AtomTool: Empowering Large Language Models with Tool Utilization Skills *

Yongle Li[1], Zheng Zhang[2], Junqi Zhang[2], Wenbo Hu[1], Yongyu Wu[2], and Richang Hong[1]

[1] Hefei University of Technology, Hefei, Anhui Province, China
[2] AtomEcho, Beijing, China
https://atomecho.cn
yongleli@mail.hfut.edu.cn, zhengzhang13@icloud.com, zhangjunqi@atomecho.xyz, wenbohu@hfut.edu.cn, wuyongyu@atomecho.xyz, hongrc.hfut@gmail.com

Abstract. In recent years, significant strides have been made in harnessing large language models (LLMs) to leverage various tools across different fields, which largely expands the application scope of LLMs. However, current research predominantly focuses on LLMs' inherent tool exploitation skills from their training data, leading to higher costs when integrating new tools. Additionally, most studies concentrate on English models, leaving a scarcity of open-source resources for other languages. This study investigates the zero-shot generalization of LLMs in tool usage, with a focus on Chinese models. We introduce AtomTool, an open-source framework for tool acquisition in LLMs, along with a dataset of 16,000 Chinese entries. This work marks the first effort to evaluate zero-shot generalization in Chinese models and provides the initial open-source framework and dataset dedicated to tool acquisition in Chinese LLMs. Our experiments show AtomTool outperforms the closed-source models like ChatGPT in zero-shot generalization in most cases. We also propose a novel dataset construction method and evaluation framework, examining prompt design and tool quantity effects on model performance. Overall, our work establishes a solid foundation for advancing tool acquisition in Chinese LLMs.

Keywords: Large Language Models · Tool Learning · Zero-Shot Generalization · Open-Source Framework · Chinese Dataset.

1 Introduction

In the current technological context, the application and challenges of Large Language Model (LLM) invocation tools have become a hot topic in the field

* Corresponding authors: Junqi Zhang and Wenbo Hu.
Yongle, Wenbo and Richang were supported by National Natural Science Foundation of China (No. 62306098), the SMP-IDATA Open Youth Fund (No. SMP-2023-iData-009) and Funds for the Central Universities (No. JZ2024HGTB0256)

of artificial intelligence [1]. Despite significant progress in tool learning with large language models, they still face significant challenges in terms of dataset diversity, complexity of API parameter handling, hallucination control, and evaluation standards:(i) *Model*: Currently, open-source large language models have relatively limited capabilities in tool usage, mainly restricted by their training data and functional range. In contrast, closed-source large language models, such as ChatGPT, perform better in tool usage due to their vast training datasets and complex model architectures.(ii) *Dataset*: At present, most open-source datasets are not suitable for API calls and have little functional difference. These datasets mostly come from platforms like Hugging Face and usually lack specific functional differentiation. Additionally, there is a very limited number of open-source datasets for Chinese, which restricts the development and application of Chinese large language models in the field of tool learning.(iii) *API*: In the field of tool learning, the API parameters currently in use are often quite simplistic, meaning that models struggle to understand and process a wide variety of API parameters. This simplicity limits the models' ability to handle complex tasks and reduces their application flexibility.(iv) *Hallucination* Phenomenon: A major issue in the field of tool learning is the "hallucination phenomenon," where models generate inaccurate or false information due to a lack of precise data or based on incorrect understanding. This phenomenon poses a challenge to the reliability and trustworthiness of models.(v) *Evaluation*: Currently, there is no standard method for evaluating tool learning. How to accurately assess the correctness of tool selection has become a problem. Without a unified standard for evaluation, it becomes difficult to compare the performance of different models.

While the challenges we face are substantial, they are not insurmountable. Below, we briefly outline our key contributions, including the development of the AtomTool framework and our novel approaches to dataset creation, API integration, hallucination control, and evaluation methods. These advancements collectively represent our targeted efforts to overcome the significant challenges in the application of large language models for tool usage.

- Model: The introduction of AtomTool, an open-source model framework, significantly enhances the capabilities of LLMs in tool usage, especially in zero-shot generalization. This framework allows users to interact directly with AtomTool using relevant tool information without needing retraining, thus addressing the limitations in current open-source LLMs that are primarily restricted by their training data and functional range.
- Dataset: We have developed a groundbreaking unsupervised method for dataset construction. This method leverages real APIs and incorporates a wide range of parameter settings, enabling the automated creation of datasets that enhance the zero-shot generalization capabilities of models. Our approach is distinct in its ability to generate data without direct human supervision, which significantly reduces the biases and limitations imposed by manually curated datasets.Capitalizing on this innovative method, we have open-sourced a high-quality Chinese dataset containing 16,000 entries. By making this dataset publicly available, we aim to furnish researchers and

developers with a robust resource that catalyzes further development and application of Chinese large language models in the field.
- API: The research includes real APIs in the dataset and incorporates a variety of parameter settings. This approach enables the models to truly learn tool usage capabilities rather than just memorizing the training data, thus addressing the challenges related to the simplicity of API parameters currently in use which limits the models' ability to handle complex tasks.
- Hallucination Phenomenon: The research effectively addresses the "hallucination phenomenon"—where models generate inaccurate or false information. This is achieved by refining prompt design, as well as employing retrieval enhancement techniques to manage the issue of lengthy prompts due to a large tool pool. These innovations improve the model's efficiency and accuracy.
- Evaluation: The study proposes novel evaluation frameworks to accurately assess the correctness of tool selection, addressing the current lack of standardized methods for evaluating tool learning. The proposed methods aim to minimize the evaluation costs while maintaining high accuracy, thus facilitating a better comparison of different models' performances.

In this paper, we systematically explore the integration of tool invocation capabilities into Large Language Models through the AtomTool framework. Section 2 details the construction of our diverse and complex dataset tailored for training LLMs. Section 3 introduces the AtomTool framework, emphasizing its efficient tool selection mechanism. Section 4 presents experimental results, demonstrating AtomTool's robust zero-shot generalization capabilities. Section 5 reviews related work, providing context for our advancements. In conclusion, Section 6 summarizes our findings and the impact on future research.

2 DATASET

We need the LLM to acquire the following three capabilities:

- **Tool Utilization:**When the user's command requires the assistance of a specific tool and the necessary tool is available in the tool pool, the LLM should accurately select and use the tool, parsing its parameters to help formulate an answer.
- **LLM reference:**When the user's command can be addressed without external tools or the required tools are not available, the LLM should rely on its own pre-trained knowledge to respond. However, the LLM's knowledge has its limits. When it is unable to provide an answer, it should respectfully decline and explain the reason. For example, when a user requests the latest weather update but there is no weather tool in the pool, the LLM should inform the user accordingly.
- **Response Optimization:**Given that the output from tools is often technical and verbose, the LLM needs to interpret these results and respond to

Table 1. The content of special tokens in data composition.

Special token	Content
System	Tools for large language models
Human	User's question
AssistantTool	Tool calls generated by the LLM
AssistantToolResult	Answers from tools
Assistant	The final answer returned to the user

the user's queries in a friendly and concise manner. For instance, if a user inquires about appropriate attire based on the day's weather, the LLM would use a weather tool. Although such tools typically offer detailed data including wind direction, temperature, and barometric pressure, the LLM should integrate this information with its knowledge to suggest suitable clothing options for the day.

Since we need the LLM to have the above capabilities, we need three kinds of data to fine-tune the LLM, which we will introduce in 2.1. In addition, we will introduce a self-supervised data set construction method in 2.2.

2.1 Dataset classification

To help the large language model understand the content of prompt and parse the language model's responses, we set up five special tokens, as shown in Table 1.

As illustrated in Fig. 1, the System provides detailed information for each tool it encompasses. For every tool, this includes the function name, function description, parameter types and their meanings, as well as the types and meanings of the return parameters.We describe the tool call function in the form of code in the System, which uses the ability of LLM to read code learned in pre-training to help LLM better understand the tool.To maintain optimal performance, the System limits the number of tools to a maximum of ten. This constraint is strategic, as overly lengthy prompts can hinder the LLM's efficiency in referencing and processing information. AssistantTool is usually an executable line of python code that contains the tool name and corresponding parameters. AssistantToolResult content is the execution result of the tool, because the result of the tool has a variety of types, the number of results is not unique, we use the form of a dictionary to store the execution result of the tool. The Assistant is the LLM's answer to the user's request. When the LLM calls the tool, the content of Assustant will be answered with reference to the AssistantToolResult. When the LLM does not need a reference tool, the content of the Assistant is the LLM's answer according to its own ability; If the LLM cannot answer the request based on its capability, the LLM will refuse to answer the request. For example, if a user requests the LLM to generate an image, but there is no tool that can generate an image in the tool pool, the LLM will refuse to complete the request.

Below, we introduce the three main types of fine-tuning data:

Fig. 1. Case on Data

Tool Selection In the first type of data, we want the LLM to be able to accurately select the tools to use based on up to ten tools available to it and user requests. Therefore, the first type of data consists of System, Human, and AssistantTool, where the System must contain tools that the LLM can use to complete the user's request.

Without Tool In the second type of data, we want the LLM to answer the user's request without using a tool. Since LLM has learned a lot of knowledge and capabilities in the pre-training stage, we hope that when the tools provided by the System cannot complete user requests, LLM can answer according to its own capabilities. Therefore, the second type of data includes System, Human, and Assistant. In order to avoid LLMS answering user requests with overconfidence, we have added some denied answer data to the second type of data for real-time questions, multimodal questions, etc.

Answer Questions In the third type of data, we want the LLM to answer user requests in Human based on the AssistantToolResult. Since the content of the AssistantToolResult is usually diverse and redundant, we hope that LLM can summarize the content of the AssistantToolResult and answer the user's request. Therefore, the third type of data includes System, Human, AssistantTool, AssistantToolResult, and Assistant.

2.2 Dataset construction

In traditional methods, constructing annotated data sets usually requires a lot of human annotation and time resources. Each data sample needs to be meticulously annotated by a human expert, which is not only costly but inefficient.

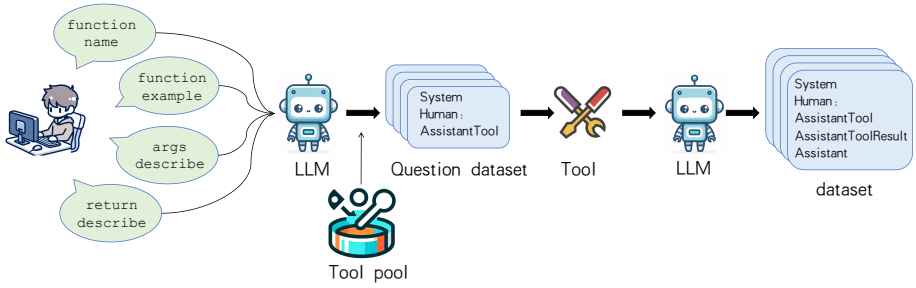

Fig. 2. Pipelining of data construction

In addition, manually labeling data is often subjective and can lead to data bias and quality issues.

In this paper, we introduce a model self-supervised data construction method based on ChatGPT [7]. This approach outperforms traditional methods in terms of efficiency and scalability. By leveraging existing data and advanced learning algorithms, the method autonomously generates new training samples. A significant benefit is its capability to rapidly produce large and diverse datasets, which is crucial for training sophisticated deep learning models. Moreover, self-supervision reduces the need for manual annotation, thereby cutting costs and enhancing the flexibility and scalability of the data construction process. This method also excels in identifying and leveraging underlying structures and patterns within the data, which often elude human annotators. As a result, it not only produces high-quality training data but also significantly improves learning efficiency and the performance of subsequent model

The model self-supervised construction method based on ChatGPT, as shown in Fig. 2, mainly consists of three steps. In the first step, we provide ChatGPT with the API description information, which includes the name of the tool API, its arguments, return values, and an example. ChatGPT then generates the user request (Human) and tool call code (AssistantTool) corresponding to the API. However, due to the presence of hallucinations in the data generated automatically by the model, we use a tool pool to verify the data generated by ChatGPT. In the second step, we call the corresponding API according to the problem dataset to obtain the tool's response (AssistantToolResult). Finally, in the third step, we provide ChatGPT with the Human and AssistantToolResult, enabling it to generate a reasonable response(Assistant).

3 AtomTool

We have provided a framework for large language models to invoke external tools, named AtomTool. Through the AtomTool framework, large language models can acquire the ability to invoke external tools with only a minimal amount of fine-tuning.

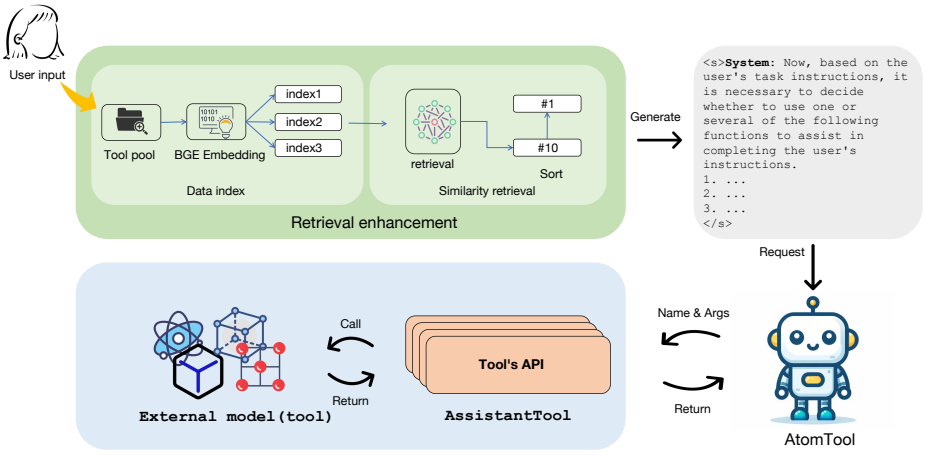

Fig. 3. AtomTool

We have developed a framework called AtomTool for large language models to invoke external tools. Through this framework, large language models can acquire the capability to call external tools with minimal fine-tuning. As shown in Figure 3, within the AtomTool framework, the system initially searches the tool pool based on user input and selects the tools most likely to be used by the LLM through similarity analysis, forming the System part of the prompt. Then, based on the specific needs of the user and the tools identified in the System, the LLM decides whether to invoke an external tool. If there is no necessity to use a tool, the LLM will satisfy the user's request using its own processing capabilities. If there is a need to invoke a tool, the LLM will access the tool's API and, based on the feedback from the tool API, summarize an appropriate answer to return to the user.

3.1 Tool pool

Prompt for large language models is usually limited in length, mainly due to the model's architecture and computational resource constraints. Rafal Jozefowicz et al. found that longer inputs provide more contextual information, which helps models understand and respond to user requests more accurately. However, too long an input can exceed the processing capacity of the model, resulting in degraded performance or an inability to process the information efficiently. Therefore, specifying an appropriate prompt length can improve the performance of the model.

In this paper, given the extensive variety of tool types available, it is impractical to incorporate every tool into the System. Therefore, we employ a retrieval enhancement strategy to selectively add tools to the System for AtomTool's use. Specifically, we utilize BGE Embedding for Chinese word vector embedding,

which enables us to effectively screen potential tools that AtomTool might use, based on their semantic similarity to the user's request.

As depicted in Figure 3, BGE Embedding is applied to vectorize a substantial dataset of tool information within the tool pool. Subsequently, we compute the semantic similarity between these embeddings and the embeddings derived from the user's request. The tools are then ranked according to their similarity scores, allowing AtomTool to make informed selections that closely align with the user's specified needs.

4 Experiments

4.1 Experimental Setup

Atom-7B Based on Llama2 [6], Atom utilizes large-scale Chinese data for continuous pre-training. The range of data includes encyclopedias, books, blogs, news articles, announcements, novels, and specialized data from fields such as finance, law, medicine, and coding. The training corpus also encompasses data from professional academic papers and datasets from Chinese natural language processing competitions. For more detailed information and resources, visit the GitHub page.[3]

Finetuning LoRA (Low-Rank Adaptation) is an advanced model tuning technique primarily used for efficient and effective fine-tuning of large pre-trained models [9]. Its core advantage lies in its ability to enhance performance by adjusting only a small part of the model weights while maintaining the original model structure and the number of parameters. This method significantly reduces the number of parameters required for tuning, thereby reducing computational burden and storage requirements.

Evaluation There is no standard answer to the choice of tools, for example, to ask LLM what the latest good movies are, you can use not only the search engine tool, but also the movie charting tool. Therefore, assessing the results of tool selection is a tricky issue.

To minimize human involvement, to minimize overhead. We use the two-layer evaluation method of chatGPT assisted evaluation for evaluation. Through experiments, we found that chatGPT is feasible to evaluate whether the tool selection can fulfill the user's request, but it is not sensitive to the problem of illusion. Therefore, for the results of LLM, we first check the tool names and parameters in the tool pool for hallucinations. We then use chatGPT to evaluate whether the LLM selected tool can fulfill the user request.

In order to verify the feasibility of the double-layer evaluation method, we verified the method manually, and we found that the accuracy of the double-layer evaluation reached 96.1%.

[3] https://github.com/LlamaFamily

Table 2. Comparison of accuracy between traditional prompts and function definition prompts in understanding code function definitions and calling corresponding APIs.

Prompt	Accuracy	Function name accuracy
Traditional prompt	67.3%	80.4%
Function definition prompt	**74.5%**	**82.3%**

Table 3. The effect of adding example to prompt on AtomTool.

Prompt	Accuracy	Function name accuracy
Without example	74.5%	82.3%
Positive example	**91.8%**	**97.8%**
Negative example	55.2%	60.3%

Benchmarks Due to the small number of benchmarks that large language models now call external tools, there is no public Chinese benchmark. Therefore, we built the Chinese evaluation datasets for the Chinese tool API. We built two kinds of evaluation datasets. One is evaluation datasets based on Chinese tool API in training data, which is used to evaluate AtomTool's ability to call external tools. The other is the evaluation datasets based on the Chinese tool API that is not in the training data, which is used to evaluate the zero-sample generalization of AtomTool.

4.2 Exploring Model Proficiency with In-Training Tool Invocation

Prompt Impact on Tool Invocation In traditional prompts, the focus is on describing the content to be referenced in more detail for the LLM to refer to [3–5], while neglecting the form of the prompt. Previous tool prompts typically provide the LLM with direct descriptions of the tool. To enable the LLM to better understand and use tools, we provide information to the LLM through function definitions and have conducted experimental comparisons.As shown in Table 2, we can see that LLMs are able to better understand function definitions in code form, thereby more effectively calling the corresponding APIs.

We added both correct examples and incorrect examples to the prompt provided to AtomTool. From the results of Experiment Table 3, we can understand: Positive examples can significantly enhance the ability of AtomTool to invoke tools, especially in terms of the Function name; Negative examples reduce the ability of AtomTool to invoke tools, making it more prone to hallucinations. From this, we can conclude: AtomTool relies largely on the content in the System rather than solely on training data, which also supports the zero-shot generalization capability of AtomTool.

Number of tools In exploring the capability of Large Language Models (LLMs) to use tools, it can be anticipated that as the number of tools in the training data increases, the model's ability to utilize tools will improve. However, acquiring high-quality tools often comes at a high cost, so we have explored the optimal

Table 4. Accuracy obtained from training with different numbers of tools.

Number of tools	Accuracy
Five	62.0%
Ten	77.6%
Twenty	88.9%
Thirty	91.6%
thirty-five	**91.8%**

number of tools in the training data. As shown in Table 4, with the increase in the number of tools used during training, the accuracy of the LLM in invoking tools also continuously increases. When the number of tools is relatively small, the rate of increase in accuracy is quite substantial. However, as the number of tools increases, the rate of increase in accuracy significantly diminishes. This suggests that a certain quantity of tools is sufficient for LLMs to learn the capability to use tools effectively. Based on the experimental results, we recommend training LLMs with thirty types of tools.

4.3 Zero-Shot Generalization in Model Tool Invocation

Zero-Shot Generalization refers to the ability of a model to perform on new categories or tasks it has never seen before. In traditional machine learning models, both training and testing of the model are conducted on the same or similar categories of data. However, in the scenario of zero-shot generalization, the model needs to deal with data that is completely different from the training data.The current version of ChatGPT is widely recognized as a model with zero-shot generalization capabilities. However, it is proprietary and closed-source. In the field of tool learning, there are currently no models with zero-shot generalization abilities.

Through conducting experiments with Zero-Shot Learning, we discovered that AtomTool possesses strong Zero-Shot Generalization capabilities. This means that for a new tool API, users only need to provide AtomTool with the corresponding API description, and AtomTool can accurately call the tool API to complete tasks based on actual situations.

Datasets To effectively validate the zero-shot generalization capability of AtomTool, we intentionally used some tool APIs in the construction of our datasets that were not present in the training samples. Additionally, we employed three different types of APIs for testing. As shown in Table 5, we categorized these APIs into three levels. The first level consists of API calls without parameters, which are less challenging as the model does not need to focus on the correctness of the content and type of parameters; the second level includes API calls with a single parameter, which are more common in training and have a higher usage rate; the third level involves API calls with multiple parameters, where the model not only needs to accurately extract parameters from user requests but also ensure the correct correspondence of parameter names and types.

Table 5. Test Data Cases for Zero-Shot Generalization Ability.

Data type	Example Human	Example AssistantTool
Without Parameters	I would like to know what's new on Bilibili recently.	BiliNews()
Single Parameter	I want to know if my QQ number "32779872" is logged in on the computer.	QqOnline(qq='32779872')
Multiple Parameters	My encryption password is 454567. Please help me encrypt the following text: "Tomorrow will be better".	EncoderForKey(key='454567', string='Tomorrow will be better')"

Table 6. AtomTool model tests zero-shot generalization capabilities on different types of data.

Data type	Accuracy	Hallucination
Without Parameters	90.6%	8.66%
Single Parameter	75.6%	19.3%
Multiple Parameters	70.4.%	25.3%

Result We conducted experiments using AtomTool on three types of data. As shown in Table 6 of the experiment results, we found that when AtomTool uses an API that is not included in its training data, it performs well if the API is relatively simple. However, if the API becomes complex, the hallucination phenomenon of AtomTool becomes very apparent.

We compared the zero-shot generalization capabilities of the proprietary ChatGPT [7] and the Chinese-trained open-source model Moss-Moon-003-SFT-Plugin [8]. The Moss-Moon-003-SFT-Plugin is a fine-tuned version of the Moss base model, developed using approximately 1.1 million rounds of dialogue data and about 300,000 rounds of multi-turn dialogue data enhanced with plugins. The plugin data includes search engines, text-to-image , calculators, and equation solvers.

From the results in Table 7, we can draw the following conclusions: AtomTool's zero-shot generalization capabilities in tool learning are close to or even better than those of the proprietary ChatGPT; the Chinese open-source model Moss-Moon-003-SFT-Plugin performs poorly in zero-shot generalization, with too obvious hallucination phenomena; the main challenge for large language models in tool learning currently remains the hallucination issue.

5 Related Work

Large Language Models In the realm of Large Language Models (LLMs), recent advancements have extended their impact beyond traditional natural language processing tasks, reaching into domains such as program synthesis [14, 6, 16]. The evolution of LLMs has been driven by techniques such as prompting [12, 20]

Table 7. Comparing the zero-shot generalization capabilities of different models.

Model	Accuracy	Hallucination
moss-moon-003-sft-plugin	4.06%	91.0%
chatGPT	75.6%	19.3%
AtomTool	**78.2%**	**18.2%**

and instruction fine-tuning, which enhance the capabilities of pre-trained models. Prompting provides contextual cues to guide the model's responses, while instruction fine-tuning refines the model's parameters to align with specific task requirements. The advent of open-sourced models like LLaMA [6], Alpaca [10], and Vicuna [11] has further democratized access to LLMs, enabling broader experimentation and deeper insights into their functionalities. These developments collectively signal a significant leap forward in the field, promising more sophisticated and adaptable models for a wide array of applications in artificial intelligence.

Tool Usage The use of external tools to enhance the capabilities of Large Language Models (LLMs) has been an area of growing interest in recent research. [17–19]The integration of tools with LLMs has been shown to improve accuracy, efficiency, and versatility for a broad spectrum of applications, ranging from knowledge retrieval [21–24] and search engines [17, 25] to calculators [26, 27], Python interpreters [20, 28], and even simulation engines. Notably, models like Toolformer have pioneered the exploration of tool usage within LLMs, incorporating functionalities such as web browsing, calculators, translation systems, and Python interpreters [29]. In contrast to existing works that often focus on specific tools or showcase the potential of LLMs with tool usage, our paper aims to explore the utilization of a wide array of tools (i.e., API calls) in an open-ended manner. We emphasize the zero-shot generalization capabilities of LLMs in tool usage, which allows them to adapt to new tools and tasks without explicit retraining. Our work seeks to bridge the gap between open-source and chatGPT by investigating the underlying mechanisms of tool-use acquisition and establishing a systematic method for evaluation and training, including fine-tuning. This approach not only enhances the understanding of LLMs' interaction with external tools but also lays the groundwork for building a comprehensive pipeline for future applications in diverse domains.

6 Conclusion

In this study, we introduced AtomTool, an innovative framework for enhancing the zero-shot generalization capabilities of large language models (LLMs) in tool usage. Our approach leverages a unique dataset construction method, prompt design, and retrieval enhancement techniques to address the challenges of dataset diversity, API parameter complexity, hallucination control, and evaluation standards in tool learning. AtomTool demonstrated superior zero-shot generalization

abilities compared to closed-source models like ChatGPT, especially in handling new tools that it has never encountered before. This advancement is crucial for the practical application of LLMs, as it enables them to adapt to a wide variety of tasks with minimal fine-tuning.

Furthermore, our research contributes to the field by providing an open-source Chinese dataset, which can serve as a valuable resource for researchers and developers. The exploration of prompt design and the impact of tool quantity in training data offer insights into optimizing LLMs for tool usage. Despite the achievements, our experiments also highlighted the ongoing challenge of hallucination phenomena in complex tool APIs, indicating a direction for future research.

In conclusion, AtomTool represents a significant step forward in the development of LLMs capable of zero-shot generalization in tool usage. Our work not only advances the understanding of tool learning with LLMs but also lays the foundation for more robust and versatile models that can seamlessly integrate with external tools to perform a broader range of tasks.

References

1. Qin, Y., Hu, S., Lin, Y., et al.: Tool learning with foundation models. arXiv preprint arXiv:2304.08354 (2023)
2. Hsieh, C.-Y., Chen, S.-A., Li, C.-L., Fujii, Y., Ratner, A., Lee, C.-Y., Krishna, R., Pfister, T.: Tool documentation enables zero-shot tool-usage with large language models. arXiv preprint arXiv:2308.00675 (2023)
3. Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., Neubig, G.: Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. ACM Computing Surveys **55**(9), 1–35 (2023). ACM, New York, NY
4. Zamfirescu-Pereira, J.D., Wong, R.Y., Hartmann, B., Yang, Q.: Why Johnny can't prompt: how non-AI experts try (and fail) to design LLM prompts. In: Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems, pp. 1–21 (2023)
5. Li, L., Zhang, Y., Chen, L.: Prompt distillation for efficient llm-based recommendation. In: Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, pp. 1348–1357 (2023)
6. Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al.: Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288 (2023)
7. Brown, T.B., Mann, B., Ryder, N., et al.: Language models are few-shot learners. arXiv preprint arXiv:2005.14165 (2020)
8. Sun, T., Zhang, X., He, Z., Li, P., Cheng, Q., Yan, H., Liu, X., Shao, Y., Tang, Q., Zhao, X., et al.: Moss: Training conversational language models from synthetic data. arXiv preprint arXiv:2307.15020, **7** (2023)
9. Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685 (2021)
10. Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., Hashimoto, T.B.: Stanford Alpaca: An Instruction-following LLaMA model. (2023)

11. Chiang, W.-L., Li, Z., Lin, Z., Sheng, Y., Wu, Z., Zhang, H., Zheng, L., Zhuang, S., Zhuang, Y., Gonzalez, J.E., et al.: Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. See https://vicuna.lmsys.org (accessed 14 April 2023), **2**(3), 6 (2023)
12. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q.V., Zhou, D., et al.: Chain-of-thought prompting elicits reasoning in large language models. Advances in Neural Information Processing Systems **35**, 24824–24837 (2022)
13. Gao, L., Madaan, A., Zhou, S., Alon, U., Liu, P., Yang, Y., Callan, J., Neubig, G.: Pal: Program-aided language models. In: International Conference on Machine Learning, pp. 10764–10799 (2023). PMLR
14. Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H.W., Sutton, C., Gehrmann, S., et al.: Palm: Scaling language modeling with pathways. Journal of Machine Learning Research **24**(240), 1–113 (2023)
15. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al.: Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971 (2023)
16. Patil, S.G., Zhang, T., Wang, X., Gonzalez, J.E.: Gorilla: Large language model connected with massive apis. arXiv preprint arXiv:2305.15334 (2023)
17. Komeili, M., Shuster, K., Weston, J.: Internet-augmented dialogue generation. arXiv preprint arXiv:2107.07566 (2021)
18. Lazaridou, A., Gribovskaya, E., Stokowiec, W., Grigorev, N.: Internet-augmented language models through few-shot prompting for open-domain question answering. arXiv preprint arXiv:2203.05115 (2022)
19. Nakano, R., Hilton, J., Balaji, S., Wu, J., Ouyang, L., Kim, C., Hesse, C., Jain, S., Kosaraju, V., Saunders, W., et al.: Webgpt: Browser-assisted question-answering with human feedback. arXiv preprint arXiv:2112.09332 (2021)
20. Gao, L., Madaan, A., Zhou, S., Alon, U., Liu, P., Yang, Y., Callan, J., Neubig, G.: Pal: Program-aided language models. In: International Conference on Machine Learning, pp. 10764–10799 (2023). PMLR
21. Borgeaud, S., Mensch, A., Hoffmann, J., Cai, T., Rutherford, E., Millican, K., Van Den Driessche, G.B., Lespiau, J.-B., Damoc, B., Clark, A., et al.: Improving language models by retrieving from trillions of tokens. In: International Conference on Machine Learning, pp. 2206–2240 (2022). PMLR
22. Guu, K., Lee, K., Tung, Z., Pasupat, P., Chang, M.: Retrieval augmented language model pre-training. In: International Conference on Machine Learning, pp. 3929–3938 (2020). PMLR
23. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al.: Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in Neural Information Processing Systems **33**, 9459–9474 (2020)
24. Yao, S., Chen, H., Yang, J., Narasimhan, K.: Webshop: Towards scalable real-world web interaction with grounded language agents. Advances in Neural Information Processing Systems **35**, 20744–20757 (2022)
25. Paranjape, B., Lundberg, S., Singh, S., Hajishirzi, H., Zettlemoyer, L., Ribeiro, M.T.: Art: Automatic multi-step reasoning and tool-use for large language models. arXiv preprint arXiv:2303.09014 (2023)
26. Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al.: Training verifiers to solve math word problems. arXiv preprint arXiv:2110.14168 (2021)

27. Parisi, A., Zhao, Y., Fiedel, N.: Talm: Tool augmented language models. arXiv preprint arXiv:2205.12255 (2022)
28. Chen, W., Ma, X., Wang, X., Cohen, W.W.: Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. arXiv preprint arXiv:2211.12588 (2022)
29. Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Hambro, E., Zettlemoyer, L., Cancedda, N., Scialom, T.: Toolformer: Language models can teach themselves to use tools. Advances in Neural Information Processing Systems **36** (2024)