	0.28	0.16	0.12	0.12	0.10	0.10
	+ AwN	0.90	0.58	0.36	0.80	0.46	0.30	0.60	0.44	0.32	0.92
	DFSDT	0.58	0.20	0.18	0.26	0.18	0.16	0.18	0.06	0.04	0.08
	+ AwN	0.88	0.60	0.46	0.90	0.52	0.36	0.64	0.46	0.38	0.94
deepseek-v3	CoT	0.44	0.40	0.20	0.24	0.28	0.24	0.10	0.14	0.14	0.30
	+ AwN	0.70	0.52	0.36	0.70	0.54	0.46	0.40	0.30	0.26	0.98
	DFSDT	0.42	0.30	0.26	0.60	0.20	0.18	0.22	0.12	0.12	0.48
	+ AwN	0.72	0.52	0.42	0.82	0.52	0.48	0.54	0.38	0.36	0.98
gemini-1.5	CoT	0.22	0.18	0.10	0.22	0.10	0.02	0.08	0.12	0.06	0.52
	+ AwN	0.86	0.40	0.18	0.74	0.24	0.08	0.58	0.28	0.22	0.68
	DFSDT	0.62	0.02	0.02	0.6	0.08	0.04	0.36	0.06	0.02	0.48
	+ AwN	0.82	0.40	0.12	0.76	0.28	0.04	0.66	0.36	0.26	0.70
claude-3.5	CoT	0.24	0.26	0.20	0.12	0.28	0.24	0.08	0.26	0.24	0.30
	+ AwN	0.54	0.5	0.5	0.32	0.30	0.24	0.34	0.34	0.26	0.88
	DFSDT	0.26	0.18	0.14	0.12	0.18	0.18	0.12	0.20	0.18	0.62
	+ AwN	0.52	0.44	0.42	0.32	0.30	0.18	0.36	0.36	0.30	0.86
O3-mini	CoT	0.00	0.16	0.16	0.00	0.16	0.14	0.00	0.04	0.04	0.00
	+ AwN	0.78	0.54	0.52	0.76	0.54	0.48	0.58	0.42	0.32	0.40
	DFSDT	0.00	0.16	0.16	0.00	0.16	0.10	0.00	0.06	0.02	0.00
	+ AwN	0.80	0.54	0.52	0.76	0.58	0.54	0.58	0.38	0.36	0.48
DeepSeek-R1	CoT	0.10	0.22	0.16	0.02	0.28	0.12	0.00	0.08	0.06	0.00
	+ AwN	0.80	0.52	0.34	0.60	0.54	0.22	0.50	0.34	0.22	0.84
	DFSDT	0.02	0.20	0.18	0.04	0.26	0.14	0.00	0.08	0.04	0.00
	+ AwN	0.86	0.62	0.42	0.62	0.42	0.26	0.54	0.38	0.22	0.76

Table 2: Assessing the accuracy of various LLMs using different prompting methods in our benchmark.

tence transformer or GPT-4o as the judge. In this section, we conduct a human annotation to validate the effectiveness of ToolEvaluator. Specifically, we randomly select 50 cases and ask annotators to assess the accuracy and efficiency, according to the evaluation metrics mentioned above. Then we compare the assessment results from ToolEvaluator and human annotators. ToolEvaluator achieves 90% accuracy, indicating its effectiveness.

5.4 Experimental Setup

We evaluated the performance of AwN against two baseline methods, chain-of-thought (CoT) (Wei et al., 2022) and depth-first search-based decision tree (DFSDT) (Qin et al., 2023b), which are two widely-used tool-learning methods. All the experiments are conducted with several LLMs as engines, gpt-3.5-turbo-0125, gpt-4-turbo-2024-04-09, gpt-4o-2024-11-20, deepseek-v3, gemini-1.5-flash-latest and claude-3-5-haiku-20241022, using the default setting. Since an ideal reaction under Instructions Beyond Tool Capabilities (IBTC) is telling the user that the query is beyond the capabilities and refusing to generate API calls, its performance in A2 and A3 are measured neither.

5.5 Main Result

We evaluate the performance of AwN as well as the baseline methods on our NoisyToolBench dataset. The accuracy-related results are shown in Table 2

Model	FrWork	IMKI		IMR		IwE		IBTC	
		Re	Steps	Re	Steps	Re	Steps	Re	Steps
gpt-3.5	CoT	-	4.46	-	4.02	-	3.90	-	2.10
	+ AwN	0.66	5.36	1.10	5.98	0.76	5.08	1.10	2.40
	DFSDT	-	12.82	-	12.80	-	13.82	-	5.50
	+ AwN	1.44	16.94	0.98	11.24	0.94	11.68	1.70	3.94
gpt-4	CoT	-	4.00	-	3.98	-	3.34	-	2.04
	+ AwN	0.16	3.94	0.20	3.94	0.36	3.46	0.04	1.16
	DFSDT	-	83.96	-	21.04	-	22.40	-	4.06
	+ AwN	0.48	9.82	0.74	13.08	0.62	9.42	0.16	2.10
gpt-4o	CoT	-	3.00	-	2.98	-	2.48	-	1.28
	+ AwN	0.62	3.86	0.70	3.96	0.46	3.18	0.00	1.10
	DFSDT	-	5.98	-	9.58	-	5.78	-	8.98
	+ AwN	0.86	6.70	1.18	7.68	0.88	8.56	0.00	1.14
deepseek-v3	CoT	-	4.20	-	3.52	-	3.12	-	1.18
	+ AwN	0.20	3.88	0.06	3.60	0.04	2.92	0.02	1.10
	DFSDT	-	59.08	-	41.70	-	24.24	-	11.64
	+ AwN	1.16	15.86	1.80	24.60	1.20	11.82	0.04	1.32
gemini-1.5	CoT	-	4.00	-	4.12	-	2.86	-	4.52
	+ AwN	0.42	6.44	0.68	6.36	0.48	4.54	0.46	1.46
	DFSDT	-	750.80	-	685.00	-	725.14	-	559.78
	+ AwN	5.34	445.16	9.08	532.56	1.94	411.18	0.46	1.46
claude-3.5	CoT	-	2.64	-	3.40	-	3.04	-	1.90
	+ AwN	0.18	3.74	0.33	1.03	0.36	3.76	0.10	1.68
	DFSDT	-	3.34	-	9.64	-	5.98	-	4.26
	+ AwN	0.76	6.74	0.80	17.46	1.04	13.08	0.14	2.76
O3-mini	CoT	-	1.88		3.08	-	2.72	-	1.08
	+ AwN	0.42	3.54	0.52	3.82	0.44	3.66	0.30	1.54
	DFSDT	-	2.04	-	4.36	-	2.88	-	1.04
	+ AwN	0.44	3.62	0.40	4.10	0.48	4.16	0.24	1.54
DeepSeek-R1	CoT	-	1.30	-	1.22	-	1.30	-	1.00
	+ AwN	0.16	2.50	0.20	1.94	0.12	2.00	0.02	1.02
	DFSDT	-	1.26	-	2.16	-	2.16	-	1.16
	+ AwN	0.22	2.54	0.36	4.08	0.4	3.66	0.00	1.02

Table 3: Assessing the efficiency of various LLMs using different prompting methods in our benchmark.

and the efficiency-related results are in Table 3.

AwN enhances the capability of LLM Agents to ask pertinent questions across different issues. For example, as is shown in Table 2, AwN improved the A1 scores from 0.52 to 0.90, from 0.18 to 0.80, from 0.12 to 0.60, and from 0.10 to 0.92 for gpt-4o-based CoT as well as from 0.58 to 0.88, from 0.26 to 0.90, from 0.18 to 0.64 and from 0.08 to 0.94 for gpt-4o-based DFSDT.

Asking the right question leads to the better generation and execution of API calls. Besides the significant improvements on A1, AwN also achieves considerable performance in generating correct API calls (A2) and returning the expected final answer (A3). For example, AwN improved the A2 scores from 0.48 to 0.58, from 0.28 to 0.46, from 0.12 to 0.44 for gpt-4o-based CoT as well as from 0.20 to 0.60, from 0.18 to 0.52, from 0.06 to 0.46 for gpt-4o-based DFSDT.

AwN can also improve the performance of reasoning models. Experimental results on two recently released strong reasoning models, OpenAI O3-mini and DeepSeek R1, showing that 1) vanilla models still cannot achieve good performance under unclear user instruction; 2) although vanilla reasoning models do not ask questions, it can sometimes produce the correct function call (A2 > A1), due to their powerful reasoning abilities; 3) AwN can significantly improve the performance by prompting LLMs to ask good questions.

AwN can improve most of the LLM agents without generating excessive unnecessary questions. As is shown in Table 3, AwN only leads to 0.16, 0.20, and 0.36 redundant questions for gpt-4-based-CoT, as well as 0.48, 0.74, and 0.62 redundant questions for gpt-4-based-DFSDT.

However, a few LLM agents tend to ask more irrelevant or redundant questions, as indicated by the higher Re scores in Table 3. For example, in Gemini-1.5-based DFSDT, where the average number of redundant questions is 5.34, 9.08, and 1.94. This suggests that while the AwN aids in identifying and addressing ambiguities in user instructions, it also leads to a less efficient querying process.

AwN can reduce the average cost of LLM's tool-using. The average number of steps measures the cost of LLMs' tool-using. As is shown in Table 3, adopting AwN can reduce the number of actions, especially for gpt-4-based DFSDT, deepseek-v3-based-DFSDT and gemini-1.5-based-DFSDT. Although AwN can lead to a higher cost for a few LLM agents, such as claude-3.5, considering the significant performance improvements achieved, the moderate increase in cost is justifiable.

6 Conclusion

This paper explores how unclear user instructions hinder LLM agents' tool usage by proposing: (1) Noisy ToolBench (NoisyToolBench), a novel benchmark for evaluating LLM performance under ambiguous instructions; (2) Ask-when-Needed (AwN), an innovative approach enabling LLMs to request clarification when uncertain; and (3) an automated evaluator (ToolEvaluator) to assess accuracy and efficiency. Experimental results show that the AwN algorithm significantly improves the performance of LLMs' tool-using under unclear user instructions.

Limitations

This paper has two limitations:

1. Although AwN can improve the performance, there is still a big gap to perfect. We hope that this work can serve as the first stepping stone, inspiring future researchers to delve deeper into this field of study.

2. The automatic evaluation process is not 100% accurate, leading to some potential false negatives and false positives. In the future, more efforts are needed to build a more reliable auto-evaluation method.

References

Chinmaya Andukuri, Jan-Philipp Fränken, Tobias Gerstenberg, and Noah D Goodman. 2024. Star-gate: Teaching language models to ask clarifying questions. *arXiv preprint arXiv:2403.19154*.

Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, et al. 2023. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *arXiv preprint arXiv:2302.04023*.

Cheng-Han Chiang and Hung yi Lee. 2023. Can large language models be an alternative to human evaluations? In *Annual Meeting of the Association for Computational Linguistics*.

Yihong Dong, Xue Jiang, Zhi Jin, and Ge Li. 2023. Self-collaboration code generation via chatgpt. *arXiv preprint arXiv:2304.07590*.

Yunhe Feng, Sreecharan Vanam, Manasa Cherukupally, Weijian Zheng, Meikang Qiu, and Haihua Chen. 2023. Investigating code generation performance of chatgpt with crowdsourcing social data. In *2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 876–885. IEEE.

Wenxiang Jiao, Wenxuan Wang, Jen-tse Huang, Xing Wang, Shuming Shi, and Zhaopeng Tu. 2023. Is chatgpt a good translator? yes with gpt-4 as the engine. *arXiv preprint arXiv:2301.08745*.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2022. Clam: Selective clarification for ambiguous questions with generative language models. *arXiv preprint arXiv:2212.07769*.

Minghao Li, Feifan Song, Bowen Yu, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023. Api-bank: A benchmark for tool-augmented llms. *arXiv preprint arXiv:2304.08244*.

Brady D Lund and Ting Wang. 2023. Chatting about chatgpt: how may ai and gpt impact academia and libraries? *Library Hi Tech News*, 40(3):26–29.

Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ram 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. *Preprint*, arXiv:2302.07842.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2023. Gorilla: Large language model connected with massive apis. *Preprint*, arXiv:2305.15334.

Keqin Peng, Liang Ding, Qihuang Zhong, Li Shen, Xuebo Liu, Min Zhang, Yuanxin Ouyang, and Dacheng Tao. 2023. Towards making the most of chatgpt for machine translation. *arXiv preprint arXiv:2303.13780*.

Cheng Qian, Bingxiang He, Zhong Zhuang, Jia Deng, Yujia Qin, Xin Cong, Zhong Zhang, Jie Zhou, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2024. Tell me more! towards implicit user intention understanding of language model driven agents. *Preprint*, arXiv:2402.09205.

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. *Preprint*, arXiv:2304.08354.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2023b. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*.

Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. 2024. Tool learning with large language models: A survey. *arXiv preprint arXiv:2405.17935*.

Sudha Rao and Hal Daumé III. 2018. Learning to ask good questions: Ranking clarification questions using neural expected value of perfect information. *arXiv preprint arXiv:1805.04655*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *EMNLP*.

Fardin Ahsan Sakib, Saadat Hasan Khan, and AHM Karim. 2023. Extending the frontier of chatgpt: Code generation and debugging. *arXiv preprint arXiv:2307.08260*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2024. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36.

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. *Preprint*, arXiv:2302.04761.

Noah Shinn, Beck Labash, and Ashwin Gopinath. 2023. Reflexion: an autonomous agent with dynamic memory and self-reflection. *arXiv preprint arXiv:2303.11366*.

Yifan Song, Weimin Xiong, Dawei Zhu, Wenhao Wu, Han Qian, Mingbo Song, Hailiang Huang, Cheng Li, Ke Wang, Rong Yao, Ye Tian, and Sujian Li. 2023. Restgpt: Connecting large language models with real-world restful apis. *Preprint*, arXiv:2306.06624.

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. *arXiv preprint arXiv:2306.05301*.

Jiayin Wang, Weizhi Ma, Peijie Sun, Min Zhang, and Jian-Yun Nie. 2024. Understanding user experience in large language model interactions. *ArXiv*, abs/2401.08329.

Wenxuan Wang, Zhaopeng Tu, Chang Chen, Youliang Yuan, Jen-tse Huang, Wenxiang Jiao, and Michael R Lyu. 2023. All languages matter: On the multilingual safety of large language models. *arXiv preprint arXiv:2310.00905*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Jie JW Wu. 2023. Does asking clarifying questions increases confidence in generated code? on the communication skills of large language models. *Preprint*, arXiv:2308.13507.

Tianyu Wu, Shizhu He, Jingping Liu, Siqi Sun, Kang Liu, Qing-Long Han, and Yang Tang. 2023. A brief overview of chatgpt: The history, status quo and potential future development. *IEEE/CAA Journal of Automatica Sinica*, 10(5):1122–1136.

Yue Wu, Xuan Tang, Tom M. Mitchell, and Yuanzhi Li. 2024. Smartplay: A benchmark for llms as intelligent agents. *Preprint*, arXiv:2310.01557.

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

Xianjun Yang, Xiao Wang, Qi Zhang, Linda Ruth Petzold, William Yang Wang, Xun Zhao, and Dahua Lin. 2023b. Shadow alignment: The ease of subverting safely-aligned language models. *ArXiv*, abs/2310.02949.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Youliang Yuan, Wenxiang Jiao, Wenxuan Wang, Jen-tse Huang, Pinjia He, Shuming Shi, and Zhaopeng Tu. 2023. Gpt-4 is too smart to be safe: Stealthy chat with llms via cipher. *arXiv preprint arXiv:2308.06463*.

Jenny Zhang, Samson Yu, Jiafei Duan, and Cheston Tan. 2023. Good time to ask: A learning framework for asking for help in embodied visual navigation. *Preprint*, arXiv:2206.10606.

Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. 2023. Toolqa: A dataset for llm question answering with external tools. *Preprint*, arXiv:2306.13304.

A Appendix

A.1 AwN Does Not Affect the Performance on Clear Instructions

Since NoisyToolBench focuses on evaluating the performance of LLM tool-use agents under unclear user instructions, the data only contains unclear instructions, which can complement existing tool-use evaluation datasets to assess the performance under imperfect user instructions. However, evaluating AwN on clear instructions is also important. So we select 200 samples from ToolBench, the user queries of which are clear, and we evaluate the performance of our Awn on this set. As is shown in Table 4, AwN can significantly improve the performance on unclear instructions and does not affect the performance under clear instructions.

Model	Clean	Noisy_IMKI	Noisy_IMR	Noisy_IwE
GPT-4o	0.54	0.34	0.16	0.10
+ AwN	0.51	0.36	0.30	0.32
DeepSeek V3	0.64	0.20	0.24	0.14
+ AwN	0.64	0.36	0.46	0.26
Claude 3.5	0.60	0.20	0.24	0.24
+ AwN	0.64	0.50	0.30	0.26
Gemini 1.5	0.36	0.10	0.02	0.06
+ AwN	0.40	0.18	0.08	0.22

Table 4: Assessing the performance of AwN on both clear and unclear instructions

A.2 Details of the Calculation of Each Metric

ToolEvaluator evaluated the model performance on A1, Re and Steps by a deterministic, rule-based algorithm that analyzes the responses. A2 and A3 are evaluated by LLM-as-a-judge.

The following are the details of the calculation of each metric.

A1: A1 is computed by analyzing whether the LLMs generate valid clarifying questions under ambiguous instructions in the NoisyToolbench dataset. A binary flag is given for each instruction; it is set to 1 if there is a clarifying question identified by similarity check. Otherwise, it remains 0.

Re: Re is computed by counting the number of redundant questions asked by LLMs. Similar to A1 situations, each clarifying question raised by LLM will be classified into relevant questions for ambiguity resolution or irrelevant questions. Re here is equal to the number of mis-clarifying questions raised by LLMs under each instruction.

Steps: Steps are computed by counting the number of total actions (generate response, ask question and call API) performed by LLM. The algorithm will go through the whole conversation history for each instruction.

A2 and A3: ToolEvaluator adopts LLMs as evaluators. We adopt GPT-4o as the evaluator to score A2 and A3 for objective 1 (whether using the correct API) and objective 2 (whether returning the correct answer). The instruction is:

As an evaluator of tool-augmented language model systems, your responsibility is to assess the models' effectiveness in using APIs to gather the necessary information to fulfill user requests. This involves reviewing the actual API calls made by the models against a given set of required API calls, including their parameters. Your evaluation focuses on two main objectives:

Objective 1: Verify the accuracy of the actual tool calls made by the models against the expected tool calls, including their arguments. A successful outcome means that the models executed all required API calls with expected arguments. Then the score of the Objective 1 should be 1. Otherwise, it is a failure for Objective 1 and the score should be 0. Here are some examples ...

Objective 2: Assess whether the model's final response correctly achieve the user's instruction and check if the final answer was indeed based on the data retrieved from these API calls, as opposed to being generated independently of these tool calls. Success in this objective means the model effectively used the API calls to achieve user's instructions. Conversely, failure suggests the response was not derived from the API data or the user's instruction is not achieved. It's important to note that any thoughts of the LLMs not based on the API response, especially regarding makeup information, are not considered valid answers. Please note that if the model does not provide the final answer, we consider it as a failure case. Here are some examples ...

A.3 User Experiments

To show the effectiveness of our method, we follow (Wang et al., 2024) to conduct a user experience study with real-world users. Specifically, we recruit 5 volunteers in Section 3.1 back, who have a Bachelor's degree, are proficient in English, and have experience using LLMs. We provided them with 10 APIs, along with the user instructions they designed to call the APIs, as well as the feedback AwN generated. Then we ask the following questions: From 1 (very disagree) to 5 (strongly agree), how much do you agree that the system's clarifying questions are relevant and helpful to solving the request? We got an average score of 4.2, showing the effectiveness of AwN from the real-world user perspectives.