CToolEval: A Chinese Benchmark for LLM-Powered Agent Evaluation in Real-World API Interactions

Zishan Guo[*] , **Yufei Huang**[*] , **Deyi Xiong** [†]
College of Intelligence and Computing, Tianjin University, Tianjin, China
{guozishan, yuki_731, dyxiong}@tju.edu.cn

Abstract

Assessing the capabilities of large language models (LLMs) as agents in decision making and operational tasks is crucial for the development of LLM-as-agent service. We propose CToolEval, a benchmark designed to evaluate LLMs in the context of Chinese societal applications, featuring 398 APIs across 27 widely-used Apps (e.g., Apps for shopping, map, music, travel, etc.) that cover 14 domains. We further present an evaluation framework that simulates real-life scenarios, to facilitate the assessment of tool invocation ability of LLMs for tool learning and task completion ability for user interation. Our extensive experiments with CToolEval evaluate 11 LLMs, revealing that while GPT-3.5-turbo excels in tool invocation, Chinese LLMs usually struggle with issues like hallucination and a lack of comprehensive tool understanding. Our findings highlight the need for further refinement in decision-making capabilities of LLMs, offering insights into bridging the gap between current functionalities and agent-level performance. To promote further research for LLMs to fully act as reliable agents in complex, real-world situations, we release our data and codes at `https://github.com/tjunlp-lab/CToolEval`.

1 Introduction

Recent years have witnessed that large language models achieve remarkable progress in planning and decision making. This promotes the emergence of numerous LLM-as-agent applications, such as ChatGPT plugins[1], AutoGPT[2], and Generative Agents (Park et al., 2023), even enabling LLM-based plugin control with natural language interface and provoking the development of LLM-based OS. In view of such rapid progresses, pressing issues are urgently emerging and soliciting attention and solutions: (i) How should we assess the capability of LLMs as agents? (ii) Which LLMs have reached agent level, and which have not? (iii)What challenges do LLMs still face when used as agents?

To address these urgent and unresolved issues, we curate a benchmark, CToolEval, for tool learning in Chinese scenarios. This benchmark is of very high quality, characterized by that (i) all APIs come from a wide and rigorous collection of open APIs from widely-used real-world Apps, making the benchmark akin to a testbed of natural language driven mobile operation system. It assesses whether an LLM-powered agent can understand and accurately respond to various functionalities within Apps, offering a high degree of real-world interactions. (ii) The generation of instructions is meticulously designed, with the use of certain tools not being completely unrelated and abrupt, but rather closely simulating all-around scenarios in our daily lives. (iii) Each instruction is annotated with a detailed reference answer and has undergone a two-stage rigorous manual review to ensure its quality.

To facilitate the agent assessment with CToolEval, we futher propose a fine-grained evaluation framework. Initially, we categorize the data from single-tool and multi-tool scenarios into four types: fixed-answer, open-ended, real-time, and operational types. In the first stage of evaluation, we assess whether an LLM-powered agent has invoked an appropriate tool, including understanding and correctly calling the API in a single-tool scenario, or whether the agent can decompose a complex task into multiple subtasks, then plan to call APIs for each subtask and understand the use of API parameters to make the correct calls in a multi-tool scenario. In the second stage of evaluation, we start from the perspective of accuracy rather than having humans (Song et al., 2023; Tang et al., 2023; Shen

[*]Equal contribution, the order of authors is determined by the first letter of their surnames.

[†]Corresponding author

[1]https://openai.com/blog/chatgpt-plugins

[2]https://github.com/Significant-Gravitas/AutoGPT

Benchmark	Statistics			Language	Tool Type		Evaluation Type	
	# Tools	# APIs	# Instances		Real-world	Multi-tool	API Call	Response
APIBench (Patil et al., 2023)	3	1,645	17,002	English	✗	✗	✓	✗
RestBench (Song et al., 2023)	2	94	157	English	✓	✓	✓	✗
API-Bank (Li et al., 2023)	53	53	274	English	✓	✗	✓	✓
ToolAlpaca (Tang et al., 2023)	400	400	3,938	English	✗	✗	✓	✓
ToolQA (Zhuang et al., 2023)	13	13	1,600	English	✓	✓	✓	✓
ToolBench1 (Qin et al., 2023a)	3,451	16,464	12,657	English	✓	✓	✓	✗
ToolBench2 (Xu et al., 2023)	8	232	2,746	English	✓	✗	✓	✗
TPTU (Ruan et al., 2023)	12	12	120	English	✓	✓	✓	✗
ToolEyes (Ye et al., 2024)	568	568	382	English	✓	✓	✓	✓
CToolEval (ours)	27	398	6,816	Chinese	✓	✓	✓	✓

Table 1: Comparison of CToolEval with existing related benchmarks.

et al., 2023) or GPT-4 (Qin et al., 2023a; Ye et al., 2024; Guo et al., 2023) judge the quality of almost all responses. Both approaches inevitably introduce bias and issues with reusability and scalability. Specifically, for real-time data, where answers are constantly changing, such as a user's query about the highest temperature tomorrow, we aim for an accurate temperature but cannot know the answer in advance. This has always been a pain point in agent evaluation because neither humans nor GPT-4 can judge whether the answers are correct. Thus, in our framework, we dynamically extract real-time answers in observations to address this problem.

With the proposed evaluation framework, we evaluate 11 LLMs with Chinese language capabilities. We find that OpenAI models generally have superior tool invocation capabilities compared to Chinese LLMs, but models with rich internal knowledge like GPT-3.5-turbo[1], when faced with conflicts between internal and external knowledge, tend to rely on internal knowledge rather than tool-invoked knowledge to answer questions. For instance, in multi-tool scenarios requiring the acquisition of a location's coordinates, it might provide coordinates it already knows internally, which are usually close but still have discrepancies, rather than those obtained through tool invocation. And we also find Chinese LLMs, like ChatGLM3-6b[2], have severe hallucination issues. They can decompose instructions into subtasks and choose the correct tool names. However, due to a lack of complete understanding of how to use the API, they cannot pass in the correct parameters to obtain the correct output from the API. Nevertheless, they still pretend to have received output from the API server and reply with fabricated answers.

Overall, our contributions are threefold:

- We propose CToolEval, a benchmark for evaluating the planning and operational capabilities of large language models in real-world application scenarios, covering 398 APIs across 14 domains in 27 Apps (please see detailed comparison of CToolEval against existing tool benchmarks.) .

- We propose a fine-grained evaluation framework, meticulously designing evaluation methods and indicators for each type of interaction from an accuracy perspective, solving the evaluation problem for real-time dynamic answers.

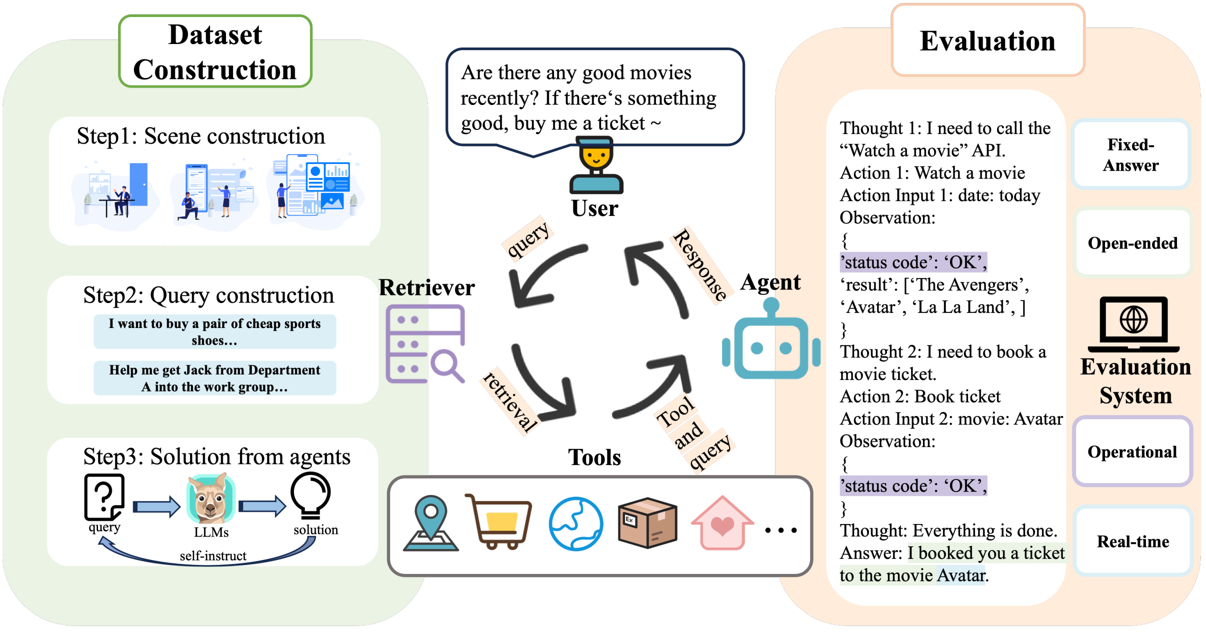
- We evaluate 11 LLMs with Chinese capabilities and conduct a deep error analysis, providing profound insights into the improvements needed for LLMs, including GPT models, to act as decision-making agents.

2 Related Work

Research on evaluating LLMs for tool manipulation and API interaction has led to benchmarks like ToolBench (Qin et al., 2023a), with its extensive collection of 16,464 APIs across 3,451 tools and 12,657 instances. The emulation of real-world scenarios is a pivotal aspect of benchmark evaluation for LLMs, as it significantly affects the relevance and applicability of evaluation findings. API-Bank (Li et al., 2023), provides an interface with 53 APIs, asserting its utility with 274 instances drawn from real-world applications. ToolQA (Zhuang et al., 2023) highlights multi-tool interactions, emphasizing their significance in LLM evaluation.

The dimension of evaluation type in benchmarking LLMs is crucial as it directly influences the interpretability and applicability of evaluation results. This dimension encompasses the accuracy (Li et al., 2023; Patil et al., 2023; Tang et al., 2023; Ruan

[1]https://platform.openai.com/docs/models/gpt-3-5-turbo
[2]https://huggingface.co/THUDM/chatglm3-6B


Dataset Construction
Step1: Scene construction
Step2: Query construction
I want to buy a pair of cheap sports shoes…
Help me get Jack from Department A into the work group…
Step3: Solution from agents
query
LLMs
solution
self-instruct
Are there any good movies recently? If there's something good, buy me a ticket ~
User
query
Response
Retriever
Agent
retrieval
Tool and query
Tools
Evaluation
Fixed-Answer
Open-ended
Evaluation System
Operational
Real-time

```
Thought 1: I need to call the "Watch a movie" API.
Action 1: Watch a movie
Action Input 1: date: today
Observation:
{
'status code': 'OK',
'result': ['The Avengers', 'Avatar', 'La La Land', ]
}
Thought 2: I need to book a movie ticket.
Action 2: Book ticket
Action Input 2: movie: Avatar
Observation:
{
'status code': 'OK',
}
Thought: Everything is done.
Answer: I booked you a ticket to the movie Avatar.
```

Figure 1: The framework of CToolEval covers interactions in real-world scenarios across 27 Apps, each containing a series of relevant APIs. The evaluation of LLMs' tool invocation capability and task completion capability is conducted from the perspective of four types of data.

et al., 2023) and execution success rate of API calls (Song et al., 2023; Xu et al., 2023; Zhuang et al., 2023; Qin et al., 2023a), the quality and relevance of the responses (Li et al., 2023; Tang et al., 2023; Zhuang et al., 2023; Huang and Xiong, 2023), and the model's ability to avoid hallucination — generating responses based on false or nonexistent information (Patil et al., 2023). Specifically, the metrics used in RestBench (Song et al., 2023) and ToolAlpaca (Tang et al., 2023) require manual evaluation, while the win rate indicator in ToolBench1 (Qin et al., 2023a) relies on scoring by GPT-4, all of which inevitably introduce elements of subjective judgment and assessment unfairness caused by the biases of human and LLMs.

A concurent work has a motivation very similar to ours. ToolEyes (Ye et al., 2024) creates a fine-grained evaluation of LLM's tool learning capabilities. Particularly in the data dimension, both of our works include real-time and operational data types. However, ToolEyes (Ye et al., 2024) still relies mostly on scoring by advanced models like GPT-4 to deliver evaluation results, inevitably encountering issues such as model bias. Our contribution, CToolEval, seeks to address these shortcomings by prioritizing accuracy, solving for the dynamic evaluation of answers that may change over time, aiming to minimize reliance on model scoring for better reuse and scalability.

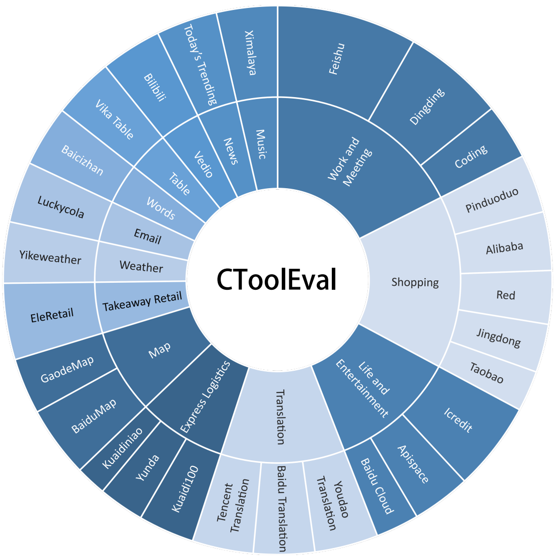


Figure 2: Different App/tool categories covered in our benchmark.

3 Benchmark Curation

Our data creation process, as illustrated in Figure 1, involves sourcing open APIs from real-world applications.

3.1 APP Pool

Against the backdrop of real interactions between users and Apps in Chinese society, we integrate APIs available from popular Apps across 14 domains. The selected Apps for integration satisify the following conditions: (i) **Open for research** - These Apps are available for research purposes; (ii) **Providing Detailed Parameter Descriptions and Usage instructions** - Each App comes with comprehensive documentation. After selection, we finally choose 27 Apps from a wide range of domains, which are essential and commonly used in daily life, including but not limited to: travel apps such as Gaode Maps and Baidu Maps; work-related apps like Dingding and Feishu; shopping apps including Taobao and Jingdong; music apps like Ximalaya; logistics apps such as Yunda and Express Bird and so on. We further sift through these app platforms to find APIs that meet our criteria. The division of all dimensions and apps can be seen in Figure 2. The shopping domain has the highest number of apps, including Taobao, Pinduoduo, Alibaba, Jingdong, and Red, with the Work and Meeting domains having the largest number of APIs, where the Feishu app alone includes 76 APIs.

3.2 API & LLM Integration

Utilizing the Langchain[1] framework, our system successfully integrated a total of 398 APIs with LLMs. This integration involves meticulously designed incorporation of each API into a comprehensive toolset pool. The LLM, initially unaware of the available APIs within the pool, analyzes user queries to identify and choose the correct tool from the pool using vector storage for tool descriptions and query embeddings, facilitating a similarity search. It then translates queries into structured data matching the selected tool's interface and executes API calls. If responses are inadequate, it iterates until achieving a satisfactory answer. This method enables easy API pool expansion. Specifically, for APIs related to database operations, we also establish essential databases, initialized with initial entries. This step is a pivotal element in constructing and utilizing the framework.

3.3 Instruction Generation

we designed queries reflecting real-life applications solvable through the detailed functionalities of these APIs. In this setup, GPT-4 is employed as an agent to orchestrate and execute API calls. Queries used for debugging the integrated code, are termed as our seed instructions. We then guide GPT-4 with meticulously crafted instructions for expansion. Further Recognizing complex user needs, we differentiated our evaluation into single-tool and multi-tool assessments, addressing the varying degrees of challenge.

[1]https://www.langchain.com

3.3.1 Single-tool Instructions

For single-tool queries, we crafted diverse inquiries per API by: (i) changing query subjects like destinations in map and delivery scenarios; (ii) varying sentence structures; (iii) creating unique contexts such as weather suitability for barbecues or hikes. This generates 3,439 single-tool instructions.

3.3.2 Multi-tool Instructions

In the generation of multi-tool instructions, instead of forcibly combining unrelated tools, our focus is on the intrinsic connections between different tools, resulting in the creation of 3,370 multi-tool instructions. The generation encompasses both multiple APIs within the same App and across different Apps.

Multiple APIs within the same App. During API integration, we observe that many APIs are interdependent. For instance, to obtain a route plan from Gaode Maps, geographical encoding of the start and end points, obtained through Gaode Maps' geocoding tool, is necessary. Typically, the number of APIs involved in these interactions is basically between 2 and 10.

Multiple APIs across different Apps. Beyond the same App, multi-API queries across different Apps also have numerous real-life applications. For instance, in the shopping category, comparing prices of the same item on Taobao and JD.com is a common practice. Based on this, we create similar cross-App multi-API queries to address more complex requirements.

3.4 Reference Answer Annotation

Upon the completion of designing each query, we deploy GPT-4 as an agent to plan and execute API calls, ensuring the feasibility of each query. The responses are then annotated based on the output of this execution. Employing the ReAct principle for the agent's implementation, the final answers encompass the decision process, actions, observations, and final answers of the agent. The sequence of API calls and responses obtained from the final API execution are meticulously annotated in our responses. However, the design of our queries poses significant challenges to GPT-4. Therefore, a rigorous manual review is conducted, ocusing on output

Statistics	Single-tool	Multi-tool	Total
# of tools/Apps	27	24	27
# of domains	14	13	14
# of APIs	185	231	398
# of API calls	3,439	10,8648	14,303
# Max APIs/inst.	1	9	9
# Max calls/inst.	1	13	13
# of Fixed-Answer	2,295	706	3,001
# of Open-Ended	189	182	371
# of Real-Time	540	481	1,021
# of Operational	415	2,008	2,423
Total	3,439	3,377	6,816

Table 2: Statistics of the CToolEval benchmark.

format and logical consistency, these reviewed answers are annotated in our benchmark. Ultimately, we retained 6,816 queries in our benchmark.

3.5 Statistics

The data statistics of our dataset can be seen in Table 2. In the end, we construct a comprehensive benchmark that covers 27 tools, featuring 398 distinct APIs, comprising a total of 6,816 queries, 14,303 API calls. Within this benchmark, the distribution of queries is as follows: 3,439 queries utilize a single tool, while 3,377 queries employ multiple tools, showing a balanced distribution between single-tool and multi-tool instances. Moreover, single-tool instances involve 27 tools and 185 APIs, while multi-tool instances cover 24 tools and 231 APIs, indicating that our application scenarios are very rich. From the perspective of evaluation method classification, we can further categorize these instances as follows: 2,423 queries in the operational category, 371 queries in the open-ended category, 3,001 queries in the fixed-answer category, and 1,021 queries in the real-time response category. For more details about the division and instances of these four types of evaluation categories, please refer to Table 2.

3.6 Quality

To ensure the quality of the data in our benchmark, we also conduct a detailed review of each data instance. Every instance in the evaluation set has been reviewed by annotators, with the review process divided into two phases. During the data generation phase, we annotate the answers with great detail and strictness. We employ GPT-4 as the agent to execute all instances. However, given that our benchmark operates in real-world Chinese-language scenarios, it also poses significant challenges to GPT-4. Throughout multiple debugging sessions, GPT-4's responses are not always satisfactory, sometimes presenting issues like incorrect tool planning or fabricated API results. Therefore, after obtaining GPT-4's output, a rigorous manual review is conducted, including scrutiny of interface compatibility, the rationality of API planning, and the congruence of the final answer with the API's actual return. We run multiple iterations until the correct answers are obtained, including manual adjustments for some responses. For instance, if after several attempts it still could not produce output in Chinese, we perform manual translations into English, and for responses containing "I am a text generation model, unable to invoke tools", we manually remove them. In the evaluation phase, to enhance the robustness of our benchmark, we first test all the data using GPT-3.5. After this stage, we eliminate 2% of the data, including instances where the API itself has issues and could not respond correctly, as well as operational instances that could not meet our repeated evaluation requirements. These stringent quality controls further improve the quality of our benchmark, allowing us to primarily assess the quality of LLMs as proxy tools in a standard and fair setting.

4 Evaluation Framework

In a comprehensive process where an LLM acts as an agent to solve problems using tools, the model initially extracts the query and the list of available tools along with their descriptions from the prompts. It then decomposes the query into smaller sub-tasks and sequentially calls the corresponding tools, using the knowledge returned by each tool to progressively deduce the correct answers.

However, prior research (Ruan et al., 2023; Kong et al., 2023; Qin et al., 2023b) reveals the inadequacies of most LLMs in tool utilization, highlighting their failure in task decomposition, tool invocation, and knowledge application, with errors frequently emerging at various stages.

Thus, to thoroughly assess LLMs' tool utilization proficiency, we scrutinize both tool invocation and task completion capabilities. Through an in-depth analysis shown in Appendix A, we evaluate LLMs' ability to apply tool-generated knowledge. These competencies are interrelated, marking ascending levels of tool usage mastery. Additionally, leveraging actual user inputs, CToolEval's instructions are classified into four categories: fixed-answer, open-ended, operational, and real-time, enabling a nuanced, multi-faceted evaluation of accuracy in these dimensions.

4.1 Data Type

Existing benchmarks predominantly feature open and fixed-answer queries, with their evaluation relying on advanced LLM-based scoring and accuracy metrics. Despite the sophisticated capabilities of such models, their assessments of answer accuracy can be unreliable, particularly for queries necessitating tool-derived knowledge.

To enhance the evaluation of large language models' tool usage proficiency, we collect real user queries that demand tool invocation, and categorize them into four distinct types based on evaluation methods: fixed-answer, open-ended, operational, and real-time, as detailed in Table 7.

(i) Fixed-answer queries entail questions with a static, fill-in-the-blank response.

(ii) Open-ended questions require detailed responses without a predetermined format.

(iii) Operational questions necessitate the execution of tool-based APIs for specific tasks.

(iv) Real-time questions involve answers that are dynamic over time.

We apply different evaluation methodologies for these question types to accurately measure task completion efficacy.

4.2 Retriever

LLMs usually have limited context capacity. Overloading LLMs with excessive irrelevant tool information can hinder their tool invocation ability. Hence, employing a retriever to assist LLMs in selecting suitable tools is essential (Qin et al., 2023b). We use Sentence-BERT (Reimers and Gurevych, 2019), based on Chinese-bert-base[1], to train a retriever that calculates the semantic similarity between the embeddings of CToolEval instructions and tool descriptions to identify the most relevant tool. The effectiveness of the retriever is measured using the NDCG metric (Järvelin and Kekäläinen, 2002). More details about the retriever are given in Appendix C.

4.3 Tool Invocation Capability

Tool invocation capability is the fundamental ability that determines whether an LLM can effectively complete tasks that involve the use of tools, which requires LLMs to decompose tasks into sub-tasks and appropriately match tools to each sub-task. The evaluation metric we employ is the "Tool Matching Rate“, which is calculated as the number of tools correctly used divided by the total number of tools that should be used. Please note that, at this stage, we exclusively examine the correctness of tool usage.

4.4 Task Completion Capability

Task completion capability is a crucial competency that assesses LLMs' ability to utilize knowledge obtained from tools to engage in reasoning and solve problems, which entails LLMs' step-by-step integration of sub-task results to deduce the final answers.

For fixed-answer queries, we employ "accuracy" as the evaluation metric. For open-ended questions, we utilize advanced LLMs to score based on the reference answers, focusing on rationality, completeness, and richness of information. Regarding operational queries, where LLMs may potentially provide deceptive responses by falsely asserting task completion without performing the necessary operations, we thoroughly check the status codes of the responses from the tools invoked for each subtask. Typically, network responses returned by different APIs contain status codes that represent the result of the request. If the operation is successful, the status code is a specific value. The evaluation metric is the number of tools returning the correct response divided by the number of tools needed as per the reference. Furthermore, as CToolEval serves as an open-source evaluation dataset, it should ensure the consistency of the backend database objects across multiple evaluations. Therefore, we design a rollback function. Specifically, after each successful invocation of a tool, the system will automatically perform the "reverse operation" of this invocation. For example, after adding a friend using an instant messaging tool, the system will delete this friend. In the case of real-time queries, we treat them as real-time fill-in-the-blank queries. When establishing reference answers for such queries, we store the API links and parameters to be called, ensuring that reference answers can be easily extracted or assembled from the returned information. This approach enables us to conduct "dynamic assessments" of real-time queries based on accuracy.

[1]https://huggingface.co/google-bert/bert-base-chinese

Model	Single-tool				Multi-tool				All
	FA	OE	OP	RT	FA	OE	OP	RT	
GPT-3.5-turbo	77.11	49.04	90.36	76.68	21.43	98.17	31.68	51.08	61.94
Qwen-7B-Chat	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Qwen-14B-Chat	1.53	3.69	3.21	2.78	0.00	0.31	0.01	0.06	1.45
Qwen-72B-Chat	25.93	21.15	32.07	11.56	19.87	7.71	4.70	3.67	15.82
InternLM-chat-7B	22.82	25.69	41.52	13.76	15.73	6.09	8.71	6.26	17.81
InternLM-chat-20B	53.47	23.67	40.74	12.63	14.25	6.50	9.17	7.73	21.02
Baichuan2-7B-Chat	4.91	8.26	11.81	5.65	4.20	5.95	2.94	2.01	5.72
Baichuan2-13B-Chat	2.40	10.09	15.88	2.42	4.72	1.56	0.00	2.37	4.93
Chinese-Alpaca-2-7B-16k	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Chinese-Alpaca-2-7B-64k	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Chinese-Alpaca-2-13B-16k	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 3: Evaluation results of the tool invocation capability of different models in each category. The metric is "Tool Matching Rate"(%), with "All" representing score over all senerios.

Model	Single-tool				Multi-tool				All
	FA	OE	OP	RT	FA	OE	OP	RT	
GPT-3.5-turbo	57.40	39.45	72.53	56.49	43.35	44.04	19.49	30.88	45.45
Qwen-7B-Chat	8.26	14.86	0.00	0.00	3.73	19.89	0.00	0.21	5.87
Qwen-14B-Chat	10.36	22.39	5.66	1.59	3.92	23.74	0.05	0.85	8.57
Qwen-72B-Chat	28.39	35.62	9.91	4.28	19.69	7.28	1.32	3.11	13.7
InternLM-chat-7B	10.91	24.40	32.16	0.00	4.60	20.88	2.49	0.00	11.93
InternLM-chat-20B	28.02	23.76	33.28	2.16	5.55	22.64	4.42	1.73	15.20
Baichuan2-7B-Chat	6.51	20.92	7.47	0.87	1.47	18.90	0.69	0.21	7.13
Baichuan2-13B-Chat	2.75	19.63	8.87	0.58	0.61	14.62	0.00	0.00	5.88
Chinese-Alpaca-2-7B-16k	23.66	20.55	0.00	0.00	8.57	20.00	0.00	0.00	9.10
Chinese-Alpaca-2-7B-64k	17.48	21.83	0.00	0.00	12.27	20.22	0.00	0.00	8.98
Chinese-Alpaca-2-13B-16k	16.34	18.53	0.00	0.00	9.89	19.89	0.00	0.00	8.08

Table 4: Evaluation results of the task completion capability of different models in each category. The metric for Fixed-Answer (FA) and Real-Time (RT) is accuracy (%), for Open-Ended (OE) is GPT-4 score (%) and for Operational is execution success rate (%), with "All" representing score over all senerios.

5 Experiments

We evaluated a variety of LLMs, including 1 closed-source LLMs and 10 open-source Chinese LLMs. As our evaluation requires LLMs to have a certain level of compliance with instructions, we selected those that have been fine-tuned with SFT and possess chat capabilities. Additionally, due to the large number of tools, and some multi-tool queries even containing more than 10 tools, the prompts fed into the LLMs can be very long, necessitating LLMs with a relatively large context window, at least above 8K. The evaluated closed-source LLMs is GPT-3.5-turbo.[1] The open-source LLMs include Qwen-7B-chat,[2] Qwen-14B-chat,[3] Qwen-72B-chat,[4] InternLM-chat-7b,[5] InternLM-chat-20b,[6] Baichuan2-13B-chat,[7] Baichuan2-13B-chat,[8] Chinese-alpaca-2-7b-16k,[9] Chinese-alpaca-2-7b-64k[10] and Chinese-alpaca-2-13b-16k.[11]

5.1 Main Results

Table 3 presents the evaluation results of the tool invocation capability of different LLMs, while Table 4 shows the task completion capabilities of different LLMs. From these results, we can observe that: (i) **The gap between open-source Chinese LLMs and the proprietary GPT-3.5-turbo remains significant.** In terms of the comparison of different model capabilities, GPT-3.5-turbo far surpasses Chinese LLMs in both tool invocation and task completion capabilities, with InternLM-Chat-20B scoring the highest among Chinese LLMs, reaching up to 53.47 points in tool invocation capability for single-tool fixed-answer data types; (ii) **Task completion scores of LLMs are higher than tool invocation scores**, especially in fixed-answer and open-ended types, indicating that many LLMs

[1]https://openai.com/docs/models/gpt-3-5
[2]https://huggingface.co/Qwen/Qwen-7B-Chat
[3]https://huggingface.co/Qwen/Qwen-14B-Chat
[4]https://huggingface.co/Qwen/Qwen-72B-Chat
[5]https://huggingface.co/internlm/internlm-chat-7b
[6]https://huggingface.co/internlm/internlm-chat-20b
[7]https://huggingface.co/Baichuan2-7B-Chat
[8]https://huggingface.co/Baichuan2-13B-Chat
[9]https://huggingface.co/chinese-alpaca-2-7b-16k
[10]https://huggingface.co/chinese-alpaca-2-7b-64k
[11]https://huggingface.co/chinese-alpaca-2-13b-16k

Scenarios	Error Type	Percentage
Single-tool	False Input Parameters Format	10.35%
	Miss Input Parameters	6.64%
	False Input Parameters	34.89%
	Parsing Error	6.38%
	Failed API Retrieval	6.07%
	Invalid API	5.34%
	False API Selection	5.22%
	Duplicate Response	13.51%
	Redundant response	11.60%
Multi-tool	Unfinished Operation	37.93%
	Skipping Previous Operations	18.82%
	API Call Logic Error	2.65%
	No API Invocation	0.12%
	Parsing Error	6.29%
	Failed API Retrieval	7.14%
	Invalid API	11.19%
	False Input Parameter Format	2.19%
	Miss Input Parameters	3.61%
	False Input Parameters	9.74%

Table 5: Distribution of errors made by GPT-3.5-turbo.

still lack the capability to invoke tools, yet they still use their internal knowledge to answer user questions. Despite demonstrating good interactivity, LLMs may deceitfully pretend to have obtained answers through tools, leading to significant issues with hallucination; (iii) **Almost all LLMs score higher in single-tool scenarios than in multi-tool scenarios**, particularly, for LLMs capable of tool invocation, the scores for operation types in single-tool scenarios often exceed those for fixed-answers and open-ended responses. This may be due to the tool descriptions and parameters being relatively brief in single-tool operations, allowing LLMs to better understand and follow the context in shorter prompts, but multi-tool scenarios still pose a greater challenge, and real-time queries are difficult for LLMs in both single- and multi-tool contexts; (iv) **Increasing model size can enhance the tool learning capability of LLMs to a certain extent, but not always.** For InternLM and Qwen, increasing the model size has improved the capability in both tool invocation and task answering, especially for Qwen-72B-Chat, which has seen a qualitative improvement compared to the other two sizes. However, for Chinese-Alpaca-2, increasing the model size can actually impair model performance. Thus, to enhance the model's tool learning capability, besides ensuring a sufficiently long context window, it is also necessary to further balance the model size and fine-tuning data.

5.2 Error Analysis

We conducted a detailed analysis on errors made by GPT-3.5-turbo in single-tool and multi-tool scenarios, and have categorized the errors in single-tool scenarios into 9 types and those in multi-tool scenarios into 10 types. The percentages of error types are presented in Table 5. In the single-tool scenario, the failure to correctly invoke an API can occur for various reasons, such as incorrect API names, incorrect or erroneous format of input parameters, the retriever failing to provide the required API or providing the needed API but making the wrong choice, and fabrications. The most common errors include false input parameters, Duplicate Response, and Redundant Response. In the multi-tool context, the failure to execute multiple tools is mainly due to the model not completing operations, only performing halfway and then outputting answers, as well as skipping the necessary preceding operations and directly relying on its internal knowledge or fabricating the required parameters' inputs. In addition, it is common in multi-tool scenarios to insert multiple required tools into a single action, leading to persistent failure. These errors suggest that GPT-3.5-turbo's learning and understanding of how to use tools, especially the ability to associate between tools, still needs improvement, and duplicate response failure indicates a deficiency in the ability to follow instructions. Moreover, we have also discovered that GPT-3.5-turbo has significant issues with temporal reasoning. Whenever our queries involve "tomorrow", it often reasons incorrectly about the input time parameters for APIs. This could be an area that GPT-3.5 needs to address in order to become a truly grounded agent.

6 Discussion

6.1 Impact of Retriever on Baseline Model Performance

To study how retrieval affects the tool invocation capabilities in LLMs, we provided gold APIs to InternLM-7B-chat, InternLM-20B-chat, and GPT-3.5-turbo and assessed their performance. Results in Table 8 and 9 in Appendix D show that gold APIs significantly enhance invocation capabilities, particularly because the retriever presents ten tool options, making prompts longer and harder for models to process useful data. Although the retriever exhibits a high recall rate, we observe that in multi-tool tasks, the tools returned by the retriever sometimes do not include those that are more critical in the earlier stages of the solution path. This absence of essential tools early in the tool invocation process imposes limitations on the LLMs.

Morever, InternLM-7B-chat and InternLM-20B-chat improve more than GPT-3.5-turbo. We speculate that providing gold APIs benefits LLMs with weaker reasoning ability in task planning, while stronger LLMs can effectively select APIs even without gold APIs.

6.2 Performance of LLMs Being Fine-tuned on CToolEval

We utilized the training set of CToolEval to fine-tune Baichuan2-7b-chat and Qwen-7b-chat models using two parameter-efficient methods, QLoRA and LoRA. Additionally, we benchmarked these models against GPT-4 in a zero-shot setting on the test set. Results, detailed in Table 10 and 11 in Appendix E, indicate significant improvements in both tool matching rate and task completion metrics for the fine-tuned Baichuan2-7B-chat and Qwen-7B-chat models. Furthermore, the Qwen-7B-chat models, fine-tuned using both QLoRA and LoRA methods, exhibit comparable abilities.

However, fine-tuning also intensifies the tendency for tool invocation hallucination. This means that after fine-tuning, the models are more likely to simulate API responses rather than performing actual tool invocations, leading to higher tool matching scores but not proportionately higher task completion scores. This is because while selecting the correct tool contributes to the tool matching rate, effective task completion depends on the LLM's ability to reason and analyze actual API responses. Since the nature of answers to real-time and open-ended queries fluctuates, without real tool invocations, task completion cannot improve. Therefore, for models to excel in both metrics, they must genuinely invoke tools similar to GPT-4.

6.3 Knowledge Utilization Capability of GPT-3.5

Knowledge utilization capability is a crucial ability that determines whether LLMs can effectively harness the knowledge provided by tools, which requires LLMs to accurately utilize the knowledge obtained from each sub-task without hallucination. However, manually annotating correct results for each sub-task can be prohibitively expensive. Therefore, we introduce an innovative control group evaluation method.

For all the tools successfully invoked in a single command execution, we will set up corresponding control groups according to the order of tool invocation, where each tool's control group will represent a command execution without using that particular tool. Specifically, if n tools are used, there will be $n-1$ control groups. By calculating the similarity between the control group's answers and the answers to be evaluated with the reference answer, we can determine whether the execution effectively utilizes the knowledge returned by each tool step. See Appendix A for results and analyses.

7 Conclusion

In this paper, we have presented CToolEval, a benchmark for assessing LLMs as agents to interact with Chinese Apps, which covers 398 real-world APIs. Through our fine-grained evaluation framework, we identify critical insights into the capabilities of LLMs, specifically in the domains of tool invocation and task completion. Our analysis indicates that while GPT-3.5-turbo demonstrates a higher proficiency in tool utilization, there remains a notable gap in the ability of Chinese LLMs to apply tools effectively. This gap is often marked by a reliance on internal knowledge and the occurrence of hallucinations, which leads to a discrepancy between the output provided and the information available through tool invocation. By dissecting the errors and pinpointing the nuances in the operational use of APIs, our work contributes to a more nuanced understanding and evaluation of the agent-level capabilities of LLMs and opens up avenues for future research and development of LLM-powered agents in real-world scenarios.

Limitations

Although we have established a Chinese benchmark that simulates real-world App interactions to conduct a comprehensive analysis of existing LLMs and outlined various existing issues, our work has two notable limitations. First, since the APIs we use are publicly accessible, there might be instances where an API interface becomes faulty over time, affecting our evaluation results, but this is an unavoidable aspect of simulating real-world tool interactions. Second, due to the token usage limitations of GPT-4, our ability to evaluate all existing LLMs is restricted. It is important to emphasize that we meticulously selected the most representative Chinese LLMs for evaluation and analysis, aiming to identify key problems with Chinese LLMs in agentization.

Ethics Statement

This paper establishes a new benchmark for evaluating the tool learning capabilities of Chinese LLMs, and herein we discuss some related ethical considerations. All App interfaces in our dataset come from open platforms accessible to researchers, without infringing upon any existing commercial software. Furthermore, to annotate data within the benchmark, annotators involved in the annotation were fully compensated for their efforts.

Acknowledgements

The present research was supported by the National Key Research and Development Program of China (Grant No. 2023YFE0116400). We would like to thank the anonymous reviewers for their insightful comments. Moreover, we would also like to thank Yue Chen, Zhuowen Han, Hangyi Jia, Jianxiang Peng, Ling Shi, Juesi Xiao, Lei Yang, and Jiang Zhou for their invaluable assistance in data collection, processing, and annotation. Their efforts have greatly contributed to the success of this work.

References

Zishan Guo, Renren Jin, Chuang Liu, Yufei Huang, Dan Shi, Linhao Yu, Yan Liu, Jiaxuan Li, Bojian Xiong, Deyi Xiong, et al. 2023. Evaluating large language models: A comprehensive survey. *arXiv preprint arXiv:2310.19736*.

Yufei Huang and Deyi Xiong. 2023. Cbbq: A chinese bias benchmark dataset curated with human-ai collaboration for large language models. *arXiv preprint arXiv:2306.16244*.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446.

Yilun Kong, Jingqing Ruan, Yihong Chen, Bin Zhang, Tianpeng Bao, Shiwei Shi, Guoqing Du, Xiaoru Hu, Hangyu Mao, Ziyue Li, et al. 2023. Tptu-v2: Boosting task planning and tool usage of large language model-based agents in real-world systems. *arXiv preprint arXiv:2311.11315*.

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, EMNLP 2023, Singapore, December 6-10, 2023*, pages 3102–3116. Association for Computational Linguistics.

Joon Sung Park, Joseph C. O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, UIST 2023, San Francisco, CA, USA, 29 October 2023- 1 November 2023*, pages 2:1–2:22. ACM.

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

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

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*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Jingqing Ruan, Yihong Chen, Bin Zhang, Zhiwei Xu, Tianpeng Bao, Guoqing Du, Shiwei Shi, Hangyu Mao, Xingyu Zeng, and Rui Zhao. 2023. Tptu: Task planning and tool usage of large language model-based ai agents. *arXiv preprint arXiv:2308.03427*.

Tianhao Shen, Renren Jin, Yufei Huang, Chuang Liu, Weilong Dong, Zishan Guo, Xinwei Wu, Yan Liu, and Deyi Xiong. 2023. Large language model alignment: A survey. *arXiv preprint arXiv:2309.15025*.

Yifan Song, Weimin Xiong, Dawei Zhu, Cheng Li, Ke Wang, Ye Tian, and Sujian Li. 2023. Restgpt: Connecting large language models with real-world applications via restful apis. *CoRR*, abs/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. *CoRR*, abs/2306.05301.

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. *CoRR*, abs/2305.16504.

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

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

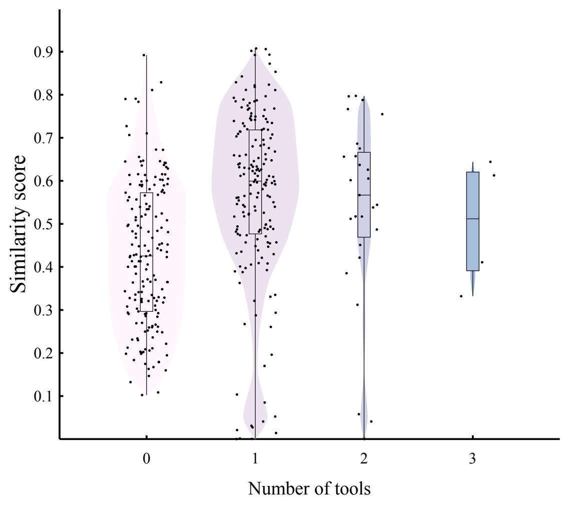


Figure 3: Violin diagram of single open-ended and multiple open-ended type instructions.

A Knowledge Utilization Capability of GPT-3.5-turbo

Figure 3 is a violin plot for GPT-3.5-turbo-1106 on multi-tool open-ended and single-tool open-ended queries, where each point represents a query execution, the horizontal axis represents the number of tools invoked, and the vertical axis represents the similarity between the answers and the reference answers. Surprisingly, the quality of LLM-generated answers does not increase with the number of tools invoked. Most queries show improvement when the first tool is invoked compared to no tool invocation; however, after invoking the second tool, the quality of answers slightly decreases for some instructions, and after invoking three tools, the quality of LLM-generated answers decreases for most instructions. We speculate that the single-open and multi-open commands in CToolEval require a large number of tokens from the returned information of the tools. After invoking a small number of tools, the LLM can obtain information to answer the question; when the number of invoked tools increases, the rapid increase in the number of tokens makes it difficult for the LLM to capture useful information, leading to a decrease in answer quality.

B Four Examples of Question Types

CToolEval classifies queries into fixed-answer, open-ended, operational, and real-time types depending on how they are evaluated. Question examples and reference answers are shown in Table 7.

Average NDCG@1	Average NDCG@3	Average NDCG@5
0.9688	0.9797	0.9831

Table 6: The average NDCG@K scores of our API retriever used in main experiments.

C Retriever

We opt to fine-tune the Chinese-bert-base[1] model based on Sentence-BERT (Reimers and Gurevych, 2019) using pairs of query-tool information data. Following ToolLLM (Qin et al., 2023b), we assess retrieval performance using NDCG (Normalized Discounted Cumulative Gain) (Järvelin and Kekäläinen, 2002), a widely used metric in information retrieval and recommendation systems to measure the quality of ranking results. NDCG accounts for the relevance of the results and assigns higher weights to more relevant results that appear earlier in the ranking, thus achieving better retrieval outcomes. The NDCG scores of the retriever used in CToolEval are as follows.

NDCG@K refers to the NDCG at a given rank position k. Average NDCG@1 is the average NDCG value for results ranked first in each query, while Average NDCG@3 and Average NDCG@5 are the average NDCG values for results ranked third and fifth, respectively. These values help evaluate the quality of a retrieval across different lengths of search. The retrieval scores indicate that the retriever used in CToolEval effectively returns the most matching tools.

D Evaluation Results of Baseline Model Provided With Gold APIs

To explore the effect of retriever on tool invocation capabilities, we provided only the gold APIs to InternLM-7B, InternLM-20B, and GPT-3.5-turbo, and documented their performance on the test set. The experimental results are shown in Table 8 and 9.

E Evaluation Results of LLMs Being Fine-tuned on CToolEval

We employed the training set of CToolEval for fine-tuning using two parameter-efficient fine-tuning methods, QLoRA and LoRA, on Baichuan2-7b-chat and Qwen-7b-chat. The experimental results on the test set are presented in Table 10 and 11.

[1]https://huggingface.co/google-bert/bert-base-chinese

Query Type	Query Example	Reference Answer
fixed-answer	请为我提供2022年10月13日印度尼西亚雅加达证券交易所的SMGR.JK（Semen Indonesia）的收盘价和交易量。 Please provide me with the closing price and trading volume of SMGR.JK on the Indonesia Stock Exchange on October 13, 2022.	7041.94091796875, 4795700.0
open-ended	在淘宝上查找流行的电子书阅读器，推荐一款，并提供其屏幕尺寸和存储容量信息。 Search for popular e-book readers on Taobao, recommend one, and provide information on its screen size and storage capacity.	推荐的电子书阅读器是“掌阅iReader Light3 电子书阅读器”。其屏幕尺寸为6英寸，内存容量为32GB。 The recommended e-book reader is the ’iReader Light3 E-book Reader’ by Zhangyue. It has a 6-inch screen size and a memory capacity of 32GB.
operational	麻烦创建一个“学生成绩”表，其中应该有“学生编号”、“姓名”、“课程”和“分数”。 Create a ’Student Grades’ table, which should include ’Student ID,’ ’Name,’ ’Course,’ and ’Score.’	已成功创建名为“学生成绩”的表格，其中包含“学生编号”、“姓名”、“课程”和“分数”四列。 The table named ’Student Grades’ has been successfully created, consisting of four columns: ’Student ID,’ ’Name,’ ’Course,’ and ’Score’.
real-time	我计划一会要去西安市雁塔区大雁塔789号，这里附近600米范围内的各拥堵路段信息怎么样? I plan to go to 789 Dayanta in Yanta District, Xi’an shortly. How is the traffic congestion information for the roads within a 600-meter radius of this location?	整体畅通，没有报告特别拥堵的路段。 The overall traffic is smooth, with no particularly congested sections.

Table 7: Four different query types of question types and their reference answers.

Model	Single-tool				Multi-tool				All
	FA	OE	OP	RT	FA	OE	OP	RT	
GPT-3.5-turbo	82.05↑	63.64↑	92.77↑	51.61↓	53.55↑	57.84↓	60.78↑	57.39↑	64.95↑
InternLM-7B-chat	11.63↑	26.36↑	32.05↓	0.00↓	5.56↑	22.94↑	2.93↑	0.00↓	12.68↑
InternLM-20B-chat	33.18↑	22.73↓	34.82↑	2.29↑	5.56↑	21.18↓	3.58↓	5.15↑	16.06↑

Table 8: The tool matching rate of diffrent models provided with gold APIs in each category. The upward arrow and the downward arrow represent the trend of changes in the tool matching rate when providing the gold API compared to providing the top 10 relevant APIs by the retriever.

Model	Single-tool				Multi-tool				All
	FA	OE	OP	RT	FA	OE	OP	RT	
GPT-3.5-turbo	61.25↑	43.64↑	80.72↑	44.08↓	30.56↑	30.00↓	40.79↑	18.56↓	43.70↓
InternLM-7B-chat	26.70↑	23.64↓	47.71↑	15.68↑	17.72↑	6.98↑	8.86↑	5.68↓	19.12↑
InternLM-20B-chat	59.27↑	24.18↑	45.05↑	14.75↑	13.85↑	10.78↑	9.84↑	6.58↓	23.04↑

Table 9: Evaluation results of the task completion capability of diffrent models provided with gold APIs in each category. The upward arrow and the downward arrow represent the trend of changes in the task completion metrics when providing the golden API compared to providing the top 10 relevant APIs by the retriever.

Model	Single-tool				Multi-tool				All
	FA	OE	OP	RT	FA	OE	OP	RT	
GPT-4	74.93	51.76	85.54	93.55	81.48	95.98	81.90	72.34	79.69
GPT-3.5-turbo	76.35	48.25	91.57	74.19	44.52	96.13	48.74	43.99	65.47
Baichuan2-7B-chat(QLoRA)	74.36↑	54.55↑	67.47↑	49.46↑	24.23↑	10.78↑	25.92↑	14.60↑	40.17↑
Qwen-7B-chat(QLoRA)	82.91↑	90.91↑	92.77↑	77.42↑	43.52↑	31.86↑	58.86↑	30.58↑	63.60↑
Qwen-7B-chat(LoRA)	83.48↑	90.91↑	87.95↑	76.34↑	45.29↑	32.84↑	57.12↑	33.51↑	63.43↑

Table 10: The tool matching rate of LLMs fine-tuned on CToolEval in each category. The upward arrow and the downward arrow represent the trend of changes in the tool matching rate after fine-tuning compared to zero-shot.

Model	Single-tool				Multi-tool				All
	FA	OE	OP	RT	FA	OE	OP	RT	
GPT-4	60.97	46.18	72.29	70.97	55.56	42.96	48.48	68.04	58.18
GPT-3.5-turbo	58.40	40.72	71.08	49.46	20.37	36.07	35.75	25.77	42.20
Baichuan2-7B-chat(QLoRA)	57.26↑	24.55↓	56.63↑	36.56↑	10.19↑	8.235↑	14.28↑	2.060↑	26.22↑
Qwen-7B-chat(QLoRA)	59.26↑	36.37↑	63.86↑	39.78↑	20.37↑	15.29↑	29.86↑	4.120↑	33.61↑
Qwen-7B-chat(LoRA)	61.25↑	40.00↑	60.24↑	43.01↑	17.59↑	36.07↑	24.72↑	4.120↑	33.50↑

Table 11: Evaluation results of the task completion capability of LLMs fine-tuned on CToolEval in each category. The upward arrow and the downward arrow represent the trend of changes in the tool matching rate after fine-tuning compared to zero-shot.