LLMs in the Imaginarium:
Tool Learning through Simulated Trial and Error

Boshi Wang♠[*] **Hao Fang**◇ **Jason Eisner**◇ **Benjamin Van Durme**◇ **Yu Su**◇
♠The Ohio State University ◇Microsoft Semantic Machines
wang.13930@osu.edu, {hao.fang,jason.eisner,bevandur,yusu2}@microsoft.com

Abstract

Tools are essential for large language models (LLMs) to acquire up-to-date information and take consequential actions in external environments. Existing work on tool-augmented LLMs primarily focuses on the broad coverage of tools and the flexibility of adding new tools. However, a critical aspect that has surprisingly been understudied is simply *how accurately an LLM uses tools for which it has been trained.* We find that existing LLMs, including GPT-4 and open-source LLMs specifically fine-tuned for tool use, only reach a correctness rate in the range of 30% to 60%, far from reliable use in practice. We propose a biologically inspired method for tool-augmented LLMs, simulated trial and error (STE), that orchestrates three key mechanisms for successful tool use behaviors in the biological system: trial and error, imagination, and memory. Specifically, STE leverages an LLM's 'imagination' to simulate plausible scenarios for using a tool, after which the LLM interacts with the tool to learn from its execution feedback. Both short-term and long-term memory are employed to improve the depth and breadth of the exploration, respectively. Comprehensive experiments on ToolBench show that STE substantially improves tool learning for LLMs under both in-context learning and fine-tuning settings, bringing a boost of 46.7% to Mistral-Instruct-7B and enabling it to outperform GPT-4. We also show effective continual learning of tools via a simple experience replay strategy.[1]

1 Introduction

Tools play an essential role in extending humans (Gibson et al., 1993) and other animals (Shumaker et al., 2011) beyond the confines of physical bodies to better perceive and exert impact on their environment. There is a recent surge of interest in augmenting large language models (LLMs) with tools to transcend the confines of their static parametric knowledge and text-in-text-out interface, empowering them to acquire up-to-date information, call upon external reasoners, and take consequential actions in external environments (Schick et al., 2023; Mialon et al., 2023; Qin et al., 2023).

Existing work on tool-augmented LLMs primarily aims to increase the ease of adding new tools, or the ability to access many tools (e.g., up to 16,000 APIs (Qin et al., 2024)). This is achieved through one of two common approaches: 1) In-context learning (ICL), which prompts frozen LLMs with API specification and tool use examples (i.e., instruction-API call pairs) (Lu et al., 2023; Song et al., 2023; Shen et al., 2023; Liang et al., 2023b), or 2) fine-tuning with tool use examples synthesized by LLMs (Schick et al., 2023; Patil et al., 2023; Qin et al., 2024; Tang et al., 2023). While coverage and flexibility are important for tool use, a critical aspect that, perhaps surprisingly, has been understudied is simply *how accurately an LLM uses tools for which it has been trained.* ICL is flexible but hard to drive to production-level accuracy. Fine-tuning can potentially lead to better accuracy by integrating a larger number of examples, but existing work mostly focuses on generalizing to unseen tools instead of optimizing an LLM's ability to use tools seen during training (Qin et al., 2024; Patil et al., 2023; Tang et al., 2023). Meanwhile, practical deployment of tool-augmented LLMs necessitates a high level of accuracy as they enable consequential actions, e.g., financial transactions or other legally binding operations. Inaccurate tool use could lead to undesired or harmful outcomes and quickly undermine user trust.

How to truly master a tool? We turn to successful precedents in the biological system such as humans (Gibson et al., 1993), apes and corvids (Emery and Clayton, 2004). Learning to

[*]Work done as an intern at Microsoft Semantic Machines.
[1]Code and data available at https://github.com/microsoft/simulated-trial-and-error.

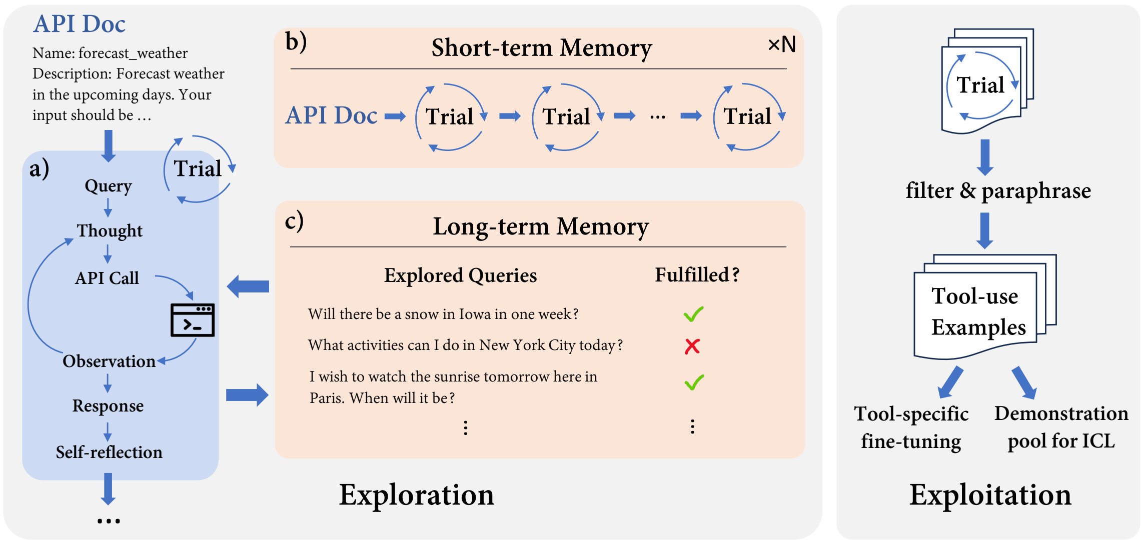

API Doc
Name: forecast_weather
Description: Forecast weather in the upcoming days. Your input should be ...
a)
Trial
Query
Thought
API Call
Observation
Response
Self-reflection
...
b)
Short-term Memory
×N
API Doc
Trial
Trial
...
Trial
c)
Long-term Memory
Explored Queries
Fulfilled?
Will there be a snow in Iowa in one week?
What activities can I do in New York City today?
I wish to watch the sunrise tomorrow here in Paris. When will it be?
Exploration
Trial
filter & paraphrase
Tool-use Examples
Tool-specific fine-tuning
Demonstration pool for ICL
Exploitation

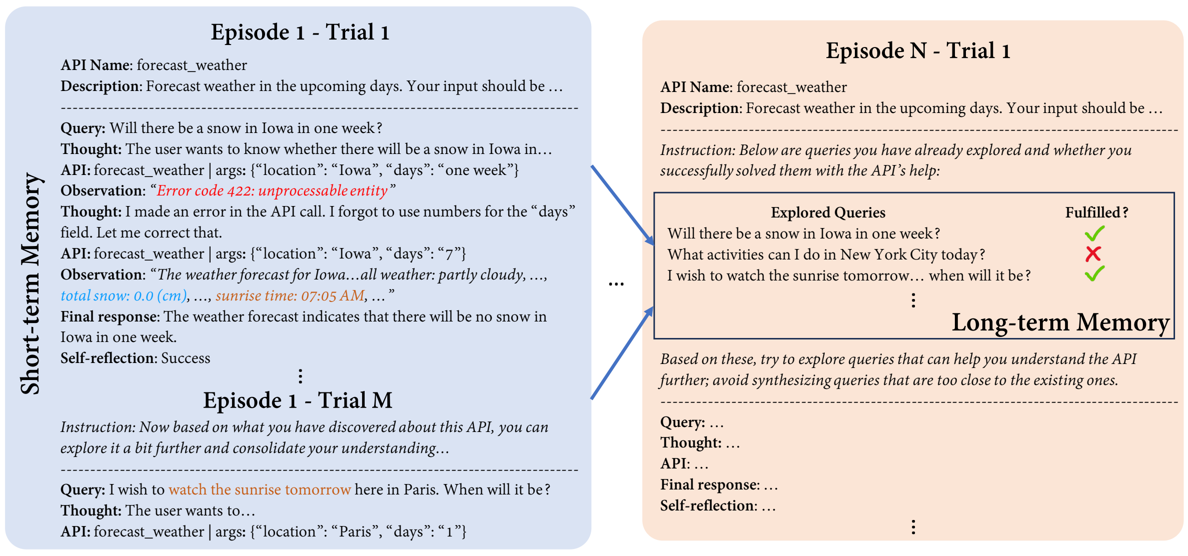
Figure 1: Illustration of simulated trial and error. In the exploration stage, an LLM interacts with the tool and progressively gathers tool-use experiences through trial and error. Specifically, a) in each trial, the LLM imagines plausible scenarios related to the target tool, iteratively interacts with the tool to fulfill the user query, and in the end self-reflects on the trial; b) a short-term memory consisting of recent trial trajectories encourages learning from fine-grained successes and failures and exploring the API in greater depth; c) a long-term memory of coarse-grained past trial and error experiences maintains progressive learning over a long time horizon. In the exploitation stage, the exploration experiences are distilled into a set of tool-use examples for either ICL or fine-tuning.

use a tool is a rather advanced cognitive function that depends on many other cognitive functions. First of all, *trial and error* is essential for tool learning (Beck, 1973; Auersperg et al., 2011). We do not master a tool solely by reading the 'user manual'; rather, we explore different ways of using the tool, observe the outcome, and learn from both successes and failures. Furthermore, intelligent animals do not just do random trial and error—we proactively *imagine* or *simulate* plausible scenarios that are not currently available to perception for exploration (Emery and Clayton, 2004; Redish, 2016). Finally, *memory*, both short-term and long-term, is instrumental for the progressive learning and recurrent use of tools (Vaesen, 2012; Emery and Clayton, 2004; Clayton and Dickinson, 1998).

To this end, we propose *simulated trial and error* (STE; illustrated in Figure 1), a biologically inspired method for tool-augmented LLMs. Given a tool (e.g., an API with its specification), STE leverages an LLM to simulate, or 'imagine', plausible scenarios (i.e., instructions) for using the tool. It then iteratively interacts with the API to fulfill the scenario by synthesizing, executing, and observing the feedback from API calls, and then reflects on the current trial (Shinn et al., 2023). We devise memory mechanisms to improve the quality of the simulated instructions. A short-term memory consisting of recent trial and error trajectories is employed to facilitate deeper exploration in a single episode, while a long-term memory containing distilled past exploration and reflections maintains progressive learning over a long horizon. In the exploitation stage, one can use the tool use examples from the explored trials to fine-tune an LLM, or simply do ICL by retrieving from those examples.

We conduct comprehensive experiments on APIs from ToolBench (Qin et al., 2024) and summarize the main findings as follows:

- Existing LLMs are far from reaching reliable tool use performance: GPT-4 (OpenAI, 2023) gets 60.8% correctness, and ToolLLaMA-v2 (Qin et al., 2024) that was specifically fine-tuned for tool use only gets 37.3%.
- STE proves to be remarkably effective for augmenting LLMs with tools, under both ICL and fine-tuning settings. STE improves the tool use capability of Mistral-Instruct-7B (Jiang et al., 2023) to 76.8% (a boost of 46.7% absolute), making it outperform GPT-4 with ICL.
- In practice, new tools are continually added. Fine-tuning (with or without STE) brings the challenge of catastrophic forgetting, where learning new tools may cause the LLM to lose its existing tool use capabilities or general language capabilities. We demonstrate that a simple experience replay strategy (Scialom et al., 2022) could largely mitigate the issue, allowing the



Episode 1 - Trial 1

API Name: forecast_weather
Description: Forecast weather in the upcoming days. Your input should be …

Query: Will there be a snow in Iowa in one week?
Thought: The user wants to know whether there will be a snow in Iowa in…
API: forecast_weather | args: {"location": "Iowa", "days": "one week"}
Observation: *"Error code 422: unprocessable entity"*
Thought: I made an error in the API call. I forgot to use numbers for the "days" field. Let me correct that.
API: forecast_weather | args: {"location": "Iowa", "days": "7"}
Observation: *"The weather forecast for Iowa…all weather: partly cloudy, …, total snow: 0.0 (cm), …, sunrise time: 07:05 AM, …"*
Final response: The weather forecast indicates that there will be no snow in Iowa in one week.
Self-reflection: Success

⋮

Episode 1 - Trial M

Instruction: Now based on what you have discovered about this API, you can explore it a bit further and consolidate your understanding…

Query: I wish to watch the sunrise tomorrow here in Paris. When will it be?
Thought: The user wants to…
API: forecast_weather | args: {"location": "Paris", "days": "1"}

…

Episode N - Trial 1

API Name: forecast_weather
Description: Forecast weather in the upcoming days. Your input should be …

Instruction: Below are queries you have already explored and whether you successfully solved them with the API's help:

Explored Queries	Fulfilled?
Will there be a snow in Iowa in one week?	✓
What activities can I do in New York City today?	✗
I wish to watch the sunrise tomorrow… when will it be?	✓
⋮	

Based on these, try to explore queries that can help you understand the API further; avoid synthesizing queries that are too close to the existing ones.

Query: …
Thought: …
API: …
Final response: …
Self-reflection: …

⋮

Figure 2: Exploration with simulated trial and error, highlighting the memory mechanisms. Each episode begins with the API specification (only in the first trial), followed by a series of trials dynamically added in the short-term memory. The long-term memory is loaded into the context at the beginning of every trial to allow the LLM to progressively imagine novel scenarios, and then offloaded afterward (omitted in the figure).

model to continually learn new tools while preserving its previously acquired skills.

2 Simulated Trial and Error

We introduce our proposed simulated trial and error (STE) for tool learning. STE consists of an exploration phase and an exploitation phase, which are discussed next.

2.1 Exploration

In the exploration phase, for each new API, the LLM interacts with the API within a budget in order to gain as much information as possible about the API. The exploration phase consists of a series of trials (Figure 1) resembling humans' progressive learning of a tool. In each trial, conditioned on the API description, the LLM 1) imagines a plausible user query relevant to the API; 2) tries to fulfill the query by interacting with the API; 3) reflects on the trial to facilitate subsequent exploration. Three core design components are integrated with the trials to enhance the validity, comprehensiveness and diversity of the exploration, introduced next.

Iterative self-refine with execution feedback. To improve the validity of the exploration, we use a strategy similar to the ideas of Chen et al. (2024); Qin et al. (2024); Madaan et al. (2023); Shinn et al. (2023) where the LLM learns from the execution feedback to refine its own outputs (Figure 2, top left). Specifically, we adopt the ReAct format (Yao et al., 2023) where during each step, the LLM first verbalizes its internal thought, then makes an action (API call) and observes the corresponding execution feedback, and then repeats the thought→action→observation process until the model decides that the API call has returned adequate information or a predefined maximum number of calls. During this stage, the LLM learns from the execution environment to correct its own syntactic and semantic errors in API calls, gathering tool-use experiences as fine-grained trial-and-error trajectories. Afterward, the model responds to the user's query and self-reflects on whether the explored query is successfully fulfilled or not.

Short-term memory. A direct implementation of the exploration where each trial is conducted in a separate episode only allows shallow explorations of the API. We augment the LLM with a short-term memory consisting of the exploration trajectories of recent trials, where the LLM is instructed to conduct subsequent trials conditioned on the memory (Figure 2, left). Each episode starts with a fresh short-term memory, where newly conducted trials are dynamically added into the memory for a certain number of trials. This allows the model to learn from recent fine-grained successes and failures (e.g., syntax/semantic errors), and also explore the API in greater depth in the coming trials based on its previous observations of the API (e.g., unexpected functionalities).

Long-term memory. Only a small number of trials can be stored in short-term memory since the fine-grained trajectories quickly consume the LLM's context capacity. We augment the LLM with a long-term memory that stores distilled trial-and-error experiences from past episodes, in order to support progressive learning over a long time horizon. Specifically, the long-term memory records the past-explored queries and whether they were judged as successfully fulfilled (Figure 2, right). It is only loaded into the context at the beginning of every new trial, where the model is instructed to imagine scenarios that are distant from previously explored ones to improve information gain. In this way, the long-term memory serves as a growing pool of past successes and failures, which allows the LLM to continually expand the exploration in order to make progress across different episodes.

2.2 Exploitation

In the exploitation stage, the trials obtained from the exploration stage are utilized to enhance the tool-use ability of an LLM via either fine-tuning or in-context learning (ICL). For each trial, we extract the synthesized user query, the LLM's last API call and its execution results, and the final response from the trial trajectory. Then, we perform filtering by using GPT-4 to judge the validity of each example, and then paraphrase the valid examples for each new API into approximately the same amount (Appendix E), which maintains a balance across different APIs and further adds linguistic variations into the synthesized tool-use examples.

For fine-tuning, we use the standard language modeling objective where the loss is computed only for the tool-use/response generation part, and do not include the API documentation in the context. For ICL, the synthesized examples are used as the demonstration pool from which in-context examples are retrieved and appended to the API documentations in the LLM's context. We use a dynamic nearest-neighbor demonstration selection strategy where the examples that are semantically closest to the test user query are retrieved as in-context examples, one of the top performing strategies for ICL (Liu et al., 2022; Rubin et al., 2022).

3 Experimental Setup

Tools. We conduct experiments using APIs from ToolBench (Qin et al., 2024), a large-scale repository of real-world APIs collected from RapidAPI and BMTools. We filter down to the APIs that are free to use with low execution latency. In the end, we obtain 50 APIs that span search engines (e.g., Google Search & Places), domain-specific information-seeking APIs (e.g., Wikipedia, Weather, Sports, Gaming), and also problem-solving ones such as WolframAlpha, Number Translator, etc. More details are in Appendix A.

Setup for exploration. In the exploration stage, we use ChatGPT (`16k-0613`) for exploration and paraphrasing, and GPT-4 (`8k-0613`) (OpenAI, 2023) for final example filtering. We set the maximum number of API calls for each trial to be 4. For each API, the exploration stage lasts for 15 episodes with 4 trials per episode, resulting in a total of 60 examples before filtering and paraphrasing. After filtering, 15 examples for each API are randomly selected into the test set, where the remaining ones are paraphrased into $\sim$140 examples, making a total of $\sim$7K tool-use examples. For the test examples, we manually examine and correct any issues, if any, to ensure test set quality.

Baselines & exploitation with STE. We experiment with **Llama-2-Chat-7B/13B** (Touvron et al., 2023), **Mistral-Instruct-7B** (Jiang et al., 2023), and **GPT-3.5-turbo/GPT-4** (ICL only) and compare their performance with and without STE. We compare with **ToolLLaMA-v2** (Qin et al., 2024) as the main baseline for existing tool learning strategies. It is based on Llama-2-7B and fine-tuned on 126K tool use examples synthesized by ChatGPT-3.5-turbo for general tool-use, covering a large number of tools from RapidAPI including the ones used in our experiments.

For ICL with nearest neighbor demonstration selection, following prior work (Liu et al., 2022; Rubin et al., 2022), we use the `paraphrase-mpnet-base-v2` model from SentenceBERT (Reimers and Gurevych, 2019) for computing the semantic similarity, and choose the top 8 examples closest to the test query as in-context demonstrations. For Llama-2 with ICL, since the token length of the full 50 API documentations (around 7K tokens) is beyond its context length (4,096),[2] we augment the model with an oracle tool retriever which retrieves the top 15 similar APIs *w.r.t.* the ground truth API using the associated documentation. We augment other models of similar scales (7B/13B) with the same

[2]While there exist variants of Llama-2 with longer context (e.g., Xiong et al. (2023)), we stick to the original model in Touvron et al. (2023) for fair comparison.

Setting	Base Model	Wellformed?	API Match	Correctness
Baseline	ToolLLaMA-v2	<u>98.1</u>	49.0	37.3
	Llama-2-Chat-7B	34.5	40.2	10.7
	Llama-2-Chat-13B	79.3	53.6	32.7
	Mistral-Instruct-7B	61.7	69.3	30.1
	GPT-3.5-turbo (16k-0613)	96.9	77.6	60.5
	GPT-4 (8k-0613)	96.1	<u>78.1</u>	<u>60.8</u>
ICL w/ STE	Llama-2-Chat-7B	58.3	86.7	41.7
	Llama-2-Chat-13B	87.5	86.6	62.9
	Mistral-Instruct-7B	69.9	88.4	47.9
	GPT-3.5-turbo (16k-0613)	97.6	90.8	75.6
	GPT-4 (8k-0613)	<u>97.7</u>	<u>92.8</u>	<u>76.3</u>
Fine-tuning w/ STE	Llama-2-Chat-7B	**<u>99.2</u>**	94.9	73.3
	Llama-2-Chat-13B	98.9	95.1	74.3
	Mistral-Instruct-7B	99.1	**<u>95.8</u>**	**<u>76.8</u>**

Table 1: Overall tool-use performance. STE is effective when used in both ICL and fine-tuning. Best overall results are **bold-faced**, and best results under each setting are <u>underscored</u>.

Setting	Wellformed?	API Match	Correctness
Full STE	99.2	94.9	73.3
– Exec. feedback	89.9	79.4	50.5
– Short. Mem.	99.7	70.6	53.9
– Long. Mem.	98.7	79.9	59.7
– Self-reflection	99.3	81.7	60.1

Table 2: Results for ablations. We separately ablate each key component of our exploration design. Exploitation is done by fine-tuning Llama-2-Chat-7B.

tool retriever (when ICL is used for exploitation) for fair comparison. LLMs fine-tuned on STE do not need such API documentation in the context, which substantially reduces the inference cost.

Evaluation metrics. We evaluate the model by matching the predicted API call against the ground truth. For APIs that have strict value ranges for the arguments, we perform string matching on the respective arguments directly. For APIs that accept free natural language inputs, we use ChatGPT to judge the correctness of the predicted API arguments. We report the overall accuracy considering both API name and arguments (**Correctness**) as the main metric, together with the percentage of examples with valid API calls and no syntax/formatting errors (**Wellformedness**) and the percentage of examples that correctly choose to use the ground-truth API (**API Match**). While it is desirable to also evaluate the model regarding whether the model resolves the user query successfully based on the execution results, the majority of the APIs in our experiments are connected to dynamic real-world information (e.g., weather 'tomorrow' where the date is contingent on the actual time of making the API call), which makes such evaluation infeasible. We leave this challenge to future research.

4 Results

4.1 Effectiveness of STE

Results are included in Table 1. We summarize the main findings below.

None of the baseline models displays satisfactory performance. For all of the baseline models that we tested, none of them achieves a satisfactory tool-use performance. The best model is GPT-4, which only achieves an overall correctness rate of 60.8%. Llama-2 and Mistral achieve a much lower performance, largely due to the model not being able to follow the specified syntactic/formatting requirements when making API calls.[3] ToolLLaMA-v2 (Qin et al., 2024), despite extensively fine-tuned for tool use, still largely underperforms GPT-3.5/4. Its performance improvement over non-fine-tuned baselines like Llama-2 seems to mainly come from wellformedness, and it still faces severe difficulties in choosing the correct tool and predicting the right arguments. This suggests that fine-tuning for general tool use is insufficient for achieving the level of performance needed for practical deployment.

STE is effective with both ICL and fine-tuning. Remarkable gains are observed under both settings. When retrieving from the tool use examples generated by STE for ICL, we see improvements across the board, with up to 30.2% (for Llama-2-Chat-13B) in correctness for open-source LLMs. It also boosts the already strong performance of GPT-3.5/4 substantially. Fine-tuning with STE examples improves the tool use capability of open-source

[3]The superior ability of GPT-3.5/GPT-4 to follow the syntax may be partially due to their special enhancement on function-calling (https://openai.com/blog/function-calling-and-other-api-updates).

LLMs by an even larger margin, boosting Mistral-Instruct-7B by 46.7% in correctness and enabling it to outperform GPT-4. Fine-tuning with STE also makes LLMs almost perfect in wellformedness and choosing the right tools. This is likely because fine-tuning allows injecting a much wider range of tool use examples into a model than ICL. While we cannot fine-tune GPT-3.5/4 due to cost and availability, it is plausible to hypothesize that STE could further improve their tool-use ability beyond their current ICL performance.

4.2 Ablation Studies

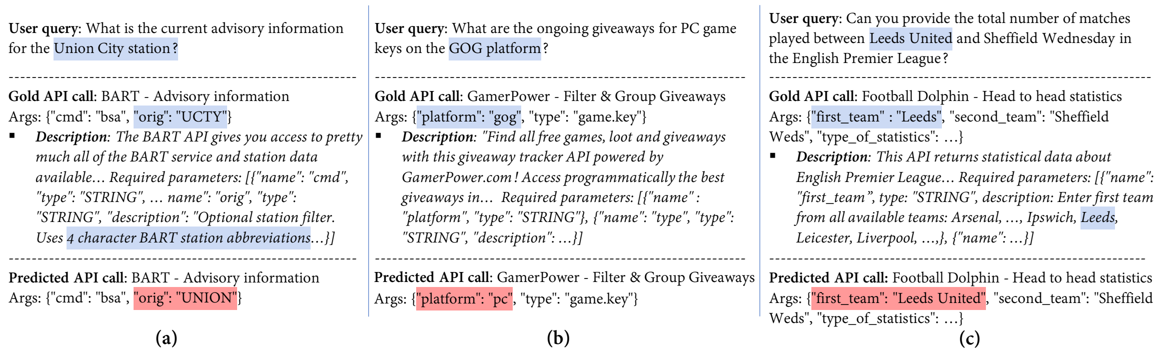
We conduct an ablation study for our exploration design, with exploitation done by fine-tuning Llama-2-Chat-7B. Specifically, we ablate the execution feedback, short/long-term memory, and the self-reflection component. We extend the number of episodes to preserve the total number of trials if needed. The results in Table 2 show that 1) exploration without execution feedback could give a notable amount of ill-formed examples where the API calls do not follow the syntax/formatting requirements; 2) both short-term and long-term memory prove to be essential for effective trial and error; 3) self-reflection is important in maintaining an informative long-term memory for exploration. To better understand the benefits of our memory design, we conduct a case study with the `forecast_weather` API (examples in Appendix C), which clearly shows that both of the memory mechanisms substantially improve the diversity and comprehensiveness of exploration:

- **Short-term memory boosts specificity and comprehensiveness.** Comparing the trials with/without short-term memory, it can be seen that the short-term memory effectively drives the LLM to comprehensively explore fine-grained information from the tool, spanning 16 different attributes (e.g., humidity, precipitation, UV index, visibility, and sunrise/sunset time) in total. Meanwhile, when short-term memory is disabled, the examples are much less specific and mostly about general weather conditions (e.g., *"What will be the weather like..."*) due to the inability to leverage newly obtained information from the execution results. In addition, exploration without short-term memory results in a significantly lower percentage of positive tool-use examples (78.3% $\rightarrow$ 51.7%), since the model cannot learn from fine-grained past errors to facilitate future trials. As an example, the model synthesizes a considerable amount of queries where the time is specified as the day of the week, which is not a supported parameter type of the API and hence constantly results in failures.
- **Long-term memory improves overall diversity over a long time horizon.** With long-term memory, the LLM explores examples covering a broader range of subjects, and maintains the progress over different episodes. When long-term memory is disabled, the trials across episodes become repetitive and less informative. For quantitative characterization, we extract the core subjects (location, time, attribute) from the queries, measure their diversity and also plot the attribute distribution (Appendix C). With long-term memory, all the queries are distinct and the trials are balanced across different attributes. Without long-term memory, only 71.7% of the trials concern distinct subjects and the distribution across attributes is much more skewed, showing the effectiveness of long-term memory in maintaining the diversity of exploration over a long time horizon.

4.3 Error Analysis

Errors of GPT-4. As one of the most capable LLMs, GPT-4 (`8k-0613`) can only achieve an overall correctness of 60.8%. We conduct an error analysis of GPT-4. We randomly sample and examine 30 error examples of GPT-4, which can be categorized into the following three types, with the corresponding percentage without $\rightarrow$ with ICL with STE examples. Examples for each category are shown in Appendix D.

- **Wrong choice of API (36.7%$\rightarrow$19.0%).** GPT-4 calls the wrong API that cannot address the user query. Table 4 shows one example where the user query is regarding parks with hiking trails in San Francisco. Here the model calls an API that retrieves the geographic coordinates of San Francisco, overlooking the ground truth "Places" API. ICL with STE examples helps resolve about half of such errors by better illustrating the fine-grained semantics of the APIs.
- **Missing/wrong arguments (26.7%$\rightarrow$10.0%).** Here, GPT-4 fails to provide the correct set of arguments despite choosing the right tool. Table 5 shows an example where the model fails to provide the required "lang" keyword. STE is particularly effective for such errors.
- **Hard-to-evaluate examples (36.7%$\rightarrow$16.7%).** We found that it is difficult to judge the correct-



User query: What is the current advisory information for the Union City station?

Gold API call: BART - Advisory information
Args: {"cmd": "bsa", "orig": "UCTY"}

- ***Description***: *The BART API gives you access to pretty much all of the BART service and station data available… Required parameters: [{"name": "cmd", "type": "STRING", … name": "orig", "type": "STRING", "description": "Optional station filter. Uses 4 character BART station abbreviations…}]*

Predicted API call: BART - Advisory information
Args: {"cmd": "bsa", "orig": "UNION"}

(a)

User query: What are the ongoing giveaways for PC game keys on the GOG platform?

Gold API call: GamerPower - Filter & Group Giveaways
Args: {"platform": "gog", "type": "game.key"}

- ***Description***: *"Find all free games, loot and giveaways with this giveaway tracker API powered by GamerPower.com! Access programmatically the best giveaways in… Required parameters: [{"name" : "platform", "type": "STRING"}, {"name": "type", "type": "STRING", "description": …}]*

Predicted API call: GamerPower - Filter & Group Giveaways
Args: {"platform": "pc", "type": "game.key"}

(b)

User query: Can you provide the total number of matches played between Leeds United and Sheffield Wednesday in the English Premier League?

Gold API call: Football Dolphin - Head to head statistics
Args: {"first_team" : "Leeds", "second_team": "Sheffield Weds", "type_of_statistics": …}

- ***Description***: *This API returns statistical data about English Premier League… Required parameters: [{"name": "first_team", type: "STRING", description: Enter first team from all available teams: Arsenal, …, Ipswich, Leeds, Leicester, Liverpool, …,}, {"name": …}]*

Predicted API call: Football Dolphin - Head to head statistics
Args: {"first_team": "Leeds United", "second_team": "Sheffield Weds", "type_of_statistics": …}

(c)

Figure 3: Error examples of Mistral-Instruct-7B after fine-tuning: (a) commonsense/world knowledge, (b) language understanding, and (c) grounding.

	Batch 1	**Batch 2**	**Batch 3**	**Batch 4**	**All APIs**	**MMLU**	**BBH**
Llama-Flan	-	-	-	-	-	37.2	39.5
CL-Round 1	80.6	-	-	-	-	39.6	36.8
CL-Round 2	$1.7 \rightarrow 76.1$	$87.7 \rightarrow 84.1$	-	-	-	40.2	38.9
CL-Round 3	$0.0 \rightarrow 70.6$	$56.9 \rightarrow 84.1$	$68.9 \rightarrow 65.6$	-	-	39.2	37.5
CL-Round 4	$0.0 \rightarrow 65.0$	$38.5 \rightarrow 88.7$	$25.0 \rightarrow 66.1$	$71.8 \rightarrow 70.3$	$34.7 \rightarrow 72.8$	38.5	39.1
Llama-FT	73.3	87.2	68.3	67.2	74.1	38.7	40.8

Table 3: Results for continual learning. Llama-Flan is the base LLM and Llama-FT is Llama-Flan fine-tuned on all the tools at once. For CL, the tools are split into four batches and the LLM needs to continually learn a new batch in each round. Scores to the left/right of each arrow ("$\rightarrow$") are the tool-use correctness without/with rehearsal. For example, 1.7$\rightarrow$76.1 means the fine-tuned model gets only 1.7% on Batch 1 tools after CL-Round 2 without rehersal, and 76.1% with rehearsal. While vanilla fine-tuning causes catastrophic forgetting, rehearsal could largely mitigate this issue and allow the model to continually learn new tools while preserving its previously acquired skills.

ness of the model predictions for around one-third of the error examples (an example included in Table 6). The main reasons behind this are 1) the existence of tools with overlapping functionalities that makes ground truth non-unique and 2) the time-sensitive nature of certain tools that prohibits consistent ground truths. Such difficulties in evaluating tool use are also noted in existing work (Qin et al., 2024; Patil et al., 2023), which is an open challenge for future work.

Errors after fine-tuning. We also examine the errors of the most performant fine-tuned model (Mistral-Instruct-7B) and summarize the notable error causes compared with GPT-4, which shed light on venues for future improvement.

- **Commonsense/world knowledge (47.4%).** Many tools require commonsense/world knowledge. Figure 3(a) shows an example where calling the API requires knowing the 4-character abbreviation of the target transit station, and here the model hallucinates a wrong abbreviation. This issue could be mitigated by scaling or additional knowledge retrieval.
- **Language understanding (31.6%).** Certain errors are caused by a lack of basic language understanding abilities. Figure 3(b) shows one example where the model misunderstands the user query which results in wrong arguments. Using a stronger base LLM could mitigate such errors.
- **Grounding (21.1%).** We find that some errors are due to a lack of grounding, where the LLM generates API calls that are semantically correct but not grounded to the API constraints. One example is given in Figure 3(c), where the model correctly extracts the target entity but fails to link it to the entity names supported by the API. This could be improved by incorporating constraints during decoding (Zhang et al., 2023; Shin et al., 2021; Fang et al., 2023) or using fuzzy-matching mechanisms.

4.4 Continual Tool Learning

While fine-tuning significantly outperforms ICL for tool use, one downside is the potential decrease of *flexibility* as discussed in §1 due to catastrophic forgetting (Kirkpatrick et al., 2017; Howard and Ruder, 2018; Kumar et al., 2022; Luo et al., 2023). Since retraining the model from scratch is costly and hurts flexibility, we explore continual learning (CL) and show that simple rehearsal (Scialom et al., 2022) seems to be sufficient for continual tool learning with STE.

We randomly split the tools into 4 consecutive batches to simulate the continual setting. For rehearsal, during each round, we add 10% tool use examples for each API from previous batches into the replay buffer. For preserving general non-tool-use capabilities, we also add in every training round 2,000 random examples from Flan-V2 (Longpre et al., 2023; Chung et al., 2022), one of the highest quality general instruction datasets (Wang et al., 2023a), and evaluate the model on MMLU (Hendrycks et al., 2021) and Big-Bench-Hard (BBH) (Suzgun et al., 2023). We use Llama-Flan as the base model to ensure a fair comparison of general capabilities on MMLU and BBH (more details in Appendix B). Results in Table 3 show that the model could drastically forget previously learned tools without rehearsal, with more distant ones being more severely forgotten. Rehearsal largely mitigates forgetting—the CL-trained model achieves comparable performance as Llama-FT. General language abilities are also retained as measured on MMLU and BBH. Overall, we extend the findings of Scialom et al. (2022) on the effectiveness of experience replay to the new realm of LLM tool learning, demonstrating a feasible way of flexibly adding new tools with the proposed STE method.

5 Related Work

Tool-augmented language models. One of the focuses of extensive research in NLP is on augmenting models with retrieval/search engines that could supplement extra knowledge (Guu et al., 2020; Lewis et al., 2020; Izacard et al., 2022; Borgeaud et al., 2022, *inter alia*). Recently, there has been a trend towards augmenting LLMs with more diverse types of tools, such as program executors, translation and QA models (Chen et al., 2023b; Gao et al., 2023; Parisi et al., 2022; Schick et al., 2023), APIs from developers and public repositories (Patil et al., 2023; Qin et al., 2024; Xie et al., 2023), and tools curated for specific environments (Gu et al., 2024) to further expand the scope of problems that LLMs can assist with.

Both fine-tuning and ICL are used to adapt an LLM to use tools. Fine-tuning-based approaches train the LLM to use tools on a set of tool-specific demonstration examples (Schick et al., 2023; Parisi et al., 2022), while ICL-based approaches (Lu et al., 2023; Song et al., 2023; Shen et al., 2023; Liang et al., 2023b; Gao et al., 2023) directly put the tool descriptions and optionally (a small amount of) tool-use demonstrations in the context. Hao et al. (2023) propose a lightweight adaptation method that expands the LLM's vocabulary with trained tool embeddings. Qin et al. (2024); Patil et al. (2023); Tang et al. (2023) explore training models to better leverage API descriptions for tool use. Our work aims to develop a framework that allows equipping LLMs with stronger tool-use abilities, motivated by how humans typically learn tools through continual trial and error.

LLMs can learn from feedback. Recent work found that LLMs are capable of improving/correcting their predictions with feedback (Shinn et al., 2023; Madaan et al., 2023; Ganguli et al., 2023; Chen et al., 2024; Peng et al., 2023; Kim et al., 2023; Pan et al., 2023). Our work is built on top of these findings and uses an LLM to progressively learn tools by leveraging feedback from the tool execution and the LLM's self-reflection.

Data synthesis & bootstrapping with LLMs. Due to LLMs' exposure to broad domains during pretraining and their rapidly improving generation abilities, recent work has explored using LLMs for dataset synthesis, which alleviates the burden of costly human annotations (Schick and Schütze, 2021; Wang et al., 2023b; Honovich et al., 2023; Li et al., 2023; Zelikman et al., 2022; Huang et al., 2023). Such model-synthesized data can then be utilized to improve models including themselves. In the tool-learning domain, similar ideas have been explored for tool-specific data synthesis (Schick et al., 2023; Patil et al., 2023; Qin et al., 2024). Our approach follows this line of work and takes a step towards better comprehensiveness and diversity of the synthesized tool-use examples.

Augmenting models with dynamic memory. Using memory mechanisms to allow models to dynamically gather and utilize experiences is an old idea, e.g., Riesbeck (1981); Schank (1983). Recent work also explores augmenting models with a growing memory of user and environment feedback (Madaan et al., 2022; Shinn et al., 2023; Zhong et al., 2023; Liang et al., 2023a; Zhao et al., 2023; Modarressi et al., 2023; Hu et al., 2023). We draw inspiration from these works and augment the LLM with fine-grained short-term memory and distilled long-term memory to enhance the LLM's progressive learning of tools.

6 Conclusions

Motivated by how humans master tools through continual interaction and reinforcement, we propose simulated trial and error, an LLM tool-learning method built upon progressive memory-based trial and error. Experiments on APIs drawn from ToolBench show the effectiveness of the proposed method, and also that rehearsal-based fine-tuning could enable continual learning of new tools with preserved previous skills.

Limitations

Iterative improvement. Currently, we use strong models for exploration and smaller weak models for exploitation. The exploration-exploitation could also be done iteratively as in prior work (Aksitov et al., 2023; Zelikman et al., 2022), where the reliance on the strong models could be diminished gradually (e.g., only as evaluators) as the capabilities of the models being enhanced improve.

Compositional tool use & planning. Another important ability in the context of tool use is composing/planning multiple tool calls to fulfill complex queries, which goes in an orthogonal direction as our focus here. Recent works show that the core abilities of LLMs are encoded and elicited from pretraining instead of injected through fine-tuning/alignment (Zhou et al., 2023; Lin et al., 2023), which suggests that extensive data preparation may not be required to adapt LLMs for complex tool use, different from our focus where extensive learning and exploration are always desired as information is gained from the tool side.

Larger memory capacity beyond context limit. The capacity of the augmented memory is limited by the context length of the LLM. There are different kinds of approaches that could be used to further scale up the memory, such as using additional retrieval modules (Wang and Li, 2023) or having more hierarchical/compressed representations of the memory (Chen et al., 2023a).

Tool unlearning? While we explored continual learning of new tools, the problem of unlearning is also important as tools could get constantly unloaded/outdated. Knowledge unlearning is generally a challenging problem (Si et al., 2023), and there could be specific designs that support easier tool unlearning, such as ToolkenGPT (Hao et al., 2023) which allows plug-and-play adaptation while enabling learning with large-scale examples.

Limitations of example-based fine-tuning. Finally, there are also inherent limitations of example-based methods for tool learning, in particular, the difficulty of teaching the model when *not* to use a tool through positive tool-use examples alone. Some potential ways of improving this issue are incorporating negative examples (e.g., using contrastive objectives) or carrying such parts of the API alongside example-based training. We leave these investigations to future work.

References

Renat Aksitov, Sobhan Miryoosefi, Zonglin Li, Daliang Li, Sheila Babayan, Kavya Kopparapu, Zachary Fisher, Ruiqi Guo, Sushant Prakash, Pranesh Srinivasan, Manzil Zaheer, Felix Yu, and Sanjiv Kumar. 2023. Rest meets react: Self-improvement for multi-step reasoning llm agent.

Alice MI Auersperg, Auguste MP Von Bayern, Gyula K Gajdon, Ludwig Huber, and Alex Kacelnik. 2011. Flexibility in problem solving and tool use of kea and new caledonian crows in a multi access box paradigm. *PloS one*, 6(6):e20231.

Benjamin B Beck. 1973. Observation learning of tool use by captive guinea baboons (papio papio). *American Journal of Physical Anthropology*, 38(2):579–582.

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.

Howard Chen, Ramakanth Pasunuru, Jason Weston, and Asli Celikyilmaz. 2023a. Walking down the memory maze: Beyond context limit through interactive reading.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023b. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2024. Teaching large language models to self-debug. In *The Twelfth International Conference on Learning Representations*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models.

Nicola S Clayton and Anthony Dickinson. 1998. Episodic-like memory during cache recovery by scrub jays. *Nature*, 395(6699):272–274.

Nathan J Emery and Nicola S Clayton. 2004. The mentality of crows: convergent evolution of intelligence in corvids and apes. *science*, 306(5703):1903–1907.

Hao Fang, Anusha Balakrishnan, Harsh Jhamtani, John Bufe, Jean Crawford, Jayant Krishnamurthy, Adam Pauls, Jason Eisner, Jacob Andreas, and Dan Klein. 2023. The whole truth and nothing but the truth: Faithful and controllable dialogue response generation with dataflow transduction and constrained decoding. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5682–5700, Toronto, Canada. Association for Computational Linguistics.

Deep Ganguli, Amanda Askell, Nicholas Schiefer, Thomas Liao, Kamilė Lukošiūtė, Anna Chen, Anna Goldie, Azalia Mirhoseini, Catherine Olsson, Danny Hernandez, et al. 2023. The capacity for moral self-correction in large language models. *arXiv preprint arXiv:2302.07459*.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR.

Kathleen R Gibson, Kathleen Rita Gibson, and Tim Ingold. 1993. *Tools, language and cognition in human evolution*. Cambridge University Press.

Yu Gu, Yiheng Shu, Hao Yu, Xiao Liu, Yuxiao Dong, Jie Tang, Jayanth Srinivasa, Hugo Latapie, and Yu Su. 2024. Middleware for llms: Tools are instrumental for language agents in complex environments. *arXiv preprint arXiv:2402.14672*.

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.

Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. 2023. ToolkenGPT: Augmenting frozen language models with massive tools via tool embeddings. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Or Honovich, Thomas Scialom, Omer Levy, and Timo Schick. 2023. Unnatural instructions: Tuning language models with (almost) no human labor. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14409–14428, Toronto, Canada. Association for Computational Linguistics.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Chenxu Hu, Jie Fu, Chenzhuang Du, Simian Luo, Junbo Zhao, and Hang Zhao. 2023. Chatdb: Augmenting llms with databases as their symbolic memory.

Jiaxin Huang, Shixiang Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2023. Large language models can self-improve. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1051–1068, Singapore. Association for Computational Linguistics.

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

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b.

Geunwoo Kim, Pierre Baldi, and Stephen McAleer. 2023. Language models can solve computer tasks. *arXiv preprint arXiv:2303.17491*.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Ananya Kumar, Aditi Raghunathan, Robbie Matthew Jones, Tengyu Ma, and Percy Liang. 2022. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *International Conference on Learning Representations*.

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.

Xian Li, Ping Yu, Chunting Zhou, Timo Schick, Luke Zettlemoyer, Omer Levy, Jason Weston, and Mike Lewis. 2023. Self-alignment with instruction backtranslation.

Xinnian Liang, Bing Wang, Hui Huang, Shuangzhi Wu, Peihao Wu, Lu Lu, Zejun Ma, and Zhoujun Li. 2023a. Unleashing infinite-length input capacity for large-scale language models with self-controlled memory system.

Yaobo Liang, Chenfei Wu, Ting Song, Wenshan Wu, Yan Xia, Yu Liu, Yang Ou, Shuai Lu, Lei Ji, Shaoguang Mao, et al. 2023b. Taskmatrix. ai: Completing tasks by connecting foundation models with millions of apis. *arXiv preprint arXiv:2303.16434*.

Bill Yuchen Lin, Abhilasha Ravichander, Ximing Lu, Nouha Dziri, Melanie Sclar, Khyathi Chandu, Chandra Bhagavatula, and Yejin Choi. 2023. The unlocking spell on base llms: Rethinking alignment via in-context learning.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, Dublin, Ireland and Online. Association for Computational Linguistics.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V. Le, Barret Zoph, Jason Wei, and Adam Roberts. 2023. The flan collection: Designing data and methods for effective instruction tuning.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. 2023. Chameleon: Plug-and-play compositional reasoning with large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. 2023. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *arXiv preprint arXiv:2308.08747*.

Aman Madaan, Niket Tandon, Peter Clark, and Yiming Yang. 2022. Memory-assisted prompt editing to improve GPT-3 after deployment. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2833–2861, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Grégoire Mialon, Roberto Dessi, Maria Lomeli, Christoforos Nalmpantis, Ramakanth Pasunuru, Roberta Raileanu, Baptiste Roziere, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, Edouard Grave, Yann LeCun, and Thomas Scialom. 2023. Augmented language models: a survey. *Transactions on Machine Learning Research*. Survey Certification.

Ali Modarressi, Ayyoob Imani, Mohsen Fayyaz, and Hinrich Schütze. 2023. Ret-llm: Towards a general read-write memory for large language models.

OpenAI. 2023. Gpt-4 technical report.

Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. 2023. Automatically correcting large language models: Surveying the landscape of diverse self-correction strategies. *arXiv preprint arXiv:2308.03188*.

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

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*.

Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, et al. 2023. Check your facts and try again: Improving large language models with external knowledge and automated feedback. *arXiv preprint arXiv:2302.12813*.

Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, Yi Ren Fung, Yusheng Su, Huadong Wang, Cheng Qian, Runchu Tian, Kunlun Zhu, Shihao Liang, Xingyu Shen, Bokai Xu, Zhen Zhang, Yining Ye, Bowen Li, Ziwei Tang, Jing Yi, Yuzhang Zhu, Zhenning Dai, Lan Yan, Xin Cong, Yaxi Lu, Weilin Zhao, Yuxiang Huang, Junxi Yan, Xu Han, Xian Sun, Dahai Li, Jason Phang, Cheng Yang, Tongshuang Wu, Heng Ji, Zhiyuan Liu, and Maosong Sun. 2023. Tool learning with foundation models.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, dahai li, Zhiyuan Liu, and Maosong Sun. 2024. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *The Twelfth International Conference on Learning Representations*.

A David Redish. 2016. Vicarious trial and error. *Nature Reviews Neuroscience*, 17(3):147–159.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Christopher Riesbeck. 1981. Failure-driven reminding for incremental learning. In *IJCAI*, pages 115–120. Citeseer.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning to retrieve prompts for in-context learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2655–2671, Seattle, United States. Association for Computational Linguistics.

Roger C. Schank. 1983. *Dynamic Memory: A Theory of Reminding and Learning in Computers and People*. Cambridge University Press, USA.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Timo Schick and Hinrich Schütze. 2021. Generating datasets with pretrained language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6943–6951, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Thomas Scialom, Tuhin Chakrabarty, and Smaranda Muresan. 2022. Fine-tuned language models are continual learners. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6107–6122, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugging-GPT: Solving AI tasks with chatGPT and its friends in hugging face. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Richard Shin, Christopher Lin, Sam Thomson, Charles Chen, Subhro Roy, Emmanouil Antonios Platanios, Adam Pauls, Dan Klein, Jason Eisner, and Benjamin Van Durme. 2021. Constrained language models yield few-shot semantic parsers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7699–7715, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Robert W Shumaker, Kristina R Walkup, and Benjamin B Beck. 2011. *Animal tool behavior: the use and manufacture of tools by animals*. JHU Press.

Nianwen Si, Hao Zhang, Heyu Chang, Wenlin Zhang, Dan Qu, and Weiqiang Zhang. 2023. Knowledge unlearning for llms: Tasks, methods, and challenges.

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. *arXiv preprint arXiv:2306.06624*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. 2023. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051, Toronto, Canada. Association for Computational Linguistics.

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

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Krist Vaesen. 2012. The cognitive bases of human tool use. *Behavioral and brain sciences*, 35(4):203–218.

Danqing Wang and Lei Li. 2023. Learning from mistakes via cooperative study assistant for large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10667–10685, Singapore. Association for Computational Linguistics.

Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Chandu, David Wadden, Kelsey MacMillan, Noah A. Smith, Iz Beltagy, and Hannaneh Hajishirzi. 2023a. How far can camels go? exploring the state of instruction tuning on open resources. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023b. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.

Tianbao Xie, Fan Zhou, Zhoujun Cheng, Peng Shi, Luoxuan Weng, Yitao Liu, Toh Jing Hua, Junning Zhao, Qian Liu, Che Liu, Leo Z. Liu, Yiheng Xu, Hongjin Su, Dongchan Shin, Caiming Xiong, and Tao Yu. 2023. Openagents: An open platform for language agents in the wild.

Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, Madian Khabsa, Han Fang, Yashar Mehdad, Sharan Narang, Kshitiz Malik, Angela Fan, Shruti Bhosale, Sergey Edunov, Mike Lewis, Sinong Wang, and Hao Ma. 2023. Effective long-context scaling of foundation models.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. STar: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*.

Kexun Zhang, Hongqiao Chen, Lei Li, and William Wang. 2023. Syntax error-free and generalizable tool use for llms via finite-state decoding.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2023. Expel: Llm agents are experiential learners.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2023. Memorybank: Enhancing large language models with long-term memory.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, LILI YU, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023. LIMA: Less is more for alignment. In *Thirty-seventh Conference on Neural Information Processing Systems*.

A API Selection

We select 50 APIs from ToolBench[4], a large collection of APIs from BMTools[5] and RapidAPI[6]. For BMTools, we manually selected 10 high-quality APIs. For RapidAPI, due to the large scale of API pool (over 16k), we perform filtering by 1) selecting APIs that are free to use without subscription needed; 2) sorting the APIs based on the rate limit and latency and selecting the top 400 ones; 3) running a small scale exploration using ChatGPT to interact with the API execution environment, and selecting the top 40 ones based on the API call success rate.

B Experimental Details

Fine-tuning. All model fine-tuning are done using the AdamW optimizer (Loshchilov and Hutter, 2019) with $\beta = (0.9, 0.999)$, $\epsilon = 10^{-8}$ and no weight decay. We fine-tune the model for one epoch with learning rate 2×10^{-5}, and a cosine scheduler with warmup ratio 0.03, and batch size 64. All fine-tuning runs are done with 4 NVIDIA A100/A6000 GPUs. The training examples are truncated to $2{,}048$ tokens.

Flan-V2 data & rehearsal. We use Flan-V2 (Chung et al., 2022; Longpre et al., 2023) for rehearsal which is one of the highest quality general-domain instruction tuning data (Wang et al., 2023a). We use the data released by Wang et al. (2023a).[7]

C Example queries for all explored trials

Table 14, 15, 16 include the queries for all explored trials for the `forecast_weather` API across different settings (as indicated in the caption). Figure 4 includes the histogram of explored trials grouped according to the inquired attribute, comparing STE and STE without long-term memory. The results clearly show that the memory mechanisms in STE substantially improve the diversity and comprehensiveness of the exploration.

D Error Examples of GPT-4 and Mistral-Instruct-7B

Table 4, 5, 6 include the error examples of GPT-4, and Table 7, 8, 9 include the error examples of fine-tuned Mistral-Instruct-7B referred by §4.2.

[4]https://github.com/OpenBMB/ToolBench

[5]https://github.com/OpenBMB/BMTools

[6]https://rapidapi.com/

[7]https://github.com/allenai/open-instruct

E Prompts

Table 10, 11 include the full prompt for simulated trial and error. Table 10 includes the prompt for self-refine with execution feedback and short-term memory, and Table 11 includes the prompt for long-term memory. Table 12 and 13 includes the prompts for example filtering and paraphrasing respectively.

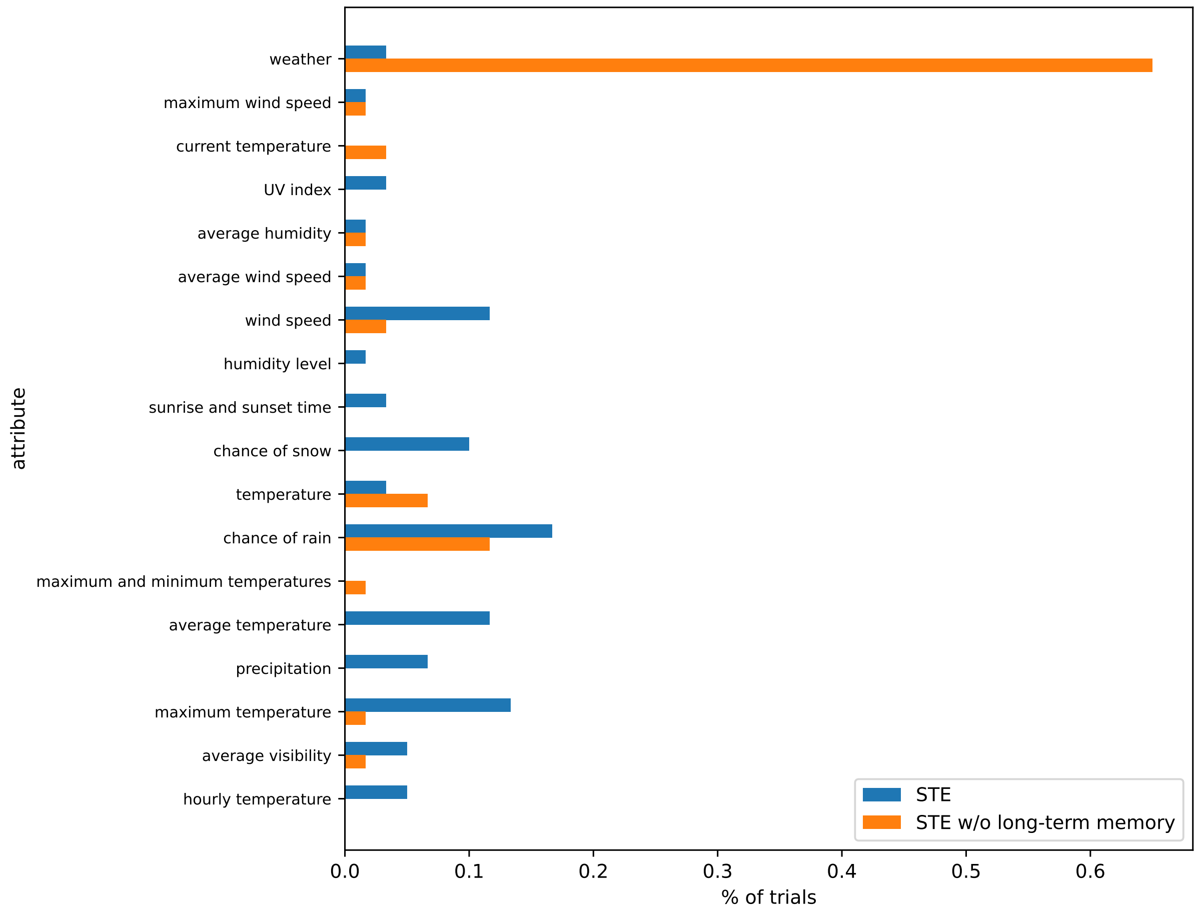


Figure 4: Histogram of explored trials grouped according to the inquired attribute. Long-term memory substantially improves the diversity of the explored attributes.

User query: Which parks in San Francisco have hiking trails?
Ground truth API name: `search_places`
Description for ground truth API: Run Places search. Your input should be a json (args json schema): {"query": string, }
Ground truth arguments: {"query": "parks in San Francisco with hiking trails"}

Predicted API: `Geographic coordinates by placename`
Description for predicted API: Returns geographic coordinates for the given placename (city, village, etc.). The method returns the place whose name is most similar to the search string.", "required_parameters": [{"name": "name", "type": "STRING", "description": "Placename", "default": "London"}, {"name": "lang", "type": "ENUM", "description": "Two-letter language code (ISO639-1). The following values are available: en (english), ru (russian)", "default": ""}], "optional_parameters": [{"name": "country", "type": "STRING", "description": "Two-letter country code, ISO-3166 (optional). Default is all countries.", "default": ""}]
Predicted Arguments: {"name": "San Francisco", "lang": "en"}

Table 4: Error example for GPT-4 (category: wrong API call). Here GPT-4 "overlooks" the Google Places API which fulfills the query, and instead calls an API that retrieves the geographic coordinates of the target place.

User query: What are the geographic coordinates for the city of Sydney, Canada?
Ground truth API name: Geographic coordinates by placename
Description for ground truth API: Returns geographic coordinates for the given placename (city, village, etc.). The method returns the place whose name is most similar to the search string. Required parameters: [{"name": "name", "type": "STRING", "description": "Placename", "default": "London"}, {"name": "lang", "type": "ENUM", "description": "Two-letter language code (ISO639-1). The following values are available: en (english), ru (russian)", "default": ""}]. Optional parameters: [{"name": "country", "type": "STRING", "description": "Two-letter country code, ISO-3166 (optional). Default is all countries.", "default": ""}]
Ground truth arguments: `{"name": "Sydney", "country": "CA", "lang": "en"}`

Predicted API: Geographic coordinates by placename
Description for predicted API: Returns geographic coordinates for the given placename (city, village, etc.). The method returns the place whose name is most similar to the search string.", "required_parameters": [{"name": "name", "type": "STRING", "description": "Placename", "default": "London"}, {"name": "lang", "type": "ENUM", "description": "Two-letter language code (ISO639-1). The following values are available: en (english), ru (russian)", "default": ""}], "optional_parameters": [{"name": "country", "type": "STRING", "description": "Two-letter country code, ISO-3166 (optional). Default is all countries.", "default": ""}]
Predicted Arguments: `[{"name": "Sydney", "country": "CA"}`

Table 5: Error example for GPT-4 (category: wrong API arguments). Here GPT-4 chooses the right tool but fails to fill in the required "lang" keyword.

User query: What are the main causes of climate change?
Ground truth API name: `search`
Description for ground truth API: The input is an exact entity name. The action will search this entity name on Wikipedia and returns the first five sentences if it exists. If not, it will return some related entities to search next. Your input should be a json (args json schema): {"entity": string, }
Ground truth arguments: {"entity": "causes of climate change"}

Predicted API: `search_general`
Description for predicted API: Run query through GoogleSearch and parse result. Your input should be a json (args json schema): {"query": string, }
Predicted Arguments: {"query": "main causes of climate change"}

Table 6: Error example for GPT-4 (category: hard to judge correctness). Here GPT-4 calls the Google Search API which could also be a valid choice besides the ground truth.

User query: What is the current advisory information for the `Union City station`?
Ground truth API name: BART - Advisory information
Description for ground truth API: The BART API gives you access to pretty much all of the BART service and station data available on the BART website. "required_parameters": [{"name": "cmd", "type": "STRING", "description": "See more examples http://api.bart.gov/docs/overview/examples.aspx", "default": "bsa"}], "optional_parameters": [{"name": "orig", "type": "STRING", "description": "Optional station filter. Uses 4 character BART station abbreviations (http://api.bart.gov/docs/overview/abbrev.aspx)", "default": ""}]
Ground truth arguments: {"cmd": "bsa", `"orig": "UCTY"`}

Predicted API: BART - Advisory information
Predicted Arguments: {"cmd": "bsa", `"orig": "UNION"`}

Table 7: Error example of Mistral-Instruct (7B) after fine-tuning. Here the model hallucinates a wrong abbreviation for the target station due to a lack of commonsense/world knowledge.

User query: What are the ongoing giveaways for PC game keys on the `GOG platform`?
Ground truth API name: GamerPower - Filter & Group Giveaways
Description for ground truth API: "Find all free games, loot and giveaways with this giveaway tracker API powered by GamerPower.com! Access programmatically the best giveaways in gaming! Filter and group platforms and giveaway types to get personalized results.", "required_parameters": [{"name": "platform", "type": "STRING", "description": "", "default": "epic-games-store.steam.android"}], "optional_parameters": [{"name": "type", "type": "STRING", "description": "", "default": "game.loot"}]
Ground truth arguments: {`"platform": "gog"`, "type": "game.key"}

Predicted API: GamerPower - Filter & Group Giveaways
Predicted Arguments: {`"platform": "pc"`, "type": "game.key"}

Table 8: Error example of Mistral-Instruct (7B) after fine-tuning. Here the model makes a mistake on understanding the user query and predicts the wrong argument for "platform".

User query: Can you provide the total number of matches played between Leeds United and Sheffield Wednesday in the English Premier League?
Ground truth API name: Football Dolphin - Head to head statistics
Description for ground truth API: This Api returns statistical data about English Premier League. Click on the link to view all endpoints in one web app https://football-dolphin-web-app.up.railway.app/. "required_parameters": [{"name": "first_team", "type": "STRING", "description": "**Enter first team from all available teams:** Arsenal, Aston Villa, Barnsley, Birmingham, Blackburn, Blackpool, Bolton, Bournemouth, Bradford, Brighton, Burnley, Cardiff, Charlton, Chelsea, Coventry, Crystal Palace, Derby, Everton, Fulham, Huddersfield, Hull, Ipswich, Leeds, Leicester, Liverpool, Man City, Man United, Middlesbrough, Newcastle, Norwich, Nott'm Forest, Portsmouth, QPR, Reading, Sheffield United, Sheffield Weds, Southampton, Stoke, Sunderland, Swansea, Tottenham, Watford, West Brom, West Ham, Wigan, Wimbledon, Wolves", "default": "Man United"}, {"name": "second_team", "type": "STRING", "description": "**Enter second team from all available teams:** Arsenal, Aston Villa, Barnsley, Birmingham, Blackburn, Blackpool, Bolton, Bournemouth, Bradford, Brighton, Burnley, Cardiff, Charlton, Chelsea, Coventry, Crystal Palace, Derby, Everton, Fulham, Huddersfield, Hull, Ipswich, `Leeds`, Leicester, Liverpool, Man City, Man United, Middlesbrough, Newcastle, Norwich, Nott'm Forest, Portsmouth, QPR, Reading, Sheffield United, Sheffield Weds, Southampton, Stoke, Sunderland, Swansea, Tottenham, Watford, West Brom, West Ham, Wigan, Wimbledon, Wolves", "default": "Liverpool"}, {"name": "type_of_statistics", "type": "STRING", "description": "**Enter one from available types of statistics:** full time result, home vs away full time result, result first half and the match,exact number of goals in the match, goals over, goals under", "default": "full time result"}]
Ground truth arguments: {"first_team": "`Leeds`", "second_team": "Sheffield Weds", "type_of_statistics": "full time result"}

Predicted API: Football Dolphin - Head to head statistics
Predicted Arguments: {"first_team": "`Leeds United`", "second_team": "Sheffield Weds", "type_of_statistics": "full time result"}

Table 9: Error example of Mistral-Instruct (7B) after fine-tuning. Here the model correctly extracts the target entity but fails to ground it to the set of supported entities of the tool.

```
Your task is to answer the user's query as best you can. You have access to the following tools which you can use via API call to help with your response:
{api_descriptions}
Now you have the chance to explore the available APIs. You can do this by 1) synthesizing some natural user query that calling the API could help, and 2) trying to respond to the user query with the help of the APIs. Here, you can focus on queries that only require calling the API once.
Now, first input your synthesized user query. You should make the query natural - for example, try to avoid using the provided API descriptions or API names in the query, as the user does not know what APIs you have access to. Also, try to make the query as specific as possible. Input just the user query alone; do NOT solve the query for now.

User Query:
=========
Now, try to respond to the query using the available APIs.
The format you use the API is by specifying 1) Action: the API function name you'd like to call 2) Action Input: the input parameters of the API call in a json string format. The result of the API call will be returned starting with "Observation:". Remember that you should only perform a SINGLE action at a time, do NOT return a list of multiple actions.
Reminder:
1) the only values that should follow "Action:" are: {api_names}
2) use the following json string format for the API arguments:
Action Input:
{
  "key_1": "value_1",
  ...
  "key_n": "value_n"
}
Remember to ALWAYS use the following format:
Thought: you should always think about what to do next
Action: the API function name
Action Input: the input parameters of the API call in json string format
Observation: the return result of the API call. This is what I will provide you with; you do not need to repeat it in your response.
... (this Thought/Action/Action Input/Observation can repeat N times)
Thought: I now know the final answer
Final Answer: the response to the user query

Begin! Remember that your response should never start with "Observation:" since that is what I will provide you with. Once you have enough information, please immediately use
Thought: I now know the final answer
Final Answer:

User Query (the same you just synthesized): {query}
=========
Do you think you successfully answered this query in the end? Respond with "Yes" or "No".
=========
Now you know a bit more about the API. You can synthesize another user query to explore the API a bit further and consolidate your understanding of the API, based on things that you discovered about this API. Again, just input the user query alone; do NOT solve the query for now.

User Query:
=========
Now try to solve the query using the API. Remember to follow the same format, i.e,
Thought:
Action:
Action Input:
Observation:
Final Answer:
```

Table 10: Prompt for self-refine with execution feedback and short-term memory.

Below are queries you have already explored and whether you successfully solved them with the API's help:

{long_term_memory}

Based on these, try to explore queries that can help you understand the API further; avoid synthesizing queries that are too close to the existing ones.

Table 11: Prompt for long-term memory.

An assistant is trying to respond to the user query with the help of some APIs. The APIs that the assistant has access to are as follows:
{api_descriptions}
Now, your task is to evaluate how well the assistant did the job. Check carefully the following aspects of the assistant's response:
1) whether the response answers the user's query in an informative way. For example, if the API calls are unsuccessful and the agent can't find the answer to the request, you should say "No."
2) whether the response is faithful with respect to the execution results of the API calls. The response should not include information that cannot be supported by the API call feedback,
3) whether the assistant used the API calls appropriately. For example, the assistant should always use relevant API calls for queries about up-to-date information or complex calculations,
For each of the three aspects, you should say "Yes" or "No" indicating whether the assistant did a good job in that aspect, and explain the reason behind your judgment. Your output should follow the format below, where "<explanation>" should be your actual explanation for the corresponding judgment:
1) Yes/No. <explanation>
2) Yes/No. <explanation>
3) Yes/No. <explanation>
Now, the user query is:
{query}
The assistant's API calls and the corresponding execution results are:
{chains}
The assistant's final response is:
{final_ans}
Now, your evaluation is (remember to follow the previous format):

Table 12: Prompt for example filtering.

Below you will be given a user query. Try to paraphrase it in a different way while preserving its meaning. The query is:
{query}
Your paraphrase of the query:
=========
Can you try to paraphrase it again in a new way? Avoid coming up with something too close to your previous ones. Your paraphrase:

Table 13: Prompt for query paraphrasing.

What is the UV index in San Francisco for the next 3 days?
What is the UV index in Sydney for the next 3 days?
What is the average humidity in Miami Beach for the next 5 days?
What is the average temperature in Los Angeles for the next 3 days?
What is the average temperature in Miami for the next 5 days?
What is the average temperature in San Francisco for the next 3 days?
What is the average temperature in Sydney for the next 10 days?
What is the average visibility in New York City for the next 3 days?
What is the average visibility in Tokyo for the next 4 days?
What is the average wind speed in Chicago for the next 7 days?
What is the chance of rain in London for the next 3 days?
What is the chance of rain in Los Angeles for the next 7 days?
What is the chance of rain in Miami Beach for the next 3 days?
What is the chance of rain in San Francisco for the next 3 days?
What is the chance of rain in San Francisco for the next 5 days?
What is the chance of rain in Seattle for the next 7 days?
What is the chance of rain in Sydney for the next 5 days?
What is the chance of rain in Tokyo for the next 7 days?
What is the chance of snow in Boston tomorrow?
What is the chance of snow in Chicago for the next 5 days?
What is the chance of snow in Denver for the next 3 days?
What is the chance of snow in New York City for the next 5 days?
What is the chance of snow in Paris for the next 3 days?
What is the forecasted wind speed in London for the next 5 days?
What is the hourly precipitation forecast in New York City for the next 24 hours?
What is the hourly precipitation forecast in Seattle for the next 24 hours?
What is the hourly temperature forecast for Chicago tomorrow?
What is the hourly temperature forecast in Los Angeles for the next 24 hours?
What is the hourly temperature forecast in New York City for the next 12 hours?
What is the hourly temperature forecast in San Francisco for the next 12 hours?
What is the hourly wind speed forecast in Miami tomorrow?
What is the maximum temperature in Sydney for the next 7 days?
What is the maximum wind speed in Los Angeles for the next 2 days?
What is the sunrise and sunset time in Paris tomorrow?
What is the sunrise and sunset time in Tokyo tomorrow?
What is the temperature range in Sydney for the next 3 days?
What is the total precipitation in Sydney for the next 10 days?
What is the total precipitation in Tokyo for the next 7 days?
What is the weather forecast in Miami for the next week?
What is the wind forecast in San Francisco for the next 7 days?
What will be the average temperature in London for the next 5 days?
What will be the average temperature in Seattle for the next 7 days?
What will be the average temperature in Tokyo for the next 7 days?
What will be the average visibility in Miami for the next 7 days?
What will be the average wind speed in San Francisco for the next 3 days?
What will be the chance of rain in San Diego for the next 7 days?
What will be the humidity level in London for the next 5 days?
What will be the maximum temperature in Chicago for the next 5 days?
What will be the maximum temperature in London for the next 10 days?
What will be the maximum temperature in Los Angeles for the next 7 days?
What will be the maximum temperature in Madrid for the next 7 days?
What will be the maximum temperature in Paris for the next 7 days?
What will be the maximum temperature in Sydney for the next 5 days?
What will be the maximum temperature in Tokyo for the next 10 days?
What will be the maximum wind speed in San Francisco for the next 5 days?
What will be the weather like in New York City for the next 5 days?
What will be the wind speed forecast in Boston for the next 3 days?
What will be the wind speed in Chicago tomorrow morning?
Will it rain in Seattle tomorrow?
Will it snow in Denver next week?

Table 14: Queries in all explored trials (prefix-sorted). Setting: STE (both short-term and long-term memory).

Is it going to rain in London tomorrow?
What will be the temperature in San Francisco on Thursday?
What will be the weather like in Barcelona on Friday?
What will be the weather like in Berlin next Wednesday?
What will be the weather like in Chicago in the next 7 days?
What will be the weather like in Denver in the next 5 days?
What will be the weather like in London in the next 2 days?
What will be the weather like in London in the next 3 days?
What will be the weather like in London in the next 5 days?
What will be the weather like in London next Friday?
What will be the weather like in London on Sunday?
What will be the weather like in Los Angeles in the next 3 days?
What will be the weather like in Los Angeles on Monday?
What will be the weather like in Madrid in the next 5 days?
What will be the weather like in Madrid on Sunday?
What will be the weather like in Miami in the next 7 days?
What will be the weather like in Miami on Friday?
What will be the weather like in Miami on Sunday?
What will be the weather like in Miami on Wednesday?
What will be the weather like in New York City in the next 5 days?
What will be the weather like in New York City next Saturday?
What will be the weather like in New York City on Thursday?
What will be the weather like in Paris in the next 3 days?
What will be the weather like in Paris next Monday?
What will be the weather like in Paris on Sunday?
What will be the weather like in San Antonio in the next 10 days?
What will be the weather like in San Francisco on Friday?
What will be the weather like in San Francisco on Saturday?
What will be the weather like in Seattle in the next 5 days?
What will be the weather like in Sydney in the next 5 days?
What will be the weather like in Sydney in the next 7 days?
What will be the weather like in Sydney next Monday?
What will be the weather like in Sydney next Tuesday?
What will be the weather like in Sydney next Wednesday?
What will be the weather like in Sydney on Monday?
What will be the weather like in Sydney on Saturday?
What will be the weather like in Tokyo in the next 3 days?
What will be the weather like in Tokyo in the next 4 days?
What will be the weather like in Tokyo in the next 7 days?
What will be the weather like in Tokyo next Sunday?
What will be the weather like in Tokyo next Tuesday?
What will be the weather like in Tokyo on Friday?
What will be the weather like in Tokyo on Monday?
What will be the weather like in Tokyo on Saturday?
What will be the weather like in Tokyo on Sunday?
What will be the weather like in Tokyo on Thursday?
What will be the weather like in Tokyo on Wednesday?
What will the weather be like in Barcelona in the next 7 days?
What will the weather be like in Miami next Monday?
What will the weather be like in Paris on Monday?
What will the weather be like in Sydney next Wednesday?
What will the weather be like in Tokyo tomorrow?
Will it be cloudy in Tokyo tomorrow?
Will it be hot in Miami next week?
Will it be rainy in Seattle tomorrow?
Will it be sunny in Madrid tomorrow?
Will it be sunny in San Francisco next Thursday?
Will it be sunny in San Francisco on Wednesday?
Will it rain in London in the next 3 days?
Will it rain in Sydney tomorrow?

Table 15: Queries in all explored trials (prefix-sorted). Setting: STE with no short-term memory.

Can I get the hourly weather forecast for tomorrow in New York City?
Can you provide the weather forecast for the next 10 days in Los Angeles?
How accurate is the weather forecast for the upcoming week?
What is the average visibility in Paris for the next 7 days?
What is the chance of rain in Los Angeles tomorrow?
What is the chance of rain in Miami for the next 3 days?
What is the chance of rain in Seattle tomorrow?
What is the chance of rain in Sydney for the next 7 days?
What is the chance of rain in Tokyo for the next 5 days?
What is the current temperature in Los Angeles?
What is the current temperature in New York City?
What is the current weather condition in Sydney?
What is the forecasted temperature for Paris tomorrow?
What is the forecasted temperature in Los Angeles for the next 10 days?
What is the forecasted weather for New York City for the next 5 days?
What is the forecasted wind speed for Los Angeles tomorrow?
What is the hourly weather forecast for Los Angeles tomorrow?
What is the hourly weather forecast for Los Angeles tomorrow?
What is the weather forecast for London tomorrow?
What is the weather forecast for Los Angeles tomorrow?
What is the weather forecast for Tokyo in the next 10 days?
What is the weather forecast for the next 10 days in London?
What is the weather forecast for the next 2 days in London?
What is the weather forecast for the next 3 days in Paris?
What is the weather forecast for the next 5 days in Los Angeles?
What is the weather forecast for tomorrow in New York City?
What is the wind speed in Paris tomorrow?
What will be the average humidity in Tokyo for the next 3 days?
What will be the average wind speed in Seattle for the next 3 days?
What will be the maximum and minimum temperatures in London tomorrow?
What will be the maximum temperature in London tomorrow?
What will be the maximum wind speed in Paris tomorrow?
What will be the temperature in London in the next 3 days?
What will be the temperature in San Francisco tomorrow?
What will be the weather conditions in London for the next 7 days?
What will be the weather forecast for London tomorrow?
What will be the weather forecast for New York City in the next 5 days?
What will be the weather forecast for Seattle over the next 5 days?
What will be the weather forecast for Sydney next week?
What will be the weather forecast for the next 5 days in New York City?
What will be the weather in New York City for the next 5 days?
What will be the weather like in London for the next 3 days?
What will be the weather like in London for the next 7 days?
What will be the weather like in New York City for the next 5 days?
What will be the weather like in New York City for the next 5 days?
What will be the weather like in New York City for the next 5 days?
What will be the weather like in New York City for the next three days?
What will be the weather like in New York City in the next 5 days?
What will be the weather like in Paris for the next 10 days?
What will be the weather like in San Francisco for the next 5 days?
What will be the weather like in San Francisco tomorrow?
What will be the weather like in Tokyo for the next 7 days?
What will the weather be like in London for the next 5 days?
What will the weather be like in London for the next 7 days?
What will the weather be like in Los Angeles for the next 3 days?
What will the weather be like in New York City for the next 5 days?
What will the weather be like in New York City for the next 5 days?
What will the weather be like in New York City tomorrow?
Will it rain in Los Angeles tomorrow?
Will it rain in San Francisco tomorrow?

Table 16: Queries in all explored trials (prefix-sorted). Setting: STE with no long-term memory.