Towards Autonomous Tool Utilization in Language Models: A Unified, Efficient and Scalable Framework

Zhi Li, Yicheng Li, Hequan Ye and Yin Zhang
Zhejiang University, China
{zhili, yichengli, yehequan, zhangyin98}@zju.edu.cn

Abstract

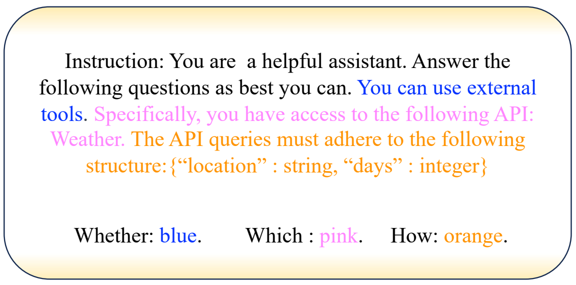
In recent research, significant advancements have been achieved in tool learning for large language models. Looking towards future advanced studies, the issue of fully autonomous tool utilization is particularly intriguing: Given only a query, language models can autonomously decide whether to employ a tool, which specific tool to select, and how to utilize these tools, all without needing any tool-specific prompts within the context. To achieve this, we introduce a unified, efficient, and scalable framework for fine-tuning language models. Based on the degree of tool dependency, we initially categorize queries into three distinct types. By transforming the entire process into a sequential decision-making problem through conditional probability decomposition, our approach unifies the three types and autoregressively generates decision processes. Concurrently, we've introduced an "instruct, execute, and reformat" strategy specifically designed for efficient data annotation. Through end-to-end training on the annotated dataset comprising 26 diverse APIs, the model demonstrates a level of self-awareness, automatically seeking tool assistance when necessary. It significantly surpasses original instruction-tuned open-source language models and GPT-3.5/4 on multiple evaluation metrics. To address real-world scalability needs, we've enhanced our framework with a dynamic rehearsal strategy for continual learning, proven to require minimal new annotations to exhibit remarkable performance.

Keywords: large language model, tool learning, autonomous, end-to-end learning

1. Introduction

In recent years, Large Language Models (LLMs) have made significant strides in natural language processing tasks, showcasing impressive capabilities and achieving state-of-the-art performance in various domains. These models leverage massive amounts of pre-training data and sophisticated training methods to generate coherent and contextually relevant outputs. However, despite these advancements, LLMs remain fundamentally constrained by the information they can store in fixed weights and the challenges posed by limited context. Consequently, a new research trend has emerged, aiming to address these challenges by allowing LLMs to leverage external tools. For example, WebGPT (Nakano et al., 2021) and Internet-augmented language models (Lazaridou et al., 2022) both employ search engines to access comprehensive information and generate more refined responses. Visual ChatGPT (Wu et al., 2023) and HuggingGPT (Shen et al., 2023) harness the power of LLMs to control AI models and effectively handle tasks involving multiple modalities.

The aforementioned studies focus primarily on the strategies and techniques involved in how to utilize various tools for language models. As illustrated in Figure 1, whether a language model should utilize tools for a given problem and selecting which tool is typically predetermined by manual prompts. Following this research path, Gorilla (Patil et al., 2023) and ToolLLM (Qin et al., 2023) have significantly expanded the number of tools by integrating with external retrieval systems. To further enhance the autonomy of tool usage in language models, GPT4Tools (Yang et al., 2023) and ToolAlpaca (Tang et al., 2023) explicitly provide multiple tools relevant to specific scenarios in the context, such as multimodal tools, and then fine-tune the models, enabling the language models to autonomously decide whether or which tool to use. However, for these methods, even aside from some potential drawbacks, such as additional retrieval steps leading to cumulative errors, the limited context length making it difficult to expand the number of tools, the language model itself still plays a passive role, requiring external prompts to guide the model in problem-solving.


Instruction: You are a helpful assistant. Answer the following questions as best you can. You can use external tools. Specifically, you have access to the following API: Weather. The API queries must adhere to the following structure:{"location" : string, "days" : integer}

Whether: blue. Which : pink. How: orange.

Figure 1: Using weather queries as an example, we showcase the prevailing methods of tool utilization.

*Corresponding Author: Yin Zhang.

Method	End to End Learning			Advantage		
	Whether	Which	How	Efficient	Scalable	Autonomous
Toolformer (Schick et al., 2023)	✓	✓	✓			✓
TRICE (Qiao et al., 2023)	✓		✓			✓
Gorilla(Patil et al., 2023)			✓	✓	✓	
ToolLLM (Qin et al., 2023)			✓	✓	✓	
GPT4Tools(Yang et al., 2023)	✓	✓	✓	✓	✓	
ToolAlpaca (Tang et al., 2023)	✓	✓	✓	✓	✓	
Our Work	✓	✓	✓	✓	✓	✓

Table 1: Comparative analysis of various studies. 'Autonomous' implies that there is no need to include any tool-related information in the prompt, as the model fully internalizes the knowledge and ability to use the tools.

We take the consideration even further, aspiring for large language models to achieve completely autonomous tool usage by fully internalizing various tool knowledge. Given only a query, language models can autonomously decide whether to employ a tool, which specific tool to select, and how to utilize these tools, all without needing any tool-specific prompts within the context. A detailed comparison reveals that, solely from the perspective of problem-setting, Toolformer (Schick et al., 2023) and TRICE (Qiao et al., 2023) focus on issues similar to what we are addressing. However, their methods have several shortcomings, such as the low efficiency of the self-supervised dataset construction method adopted by Toolformer (Schick et al., 2023), the limited types of tools examined, for instance, TRICE (Qiao et al., 2023) only investigated one calculator tool, and both lack discussion and analysis of scalability.

In pursuit of autonomous tool utilization, we propose a unified, efficient, and scalable framework. Specifically, our contributions are encapsulated as follows:

1. We've categorized queries into three precise and reasonable categories based on the extent of tool-reliance: 1. Directly solvable issues, like translations; 2. Challenges needing validation, such as intricate mathematical computations; 3. Unsolvable due to inherent limitations, exemplified by real-time queries. By transforming the entire process into a sequential decision-making problem through conditional probability decomposition, our approach unifies the three types and autoregressively generates decision processes.

2. Serving our framework, we have pioneered an "*instruct, execute, and reformat*" strategy for efficient dataset construction. Beginning with foundational seed queries, we enhance them via self-instruction. Subsequently, we guide a language model to devise the API call, taking cues from both the query and the tool instructions, as depicted in Figure 1. If the API call is executed successfully, a dedicated crowdsourcing team evaluates the data instance. In the concluding phase, we meticulously reformat the data to adhere to specified requirements by transitioning tool instructions from an input to an output role.

3. Considering real-world scenarios, we further extend our framework with a dynamic balanced rehearsal strategy for continual learning, which starts with a focus on new API categories, gradually increasing the proportion of old API categories until the balance is achieved. Experiments indicate that our framework requires minimal new tool-annotated data to exhibit remarkable performance while maintaining its existing tool capabilities.

4. Experiments show that the fine-tuned model exhibits a degree of self-awareness and can automatically seek tool assistance when needed and dramatically surpasses instruction-tuned both open-source language models and GPT models in tool selection, accurate tool usage, and overall problem-solving efficacy. Further ablation shows that our unified framework can effectively promote the model's learning across different tools.

2. Related Work

2.1. Instruction Tuning

Recent research (Ouyang et al., 2022; Chung et al., 2022) indicates that pre-trained language models, when fine-tuned with task-specific instruction data, excel at following natural language directives and accomplishing real-world tasks. In particular, models such as InstructGPT (Ouyang et al., 2022) and FLAN-T5 (Chung et al., 2022) have shown remarkable results through instruction-based fine-tuning.

The self-instruction methodology, as presented by Wang et al. (2022), posits that language models can bolster their aptitude for instruction adherence using self-generated instructional data. This concept has further motivated the utilization of data from powerful language models like Davinci [1], ChatGPT[2], GPT-4 (OpenAI, 2023) to refine open-source language models, such as Alpaca[3].

2.2. In-Context Tool Learning

LLMs have shown impressive in-context learning capabilities (Dong et al., 2022). This has paved the way for integrating tools by offering contextual tool descriptions and demonstrations (Paranjape et al., 2023; Gou et al., 2023; Hsieh et al., 2023; Chen et al., 2023). Building upon this, the introduction of reasoning chains has facilitated the tackling of more intricate problems. This methodology has catalyzed the creation of widely recognized industry solutions, such as ChatGPT plugins [4] and Langchain [5], with proven applications in significant research areas. Furthermore, PaLM-SAYCAN (Brohan et al., 2023) and PaLM-E (Driess et al., 2023) leverage high-level robotics APIs to command robots to execute real-world objectives. There have also been promising advances in the deployment of LLMs as coordinators of multiple neural models in multimodal reasoning tasks (Surís et al., 2023; Wu et al., 2023; Shen et al., 2023).

2.3. Supervised Tool Learning

Nevertheless, methods based on in-context learning exhibit certain drawbacks. Even for today's state-of-the-art LLMs such as GPT-4, there are challenges to generating accurate input arguments and eliminating the tendency to hallucinate the wrong usage of an API call. The alternative approach emphasizes the fine-tuning of LLMs to utilize tools. REALM (Guu et al., 2020), RAG (Lewis et al., 2020), RETRO (Borgeaud et al., 2022), and Atlas (Izacard et al., 2022) equip LLMs with an external retriever. TALM (Parisi et al., 2022) adapts these models for a handful of universal tools, such as QA models, calculators, and translators. Recent works mirroring our timeline are also advancing in this direction. Gorilla (Patil et al., 2023) and ToolLLM (Qin et al., 2023) have significantly expanded the number of tools by integrating with external retrieval systems. While GPT4Tools (Yang et al., 2023) and ToolAlpaca (Tang et al., 2023) finetune models to independently assess the need and selection of tools, there remains a requisite to explicitly provide all relevant tools and usage directives within the context. Diverging from these methodologies, our approach takes innovation a step further. We envision large language models that are capable of *fully autonomous tool utilization without any tool-specific prompts in the context*, achieved by thoroughly internalizing diverse tool knowledge.

[1]https://platform.openai.com/docs/models/gpt-3

[2]https://chat.openai.com/

[3]https://crfm.stanford.edu/2023/03/13/alpaca.html

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

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

3. Framework

3.1. Problem Formalization

Let's first agree on the abbreviations for the terms:

- Y denotes the target response.
- X stands for the query.
- $\mathrm{W_e}$ determines whether to use the tool.
- $\mathrm{W_i}$ indicates which tool to use.
- H indicates how to use the tool.

The aforementioned studies focus primarily on the strategies and techniques involved in how to utilize various tools for language models. The decision of whether a language model should deploy a particular tool, and which one to choose, is often guided by manual prompts. This decision-making process can be encapsulated by the following equation:

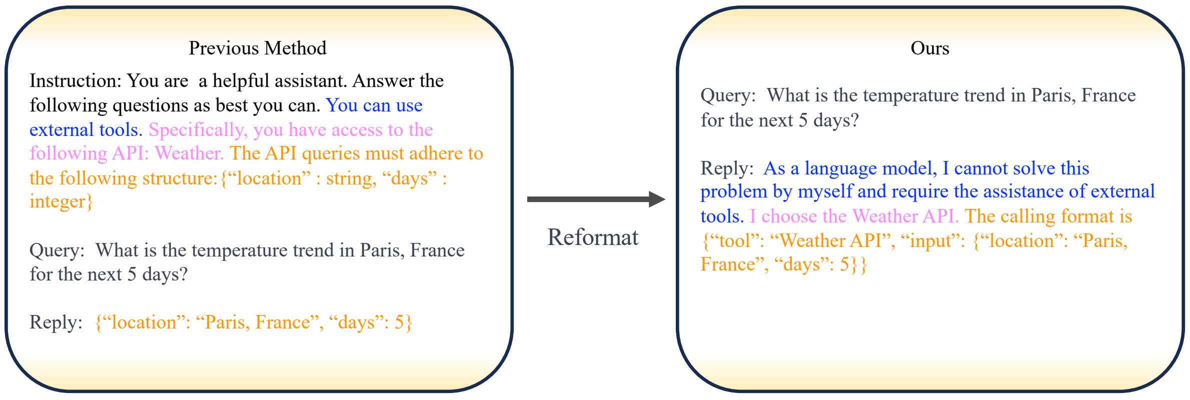
$$P(\mathrm{Y} \mid \mathrm{W_e}, \mathrm{W_i}, \mathrm{H}, \mathrm{X}) \tag{1}$$

In the near future, as the array of tools available to language models expands, the task of discerning and choosing the right tool from a vast array will undoubtedly become more challenging. While retrieval modules such as the BM25 and GPT embeddings often facilitate this selection process, they are not perfect. These modules sometimes fail because they lack comprehensive end-to-end optimization, sometimes leading to the selection of inappropriate tools. Such shortcomings can result in error propagation, with subsequent outputs failing to correct initial missteps. More critically, we assert that language models have untapped potential. Considering more advanced models or intelligent agents, they should be capable of independent decision-making. Specifically, in the absence of clear tool directives in a user's query, they should autonomously determine the necessity of a tool, identify the most fitting one for the query, and ascertain the correct method of invocation. This process can be represented by the following formula:

$$P(\mathrm{Y}, \mathrm{W_e}, \mathrm{W_i}, \mathrm{H} \mid \mathrm{X}) \tag{2}$$

3.2. Unified Conditional Probability Decomposition

Based on the degree of reliance on tools when solving problems with a large language model, we


Previous Method

Instruction: You are a helpful assistant. Answer the following questions as best you can. You can use external tools. Specifically, you have access to the following API: Weather. The API queries must adhere to the following structure:{"location" : string, "days" : integer}

Query: What is the temperature trend in Paris, France for the next 5 days?

Reply: {"location": "Paris, France", "days": 5}

Reformat

Ours

Query: What is the temperature trend in Paris, France for the next 5 days?

Reply: As a language model, I cannot solve this problem by myself and require the assistance of external tools. I choose the Weather API. The calling format is {"tool": "Weather API", "input": {"location": "Paris, France", "days": 5}}

Figure 2: Comparison of specific data instances using the previous methods and our method.

divide queries into three categories:

- Solvable Problems: These are issues that the large language model can address comprehensively in theory. They primarily encompass pure natural language processing tasks such as translation, sentiment analysis, and summarization.
- Verifiable Problems: These are challenges that the language model might have the capacity to tackle to a certain extent, but it is advisable to verify the results. This category includes complex mathematical reasoning and factual matters prone to hallucinations. For such problems, the language model initially tries to provide an answer on its own, but ideally should be cross-checked with external tools for confirmation.
- Unsolvable Problems: These are dilemmas that the language model simply cannot handle on its own. Examples are real-time queries and multimodal issues. Without the aid of external tools, the language model is entirely inept at providing solutions.

Considering the inherent sequential nature of the three questions, we introduce a unified approach: by leveraging the decomposition of conditional probabilities, we unify three types of queries and cast the entire process as a sequential decision-making problem. This lets the model autoregressively generate its decisions: first discerning the necessity of a tool, then choosing the right one, and finally determining its application method. The approach can be represented by the following formula:

$$\begin{aligned} &P(\mathrm{Y}, \mathrm{W_e}, \mathrm{W_i}, \mathrm{H} \mid \mathrm{X}) \\ &\quad = P(\mathrm{W_e} \mid \mathrm{X}) \\ &\quad \times P(\mathrm{W_i} \mid \mathrm{W_e}, \mathrm{X}) \\ &\quad \times P(\mathrm{H} \mid \mathrm{W_e}, \mathrm{W_i}, \mathrm{X}) \\ &\quad \times P(\mathrm{Y} \mid \mathrm{H}, \mathrm{W_e}, \mathrm{W_i}, \mathrm{X}) \end{aligned} \tag{3}$$

To better understand our methodology, Figure 2 illustrates a data instance identified as an unsolvable problem. Our framework efficiently simplifies the process of end-to-end learning, covering both the discernment of appropriate tools and their prime utilization. It is also distinguished by its significant transparency and interpretability. In practical applications, tools dealing with the same type of problems share common templates, fostering mutual enhancement of model convergence and establishing a foundation for the assimilation of new tools.

3.3. Efficient Dataset Construction

Relying solely on crowdsourcing for data annotation can incur significant costs, necessitating experts with in-depth knowledge of distinct API functionalities. Conversely, employing self-supervised methods such as those described by Schick et al. (2023) tends to be less efficient. Given the extensive array of tools accessible for language models, it's evident that a more cost-effective and efficient approach to dataset construction is imperative. Given these challenges, we have introduced the "instruct, execute, and reformat" methodology. As shown in Figure 3, we first use the commonly known self-instruct method to build the data, and then construct the desired dataset by reformatting. Specific steps include:

1. For any given API, our first step involves detailed labeling of seed queries. To illustrate,

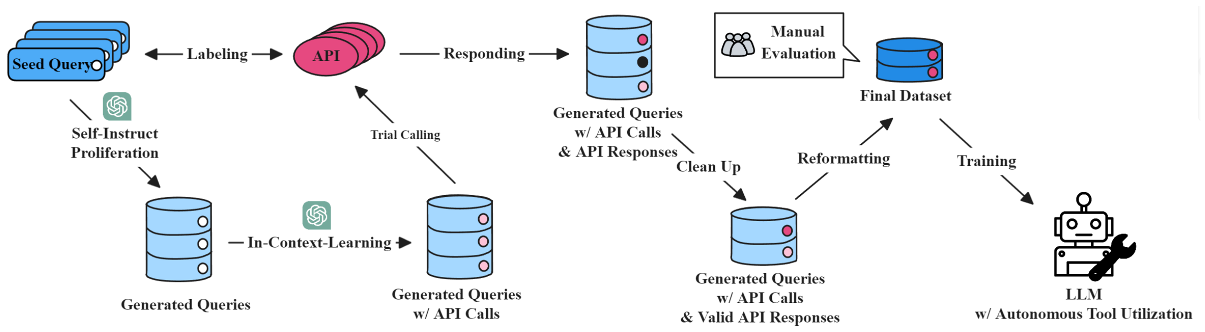

Seed Query
Labeling
API
Responding
Generated Queries w/ API Calls & API Responses
Manual Evaluation
Final Dataset
Self-Instruct Proliferation
Trial Calling
Clean Up
Reformatting
Training
Generated Queries
In-Context-Learning
Generated Queries w/ API Calls
Generated Queries w/ API Calls & Valid API Responses
LLM w/ Autonomous Tool Utilization

Figure 3: The “instruct, execute, and reformat” methodology for constructing datasets. It begins with detailed labeling of seed queries for a given API, followed by the application of a self-instruct strategy to generate more relevant queries. Using in-context learning, the language model is directed to create an appropriate API invocation. Subsequently, the data is reformatted to adhere to our specifications, as illustrated in Figure 2. Finally, human evaluation can be conducted on the reformatted data, along with training the language model to automate tool invocation.

we use the "Get weather today API" as an example, with seed queries like: “What is the current temperature and wind speed in London, UK?” and "Can you tell me how windy it is in Berlin, Germany today?”

2. We then employ a standard self-instruct strategy to proliferate relevant queries.
3. Using the in-context learning approach, the language model is directed to formulate the appropriate API invocation based on the query and provided API instructions.
4. This generated invocation is subsequently executed in the environment to obtain the API’s responses. Should the method be flawed, rendering the execution unsuccessful, such data is excluded, drastically reducing the subsequent manual intervention.
5. These data, comprising query, API instructions, generated API invocation, and API responses, will be reformatted to meet our desired specifications, as depicted in Figure 2.
6. Finally, we can sample the dataset for human evaluation while concurrently training the language model to to learn automatic tool utilization.

3.4. Dynamic Rehearsal for Scalability

Given the dynamically evolving nature of real-world API environments, which are characterized by an abundant and perpetually expanding collection of API tools, a limited set of API tools for training, as utilized in our experiments, is inadequate to mirror the complexities of the actual scenario. Hence, we delved deeper into the model’s scalable capabilities in continual learning. We propose a dynamic balanced rehearsal strategy, which start with a focus on new API categories, gradually increasing the proportion of old API categories until balance is achieved. Below is the formal description :

Let’s define the following variables:

- C_{old} represents the number of API categories already learned,
- C_{new} represents the number of newly introduced API categories,
- C_{total} represents the total number of API categories, where $C_{\text{total}} = C_{\text{old}} + C_{\text{new}}$,
- $\alpha(t)$ represents the proportion of sampling from old API categories at time t,
- t represents the current training epoch.

Our goal is for $\alpha(t)$ to start at 0 (initially, there are only new API categories) and, over time, balance out between new and old API categories until each category has an equal chance of being sampled. This can be achieved as follows:

$$\alpha(t) = \min\left(\frac{C_{old}}{C_{total}}, \gamma \cdot t\right)$$

Here, γ is a hyperparameter that dictates the amount by which $\alpha(t)$ increases each epoch until it equals C_{old}/C_{total}. This strategy ensures that while the model adapts to new APIs, it gradually integrates the knowledge of old APIs, achieving effective continual learning.

4. Experiment

4.1. API and Dataset Details

Table 2 shows our dataset of 26 APIs. Below is a detailed description of each API:

Type	Domain	Task	API
Solvable	NLP	Translation Sentiment Analysis Story Generation Summarization Grammar Correction etc	Language model itself
Verifiable	Reason	Math	Wolfram Alpha
	Fact	Entertainment	Bing Search
		Education	
		Culture and art	
		Politics	
		Sports	
		Map	Get route
			Get coordinates
			Get distance
			Search nearby
Unsolvable	Time	Recent Paper	Arxiv
		Stock	Get daily prices
			Get exchange rate
			Get open info
		Weather	Get weather today
			Forecast weather
	Mutimodal	Image Understanding	Image captioning
			Detect the Given Object
			Answer Question About The Image
			Segment the given object
			Remove Something From The Photo
		Image Generation	Generate Image Condition On Text
			Generate Image Condition On Normal Map
			Generate Image Condition On Pose Image
			Generate Image Condition On Depth
			Generate Image Condition On Sketch Image
			Generate Image Condition On Segmentations
			Generate Image Condition On Soft Hed Boundary Image
			Generate Image Condition On Canny Image

Table 2: Statistical overview of type, domain, task, and API.

- Wolfram Alpha API [6]: This API grants developers access to Wolfram Alpha's unique knowledge engine for computational tasks.
- Bing Search API [7]: It facilitates the retrieval of factual information across various domains, including entertainment, education, sports, and more.
- Map APIs [8]. Designed to support user queries regarding routes, distances, latitude coordinates, and points of interest in proximity.
- Stock APIs [9]: This API provides daily stock prices, exchange rate information between two currencies, and opening details for specific stocks.

[6]https://www.wolframalpha.com/

[7]https://www.microsoft.com/en-us/bing/apis/bing-web-search-api

[8]https://learn.microsoft.com/en-us/bingmaps/

[9]https://www.alphavantage.co/documentation/

- ArXiv API [10]: Useful for fetching the latest academic contributions from the arXiv repository.
- Weather APIs [11]: Offer real-time weather conditions for any given city and forecast up to N days ahead.
- Multimodal APIs are constructed based on a variety of pretrained multimodal models. For image understanding, these encompass the following: Image captioning (Li et al., 2022) , Detect the Given Object (Carion et al., 2020), Answer Question About The Image(Li et al., 2022), Segment the given object (Cheng et al., 2021a), Remove Something From The Photo(Cheng et al., 2021b; Rombach et al., 2022a), Regarding image generation models, stable diffusion (Rombach et al., 2022b) is used for Generate Image Condition On Text.

[10]https://info.arxiv.org/help/api/basics.html

[11]https://www.weatherapi.com/

Other image generation models primarily originate from ControlNet(Zhang and Agrawala, 2023). These mutimodal models are collected from several sources including HuggingFace Transformers[12], Maskformer[13], and ControlNet[14].

We generated 1000 GPT-4 query-response pairs per API, dividing them into 800 training, 100 validation, and 100 testing pairs. The dataset is publicly available [15].

4.2. Environment

We've established an environment that incorporates the mentioned API, offering the following capabilities:

1. Using regular expressions, it extracts tool names and associated parameters from the outputs produced by the large language model.
2. It seamlessly executes tool scripts based on these tool names and parameters.
3. The content returned by these tools is transformed into text. In the case of images, both the filename and storage path are recorded.
4. The interpreted results are then fed back to the large language model.

4.3. Baseline

Without instruction tuning the model according to the guidelines from Equation 3, simply submitting a query and expecting that mainstream models will autonomously decide and output whether, which, and how to utilize tools proves to be a formidable challenge. As a result, we turn to a more robust baseline methodology. When we input a query into the large language model, we also incorporate all the tool usage instructions from Table 2. In our prompt, we explicitly instruct the model first to ascertain if a tool is necessary, then to select the appropriate tool, and lastly to decide the correct invoking method based on the query and chosen tool. The prompt will be found in the appendix of the final version. We consider the following language model: Alpaca, Vicuna, Llama2Chat, GPT-3.5 and GPT-4. We consider the size of the 7B parameter for the open-source language model.

[12]https://huggingface.co/

[13]https://github.com/facebookresearch/MaskFormer

[14]https://github.com/lllyasviel/ControlNet

[15]https://github.com/zhilizju/reformatted-tool-data

Model	Whether	Which	How	Success
Alpaca	18.9	10.3	9.3	7.5
Vicuna	45.8	22.3	15.6	13.9
Llama2Chat	56.9	29.7	25.2	23.3
GPT-3.5	72.8	69.9	65.5	62.8
GPT-4	89.7	86.5	82.3	80.4
Ours	98.6	97.6	92.3	91.2

Table 3: Accuracy of 6 models across four critical aspects.

4.4. Human Evaluation on Reformatted Dataset

Before training model with the data, we conducted a human evaluation on a sampled subset of the reformatted data. It's worth noting that the majority of cases were found to be correct. Among the few cases that exhibited common patterns of inadequacy, the limitations of certain APIs were identified as a contributing factor. In such instances, the queries generated via self-instruction were not fully suitable for resolving the specific limitations of the selected API. For example, not all mathematical questions can be completely and accurately resolved by the Wolfram Alpha API, leading to API responses that do not genuinely address the issue. Importantly, the presence or absence of these types of data does not significantly affect the overall training process, as the model can still acquire the "whether, which, how" decision-making process from these cases.

4.5. Training

Using only 1,000 annotated data for NLP tasks in our dataset, it is challenging to make the language model itself good at handling NLP tasks. Therefore, we chose Alpaca, which already has the capability to process NLP tasks, as our base model to continue training on our dataset. Invoking the language model itself means that it directly generates an answer once it determines that it can solve the task rather than generating additional commands. We used the Alpaca training code [16] to fine-tune the base model on 8 RTX A6000 GPUs. We adopted a batch size of 128. The learning rate was set at 2e-5 and we trained the model for 3 epochs. The maximum sequence length is 512 and the weight decay is 0.

4.6. Model Evaluation

We employ accuracy as our primary evaluation metric, which includes four critical aspects.

[16]https://github.com/tatsu-lab/stanford_alpaca

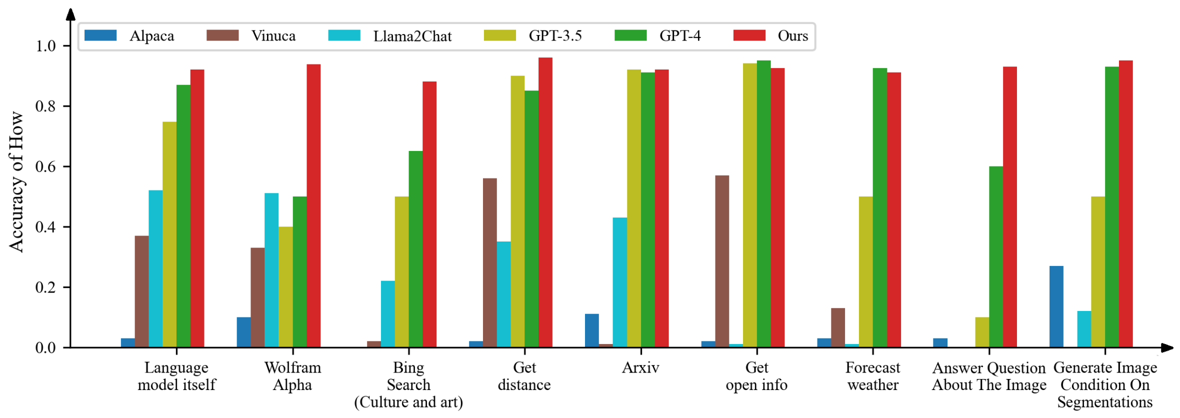


Figure 4: The performance of different models on various APIs.

- Whether: The ability to ascertain, based on a given query, whether a tool is necessary.
- When: If a tool is deemed necessary, the proficiency in selecting the appropriate one.
- How: After choosing the right tool, the capability to dictate the correct method for its invocation.
- Success: The accuracy of the results after invocation.

The initial three criteria gauge the prowess of the language model. The final metric evaluates the holistic success rate, considering not just the model but also the ability of integrated APIs. As an example, for intricate mathematical problems, the Wolfram Alpha API may not always yield accurate results. And the assessment method is human evaluation. All the experimental results in this paper are the statistical average of five crowd-sourced workers. It should be noted that when evaluating, we consider Equation 3. If the model makes a mistake at any intermediate step, even if the subsequent output is correct, we do not take it into account.

4.7. Results

Table 3 showcases the accuracy of six distinct models in four key aspects. The results underscore two pivotal insights: 1. Empowering large-scale language models to autonomously determine 'Whether', 'Which', or 'How' through in-context learning poses significant challenges, even for the robust GPT-4. Scrutiny of the data reveals a primary shortcoming of GPT models in discerning the necessity for tool usage, attributable mainly to two factors: (1) Hallucination, manifesting as overconfidence in responding to real-time and factual queries, among others; and (2) A disregard for provided candidate tools, deeming them unprocessable, especially in image-related queries. 2. Our methodology markedly exceeds the baseline, especially compared to the base model, Alpaca, highlighting the profound impact of our proposed end-to-end training on enhancing model performance significantly. Figure 4 further delineates the varying performance of these models in selected APIs. For a comprehensive view, the Appendix will illustrate the performance metrics of all APIs in a histogram format. The above result highlights the model's ability to generalize across varied instances within the same domain and API settings, tested against a standard dataset.

To present a rigorous analysis of model generalization, we consider two distinct but challenging scenarios. The first scenario pertains to the model's capacity to generalize across different domains while employing the same API. We chose the Bing Search API as a typical example. The training data covered four specific domains: Entertainment, Culture and Art, Politics, and Sports. The test was then expanded to an extra domain: Education, not present in the training data. This enables us to assess the model's generalization in an unknown domain with the same API. The success rate in trained domains is 92%, compared to 88% in Education. Despite a minor decline, the high rate persists, indicating the model's competent generalization across diverse domains.

The second scenario involves the model's generalization ability within the same type across different APIs. To assess this, we remove two multimodal APIs: 'Detect the Given Object' and 'Generate Image Based on Sketch Image'. It is evident that the baseline model is incapable of correctly producing the associated call methods of unseen API. Therefore, we gauge whether the model can accurately identify the given queries belongs to unsolvable problems. Tests showed a 95% average accuracy for two unseen APIs, demonstrating our model's effective knowledge generalization within the type.

4.8. Ablation

Method	How	Success
Ours	92.3	91.2
w/o Whether	78.7	75.4
w/o Which	66.4	62.0
w/o Whether+Which	55.6	52.9

Table 4: An in-depth ablation analysis of our approach.

Table 4 presents the results of our ablation study. Removing the modeling of 'Whether' and 'Which' from end-to-end learning, means that the training dataset lacks text information for 'Whether' and 'Which', and the model is tasked directly with predicting 'How', leading to a substantial decline in performance. This highlights the critical nature of each step in Equation 3 for the success of the end-to-end learning process. A unified architecture, together with a standardized template, significantly enhances the model's tool learning efficiency.

4.9. Continual Learning

We explored the model's scalable potential in continual tool learning. From Table 2, we selected three APIs—Get Distance, Generate Image Based On Sketch Image, and Arxiv—as new tools. The model was first trained using the training set of other "old" tools. For each new tool, we provided only 10 samples and set the total continual learning dataset to consist of 30 data samples. The ratio of new to old tools strictly followed the guidelines in section 3.4, with γ set at 0.5, over three training epochs. Surprisingly, Table 5 reveals that with very few annotated samples of new tools, the model can swiftly master new tools. We attribute this to a unified framework and standardized template. Additionally, the replay strategy ensures that the model maintains the effectiveness of previous APIs. Due to space constraints, an analysis of the impact of sample size and hyperparameters on the final results will be included in the appendix of the final version.

Strategy	New APIs	Old APIs
without Replay	91.6	84.7
with Replay	91.1	90.88

Table 5: The success rate under the scenario of continual learning.

4.10. Functionality Overlap

We explore API functionality overlap, highlighting differences between general APIs like Bing Search and specialized map APIs. Prioritizing map APIs for geographic responses is crucial. We have discovered that a meticulously curated balanced training dataset can accomplish this objective. For instance, Bing Search's ability to answer questions across diverse vertical domains is emphasized in the training dataset. This dataset is then crafted to include as many of these domains as possible, thereby reducing the model's dependency on Bing Search for geographical queries. Conversely, we keep queries for a specialized map API strictly geography-related. Our tests indicate that, with balanced training data for both, the model favors the map API for geography queries, employing it in nearly 98% of these cases.

5. Conclusion

In this paper, we explore full autonomy in tool usage within Large Language Models (LLMs). Our approach unifies different types of queries through the sequential decision-making framework, employs an efficient dataset construction method, and integrates new tools and APIs via a dynamic balanced rehearsal strategy. Experimental results demonstrate that our model has achieved significant success, marking a pivotal effort toward more autonomous AI systems.

Acknowledgements

This work was supported by the NSFC project (No. 62072399), the Zhejiang Provincial Natural Science Foundation of China under Grant No. LZ23F020009, MoE Engineering Research Center of Digital Library, China Research Centre on Data and Knowledge for Engineering Sciences and Technology, the CALT2022 project, and the Fundamental Research Funds for the Central Universities. We also express our sincere gratitude to anonymous reviewers for their invaluable feedback and constructive comments.

6. Bibliographical References

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-

Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. 2022. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR.

Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, et al. 2023. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on Robot Learning*, pages 287–318. PMLR.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer.

Zhipeng Chen, Kun Zhou, Beichen Zhang, Zheng Gong, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Chatcot: Tool-augmented chain-of-thought reasoning on\\chat-based large language models. *arXiv preprint arXiv:2305.14323*.

Bowen Cheng, Alex Schwing, and Alexander Kirillov. 2021a. Per-pixel classification is not all you need for semantic segmentation. *Advances in Neural Information Processing Systems*, 34:17864–17875.

Bowen Cheng, Alex Schwing, and Alexander Kirillov. 2021b. Per-Pixel Classification is Not All You Need for Semantic Segmentation. In *Advances in Neural Information Processing Systems*, volume 34, pages 17864–17875. Curran Associates, Inc.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2022. A survey for in-context learning. *arXiv preprint arXiv:2301.00234*.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. 2023. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2023. Critic: Large language models can self-correct with tool-interactive critiquing. *arXiv preprint arXiv:2305.11738*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.

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

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2022. Few-shot learning with retrieval augmented language models. *arXiv preprint arXiv:2208.03299*.

Angeliki Lazaridou, Elena Gribovskaya, Wojciech Stokowiec, and Nikolai Grigorev. 2022. Internet-augmented language models through few-shot prompting for open-domain question answering. *CoRR*, abs/2203.05115.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. 2022. BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2021. Webgpt: Browser-assisted question-answering with human feedback. *CoRR*, abs/2112.09332.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Bhargavi Paranjape, Scott M. Lundberg, Sameer Singh, Hannaneh Hajishirzi, Luke Zettlemoyer, and Marco Túlio Ribeiro. 2023. ART: automatic multi-step reasoning and tool-use for large language models. *CoRR*, abs/2303.09014.

Aaron Parisi, Yao Zhao, and Noah Fiedel. 2022. Talm: Tool augmented language models. *arXiv preprint arXiv:2205.12255*.

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*.

Shuofei Qiao, Honghao Gui, Huajun Chen, and Ningyu Zhang. 2023. Making language models better tool learners with execution feedback. *arXiv preprint arXiv:2305.13068*.

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*.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bjorn Ommer. 2022a. High-Resolution Image Synthesis with Latent Diffusion Models. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10674–10685, New Orleans, LA, USA. IEEE.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022b. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugginggpt: Solving ai tasks with chatgpt and its friends in huggingface. *arXiv preprint arXiv:2303.17580*.

Dídac Surís, Sachit Menon, and Carl Vondrick. 2023. Vipergpt: Visual inference via python execution for reasoning. *arXiv preprint arXiv:2303.08128*.

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

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language models with self-generated instructions. In *Annual Meeting of the Association for Computational Linguistics*.

Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. 2023. Visual chatgpt: Talking, drawing and editing with visual foundation models. *arXiv preprint arXiv:2303.04671*.

Rui Yang, Lin Song, Yanwei Li, Sijie Zhao, Yixiao Ge, Xiu Li, and Ying Shan. 2023. Gpt4tools: Teaching large language model to use tools via self-instruction. *arXiv preprint arXiv:2305.18752*.

Lvmin Zhang and Maneesh Agrawala. 2023. Adding Conditional Control to Text-to-Image Diffusion Models.