Can Tool-augmented Large Language Models be Aware of Incomplete Conditions?

Anonymous ACL submission

Abstract

Recent advancements in integrating large language models (LLMs) with tools have allowed the models to interact with real-world environments. However, these *tool-augmented LLMs* often encounter incomplete scenarios when users provide partial information or the necessary tools are unavailable. Recognizing and managing such scenarios is crucial for LLMs to ensure their reliability, but this exploration remains understudied. This study examines whether LLMs can identify incomplete conditions and appropriately determine when to refrain from using tools. To this end, we address a dataset by manipulating instances from two datasets by removing necessary tools or essential information for tool invocation. We confirm that most LLMs are challenged to identify the additional information required to utilize specific tools and the absence of appropriate tools. Our research can contribute to advancing reliable LLMs by addressing scenarios that commonly arise during interactions between humans and LLMs. Our code and dataset will be made publicly available.

1 Introduction

Recently, there has been significant improvement in integrating large language models (LLMs) with tools (Li et al., 2023; Qin et al., 2023; Schick et al., 2024; Lu et al., 2024; Hao et al., 2024). These *tool-augmented LLMs* can perceive up-to-date information, acquire real-world interaction capabilities, and perform complex tasks (Wang et al., 2024). The tool-augmented models have enhanced user experiences across various applications (Yang et al., 2023; Hong et al., 2023).

LLMs should select and use the necessary tools when they interact with users. However, users often lack comprehensive knowledge about all available tools, making it challenging to initially provide the required information to models. Furthermore, the necessary tools may not be provided to models or

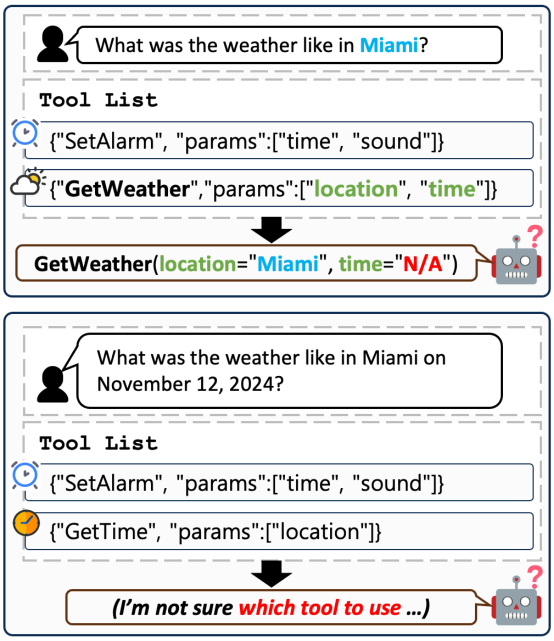

Figure 1: An illustration of incomplete conditions for tool invocation by tool-augmented LLMs.

even do not exist to address user requests. When LLM agents use various tools, it is common to encounter issues where the necessary tools are absent, or the user's input is incomplete, lacking the information needed to use them. Despite significant advances in tool-augmented LLM, research has been conducted on the premise of only a complete scenario (Huang et al., 2023).

Our work investigates whether tool-augmented LLMs can recognize incomplete conditions where no appropriate tool is available for the user's request, or the given information is insufficient to utilize the tool, as depicted in Figure 1. To this end, we construct a dataset by manipulating instances from datasets for tool-augmented LLMs (Qin et al., 2023; Li et al., 2023) to simulate the scenarios. Further human verification ensures only instances where tools could not be used are included.

Based on the dataset, we evaluate whether LLMs can recognize conditions where desirable tool invocation is unavailable. Experimental results indicate that most LLMs struggle to identify the additional information required to utilize specific tools or the absence of appropriate tools. Furthermore, our further analysis indicates that LLMs cannot identify incomplete scenarios, particularly when users provide insufficient information or utilize real-world tools. Our work contributes to the development of reliable tool-augmented LLMs.

2 Related Work

Recent studies have proposed to augment LLMs with external tools (Wang et al., 2024). These tools range from simple retrieval systems (Chen et al., 2017) and arithmetic operations (Schick et al., 2024) to complex programming languages (Cai et al., 2023) and real-world APIs (Xu et al., 2023; Yuan et al., 2024; Guo et al., 2024). Various strategies and benchmarks have been addressed to enable LLMs to utilize such tools more effectively. For instance, Huang et al. (2023) evaluates LLMs' ability to accurately decide when and which tools to use. In contrast, our study evaluates LLMs in conditions where tool usage is required, but no appropriate tool is available or information for proper tool invocation is insufficient. While research in question answering and retrieval systems address scenarios with missing external knowledge or ambiguous user requests (Feng et al., 2020; Min et al., 2020; Kamath et al., 2020; Cole et al., 2023; Jeong et al., 2024), the challenge of LLMs recognizing when tool usage is impractical remains unexplored.

3 How to Evaluate the Awareness of Tool-augmented LLMs

To simulate incomplete scenarios in which the necessary tools are unavailable or users provide partial information, we gather and manipulate the test set of two benchmarks (Li et al., 2023; Qin et al., 2023). The two datasets are designed to evaluate whether they can provide appropriate responses to user requests by utilizing APIs.[1] An API is always available to address the user's request, and sufficient information is provided to invoke the API.

Replacement of Relevant API By leveraging a dense retriever (Gao et al., 2021), we replace the appropriate APIs that could handle the user's requests with similar but irrelevant APIs. This replacement ensures that the desirable tool is unavailable, simulating a scenario where LLMs should decide not to use any given tools.

[1]Afterward, we refer to Tool and API interchangeably.

	API Replacement	**Utterance Removal**
APIBank	423	304
ToolBench	477	406

Table 1: **Data Statistics**. The *API Replacement* and *Utterance Removal* denote *Relevant API Replacement* and *Partial Removal of User Utterance*, respectively.

Partial Removal of User Utterance We removed the user utterances in a conversation to mimic scenarios where (1) The request is unclear, making it impossible to use APIs, or (2) the request is clear but lacks the necessary information for API invocation. To automatically create such instances, we use a proprietary LLM (i.e., GPT-4 (Achiam et al., 2023)) to automate the creation of naturally corrupted dialogues. We instruct the model to follow a detailed guideline and generate manipulated conversations. Additional information on both processes is provided in Appendix B.

Data Source We create a new dataset utilizing two datasets: APIBank (Li et al., 2023) and ToolBench (Qin et al., 2023). APIBank contains 73 manually implemented APIs and conversations written by human annotators. ToolBench includes over 16,000 real-world APIs and user instructions annotated by LLMs. Each instance comprises an available API list, a conversation between a user and AI, and the relevant APIs. These instances, where LLMs can immediately invoke the tool, are considered tool-callable. The number of selected instances for APIBank and ToolBench are 450 and 764, respectively. More dataset details are in Appendix A.

Data Verification To verify the validity of the dataset, we manually examine all the data. We eliminate unnatural conversations or scenarios where user requests can be handled through given APIs. Common cases in removed instances include (1) instances with unchanged or completely removed user utterances, (2) instances where the replaced API was still relevant, and (3) any other instances where the tool could still be invoked correctly. Further details are provided in Appendix B.

	Mistral-7B	Claude-3	ChatGPT	GPT-4
API Replacement				
Acc.	64.69/67.91	68.42/61.90	66.03/64.40	**79.99/82.29**
F1	47.98/52.13	60.78/36.81	54.12/45.14	**74.68/80.93**
Utterance Removal				
Acc.	51.29/51.42	57.59/51.25	52.71/57.90	**73.21/77.44**
F1	14.12/12.33	40.05/5.41	19.80/28.64	**58.20/72.30**

Table 2: Performance evaluation results of LLM by manipulation type. The accuracy (Acc.) and F1 score (F1) are used for evaluation metrics. Both the zero-shot and four-shot performance are presented in a format of (0-shot/4-shot). The highest and the second-highest scores in each metric are highlighted in **bold** and underlined.

Statistics The dataset includes 727 instances from APIBank and 883 from ToolBench, resulting in 1,610. These consist of 900 instances generated by replacing relevant APIs and 710 instances created by removing parts of the utterances, as shown in Table 1.

4 Experimental Results

We set the task where LLMs recognize incomplete conditions as a binary classification problem. We ask the LLM to determine whether APIs can be invoked to fulfill user requests based on conversations and available APIs. To assess whether LLMs can recognize both complete and incomplete conditions, we utilize the instances of two datasets(APIBank and ToolBench) and corresponding manipulated samples. LLMs should answer "Yes" or "No" for the original and manipulated instances, respectively.

For the experiments, we adopt both open-source and proprietary LLMs. The open-source models used are Llama-2 (13B-chat, 70B-chat) (Touvron et al., 2023), Vicuna-v1.5 (13B) (Zheng et al., 2024), and Mistral-Instruct-v0.2 (7B) (Jiang et al., 2023). The following proprietary LLMs are used: Claude-3-Haiku (Anthropic, 2024), ChatGPT (OpenAI, 2023a), and GPT-4 (Achiam et al., 2023). In experiments with instances from ToolBench, we select models with a context length exceeding 8192 tokens to handle the frequently encountered lengthy API descriptions.

Most LLMs struggle to identify incomplete conditions, and proprietary models perform better than Mistral-7B in Table 2. Among the open-source models, Mistral-7B performs the best in scenarios where appropriate APIs are not provided. At the same time, Llama-2-70B excels in scenarios where the user did not provide sufficient information to call APIs. Regarding models from the same family with different sizes, larger models generally exhibit better performance.

[Conversation]
User: Can you check a reminder to call my mom on March 16th?
AI: Sure, Can you confirm your username and password?
User: My username is JohnDoe and my password is pass123.

[API List]
~~{"name":"GetUserToken", "description":"Get the user token", "parameters":["username", "password"]}~~ *(removed)*
{"name":"RegisterUser", "description":"Register a account" "parameters":["name", "password", "email"]} *(inserted)*
{"name":"QueryReminder", "description":"Query a reminder", "parameters":["token", "content", "time"]}

[Explanation]
The available APIs do not include an API for setting reminders. The only relevant API is 'QueryReminder' which is used for querying reminders, but there is no API for setting reminders.

[Conversation]
User: Can you show me the historical events on ~~December 10th~~? *(removed)*

[API List]
{"name":"QueryHistoryToday", "description":"Query the history of the given date", "parameters":["date"]}
{"name":"Calculator", "description":"Provide basic arithmetic operations", "parameters":["formula"]}

[Explanation]
The available APIs don't include an API for showing historical events. The only API provided is for the calculator to perform arithmetic operations.

Figure 2: **Examples of Incorrect Explanation**. The upper illustrates a case of erroneous explanation in API Replacement, while the lower shows in Utterance Removal. Additional examples can be found in the Appendix H

In most cases, models are more vulnerable to Utterance Removal compared to API Replacement. In a zero-shot experiment, the F1 score for Mistral-7B decreases from 47.98% to 14.12%. Similarly, ChatGPT drops from 54.12% to 19.80%, and GPT-4 declines from 74.68% to 58.2%.

We examine the impact of providing the model with a set of examples on its performance. To this end, we provide manually curated few-shot examples, ensuring the classes are evenly balanced. Surprisingly, there is no clear tendency indicating either performance improvement or degradation when using few-shot learning samples. LLMs can be challenging to comprehend incomplete conditions because of the complexity of tool usage conditions (Khot et al., 2022).

5 Analysis

Data Sources We investigate performance by categorizing it based on data sources in Table 3 and Table 4. The performance of models on instances from ToolBench is generally lower compared to their performance on APIBank. Even the most powerful model known to be, GPT-4, shows a low

	Llama-2-13B	Llama-2-70B	Vicuna-13B	Mistral-7B	Claude-3	ChatGPT	GPT-4
API Replacement							
Acc.	53.75/57.07	59.66/58.30	61.87/66.17	69.25/74.91	68.27/65.19	67.77/70.85	**87.58/85.49**
F1	14.55/32.50	36.19/32.34	45.99/64.15	61.66/69.28	62.61/46.30	62.03/64.99	**87.30/84.79**
Utterance Removal							
Acc.	51.29/52.15	52.15/51.12	51.12/54.04	52.32/52.32	61.96/52.50	55.94/65.40	**89.33/88.64**
F1	5.98/18.71	16.77/12.88	18.39/44.95	22.84/20.17	52.27/9.80	36.63/55.23	**89.27/88.70**

Table 3: The performance of models on instances from APIBank.

	Mistral-7B	Claude-3	ChatGPT	GPT-4
API Replacement				
Acc.	60.13/60.90	68.57/58.60	64.29/57.94	**72.40/79.08**
F1	34.30/34.97	58.94/27.31	46.20/25.29	**62.05/77.07**
Utterance Removal				
Acc.	50.26/50.52	53.22/50.00	49.48/50.39	**57.09/66.24**
F1	5.39/4.48	**27.83**/1.02	2.97/2.04	27.13/**55.89**

Table 4: The performance of models on instances from ToolBench.

	API Replacement		Utterance Removal	
	Num.	Acc.	Num.	Acc.
Mistral-7B	149	87.25	19	47.37
ChatGPT	182	85.16	52	48.08
GPT-4	360	97.5	273	99.63

Table 5: The results of explanation for the incomplete scenario. *Num.* represents the count of accurately identified incomplete instances, which corresponds to the number of instances evaluated in the explanation assessment. *Acc.* denotes accuracy of explanations judged by GPT-4, respectively.

F1 score in a zero-shot experiment, with 62.05% in the API Replacement and 27.13% in the Utterance Removal. These tendencies can stem from the complexity of the API descriptions in ToolBench, which consists of real-world APIs that frequently contain unnecessary information or omit essential details (Yuan et al., 2024).

Can Tool-augmented LLMs Correctly Explain for Incomplete Conditions? We probe whether LLMs can accurately explain their decision-making process when they correctly identify incomplete conditions. To this end, we instruct the models to generate explanations for their decisions and assess if these explanations correctly identify why tools cannot be used. We adopt the Judge LLM (i.e., GPT-4) (Zheng et al., 2024) to assess the correctness of the explanations. The agreement between Judge LLM and human annotators for randomly sampled 100 explanations is 82%, detailed in Appendix D.

In Table 5, we observe that ChatGPT achieves an accuracy of 48.08% for Utterance Removal, while Mistral-7B shows a similar performance with an accuracy of 47.37%. For API Replacement, the accuracy rates of ChatGPT and Mistral-7B are 85.16% and 87.25%. This indicates that it is more difficult for LLMs to provide accurate explanations when users offer insufficient context compared to when the necessary tools are not provided.

Additionally, we examine instances where the LLMs generated incorrect reasoning, as shown in Figure 2. In API Replacement, LLMs often misunderstand the user's intent, leading to inaccurate assertions that the existing APIs are insufficient. Conversely, in Utterance Removal, the predominant errors come from incorrect explanations stating that no appropriate APIs are available despite the presence of appropriate APIs.

6 Conclusion

In this work, we investigate the capability of tool-augmented LLMs to recognize incomplete conditions. Specifically, we examine scenarios where users only provide partial information, or the required tools are inaccessible. Experimental results, based on carefully manipulated datasets to simulate scenarios with missing tools or essential user information, reveal a significant issue: most LLMs struggle to detect and elaborate incompleteness, particularly when dealing with complex tools. It shows that there is still room for improvement in their ability to identify and explain the reasons behind tools' unusability consistently. We believe addressing these issues is crucial for advancing the reliability and safety of LLMs, thereby preventing potential problems such as hallucination and risky actions in real-world applications.

Limitations

While we explore the ability of tool-augmented LLMs to abstain from tool usage in incomplete conditions, our study has several limitations. First, the data annotation process primarily relied on model-based annotations, although human verification was conducted. This approach might differ from actual cases where humans provide incomplete information regarding tool usage. Additionally, our focus was solely on API-based tools, which, although significant, do not encompass the full spectrum of tools such as plugins, robotic systems, and other interactive systems. Despite these limitations, our research underscores the importance of developing reliable tool-augmented LLMs, highlighting the need for further advancements in this area.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

AI Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku. *Claude-3 Model Card*.

Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. 2023. Large language models as tool makers. In *The Twelfth International Conference on Learning Representations*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

Jeremy Cole, Michael Zhang, Daniel Gillick, Julian Eisenschlos, Bhuwan Dhingra, and Jacob Eisenstein. 2023. Selectively answering ambiguous questions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 530–543, Singapore. Association for Computational Linguistics.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. 2023. Chain-of-verification reduces hallucination in large language models. *arXiv preprint arXiv:2309.11495*.

Yulan Feng, Shikib Mehri, Maxine Eskenazi, and Tiancheng Zhao. 2020. "none of the above": Measure uncertainty in dialog response retrieval. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2013–2020, Online. Association for Computational Linguistics.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. 2024. Stabletoolbench: Towards stable large-scale benchmarking on tool learning of large language models. *arXiv preprint arXiv:2403.07714*.

Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. 2024. Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings. *Advances in neural information processing systems*, 36.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2023. Metagpt: Meta programming for a multi-agent collaborative framework. *Preprint*, arXiv:2308.00352.

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: Deciding whether to use tools and which to use. In *The Twelfth International Conference on Learning Representations*.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C Park. 2024. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. *arXiv preprint arXiv:2403.14403*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Amita Kamath, Robin Jia, and Percy Liang. 2020. Selective question answering under domain shift. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5684–5696, Online. Association for Computational Linguistics.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2022. Decomposed prompting: A modular approach for solving complex tasks. In *The Eleventh International Conference on Learning Representations*.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023. API-bank: A comprehensive benchmark for tool-augmented LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3102–3116, Singapore. Association for Computational Linguistics.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2023. Awq: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978*.

Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. 2024. Chameleon: Plug-and-play compositional reasoning with large language models. *Advances in Neural Information Processing Systems*, 36.

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. AmbigQA: Answering ambiguous open-domain questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5783–5797, Online. Association for Computational Linguistics.

OpenAI. 2023a. Chatgpt: A large language model developed by openai. `https://www.openai.com/chatgpt`. Accessed: 2024-06-16.

OpenAI. 2023b. Prompt engineering. `https://platform.openai.com/docs/guides/prompt-engineering`. Accessed: 2024-06-16.

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

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

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.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Zhiruo Wang, Zhoujun Cheng, Hao Zhu, Daniel Fried, and Graham Neubig. 2024. What are tools anyway? a survey from the language model perspective. *arXiv preprint arXiv:2403.15452*.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. 2022. Large language models are better reasoners with self-verification. *arXiv preprint arXiv:2212.09561*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

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*.

Hui Yang, Sifu Yue, and Yunzhong He. 2023. Auto-gpt for online decision making: Benchmarks and additional opinions. *arXiv preprint arXiv:2306.02224*.

Siyu Yuan, Kaitao Song, Jiangjie Chen, Xu Tan, Yongliang Shen, Ren Kan, Dongsheng Li, and Deqing Yang. 2024. Easytool: Enhancing llm-based agents with concise tool instruction. *arXiv preprint arXiv:2401.06201*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2024. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36.

	API Replacement		Utterance Removal	
	APIBank	ToolBench	APIBank	ToolBench
Avg. Turns	6.18	1	5.18	1
Avg. APIs	2.13	5.13	2.05	5.51
Avg. Utterance Words	17.66	52.38	18.87	51.15
Avg. API Length	434.39	713.35	443.38	747.53

Table 6: **Dataset Statistics**. *Avg. Turns* denote the number of utterances in a conversation. *Avg. APIs* denote the number of APIs in the instance. *Avg. API Length* denotes the string length of each API description.

A Data Source

APIBank Li et al. (2023) consists of 73 APIs and 314 manually annotated multi-turn conversations between human and AI systems. We use 450 instances in the dataset's test split, except for those that require a tool-retrieval module.

ToolBench Qin et al. (2023) is constructed based on 16,000 real-world APIs in 49 categories from RapidAPI Hub[2]. Based on the dataset, Guo et al. (2024) releases more cleaned and solvable instances that user requests can be addressed by tool invocation. We use the 764 test instances in different categories (i.e., single-tool instructions (I1), intra-category multi-tool instructions (I2), and intra-collection multi-tool instructions (I3)) filtered by Guo et al. (2024) as the original instances.

B Data Manipulation

Relevant API Replacement The relevant APIs in the available API list are replaced with other APIs in API pools. To find semantically similar to the relevant APIs, we adapt a sentence encoder (i.e., Gao et al. (2021)[3]) (Patil et al., 2023; Li et al., 2023; Qin et al., 2023). Specifically, we first concatenate the name and description of each API in a dataset and convert this text into a fixed-size vector. We then choose one of the most similar 10 APIs by calculating the cosine similarity between relevant APIs and all APIs presented in the dataset. Note that the API pools for this retrieval process are separately constructed for each dataset. The modified instances, successful and failed cases, are presented in Fig. 4 to Fig. 7.

Partial Removal of User Utterance This manipulation aims to eliminate essential information from user utterances to make appropriate API invocation infeasible. In this procedure, we utilize the advanced proprietary language model, GPT-4. The model identifies and removes critical information required for tool invocation within the user's utterance. We incorporate a reasoning prompt and provide five manually designed demonstration examples to enhance the quality of the dataset. The modified instance samples are presented in Fig. 8 to Fig. 11. Detailed dataset statistics are presented in Table 6.

Human Verification In a preliminary study, we observed that the instances automatically generated by the processes above (Appendix B) sporadically do not represent incomplete scenarios for tool-augmented LLMs. For instance, the newly replaced APIs can still address the user instruction by executing similar operations. In the case of Utterance Removal, the modified utterance may still contain sufficient information to invoke the tool accurately. Additionally, the manipulated conversations generated by the LLM occasionally fail to maintain a natural flow, completely removing an utterance.

To handle these issues, we manually conduct a validation process to ensure that only incomplete conditions remain in the manipulated instances. The detailed filtering criteria for the verification of *Relevant API Replacement* are as follows: (1) The user's request can be fulfilled using alternative APIs. (2) The user's request can be fulfilled using the non-replaced APIs for instances from ToolBench. The detailed filtering criteria for the verification of *Partial Removal of User Utterance* are as follows: (1) Only non-essential information is removed, allowing the APIs to be invoked to fulfill the user's request. (2) One of the user's multiple requests is removed, allowing the remaining requests to be resolved through existing API invocations. (3) The conversation context becomes unnatural or grammatically incorrect. (4) The result is identical to the original. (5) It is unnatural for the model to execute an API call in the subsequent turn because the subject of the last utterance in the conversation is not the user. Two authors with a bachelor's degree in relevant fields participated in the verification process with active discussion.

C Models

Both open-source and proprietary LLMs are used in our experiments. All open-source LLMs are implemented with the Transformers library (Wolf et al., 2020). The quantized checkpoints of Llama-2-70B-hf and Llama-2-70B-chat are used for ex-

[2] https://rapidapi.com/hub
[3] princeton-nlp/sup-simcse-roberta-large

periments (Lin et al., 2023). For proprietary models, we use `claude-3-haiku-20240307`, `gpt-3.5-turbo-0125`, and `gpt-4-0613` for Claude-3-Haiku, ChatGPT, and GPT-4, and the temperature is set to 0. We use two original and two manipulated instances for the evaluation with few-shot examples.

D Implementation Details in Analysis

Our approach involves instructing the models to determine whether they could call the APIs and the reasoning behind their decision on incomplete samples of APIBank. To assess the validity of these explanations, we employ GPT-4, a powerful model, as the Judge LLM (Zheng et al., 2024). As illustrated in Fig. 19, the Judge LLM determines the correctness of the responses generated by tool-augmented LLMs. We manually crafted four-shot examples with a balanced class distribution for a more reliable evaluation.

To verify Judge LLM's reliability in assessing explanation validity, we manually annotate the 100 randomly sampled instances from ChatGPT predictions, evenly divided between API Replacement and Utterance Removal. The agreement rate between human evaluators and Judge LLM is 82%. This high level of agreement indicates that Judge LLM produces evaluation results that are closely aligned with human evaluators, establishing Judge LLM as credible and effective.

E Results of LLMs without Instruction Tuning

We compare the performance of pre-trained models and instruct-tuned models on the evaluation setup described in Section 4. We compare Llama-2-13B-chat and Llama-2-70B-chat against Llama-2-13B-hf and Llama-2-70B-hf. Results are shown in Table 10. The models not trained on instruction data generally outperform their counterparts. These trends are consistently shown in different model sizes and the number of demonstration examples. More exploration of the relationship between the awareness of incompleteness for tool invocation and the training on instruction-following datasets are left as our future work.

F Token probability of "Yes"

We analyze the predictive distribution of the models on the "Yes" token to understand the tendencies of LLMs in incomplete conditions. Specifically, we examine the ratio at which LLMs indicate they can call APIs in our main experiments. As shown in Table 8 and Table 9, most models tend to determine that tools can be appropriately called, which is especially noticeable when the models perform poorly.

We also visualize the predictive distribution from two LLMs on the complete and incomplete instances. We use the probability of verbalized token predicted as if tool invocation is available. As shown in Fig. 3, GPT-4 clearly distinguishes complete and incomplete conditions, while ChatGPT fails to discern corrupted instances.

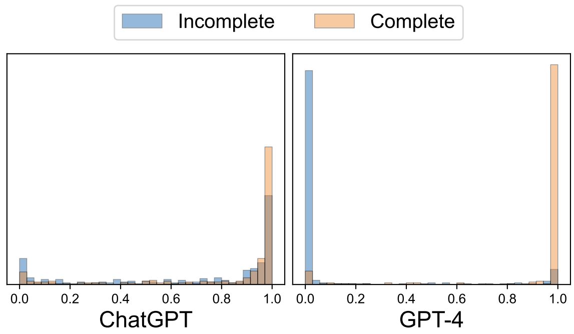

Figure 3: The probability distribution of "Yes" token for complete and incomplete instances. The utterance manipulation instances from APIBank are used for visualization.

Utterance Removal	API Call	ChatGPT	GPT-4
O	**Error**	96.0	86.0
O	**Hallucination**	7.0	67.0
X	**Error**	13.0	99.0

Table 7: The results for the self-verification. We use accuracy as an evaluation metric.

G Self-verification with API Invocation Results

LLMs can improve their performance by verifying their own responses (Weng et al., 2022; Dhuliawala et al., 2023). This capability is also applied to scenarios where LLMs call external tools to address users' requests (Qin et al., 2023). Building on this foundation, we investigate whether the self-checking ability of tool-augmented LLMs can be further refined to enhance their capacity for recognizing incomplete conditions.

First, we randomly sample 100 instances from our main experiment, in which ChatGPT correctly identified in Utterance Removal. We then present these instances to the LLMs with the following

	Llama-2-13B	Llama-2-70B	Vicuna-13B	Mistral-7B	Claude-3	ChatGPT	GPT-4
API Replacement							
Yes Ratio (%)	24.35/0.0	57.8/62.0	78.13/54.85	68.56/67.27	64.62/84.16	64.18/65.84	51.65/53.78
Utterance Removal							
Yes Ratio (%)	24.34/0.0	65.62/62.71	88.82/65.13	87.04/89.28	69.57/96.88	78.29/72.20	50.49/49.34

Table 8: Predictive distribution on main experiments of APIBank. We measure *Yes Ratio*, which represents the proportion where the model predicts that it can invoke the APIs. We report the distribution for both zero-shot and four-shot in a format of (0-shot/4-shot).

	Mistral-7B	Claude-3	ChatGPT	GPT-4
API Replacement				
Yes Ratio	87.74/88.79	70.69/91.91	81.37/92.74	75.74/57.25
API Replacement				
Yes Ratio	97.16/98.22	84.41/99.51	97.65/99.13	91.09/73.33

Table 9: Predictive distribution on main experiments of ToolBench. The indicators are the same as Table 8.

scenarios: 1) *Error in Utterance Removal*: Cases where an API call results in an error due to missing information. 2) *Hallucination in Utterance Removal*: Instances where APIs are invoked with information not provided in the dialogue context, but the API calls are successful. 3) *Error in Complete Condition*: Conditions where the API could be appropriately called, but the call failed, resulting in an error.

For scenarios 1 and 3, we ask the LLMs to determine whether they can now call APIs. For scenario 2, we ask the LLMs to determine whether they can now respond appropriately to users. Specifically, in scenario 1, the LLMs should recognize that an API invocation cannot be made. In scenario 2, they should acknowledge that they cannot provide a suitable response to the user, and in scenario 3, they should identify that API calls can be made.

When API call results are erroneous, ChatGPT tends to determine that the API cannot be invoked, regardless of the actual feasibility of making appropriate API calls, as shown in Table 7. Conversely, GPT-4 exhibits a superior capacity to ChatGPT for assessments of the feasibility of API calls. On the other hand, when necessary information to invoke APIs is not provided but the API call is still successful, both ChatGPT and GPT-4 demonstrate limited awareness of hallucinations. This indicates that when the models make API calls based on incomplete information, but no error occurs, they struggle to recognize the inaccuracy of the ostensibly successful result.

H Samples of Incomplete Instances and Explanation

We present the manipulated instances with different strategies are presented from Fig. 4 to Fig. 11. Both accurately and inaccurately modified instances resulting from our dataset construction method are provided.

We also present additional explanation examples generated by LLMs when they recognize incomplete conditions on manipulated samples. Examples of explanations in API Replacement are illustrated in Fig. 12 and Fig. 13. Examples of explanations in Utterance Removal are depicted in Fig. 14 and Fig. 15.

I Prompt Templates

The text prompt used in the dataset construction is presented in Fig. 16. The text prompt used in the main experiments (Table 2, Table 3 and Table 4) is presented in Fig. 17. The text prompts used in the experiments of LLM explanation and LLM judgment are presented in Fig. 18 and Fig. 19 respectively. The text prompts used in Section G are in Fig. 20 and Fig. 21.

When implementing few-shot prompting, we follow the approach of setting up interactions between a human and an AI assistant to provide examples (OpenAI, 2023b).

	Llama-2-13B	Llama-2-13B$_{\text{BASE}}$	Llama-2-70B	Llama-2-70B$_{\text{BASE}}$
API Replacement				
Acc.	53.75/57.07	54.86/49.57	59.66/58.30	60.52/62.98
F1	14.55/32.50	64.05/66.28	36.19/32.34	56.68/57.43
Utterance Removal				
Acc.	51.29/52.15	53.01/49.74	52.15/51.12	54.91/60.59
F1	5.98/18.71	62.65/66.44	16.77/12.88	45.87/54.11

Table 10: Experiments on a variety of Llama-2 for instances from APIBank. We measure accuracy (Acc.) and F1 score (F1). We report both the zero-shot and four-shot performance in a format of (0-shot/4-shot). The models not trained on supervised instruction-following datasets are marked with subscript (i.e., BASE).

[Conversation]

User: Can you tell me about the stock price of SQ on March 15th, 2022?

[Relevant API (Removed)]


```
{\"name\": \"QueryStock\", \"description\": \"This API queries the
stock price of a given stock code and date.\", \"input_parameters\":
{\"stock_code\": {\"type\": \"str\", \"description\": \"The stock
code of the given stock.\"}, \"date\": {\"type\": \"str\",
\"description\": \"The date of the stock price. Format: %Y-%m-%d\"}},
\"output_parameters\": {\"stock_price\": {\"type\": \"float\",
\"description\": \"The stock price of the given stock.\"}}}
```


[Replaced Irrelevant API]


```
{'name': 'QueryBalance', 'description': 'This API queries the
balance of a given user.', 'input_parameters': {'token': {'type':
'str', 'description': 'The token of the user.'}},
'output_parameters': {'balance': {'type': 'float', 'description':
'The balance of the user.'}}}
```


Figure 4: *API Replacement* Successful case from APIBank instances.

[Conversation]

User: Can you tell me today's date?

[API List]


```
{'name': 'AddAgenda', 'description': 'The API for adding a schedule
item includes parameters for token, content, time, and location.',
'input_parameters': {'token': {'type': 'str', 'description': "User's
token."}, 'content': {'type': 'str', 'description': 'The content of
the agenda.'}, 'time': {'type': 'str', 'description': 'The time for
agenda. Format: %Y-%m-%d %H:%M:%S'}, 'location': {'type': 'str',
'description': 'The location of the agenda.'}}, 'output_parameters':
{'status': {'type': 'str', 'description': 'success or failed'}}}
```


[LLM Explanation] *(ChatGPT)*

The available APIs do not include an API specifically for retrieving today's date. The AddAgenda API is for adding a schedule item and does not provide information on the current date.

Figure 5: *API Replacement* Successful case from ToolBench instances.

[Conversation]

User: Can you search the definition of the word \"plethora\" for me?
AI: Sure, searching for \"plethora\"...
API-Request: [Dictionary(keyword='plethora')]->[{'partOfSpeech': 'noun', 'definitions': [{'definition': '(usually followed by of) An excessive amount or number; an abundance.', 'synonyms': [], 'antonyms': [], 'example': 'The menu offers a plethora of cuisines from around the world.'}, {'definition': 'An excess of red blood cells or bodily humours.', 'synonyms': [], 'antonyms': []}], 'synonyms': ['glut', 'myriad', 'slew', 'superfluity', 'surfeit'], 'antonyms': []}]
AI: Here's what I found: \"a large or excessive amount of something.\"
Can I help you with anything else?\
User: Yes, can you give me the synonyms for \"plethora\"?

[Relevant API (Removed)]

```
{\"name\": \"Dictionary\", \"description\": \"This API searches the dictionary for a given keyword.\", \"input_parameters\": {\"keyword\": {\"type\": \"str\", \"description\": \"The keyword to search.\"}}, \"output_parameters\": {\"results\": {\"type\": \"list\", \"description\": \"The list of results. Format be like [{\\\"partOfSpeech\\\": \\\"xxx\\\", \\\"definitions\\\": [{\\\"definition\\\": \\\"xxx\\\", \\\"example\\\": \\\"xxx\\\", \\\"synonyms\\\": [\\\"xxx\\\", \\\"xxx\\\"]}, ...]\"}}}
```

[Replaced Irrelevant API]

```
{'name': 'SearchEngine', 'description': 'This API searches for a given keyword for search engine.', 'input_parameters': {'keyword': {'type': 'str', 'description': 'The keyword to search.'}}, 'output_parameters': {'results': {'type': 'list', 'description': 'The list of results.'}}}
```

Figure 6: *API Replacement* Failed case from APIBank instances.

[Conversation]
User: I'm creating a database and I need unique identifiers for each entry. Can you help me generate UUIDs?

[Relevant API (Removed)]

```
{\"category_name\": \"Tools\", \"tool_name\": \"Helper Function\",
\"api_name\": \"Random\", \"api_description\": \"Generate Random
String with custom length\\nBASIC: 20 \\nPRO: 30\\nULTRA: 70\\nMEGA:
150\", \"required_parameters\": [], \"optional_parameters\":
[{\"name\": \"length\", \"type\": \"NUMBER\", \"description\": \"\",
\"default\": \"20\"}], \"method\": \"GET\", \"template_response\":
{\"result\": \"str\"}}
{\"category_name\": \"Tools\", \"tool_name\": \"Helper Function\",
\"api_name\": \"Generate UUID\", \"api_description\": \"Generate
UUID v4 with dash or not.\\nmax\\nBASIC: 3\\nPRO: 20\\nULTRA:
50\\nMEGA: 150\", \"required_parameters\": [],
\"optional_parameters\": [{\"name\": \"timestamp_first\", \"type\":
\"BOOLEAN\", \"description\": \"\", \"default\": \"false\"},
{\"name\": \"remove_dash\", \"type\": \"BOOLEAN\", \"description\":
\"\", \"default\": \"true\"}, {\"name\": \"count\", \"type\":
\"STRING\", \"description\": \"\", \"default\": \"5\"}], \"method\":
\"GET\", \"template_response\": {\"result\": [\"list of str with
length 3\"]}}
```


[Replaced Irrelevant API]

```
{'category_name': 'Tools', 'tool_name': 'Random Strings',
'api_name': 'UUID', 'method': 'GET', 'required_parameters': [],
'optional_parameters': [], 'api_description': 'Generates a random
version 4 UUID.’}
{'category_name': 'Tools', 'tool_name': 'QR Code Generator',
'api_name': 'Generate Basic - Direct Image', 'method': 'GET',
'required_parameters': [{'name': 'data', 'type': 'STRING',
'description': '', 'default': '1234'}], 'optional_parameters':
{'name': 'size', 'type': 'NUMBER', 'description': '', 'default':
'500'}], 'api_description': \"Generates a QR code as a direct image
with limited settings. (NOTE: doesn't show correctly in RapidAPI)\"}
```


Figure 7: *API Replacement* Failed case from ToolBench instances.

[Conversation]
User: Hi, can you help me check my alarm for March 20th, 2023 at 6:30AM?
AI: Sure, to access your alarm information, I'll need to authenticate. Can you please provide me with your email, username, and password?
User: My email is janesmith@example.com, username is JaneSmith, and password is password.

[Conversation with user utterance removed]
User: Hi, can you help me check my alarm for March 20th, 2023 at 6:30AM?
AI: Sure, to access your alarm information, I'll need to authenticate. Can you please provide me with your email, username, and password?
User: My email is janesmith@example.com, username is JaneSmith.

[Relevant API]

```
{\"name\": \"GetUserToken\", \"description\": \"Get the user token
by username and password.\", \"input_parameters\": {\"username\":
{\"type\": \"str\", \"description\": \"The username of the user.\"},
\"password\": {\"type\": \"str\", \"description\": \"The password
of the user.\"}}, \"output_parameters\": {\"token\": {\"type\":
\"str\", \"description\": \"The token of the user.\"}}}
{\"name\": \"QueryAlarm\", \"description\": \"The API for querying
alarm clock, help user to check the alarm clock they have set.\",
\"input_parameters\": {\"token\": {\"type\": \"str\",
\"description\": \"User's token.\"}, \"time\": {\"type\": \"str\",
\"description\": \"The time for alarm. Format: %Y-%m-
%d %H:%M:%S\"}}, \"output_parameters\": {\"info\": {\"type\":
\"json\", \"description\": \"alarm info including username, time,
content and location\"}}}
```


Figure 8: Successful case in *Utterance Removal*

[Conversation]
User: Can you help me analyze the number of pages indexed in Google for my company's website? The URL is 'www.example.com'.

[Conversation with user utterance removed]
User: Can you help me analyze the number of pages indexed in Google for my company's website?

[Relevant API]

```
{\"category_name\": \"Tools\", \"tool_name\": \"SEO - Count website
pages in Google index\", \"api_name\": \"countPages\",
\"api_description\": \"Count website pages in Google index\",
\"required_parameters\": [{\"name\": \"url\", \"type\": \"STRING\",
\"description\": \"\", \"default\": \"\"}], \"optional_parameters\":
[], \"method\": \"GET\"}
```


Figure 9: Successful case in *Utterance Removal*.

[Conversation]
User: Can you search for news articles about sustainable development goals?
AI: Sure, what specific keyword are you looking for?
User: the sustainable development.

[Conversation with user utterance removed]
User: Can you search for news articles about sustainable development goals?
AI: Sure, what specific keyword are you looking for?
User: Can you help me find information?

[Relevant API]

```
{\"name\": \"SearchEngine\", \"description\": \"This API searches
for a given keyword for search engine.\", \"input_parameters\":
{\"keyword\": {\"type\": \"str\", \"description\": \"The keyword to
search.\"}}, \"output_parameters\": {\"results\": {\"type\":
\"list\", \"description\": \"The list of results.\"}}}
```


Figure 10: Failed case in *Utterance Removal*.

[Conversation]
User: I am working on a project that requires unique IDs for each document. Can you assist me in generating GUIDs for the documents? I would need 50 GUIDs in total.

[Conversation with user utterance removed]
User: I am working on a project that requires unique IDs for each document. Can you assist me in generating GUIDs for the documents?

[Relevant API]

```
{\"category_name\": \"Tools\", \"tool_name\": \"GUID generator\",
\"api_name\": \"GenerateGuid\", \"api_description\": \" \",
\"required_parameters\": [], \"optional_parameters\": [],
\"method\": \"GET\"} {\"category_name\": \"Tools\", \"tool_name\":
\"GUID generator\", \"api_name\": \"BulkGenerateGuids\",
\"api_description\": \" \", \"required_parameters\": [],
\"optional_parameters\": [{\"name\": \"batch_size\", \"type\":
\"NUMBER\", \"description\": \"The number of GUIDs to return. Must
be between 1 and 10000. If the parameter is not provided, the
default batch size is 20.\", \"default\": \"\"}], \"method\":
\"GET\"}
```


Figure 11: Failed case in *Utterance Removal*.

[Conversation]
User: Can you tell me today's date?

[API List]
```
{'name': 'AddAgenda', 'description': 'The API for adding a schedule
item includes parameters for token, content, time, and location.',
'input_parameters': {'token': {'type': 'str', 'description': "User's
token."}, 'content': {'type': 'str', 'description': 'The content of
the agenda.'}, 'time': {'type': 'str', 'description': 'The time for
agenda. Format: %Y-%m-%d %H:%M:%S'}, 'location': {'type': 'str',
'description': 'The location of the agenda.'}}, 'output_parameters':
{'status': {'type': 'str', 'description': 'success or failed'}}}
```

[LLM Explanation] *(ChatGPT)*
The available APIs do not include an API specifically for retrieving today's date. The AddAgenda API is for adding a schedule item and does not provide information on the current date.

Figure 12: Explanation Success Case in *API Replacement*. The model clearly states and explains that an API to know current date is not provided.

[Conversation]
User: Can you record my blood pressure and heart rate data?
AI: Sure, I can help you with that. Please provide me with your user ID, time, blood pressure, and heart rate data.
User: My user ID is 1234, time is 2022-01-01 10:00:00, blood pressure is 110/70, and heart rate is 90.

[API List]
```
{'name': 'QueryHealthData', 'description': 'This API queries the
recorded health data in database of a given user and time span.',
'input_parameters': {'user_id': {'type': 'str', 'description': 'The
user id of the given user. Cases are ignored.'}, 'start_time':
{'type': 'str', 'description': 'The start time of the time span.
Format: %Y-%m-%d %H:%M:%S'}, 'end_time': {'type': 'str',
'description': 'The end time of the time span. Format: %Y-%m-
%d %H:%M:%S'}}, 'output_parameters': {'health_data': {'type':
'list', 'description': 'The health data of the given user and time
span.'}}}
```

[LLM Explanation] *(ChatGPT)*
The available API "QueryHealthData" allows querying recorded health data based on user ID and a time span. However, the conversation between the user and AI only provided the user ID, time, blood pressure, and heart rate data for a specific timestamp (2022-01-01 10:00:00). The API requires a start and end time for the time span, which are not provided in the conversation. Therefore, the current information is not sufficient to generate API calls to record the blood pressure and heart rate data.

Figure 13: Example of failed explanation in *API Replacement*. The user wants to record health data, but only the *QueryHealthData* API, which is for searching health data, is provided instead of an API for recording health data. The model incorrectly assumes that it must use the *QueryHealthData* API and provides an erroneous explanation that it cannot call the API due to insufficient information.

[Conversation]

User: Can you help me find possible diseases related to a symptom I'm experiencing?
AI: Sure, I can search for possible diseases related to fatigue. Please provide the symptom you want to search for.
User: I prefer not to specify right now.

[API List]

{'name': 'SymptomSearch', 'description': 'This API searches for a given symptom.', 'input_parameters': {'symptom': {'type': 'str', 'description': 'The symptom to search.'}}, 'output_parameters': {'results': {'type': 'list', 'description': 'The list of results. Format be like [{"name":possible disease name, "description": disease details},...]'}}}
{'name': 'AppointmentRegistration', 'description': 'This API registers an appointment of hospital.', 'input_parameters': {'patient_name': {'type': 'str', 'description': 'The name of patient.'}, 'date': {'type': 'str', 'description': 'The date of appointment. Format be like %Y-%m-%d'}, 'doctor_name': {'type': 'str', 'description': 'The name of appointed doctor.'}}, 'output_parameters': {'appointment_id': {'type': 'str', 'description': 'The ID of appointment.'}}}

[LLM Explanation] *(ChatGPT)*

The available API "SymptomSearch" requires the user to provide the specific symptom to search for. Since the user preferred not to specify the symptom at the moment, we do not have enough information to make an API call to search for possible diseases related to the symptom.

Figure 14: Example of successful explanation in *Utterance Removal*. The model recognizes the need to call the *SymptomSearch* API to address the user's requirements. Additionally, it clearly acknowledges and explains that the necessary symptom information required to call the API has not been provided.

[Conversation]

User: Can you please check my account balance?
AI: Sure, I can help you with that. I just need your token. Do you have your token with you?
User: No, I don't have it.
AI: That's alright. I can get your token for you. Can you please confirm your username, password, and email address?
User: My username is user2 and email is user2@example.com.

[API List]

{'name': 'OpenBankAccount', 'description': 'This is an API for opening a bank account for a user, given the account, password and name.', 'input_parameters': {'account': {'type': 'str', 'description': 'The account for the user.'}, 'password': {'type': 'str', 'description': 'The password.'}, 'name': {'type': 'str', 'description': 'account holder name.'}}, 'output_parameters': {'status': {'type': 'str', 'description': 'success or failed'}}}
{'name': 'QueryStock', 'description': 'This API queries the stock price of a given stock code and date.', 'input_parameters': {'stock_code': {'type': 'str', 'description': 'The stock code of the given stock.'}, 'date': {'type': 'str', 'description': 'The date of the stock price. Format: %Y-%m-%d'}}, 'output_parameters': {'stock_price': {'type': 'float', 'description': 'The stock price of the given stock.'}}}
{'name': 'QueryBalance', 'description': 'This API queries the balance of a given user.', 'input_parameters': {'token': {'type': 'str', 'description': 'The token of the user.'}}, 'output_parameters': {'balance': {'type': 'float', 'description': 'The balance of the user.'}}}
{'name': 'GetUserToken', 'description': 'Get the user token by username and password.', 'input_parameters': {'username': {'type': 'str', 'description': 'The username of the user.'}, 'password': {'type': 'str', 'description': 'The password of the user.'}}, 'output_parameters': {'token': {'type': 'str', 'description': 'The token of the user.'}}}

[LLM Explanation] *(ChatGPT)*

The conversation between the user and AI did not involve obtaining the user's token, which is required to query the account balance using the "QueryBalance" API. The user provided their username and email address, but without the token, we cannot make an API call to check the account balance.

Figure 15: Example of failed explanation in *Utterance Removal*. The user's requirement is to check the account balance, which necessitates a token. However, the user has not provided the password required to obtain the token. Although an API exists to obtain the token, the model fails to recognize this and incorrectly states that the API call cannot be made due to the absence of the token.

[System Prompt]
You are an expert in data annotation with a specific task of processing user requests, corresponding API descriptions, and API calls.
Your goal is to remove specific information from the user requests to sound natural while ensuring the API call fails due to missing information.
Follow these steps to achieve this:

1. Analyze User Requests and API Information:
- Examine the relationship between the user's request, the API description, and the actual API call.
- Identify the parameters and details the API requires to function correctly.

2. Generate the Thought Process:
- Before making any modifications, articulate your thought process.
- Select information from the user utterances that can be removed while keeping the dialogue context intact.
- Ensure you only remove information relevant to the API descriptions and calls provided.

3. Modify the User Request:
- Adjust the user request so that it remains natural-sounding but lacks the necessary information for the API call.
- Do not alter or remove any parts of the dialogue other than the essential information identified.

Output the results in the following JSON format:

```
{
"thought": "YOUR_THOUGHT_PROCESS",
"answer": "MODIFIED_CONVERSATION"
}
```

[User Prompt]
API_Description:
{api_description}

Conversation:
{conversation}

Figure 16: Data Construction Prompt Template.

[System Prompt]
I will provide you with a user's query, and you should assess, based on your own capabilities, whether the current information is sufficient to generate API calls for addressing the user's query. The information includes (a) conversation between user and AI (you) and (b) the list of available APIs.
You can't use APIs other than the given APIs.
Based on the given query, if you think you can successfully address the user's request by making an API call, please respond with "Yes"; otherwise, respond with "No".

[User Prompt]
Here is the previous conversation between User and AI (you):
[Conversation Begins]
{conversation}
[Conversation Ends]

Here is the list of available APIs:
[List of APIs Begins]
{api_list}
[List of APIs Ends]

Can you address the user's request by making an API call now?
Please answer in "Yes" or "No".

Figure 17: Main Experiments Prompt Template.

[System Prompt]
I will provide you with a user's query, and you should assess, based on your own capabilities, whether the current information is sufficient to generate API calls for addressing the user's query. The information includes (a) conversation between user and AI (you) and (b) the list of available APIs.
You can't use APIs other than the given APIs.
Based on the given query, if you think you can successfully address the user's request by making an API call, please respond with "Yes"; otherwise, respond with "No".
Begin your response your decision ("Yes" or "No") and then output explanations behind your decision.
Follow this format: Decision: <Yes/No> Explanation: <your explanation>

[User Prompt]
Here is the previous conversation between User and AI (you):
[Conversation Begins]
{conversation}
[Conversation Ends]

Here is the list of available APIs:
[List of APIs Begins]
{api_list}
[List of APIs Ends]

Figure 18: Prompt template of explanation.

[System Prompt]
You are a judge evaluating the appropriateness of the AI Assistant's response.
The AI Assistant was tasked with determining whether it could create an API call to solve a user's query, based on the provided APIs and conversations.
There isn't enough information available to actually perform the API call.
Your role is to assess if the AI Assistant's reasoning for being unable to create the API call is valid.
If the necessary information is available in the APIs and conversations but the AI Assistant claims it isn't, or if the AI Assistant states that unnecessary information to call API is required, you should judge the response as incorrect.
Output "Correct" if the AI Assistant's reasoning is valid, and "Wrong" if it is not.

[User Prompt]
[Conversation Begins]
{conversation}
[Conversation Ends]

[List of APIs Begins]
{api_list}
[List of APIs Ends]

[AI Assistant Explanation Begins]
{explanation}
[AI Assistant Explanation Ends]

Figure 19: Prompt template of Judge LLM.

[System Prompt]
I will provide you with a user's query, and you should assess, based on your own capabilities, whether the current information is sufficient to generate API calls for addressing the user's query. The information includes (a) conversation between user and AI (you), (b) the list of available APIs and (c) the previous API call made to address the user's request (if applicable).
You can't use APIs other than the given APIs.
Based on the given query, if you think you can successfully address the user's request by making an API call, please respond with "Yes"; otherwise, respond with "No".

[User Prompt]
Here is the previous conversation between User and AI (you):
[Conversation Begins]
{conversation}
[Conversation Ends]

Here is the list of available APIs:
[List of APIs Begins]
{api_list}
[List of APIs Ends]

This is not your first attempt, and the previous attempt has failed.
In your previous attempt, you determined that you could create an API call to address user's request, and you created an API Call in subsequent steps.
Before determining if you can now address the user's request with an API call, you must review and analyze the API calls you have previously made:
[Previous API Call Begins]
{api_call}
[Previous API Call Ends]

Can you address the user's request by making an API call now?
Please answer in "Yes" or "No".

Figure 20: Prompt Template of self-verification for API invocation error. We use the same prompt for error in utterance removal and complete scenario. Both receive information about an erroneous API call result and are asked whether the API call is currently feasible.

[System Prompt]
I will provide you with a user's query, and you should assess, based on your own capabilities, whether the current information is sufficient and appropriate to answer the user's query.
The information includes (a) conversation between user and AI (you), (b) the list of available APIs, and (c) the API call generated by AI(you) in the previous step to address the user's query.
No API other than the given API is available.
Based on the given information, if you think you can provide an appropriate answer to user's request, please response with "Yes"; otherwise, respond with "No".

[User Prompt]
Here is the previous conversation between User and AI (you):
[Conversation Begins]
{conversation}
[Conversation Ends]

Here is the list of available APIs:
[List of APIs Begins]
{api_list}
[List of APIs Ends]

Here is the API calls you have previously made
[Previous API Call Begins]
{api_call}
[Previous API Call Ends]

Can you provide an appropriate answer to the user now?
Please answer in "Yes" or "No".

Figure 21: Prompt Template of self-verification for hallucination.