# Making Language Models Better Tool Learners with Execution Feedback

**Shuofei Qiao[1,2], Honghao Gui[1,2], Chengfei Lv[4], Qianghuai Jia[4], Huajun Chen[1,2,3], Ningyu Zhang[1,2]***

[1]College of Computer Science and Technology, Zhejiang University
[2]ZJU-Ant Group Joint Research Center for Knowledge Graphs, Zhejiang University
[3]ZJU-Hangzhou Global Scientific and Technological Innovation Center, Zhejiang University
[4]Alibaba Group
{shuofei,guihonghao,huajunsir,zhangningyu}@zju.edu.cn

## Abstract

Tools serve as pivotal interfaces that enable humans to understand and reshape the environment. With the advent of foundation models, AI systems can utilize tools to expand their capabilities and interact with the real world. Existing tool learning methodologies, encompassing supervised fine-tuning and prompt engineering approaches, often induce large language models to utilize tools indiscriminately, as complex tasks often exceed their own competencies. However, introducing tools for simple tasks, which the models themselves can readily resolve, can inadvertently propagate errors rather than enhance performance. This leads to the research question: *can we teach language models when and how to use tools?* To meet this need, we propose **T**ool lea**R**ning w**I**th exe**C**ution f**E**edback (**TRICE**), a two-stage end-to-end framework that enables the model to continually learn through feedback derived from tool execution, thereby learning when and how to use tools effectively. Experimental results, backed by further analysis, show that TRICE can make the large language model selectively use tools by improving the accuracy of tool usage while enhancing insufficient tool learning and mitigating excessive reliance on tools[1].

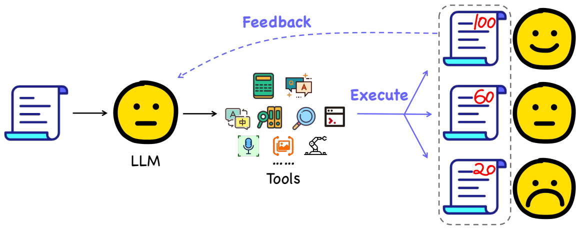


Figure 1: Large language model learns to use tools from execution feedback.

## 1 Introduction

The recent rapid advancement of foundation models (Brown et al., 2020; Ouyang et al., 2022; Chowdhery et al., 2022; Qiao et al., 2023; Zhao et al., 2023b) makes it practical for AI machines to create (Cai et al., 2023; Qian et al., 2023) and utilize tools effectively (Shen et al., 2023; Lu et al., 2023), which greatly transcends their inherent limitations in various underlying areas, including arithmetic (Cobbe et al., 2021; Parisi et al., 2022), knowledge updating (Sun et al., 2023; Zhao et al., 2023a), multi-modal semantic analysis (Wu et al., 2023; Driess et al., 2023), etc. Existing research has shed light on the potential of Large Language Models (LLMs) to exhibit a promising level of dexterity and finesse in tool use (Qin et al., 2023a; Wang et al., 2023). Prior works view tools as external resources to augment LLMs for better performance (Schick et al., 2023; Hao et al., 2023; Patil et al., 2023; Tang et al., 2023) or employ LLMs as a hub for human-tool interaction, responsible for orchestrating the deployment and usage of tools (Shen et al., 2023; Ge et al., 2023; Lu et al., 2023; Driess et al., 2023).

Despite the empirical success of previous work, a critical issue remains: LLMs often do not understand **when and how to properly use which tools**. On one hand, the use of tools is necessary to augment LLMs when facing complex problems that surpass their inherent capabilities. On the other hand, for simpler problems that can readily be solved by the models themselves, introducing tools can paradoxically propagate errors rather than enhance performance. These errors can include but are not limited to, improper selection of tool types, generation of incorrect tool inputs, and ineffective utilization of tool return results. Intuitively, it's crucial for LLMs to develop an awareness of when tools are necessary and when they are not, and to be able to make decisions about selecting the most appropriate tools for the task at hand.

To address the above issues, we propose **T**ool

* Corresponding Author.
[1]Code: https://github.com/zjunlp/TRICE.



*Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3550–3568
June 16-21, 2024 ©2024 Association for Computational Linguistics

| Method | LM | Model Scale | Mechanism | Feedback | Peft | Teacher | Unseen |
|---|---|---|---|---|---|---|---|
| Toolformer (Schick et al., 2023) | GPT-J | 6B | instruct-tuning | ✗ | ✗ | ✗ | ✗ |
| ToolkenGPT (Hao et al., 2023) | LLaMA | 13B, 30B | fine-tuning | ✗ | ✗ | ✗ | ✗ |
| HuggingGPT (Shen et al., 2023) | ChatGPT | >=100B | prompt | ✗ | ✗ | ✗ | ✔ |
| Chameleon (Lu et al., 2023) | GPT-4 | >=100B | prompt | ✗ | ✗ | ✗ | ✔ |
| ChatCoT (Chen et al., 2023) | ChatGPT | >=100B | prompt | ✗ | ✗ | ✗ | ✔ |
| Gorilla (Patil et al., 2023) | LLaMA | 7B | instruct-tuning | ✗ | ✗ | ✔ | ✔ |
| ToolLLaMA (Qin et al., 2023b) | LLaMA | 7B | instruct-tuning | ✗ | ✗ | ✔ | ✔ |
| GPT4Tools (Yang et al., 2023) | Vicuna | 13B | instruct-tuning | ✗ | ✔ | ✔ | ✔ |
| TRICE (ours) | ChatGLM<br>Alpaca<br>Vicuna | 6B, 7B | instruct-tuning<br>reinforcement learning | ✔ | ✔ | ✔ | ✔ |

Table 1: Comparison of related works. **Mechanism** denotes how the LM learns to invoke tools. **Feedback** stands for whether the LM learns from execution feedback. **Peft** means the parameter efficient tuning. **Teacher** expresses whether learning from a powerful teacher like ChatGPT. **Unseen** indicates the zero-shot capability on unseen tools.

lea**R**ning w**I**th exe**C**ution f**E**edback (**TRICE**) as shown in Figure 1, a two-stage end-to-end framework that enables the model to continually learn through feedback derived from execution, thereby learning when and how to use tools effectively. Specifically, we first prepare a dataset that helps discern when tool usage is necessary for LLMs and when it is not. Given the lack of gold labels, we utilize ChatGPT (OpenAI, 2022) to automatically generate tool usage APIs. Then, we introduce a two-stage training strategy to teach the model when to use tools: 1) **Behavior Cloning**. We conduct instruct-tuning on the dataset to let the model imitate the tool-using behavior. 2) **Reinforcement Learning with Execution Feedback (RLEF)**. We further reinforce the model with execution feedback by aligning it with desirable candidate responses, guiding the model to selectively use tools to avoid error propagation. We detail the main difference of TRICE with related works in Table 1.

We train and evaluate TRICE on various tasks and backbone models. Experimental results and further analyses demonstrate that TRICE successfully instructs the model to judiciously use tools, simultaneously enhancing insufficient tool learning, reducing excessive reliance on tools, and improving the accuracy of tool usage. In summary, the key contributions of our study are as follows:

- We introduce TRICE, a two-stage end-to-end training framework that leverages execution feedback to help LLMs become more proficient tool learners.
- We perform superior on eight benchmark datasets of four tasks with various models.
- Extensive empirical analysis demonstrates that TRICE can guide the model in judicious tool use, thereby enhancing insufficient tool use, reducing excessive dependency on tools, and improving the effectiveness of tool use.

## 2 Related Work

**Tool Learning.** Though possessing remarkable capabilities (Qiao et al., 2023; Yao et al., 2023), LLMs still struggle in many basic aspects where much smaller and simpler tools may precisely excel. Under this circumstance, a new paradigm, called Tool Learning (Qin et al., 2023a), is born to combine the strengths of both LLMs and specialized tools. Some works (Driess et al., 2023; Shen et al., 2023; Lu et al., 2023) regard LLMs as a decision-making hub for compositional tool using which can be called Tool-Oriented Learning (Qin et al., 2023a), while others (Gao et al., 2022; Liu et al., 2023; Schick et al., 2023) treat tools as complementary resources to extend the power of LLMs which can be called Tool-Augmented Learning (Mialon et al., 2023; Qin et al., 2023a). Despite their success, tool-augmented approaches tend to force LLMs to use tools mindlessly regardless of whether they actually need tools for help. This may, in some scenarios, steer LMs to erroneously choose the type of tools or the way to use tools, making the loss outweighs the gain. Compared to previous works, we try to **make LMs better tool learners by teaching them to use tools selectively instead of blindly**.

**Learning from Feedback.** An intuitive approach of tool learning is to fit LLMs on examples with human-labeled tools directly (Torabi et al., 2018; Li et al., 2022). However, this is often impractical to annotate every possible scenario (Codevilla et al., 2019) and difficult to generalize to new ones. It is worth noting that humans generally have the ability to correct and reflect on their own behavior from

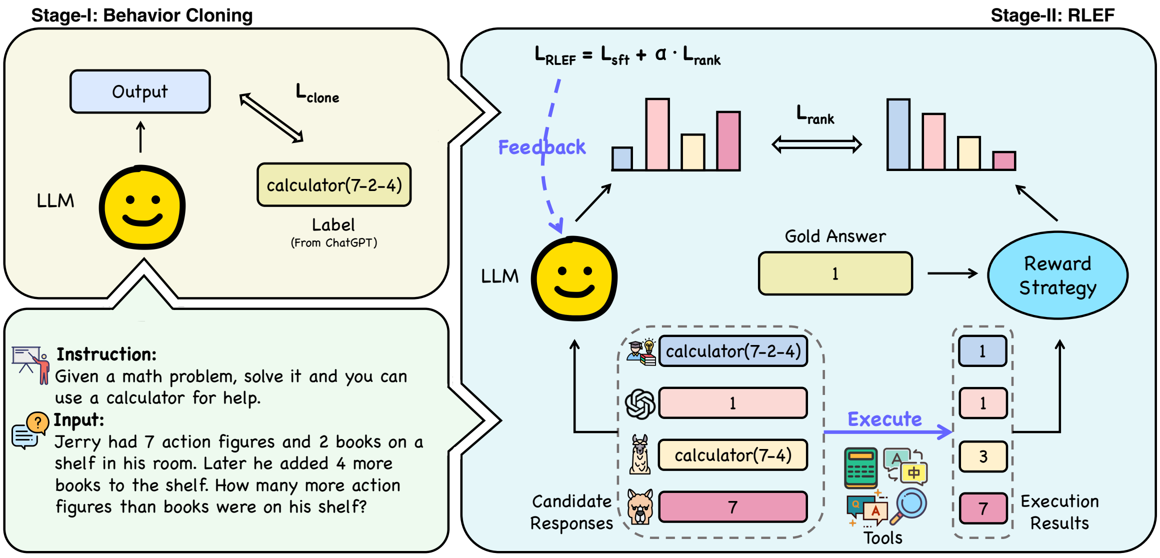


Figure 2: The overview of our proposed framework TRICE. In stage-I (**Behavior Cloning**), We conduct instruct-tuning on the dataset to let the model imitate the tool-using behavior. In stage-II (**RLEF**), we further reinforce the model with tool execution feedback by aligning it with desirable candidate responses.

trial and error (Allen et al., 2019). Intuitively, feedbacks from the environments or humans enable LLMs to understand the impact of their actions and adapt their behavior accordingly. Reinforcement learning (RL) excels at enabling models to learn decision-making abilities in complex environments through feedback (Schrittwieser et al., 2020; Yao et al., 2022; Ge et al., 2023). Ouyang et al. (2022) apply a state-of-the-art RL algorithm, PPO (Schulman et al., 2017), to align LLMs with human feedback. Liu et al. (2022) reinforce knowledge for commonsense question answering with a fixed QA model providing feedback. Yuan et al. (2023); Lee et al. (2023) offer a promising alternative that leverages powerful off-the-shelf LLMs to generate preferences. Compared to the previous feedback framework, we introduce **RLEF** for tool learning which reinforces the LLMs with the execution result of tools.

## 3 Methodology

**Problem Overview.** We mainly focus on four kinds of tasks, with each instance in the format of $x = (s, q, t, a)$, where $s$ denotes the specialized instruction of each task, $q$ refers to the question, $t$ stands for the tool API and $a$ is the gold answer. Following an instruction-following paradigm, the complete input of the LLM is as follows:

$$\texttt{input} = [s, q], \tag{1}$$

where [,] stands for text concatenating. In terms of the output, when LLM deems that no tool is necessary, it generates the answer $a$. Conversely, if the model identifies the need for a tool, it produces the tool API $t$, which encompasses the specific type of tool and its corresponding input:

$$\texttt{output} = \begin{cases} a & use\_tool = \texttt{false} \\ t & use\_tool = \texttt{true} \end{cases} \tag{2}$$

The detailed format of each task is shown in A.1.

Given the problem, the **main challenges** lie in 1) determining the LLM when to or not to harness tools for help and 2) how to impart the ability to the LLM to make selective use of tools. For the **former**, we allow the untrained model to directly infer answers, considering the correct ones as not requiring tools and the incorrect ones as indicating the need for tool assistance. For the **latter**, we adopt **TRICE**, a two-stage training strategy. In the first stage, we use **Behavior Cloning** to make the model imitate tools invoking. Building upon this, we continue to train the model for selective tool usage with **RLEF** in the second stage. The overview of our method is illustrated in Figure 2. **Please note that all symbols are globally defined in sections 3&4.**

**Data Preparation.** The data preparation follows the principles outlined in Eq.1&2. Given a raw initial dataset $\mathcal{D}_{\text{init}} = \{(q, a)\}_{i=1}^{|\mathcal{D}_{\text{init}}|}$ from the benchmark, we utilize LLM without fine-tuning to generate predictions. Since we do not have gold labels for tool APIs, we employ ChatGPT (OpenAI, 2022) to generate pseudo-labels under few-shot prompting. Specifically, we generate tool API labels $t = \texttt{tool}_{\texttt{name}}(\texttt{tool}_{\texttt{input}})$ for questions where the generated predictions are incorrect. For questions with correct predictions, we directly set $t = \texttt{None}$ to indicate that tool APIs are not required. We design particular instructions $s$ tailored to each task. In the end, we obtain $\mathcal{D}_{\text{tool}} = \{(s, q, t, a)\}_{i=1}^{|\mathcal{D}_{\text{tool}}|}$ according with Eq.1&2 containing the tool demand information of the specific LLM as we desire[2].

**Training.** As shown in Figure 2, based on $\mathcal{D}_{\text{tool}}$, we conduct a two-stage training approach: I) **Behavior Cloning** (§3.1). In this stage, we teach the model to imitate the tool usage behavior by fine-tuning it on $\mathcal{D}_{\text{tool}}$ in an instruct-tuning manner. This empowers the model with preliminary functionality of tool API invocation. II) **Reinforcement Learning with Execution Feedback** (§3.2). Drawing inspiration from fine-tuning with human feedback (Ouyang et al., 2022), we continue to reinforce our model obtained in stage I with execution feedback by steering it to align with desirable candidate responses.

### 3.1 Training Stage I: Behavior Cloning

During the behavior cloning stage, we aim to enable the LLM to master the schema of tool API invocation and develop preliminary skills in selectively utilizing tools. We perform supervised fine-tuning on $\mathcal{D}_{\text{tool}}$ in this stage.

Specifically, for the model $p_{\text{LM}}$ with tunable parameters $\theta$, the training loss of stage I can be formulated as:

$$\mathcal{L}_{\text{clone}}(\theta) = \sum_{(s,q,t,a)\in\mathcal{D}_{\text{tool}}} -\log p_{\text{LM}}(o|s, q; \theta), \tag{3}$$

where $o$ is the specified output of the model as defined in Eq.2. The final parameterized model of this stage is denoted as $\theta_{\text{clone}}$.

### 3.2 Training Stage II: RLEF

In stage II, we continue to optimize $\theta_{\text{clone}}$ with execution feedback, so as to enhance its capability to selectively utilize tools and improve the accuracy of decision-making regarding tool types and corresponding inputs.

**Overall Loss.** Following Yuan et al. (2023), for each question $q$, we have $k$ different candidate responses $y_i$ $(1 \leq i \leq k)$ marshaled from other LLMs (e.g. ChatGPT, LLaMA) or human experts. We apply a reward strategy to score each $y_i$ with $r_i = R(a, y_i)$ where $a$ is the gold answer of question $q$. Our goal is to instruct the LLM to determine the more desirable response by aligning with scores $\{r_i\}_k$. So we then score each $y_i$ with the LLM:

$$p_i = \frac{\sum_t \log p_{\text{LM}}(y_{i,m}|q, y_{i,<m}; \theta_{\text{clone}})}{||y_i||}, \tag{4}$$

where $m$ denotes the $m$th token of $y_i$, $p_i$ is the conditional log probability of $y_i$ and $||y_i||$ refers to the length-normalized factor.

To facilitate the LLM in learning the correct score ordering of different $y_i$, we introduce a ranking loss during training:

$$\mathcal{L}_{\text{rank}} = \sum_{r_i<r_j} \max(0, p_i - p_j). \tag{5}$$

Meanwhile, in order to prevent the model from deviating too far from the original parameters and generating unreasonable tool API invocation structure, we reintroduce the supervised fine-tuning loss:

$$\mathcal{L}_{\text{sft}} = -\sum_m \log p_{\text{LM}}(o_m|s, q, o_{<m}; \theta_{\text{clone}}). \tag{6}$$

Finally, the overall RLEF loss is defined as follows:

$$\mathcal{L}_{\text{RLEF}} = \alpha \cdot \mathcal{L}_{\text{rank}} + \mathcal{L}_{\text{sft}}, \tag{7}$$

where $\alpha$ is a hyperparameter that determines the proportion of the rank loss.

**Reward Strategy.** The reward strategy aims to give each $y_i$ an $r_i$ and rank them accordingly for a given question $q$. We view the output $o$ regulated in $\mathcal{D}_{\text{tool}}$ as the pseudo-human-expert (gold) response. Then the reward strategy is derived from two indicators: 1) **accuracy of the answer** and 2) **consistency of tool usage with the gold response.** Specifically, we employ a five-level scoring strategy. We assign the gold response with the maximum score. For the remaining, assuming that the correctness of the response is denoted as `True` for correct answers and `False` for incorrect answers, and whether the use of tools aligns with the gold

[2]For more details of data preparation, please refer to Appendix A.2.

| Task | Tool | Datasets |
|---|---|---|
| **Math Reasoning** | Calculator | ASDiv (Miao et al., 2020)<br>SVAMP (Patel et al., 2021)<br>GSM8K (Cobbe et al., 2021) |
| **Question Answering** | WikiSearch | WebQuestions (Berant et al., 2013)<br>NaturalQuestions (Kwiatkowski et al., 2019)<br>TriviaQA (Joshi et al., 2017) |
| **LAMA** | QA Model | T-REx (Petroni et al., 2019) |
| **Multilingual QA** | Translator | MLQA (Lewis et al., 2020) |

Table 2: Tasks, datasets and the corresponding tools.

response is denoted as `Yes` for alignment and `No` for misalignment, to ensure accurate and selective tool usage, our scoring is prioritized as follows:

$$\texttt{TrueYes} > \texttt{TrueNo} > \texttt{FalseYes} > \texttt{FalseNo}.$$

If two responses share the same state, they would receive the same score.

# 4 Experiments

## 4.1 Experimental Settings

**Tasks and Tools.** As shown in Table 2, we mainly evaluate our method on four tasks with each task specified to an external tool. Due to limited computational resources, we randomly sample train and test sets from each dataset to reduce the data scale. We display the detailed data distribution for each task in Figure 11. Following Schick et al. (2023), we use a more lenient evaluation criterion than exact match. We simply check for the last number predicted by the model for the math reasoning task and check whether the correct answer is within the first twenty words for other tasks. The QA model we use for LAMA is a retrieval-augmented LM fine-tuned on Natural Questions (Kwiatkowski et al., 2019) named *Atlas* (Izacard et al., 2023). We use the 600M parameter NLLB (Costa-jussà et al., 2022) as our machine Translation model for Multilingual QA.

**Candidate Response Generation.** We collect five responses for each question from four different models, e.g. ChatGPT, InstuctGPT, Vicuna-7B, Alpaca-7B, and the output regulated in $\mathcal{D}_{\text{tool}}$ as the gold response. To differentiate whether or not to use tools among candidate responses, we compel ChatGPT and InstructGPT to utilize tools while allowing Alpaca and Vicuna to make the choice of using tools. For ChatGPT and InstructGPT, we prompt them with instructions and few-shot examples, and for Alpaca-7B and Vicuna-7B, we fine-tune them on $\mathcal{D}_{\text{tool}}$ with LoRA (Hu et al., 2022) for a few steps in order to equip them with initial abilities for question answering and tool generation[3].

**Baselines.** We mainly experiment with the following LLMs: 1) **GPT-3.5** (OpenAI, 2022). We utilize the `text-davinci-003` version of GPT series. 2) **ChatGLM-6B** (Du et al., 2022), a general language model pre-trained with an autoregressive blank-filling objective. 3) **Alpaca-7B** (Taori et al., 2023), a model further trained on LLaMA-7B (Touvron et al., 2023) with self-instruction. 4) **Vicuna-7B** (Chiang et al., 2023), an open-source chatbot trained by fine-tuning LLaMA-7B. For GPT-3.5, we directly utilize the OpenAI API, while for other models, we train them all with LoRA (Hu et al., 2022) for efficiency in both stage-I&II.

Based on the differences in training status, we classify the baselines of our primary experiment into three categories (see Table 3&4): 1) *Prompt-Based.* Models are directly evaluated without training under **Zero-Shot** or **Few-Shot** manners. 2) *Supervised Fine-Tuning.* Models are trained purely on question-answer paired data (**0% Tool** usage) or trained purely on question-tool paired data (**100% Tool** usage). 3) TRICE-*Based.* Models are trained separately for each task (**TRICE-SPLIT**) or by combining training data from all tasks (**TRICE-MIX**) with TRICE. Furthermore, we observe the role of each training stage (see Figure 3): 1) **TRICE-I.** Models are trained only by the Behavior Cloning stage. 2) **TRICE-II.** Models are trained only by the RLEF stage. 3) **TRICE-ALL.** Models are trained by both TRICE-I and TRICE-II. In our analysis, we use arrows to indicate ↑positive and ↓negative performance compared to the specific baseline.

**Setups.** We fine-tune all our models with LoRA (Hu et al., 2022) in the format proposed in Alpaca (). All the models are trained for 5 epochs in stage-I and 2 epochs in stage-II. We use the learning rates of {2e-5, 1e-4, 3e-4} for ChatGLM-6B and {2e-5, 1e-4} for Alpaca-7B and Vicuna-7B. The $\alpha$ is set to {0.01, 0.1, 1} for all the models. The detailed hyper-parameters we use are shown in Appendix A.4. Since sampling responses and training are separated, our whole training procedure only needs to load one model. All our training can be completed on one 80G A800 GPU within 10 hours.

[3] For more details of candidate response generation, please refer to Appendix A.3.

| Setting | Model | Math Reasoning | | | Question Answering | | | LAMA | Multilingual QA | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|
| | | ASDiv | SVAMP | GSM8K | WebQ | NaturalQ | TriviaQA | T-REx | MLQA | |
| | GPT-3.5 | 64.6 | 62.0 | 19.8 | **46.4** | 15.0 | 41.3 | **58.7** | 34.4 | 42.8 |
| *Prompt Based* | ChatGLM (Zero-Shot) | 30.8 | 30.5 | 6.3 | 12.1 | 1.6 | 3.9 | 21.8 | 36.5 | 17.9 |
| | ChatGLM (Few-Shot) | 34.5 | 30.5 | 7.1 | 11.9 | 1.9 | 3.5 | 23.5 | 36.7 | 18.7 |
| *Supervised Fine-Tuning* | ChatGLM (0% Tool) | 44.2 | 35.5 | 7.2 | 14.9 | 9.5 | 11.2 | 30.6 | 37.7 | 23.9 |
| | ChatGLM (100% Tool) | 68.2 | 59.5 | 11.8 | 12.5 | 9.9 | 13.8 | 26.8 | 35.9 | 29.8 |
| **TRICE Based** | **ChatGLM (TRICE-SPLIT)** | 72.9 | <u>64.0</u> | 12.4 | 15.2 | 11.6 | 15.2 | 32.7 | 37.3 | 32.7 |
| | **ChatGLM (TRICE-MIX)** | <u>75.6</u> | **65.5** | 15.8 | 18.5 | 13.7 | 29.0 | 34.7 | 41.7 | 36.8 |
| *Prompt Based* | Alpaca (Zero-Shot) | 31.2 | 22.0 | 3.5 | 32.8 | 5.3 | 15.0 | 39.7 | 37.7 | 23.4 |
| | Alpaca (Few-Shot) | 38.3 | 23.5 | 4.3 | 33.9 | 6.0 | 16.6 | 41.1 | 45.5 | 26.2 |
| *Supervised Fine-Tuning* | Alpaca (0% Tool) | 44.0 | 23.0 | 5.8 | 37.6 | 10.3 | 20.4 | 53.1 | 48.9 | 30.4 |
| | Alpaca (100% Tool) | 68.6 | 44.5 | 15.6 | 35.9 | 16.4 | 32.6 | 41.7 | 46.6 | 37.7 |
| **TRICE Based** | **Alpaca (TRICE-SPLIT)** | 73.4 | 45.0 | 16.3 | 38.2 | 18.6 | 37.8 | 54.6 | 48.2 | 41.5 |
| | **Alpaca (TRICE-MIX)** | 75.2 | 58.0 | <u>21.5</u> | 41.4 | <u>20.7</u> | <u>41.4</u> | 55.2 | **52.0** | <u>45.7</u> |
| *Prompt Based* | Vicuna (Zero-Shot) | 50.4 | 33.0 | 6.4 | 34.9 | 7.7 | 16.7 | 42.5 | 35.9 | 28.4 |
| | Vicuna (Few-Shot) | 56.1 | 35.5 | 6.9 | 36.2 | 8.8 | 17.6 | 44.2 | 38.5 | 30.5 |
| *Supervised Fine-Tuning* | Vicuna (0% Tool) | 52.3 | 38.5 | 8.1 | 38.8 | 11.5 | 20.8 | 52.9 | 44.3 | 33.4 |
| | Vicuna (100% Tool) | 69.4 | 48.0 | 15.8 | 37.1 | 17.5 | 33.9 | 45.7 | 42.1 | 38.7 |
| **TRICE Based** | **Vicuna (TRICE-SPLIT)** | 72.6 | 49.0 | 16.6 | 43.2 | <u>20.7</u> | 40.8 | 54.1 | 42.6 | 42.4 |
| | **Vicuna (TRICE-MIX)** | **81.2** | 60.5 | **21.8** | <u>44.1</u> | **21.2** | **41.6** | <u>55.4</u> | <u>49.7</u> | **46.9** |

Table 3: Performance of TRICE across various tasks with different backbone models. **Zero-Shot:** models are directly evaluated. **Few-Shot:** models are prompted by 3 examples during evaluation. **0% Tool:** models are trained purely on question-answer paired data. During the above settings, the model does not rely on tools. **100% Tool:** models are trained purely on question-tool paired data. **TRICE-SPLIT:** models are trained with TRICE separately for each task. **TRICE-MIX:** models are trained with TRICE by combining training data from all tasks.

## 4.2 Main Results

**Selective Tool Learning of Single Tool.** Within each task, we train the model to learn the corresponding tool as shown in Table 2, thereby evaluating the model's proficiency in handling a single tool. From the rows labeled TRICE-SPLIT in Table 3, it is evident that training by TRICE, Alpaca and Vicuna perform on par with GPT-3.5, exhibiting only a slight decrease of ↓1.3% and ↓0.4% on average. Meanwhile, across all backbone models, TRICE-SPLIT demonstrates significant improvements compared to the prompt-based baselines, surpassing the few-shot setting with ↑14.0% for ChatGLM, ↑15.3% for Alpaca, and ↑11.9% for Vicuna. This indicates that TRICE consistently empowers LLMs to use tools effectively, irrespective of the model architecture and scale (ChatGLM-6B is encoder-decoder, while Alpaca-7B and Vicuna-7B are decoder-only). Moreover, whether it is completely independent (0% Tool) or dependent (100% Tool) on tools, supervised fine-tuning fails to beat TRICE-based training, which highlights the necessity and efficacy of judicious tool learning.

**Selective Tool Learning of Multi-Tools.** Across all tasks, we train the model to simultaneously learn all the tools, assessing its capabilities in multi-tool learning. As indicated in the rows labeled TRICE-MIX in Table 3, training across tasks achieves state-of-the-art performance by further exceeding the TRICE-SPLIT with over ↑4.0% average score gains across different models. Meanwhile, both Alpaca and Vicuna outperform GPT-3.5, exhibiting improvements of ↑2.9% and ↑4.1%, respectively. These results declare the potential of TRICE in selective multi-tool learning, which paves the way for expanding the capabilities of LLMs to wisely handle more complex and diverse types of tools.

| Model | Unseen Dataset | | | Unseen Tool |
|---|---|---|---|---|
| | Calculator | | QA Model | Retriever |
| | MultiArith | AddSub | SQuAD | HotpotQA |
| GPT-3.5 | 51.1 | 59.5 | **45.2** | **36.7** |
| Vicuna (Zero-Shot) | 42.3 | 44.1 | 28.6 | 19.7 |
| Vicuna (Few-Shot) | 45.5 | 49.1 | 31.2 | 20.6 |
| **Vicuna (TRICE-SPLIT)** | <u>63.1</u> | <u>75.2</u> | 30.9 | — |
| **Vicuna (TRICE-MIX)** | **66.6** | **80.5** | <u>35.7</u> | <u>27.3</u> |

Table 4: Performance to unseen datasets and tools. The presence of an empty value in the Unseen Tool TRICE-APLIT section is due to our reliance on the generalization achieved through mixed-tool training (MIX). Testing the new tool individually through separate training (SPLIT) using any single tool would not be suitable.

**Generalization of Tool Learning.** To analyze the generalization ability of TRICE, we extend the trained model to unseen datasets and tools. As illustrated in Table 4, we evaluate Vicuna on another two math reasoning datasets (MultiArith (Roy and Roth, 2015) and AddSub (Hosseini et al., 2014))

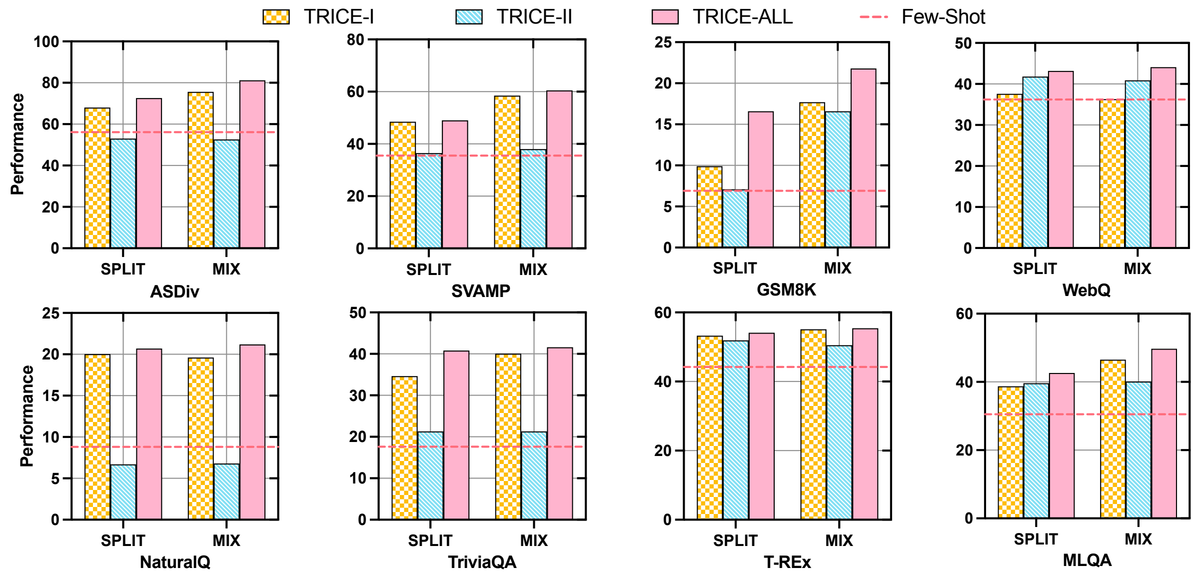


Figure 3: Performance of TRICE across all tasks at different training stages. **TRICE-I:** only train by Behavior Cloning (instruct-tuning) stage. **TRICE-II:** only train by RLEF (reinforcement learning with execution feedback) stage. **TRICE-ALL:** train by both TRICE-I and TRICE-II.

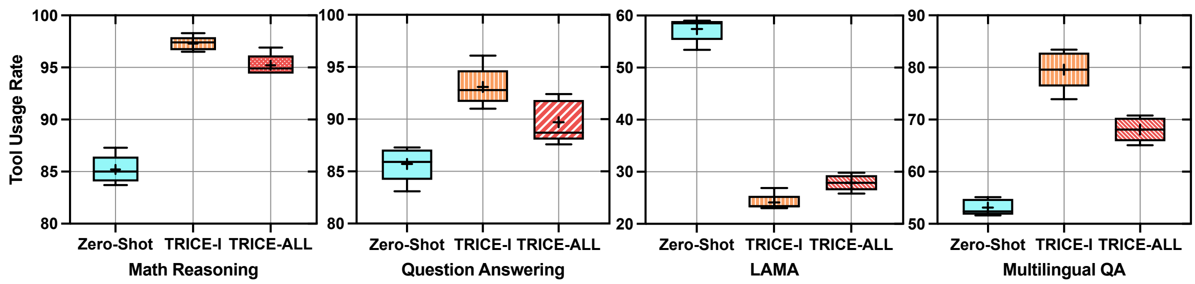


Figure 4: Comparison of tool use rate statistics among different training stages. In the Zero-Shot stage, we consider a need for tools when the model reaches a wrong answer.

and one LAMA dataset (SQuAD (Petroni et al., 2019)). Our approach enables continuous optimization of the model's performance on unseen datasets, with TRICE-MIX yielding superior results compared to TRICE-SPLIT. This suggests that TRICE equips the model with general tool usage capabilities. Furthermore, we steer the model towards unseen tools by simply modifying the instructions. The performance of Vicuna (TRICE-MIX) augmented by a retriever on HotpotQA (Yang et al., 2018) advances ↑6.7% than the few-shot manner. Despite the disparities between GPT-3.5 on certain datasets, these findings highlight the promise of multi-tool training based on TRICE for facilitating the generalization of tool learning.

### 4.3 Analyses of Selective Tool Learning

**Stage-I is the Foundation of Stable Selective Tool Learning.** Figure 3 showcases the performance of TRICE at different training stages, with Vicuna as the representative. It is evident that only trained in stage I (TRICE-I), the model acquires efficacious tool usage capabilities, resulting in a substantial performance improvement. Upon further training in stage II (TRICE-ALL), the model experiences additional performance enhancements in both the SPLIT and MIX training settings. However, the results obtained solely from stage II (TRICE-II) are unsatisfactory, indicating that the initial tool generation ability bestowed upon the model during stage I is crucial for more stable training.

**Stage-II Plays a Pivotal Role in Selective Tool Learning.** To investigate how the models learn to use tools selectively, we analyze the tool usage rate statistics of Vicuna during each training stage in Figure 4. After stage I, we notice that the model's reliance on tools has significantly deepened on most tasks. This indicates that the model effectively learns the pattern of tool usage in stage I. Still, due to the imbalanced data distribution regard-

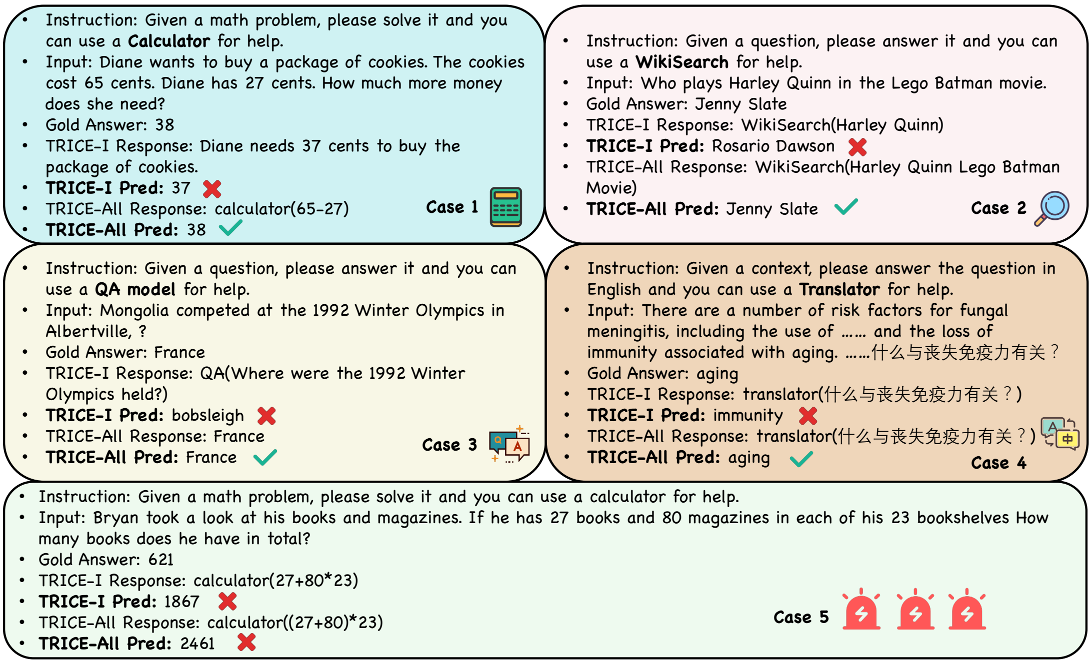


Figure 5: Case study. We mainly show the responses and predictions of stages I and All.

ing the presence or absence of tools in the training set, instruct-tuning tends to make the model overly dependent on tools. However, after stage II, the model not only shows performance improvement (see Figure 3) but visibly reduces its dependency on tools, which illustrates that the execution feedback can help mitigate the model's excessive reliance on tools and alleviate error propagation in tool usage. Moreover, it cannot be ignored that the fluctuation of LAMA differs from others. The decision-making process for invoking the QA model poses challenges, leading to insufficient tool learning during stage I. The improvement in tool usage rate during stage II implies that the execution feedback can help address the issue of inadequate tool learning. The above two phenomena highlight the validity of TRICE for selective tool usage.

**Case Study.** In Figure 5, we present several cases featuring responses and predictions from different stages. Case 1 suggests that stage II can alleviate the insufficient tool learning in stage I, urging the model to seek assistance from tools for questions it struggles to answer. Though stage I equips the model with a certain level of tool generation capability, it may not excel in making optimal decisions about the tool's input, as shown in Case 2. Stage II mitigates this limitation and enhances the accuracy of tool use. Case 3 confirms that our proposed method enables the model to use tools judiciously. In Case 4, despite having the same tool invocation in both stages I and II, the model may generate completely opposite answers. This indicates that stage II can further optimize the model's ability to leverage the return results of tools. However, as shown in Case 5, our model still exhibits certain flaws leading to errors in tool usage. We speculate that this could be attributed to the scale of our backbone models, which generally range from 6-7B, potentially limiting their tool learning ability.

## 5 Discussion

**Knowledge Conflicts.** In tool learning, LLMs manipulate tools and respond to users conditioned on a variety of knowledge sources. One particularly challenging issue in tool learning is the problem of knowledge conflicts (Qin et al., 2023a) which may derive from the conflicts between model knowledge and augmented knowledge from tools, and among augmented knowledge from different tools. This may lead to a lack of explainability in model prediction and planning. LLMs need to have the ability to differentiate knowledge from various sources and discern which ones are valuable, which ones are irrelevant, and even which ones may be harmful. This ability becomes even more critical in highly specialized fields such as biomedical research and legal assistance, where the accuracy and reliability of knowledge are of utmost importance. Our approach leverages the feedback loop of trial and error (see Figure 2) to learn when to use tools and when not to. The model learns to recognize situations where relying solely on its intrinsic knowledge may not be sufficient and utilizing tools is more reliable. Similarly, it learns to identify scenarios where its own learned knowledge is capable of solving the problem without the need for extensive tool usage. This learning process allows the model to adapt and make informed decisions about when to rely on its own capabilities and when to utilize tools effectively (see Figure 4).

**Interactive Learning.** Recent NLP has witnessed rapid advancement in interactive learning which considers language models as agents capable of observing, acting, and receiving feedback in a loop with external objects such as humans, knowledge bases, tools, models, and environments (Wang et al., 2023). Collaboration among multi-agents (Lin et al., 2023; Liang et al., 2023) and learning from feedback (Chen et al., 2022; Ichter et al., 2022) are the keys to achieving general embodied intelligence as of now. Our approach is a preliminary endeavor to explore the incorporation of embodied methods into tool learning. By leveraging feedback obtained through interactions between the environment (tools) and multi-agents with varying capabilities, we enable language models to learn more desirable execution strategies (see Figure 5). However, our current method is unable to learn the usage of multi-tool compositions. In the future, more sophisticated trial-and-error processes and feedback mechanisms will be necessary for LLMs to better utilize tools (e.g. learning multi-tool compositions) and even create new tools.

## 6 Conclusion

In this paper, we focus on addressing the challenge of selective utilization of tools by LLMs and propose a two-stage end-to-end training framework dubbed TRICE to make LLMs better tool learners with execution feedback. Through comprehensive experiments, we show that our method can achieve better performance compared to GPT-3.5. Further analyses illustrate that TRICE can selectively use tools by improving the accuracy of tool usage while enhancing insufficient tool learning and mitigating excessive reliance on tools.

## Limitations

In this paper, we focus on addressing the challenge of selective utilization of external tools by LLMs and propose a two-stage end-to-end training framework dubbed TRICE to make LLMs better tool learners with execution feedback. Despite our best efforts, there may be still some limitations that remain in this paper.

**Method.** Our approach can be applied to any tool-learning scenario, including embodied robotics. However, due to the iterative nature of execution feedback, which relies on continuous trial-and-error, it is typically more suitable for computationally feasible virtual environments, while real-world scenarios often require a significant time investment. In the future, we will explore more scientific and intricate feedback mechanisms to address the limitations above.

**Language Models.** Given our limited computational resources, we only conduct experiments on three backbone models with scales of 6-7B. In the future, we may advent on LLMs with different architectures and larger scales.

**Tasks and Datasets.** Due to the limited resources, we only experiment on four tasks containing eight datasets. There are also numerous tasks and scenarios that require the utilization of more diverse and complex tools. In the future, we will embark on further research endeavors.

## Acknowledgements

We would like to express gratitude to the anonymous reviewers for their kind comments. This work was supported by the National Natural Science Foundation of China (No. 62206246), the Fundamental Research Funds for the Central Universities (226-2023-00138), Zhejiang Provincial Natural Science Foundation of China (No. LGG22F030011), CAAI-Huawei MindSpore Open Fund, Ningbo Natural Science Foundation (2021J190), Yongjiang Talent Introduction Programme (2021A-156-G), CCF-Tencent Rhino-Bird Open Research Fund, and Information Technology Center and State Key Lab of CAD&CG, Zhejiang University.

## References

Kelsey R. Allen, Kevin A. Smith, and Joshua B. Tenenbaum. 2019. The tools challenge: Rapid trial-and-error learning in physical problem solving. *CoRR*, abs/1907.09620.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1533–1544. ACL.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. 2023. Large language models as tool makers. *CoRR*, abs/2305.17126.

Boyuan Chen, Fei Xia, Brian Ichter, Kanishka Rao, Keerthana Gopalakrishnan, Michael S. Ryoo, Austin Stone, and Daniel Kappler. 2022. Open-vocabulary queryable scene representations for real world planning. *CoRR*, abs/2209.09874.

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. *CoRR*, abs/2305.14323.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways. *CoRR*, abs/2204.02311.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

Felipe Codevilla, Eder Santana, Antonio M. López, and Adrien Gaidon. 2019. Exploring the limitations of behavior cloning for autonomous driving. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 9328–9337. IEEE.

Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Y. Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loïc Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. 2022. No language left behind: Scaling human-centered machine translation. *CoRR*, abs/2207.04672.

Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. 2023. Palm-e: An embodied multimodal language model. *CoRR*, abs/2303.03378.

Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2022. GLM: general language model pretraining with autoregressive blank infilling. pages 320–335.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2022. PAL: program-aided language models. *CoRR*, abs/2211.10435.

Yingqiang Ge, Wenyue Hua, Jianchao Ji, Juntao Tan, Shuyuan Xu, and Yongfeng Zhang. 2023. Openagi: When LLM meets domain experts. *CoRR*, abs/2304.04370.

Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. 2023. Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings. *CoRR*, abs/2305.11554.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 523–533. ACL.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Brian Ichter, Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, Dmitry Kalashnikov, Sergey Levine, Yao Lu, Carolina Parada, Kanishka Rao, Pierre Sermanet, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Mengyuan Yan, Noah Brown, Michael Ahn, Omar Cortes, Nicolas Sievers, Clayton Tan, Sichun Xu, Diego Reyes, Jarek Rettinghouse, Jornell Quiambao, Peter Pastor, Linda Luu, Kuang-Huei Lee, Yuheng Kuang, Sally Jesmonth, Nikhil J. Joshi, Kyle Jeffrey, Rosario Jauregui Ruano, Jasmine Hsu, Keerthana Gopalakrishnan, Byron David, Andy Zeng, and Chuyuan Kelly Fu. 2022. Do as I can, not as I say: Grounding language in robotic affordances. In *Conference on Robot Learning, CoRL 2022, 14-18 December 2022, Auckland, New Zealand*, volume 205 of *Proceedings of Machine Learning Research*, pages 287–318. PMLR.

Gautier Izacard, Patrick S. H. Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. Atlas: Few-shot learning with retrieval augmented language models. *J. Mach. Learn. Res.*, 24:251:1–251:43.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1601–1611. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *Trans. Assoc. Comput. Linguistics*, 7:452–466.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Lu, Thomas Mesnard, Colton Bishop, Victor Carbune, and Abhinav Rastogi. 2023. RLAIF: scaling reinforcement learning from human feedback with AI feedback. *CoRR*, abs/2309.00267.

Patrick S. H. Lewis, Barlas Oguz, Ruty Rinott, Sebastian Riedel, and Holger Schwenk. 2020. MLQA: evaluating cross-lingual extractive question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7315–7330. Association for Computational Linguistics.

Shuang Li, Xavier Puig, Chris Paxton, Yilun Du, Clinton Wang, Linxi Fan, Tao Chen, De-An Huang, Ekin Akyürek, Anima Anandkumar, Jacob Andreas, Igor Mordatch, Antonio Torralba, and Yuke Zhu. 2022. Pre-trained language models for interactive decision-making. In *NeurIPS*.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Zhaopeng Tu, and Shuming Shi. 2023. Encouraging divergent thinking in large language models through multi-agent debate. *CoRR*, abs/2305.19118.

Bill Yuchen Lin, Yicheng Fu, Karina Yang, Prithviraj Ammanabrolu, Faeze Brahman, Shiyu Huang, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2023. Swiftsage: A generative agent with fast and slow thinking for complex interactive tasks. *CoRR*, abs/2305.17390.

Jiacheng Liu, Skyler Hallinan, Ximing Lu, Pengfei He, Sean Welleck, Hannaneh Hajishirzi, and Yejin Choi. 2022. Rainier: Reinforced knowledge introspector for commonsense question answering. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 8938–8958. Association for Computational Linguistics.

Ruibo Liu, Jason Wei, Shixiang Shane Gu, Te-Yen Wu, Soroush Vosoughi, Claire Cui, Denny Zhou, and Andrew M. Dai. 2023. Mind's eye: Grounded language model reasoning through simulation. In *The Eleventh 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. *CoRR*, abs/2304.09842.

Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ramakanth Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, Edouard Grave, Yann LeCun, and Thomas Scialom. 2023. Augmented language models: a survey. *CoRR*, abs/2302.07842.

Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2020. A diverse corpus for evaluating and developing english math word problem solvers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July*

*5-10, 2020*, pages 975–984. Association for Computational Linguistics.

OpenAI. 2022. Chatgpt: Optimizing language models for dialogue. https://openai.com/blog/chatgpt/.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *NeurIPS*.

Aaron Parisi, Yao Zhao, and Noah Fiedel. 2022. TALM: tool augmented language models. *CoRR*, abs/2205.12255.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 2080–2094. Association for Computational Linguistics.

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

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick S. H. Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander H. Miller. 2019. Language models as knowledge bases? 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 2019, Hong Kong, China, November 3-7, 2019*, pages 2463–2473. Association for Computational Linguistics.

Cheng Qian, Chi Han, Yi R. Fung, Yujia Qin, Zhiyuan Liu, and Heng Ji. 2023. CREATOR: disentangling abstract and concrete reasonings of large language models through tool creation. *CoRR*, abs/2305.14318.

Shuofei Qiao, Yixin Ou, Ningyu Zhang, Xiang Chen, Yunzhi Yao, Shumin Deng, Chuanqi Tan, Fei Huang, and Huajun Chen. 2023. Reasoning with language model prompting: A survey. In *ACL*. The Association for Computational Linguistics.

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. 2023a. Tool learning with foundation models. *CoRR*, abs/2304.08354.

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

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1743–1752. The Association for Computational Linguistics.

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. *CoRR*, abs/2302.04761.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy P. Lillicrap, and David Silver. 2020. Mastering atari, go, chess and shogi by planning with a learned model. *Nature.*, 588(7839):604–609.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

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. *CoRR*, abs/2303.17580.

Zhiqing Sun, Xuezhi Wang, Yi Tay, Yiming Yang, and Denny Zhou. 2023. Recitation-augmented language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

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

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

Faraz Torabi, Garrett Warnell, and Peter Stone. 2018. Behavioral cloning from observation. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 4950–4957. ijcai.org.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

Zekun Wang, Ge Zhang, Kexin Yang, Ning Shi, Wangchunshu Zhou, Shaochun Hao, Guangzheng Xiong, Yizhi Li, Mong Yuan Sim, Xiuying Chen, Qingqing Zhu, Zhenzhu Yang, Adam Nik, Qi Liu, Chenghua Lin, Shi Wang, Ruibo Liu, Wenhu Chen, Ke Xu, Dayiheng Liu, Yike Guo, and Jie Fu. 2023. Interactive natural language processing. *CoRR*, abs/2305.13246.

Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. 2023. Visual chatgpt: Talking, drawing and editing with visual foundation models. *CoRR*, abs/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. *CoRR*, abs/2305.18752.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2369–2380. Association for Computational Linguistics.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. In *NeurIPS*.

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*.

Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, Songfang Huang, and Fei Huang. 2023. RRHF: rank responses to align language models with human feedback without tears. *CoRR*, abs/2304.05302.

Ruochen Zhao, Xingxuan Li, Shafiq Joty, Chengwei Qin, and Lidong Bing. 2023a. Verify-and-edit: A knowledge-enhanced chain-of-thought framework. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 5823–5840. Association for Computational Linguistics.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023b. A survey of large language models. *CoRR*, abs/2303.18223.

| Name | Stage-I | Stage-II |
|---|---|---|
| lora_r | 8 | 8 |
| lora_alpha | 16 | 16 |
| lora_target_modules | q_proj v_proj | q_proj v_proj |
| lora_dropout | 0.05 | 0.05 |
| max_length | 2048 | 2048 |
| batch_size_per_device | 48 | 8 |
| gradient_accumulation_steps | 8 | 32 |
| warmup_steps | 0 | 0 |
| epochs | 5 | 2 |
| lr | 1e-4, 3e-4 | 1e-4, 2e-5 |
| $\alpha$ | — | 0.01, 0.1, 1 |

Table 5: Hyperparameters to train Chatglm-6B.

| Name | Stage-I | Stage-II |
|---|---|---|
| lora_r | 8 | 8 |
| lora_alpha | 16 | 16 |
| lora_target_modules | q_proj v_proj | q_proj v_proj |
| lora_dropout | 0.05 | 0.05 |
| max_length | 512 | 512 |
| batch_size_per_device | 128 | 8 |
| gradient_accumulation_steps | 8 | 32 |
| warmup_steps | 0 | 0 |
| epochs | 5 | 2 |
| lr | 1e-4, 2e-5 | 1e-4, 2e-5 |
| $\alpha$ | — | 0.01, 0.1, 1 |

Table 6: Hyperparameters to train Alpaca-7B and Vicuna-7B.

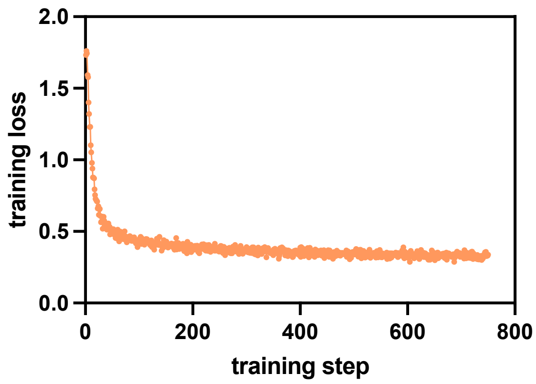


Figure 6: Training loss variations of Vicuna-7B in stage I of TRICE-MIX.

# A Appendix

## A.1 Task Format

We mainly evaluate our method on four kinds of tasks as shown in Table 2. Eq.1&2 formally define the input and output of each task in general. Here is the detailed format of each task.

**Math Reasoning** :

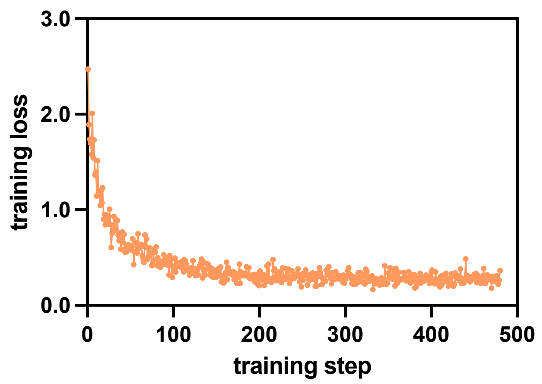


Figure 7: Training loss variations of Vicuna-7B in stage II of TRICE-MIX.

Instruction $s$: `Given a math problem, please solve it and you can use a calculator for help.`

Question $q$: `Mrs. Hilt has 50 cents. A pencil costs 5 cents. How many pencils can she buy with the money she has?`

Tool API $t$ (if needed): `calculator(50/5)`

Gold Answer $a$: `10`

**Question Answering** :

Instruction $s$: `Given a question, please answer it and you can use a WikiSearch for help.`

Question $q$: `Where are sunbeam microwaves made?`

Tool API $t$ (if needed): `WikiSearch(Sunbeam microwaves manufacturing location)`

Gold Answer $a$: `Florida`

**LAMA** :

Instruction $s$: `Given a question, please answer it and you can use a QA model for help.`

Question $q$: `Winners of the festivals «Chervona Ruta» (Ukraine), «Pearls of the Season» (Ukraine), «Boards» (Moscow), «Woodstock» ( ?`

Tool API $t$ (if needed): `QA(Where is the Woodstock festival held?)`

Gold Answer $a$: `Poland`

**Multilingual QA** :

Instruction $s$: `Given a context, please answer the question in English and you can use a translator for help.`

Question $q$: `Context: Over the next decade, she went on more than 40 field missions, meeting with refugees and internally displaced persons in over 30 countries. In 2002, when asked what she hoped to accomplish, she stated, “Awareness of the plight of these people. I think they should be commended for what they have survived, not looked down upon.” To that end, her 2001-02 field visits were chronicled in her book Notes from My Travels, which was published in October 2003 in conjunction with the release of her humanitarian drama Beyond Borders. Question: 她在10年内完成了多少任务?`

Tool API $t$ (if needed): `translator(她在10年内完成了多少任务?)`

Gold Answer $a$: `more than 40`

### A.2 Data Preparation

We present the prompt used to generate tool APIs for Math Reasoning, Question Answering, and LAMA in Figure 8-10. Since the sentence to be translated happens to be the provided question, Multilingual QA does not require ChatGPT to generate tool APIs. Due to limited computational resources, we randomly sample train and test sets from each dataset to reduce data scale and training/testing costs. The final data distribution for each task is illustrated in Figure 11.

### A.3 Response Generation

We show the prompt used to generate candidate responses for ChatGPT and GPT-3.5 in Figure 12-15. We use the same instructions in Figure 5 to generate candidate responses for Vicuna and Alpaca.

### A.4 Training Details

The hyperparameters we use to train ChatGLM-6B are shown in Table 5 and Alpaca-7B, Vicuna-7B are shown in Table 6. We present the training loss variations of Vicuna-7B in stages I and II of TRICE-MIX in Figure 6&7. During training, we observe that despite the decrease in training loss, prolonged reinforcement learning training will result in a significant performance loss. Typically, the model achieves optimal performance within the first 10-40 steps.

I will provide you with a math Question and a Golden answer. I need you to write “calculator(formula)” to invoke the API for assistance in solving the question, where “formula” is the formula to reach the Golden answer. Here are some examples:

Question: Natalia sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May?
Golden answer: 72
Output: calculator(48+48/2)

Question: Weng earns $12 an hour for babysitting. Yesterday, she just did 50 minutes of babysitting. How much did she earn?
Golden answer: 10
Output: calculator((12/60)*50)

Question: Weng earns $12 an hour for babysitting. Yesterday, she just did 50 minutes of babysitting. How much did she earn?
Golden answer: 10
Output: calculator((12/60)*50)

Question: Betty is saving money for a new wallet which costs $100. Betty has only half of the money she needs. Her parents decided to give her $15 for that purpose, and her grandparents twice as much as her parents. How much more money does Betty need to buy the wallet?
Golden answer: 5
Output: calculator(100-100/2-15-15*2)

Question: {question}
Golden answer: {answer}
Output:

Figure 8: Prompt used for Math Reasoning to generate tool APIs.

I will provide you with a Question, Golden answers. I need you to write "WikiSearch(term)" to invoke the API for assistance in answering the Question, where "term" is the search term you want to look up to obtain the Golden answers. Here are some examples:

Question: Where are sunbeam microwaves made?
Golden answers: ['Florida']
Output: WikiSearch(Sunbeam microwaves manufacturing location)

Question: What type of car does Michael Weston drive?
Golden answers: ['Wishcraft']
Output: WikiSearch(Michael Weston car)

Question: What is Nina Dobrev nationality?
Golden answers: ['Bulgaria']
Output: WikiSearch(Nina Dobrev nationality)

Question: What religion are people in Russia?
Golden answers: ['Islam', 'Russian Orthodox Church']
Output: WikiSearch(Religion in Russia)

Question: {question}
Golden answers: {answers}
Output:

Figure 9: Prompt used for Question Answering to generate tool APIs.

I will provide you with a Question, Golden answers. I need you to write "QA(question)" to invoke the API for assistance in answering the Question, where "question" is the question you want to ask to obtain the Golden answers. Here are some examples:

Question: The army held Rome for a brief time, but was then forced to retreat to the city of Perusia (modern Perugia, ?
Golden answers: ['Italy']
Output: QA(Which country is Perusia, or modern Perugia, located in ?)

Question: Winners of the festivals «Chervona Ruta» (Ukraine), «Pearls of the Season» (Ukraine), «Boards» (Moscow), «Woodstock» ( ?
Golden answers: ['Poland']
Output: QA(Where is the Woodstock festival held?)

Question: It is native to the Alps and the Pyrenees Mountains of Europe (Spain, France, Italy, Switzerland, Austria and ?
Golden answers: ['Germany']
Output: QA(Which country is mentioned as being native to the Alps and the Pyrenees Mountains alongside Spain, France, Italy, Switzerland, and Austria ?)

Question: Heorhiy Kyrylovych Tkachenko (May 5, 1898 in Hlushkovo, Kursk region of the Russian Empire – 1993 in Kiev, ?
Golden answers: ['Ukraine']
Output: QA(Where did Heorhiy Kyrylovych Tkachenko die ?)

Question: {question}
Golden answers: {answers}
Output:

Figure 10: Prompt used for LAMA to generate tool APIs.

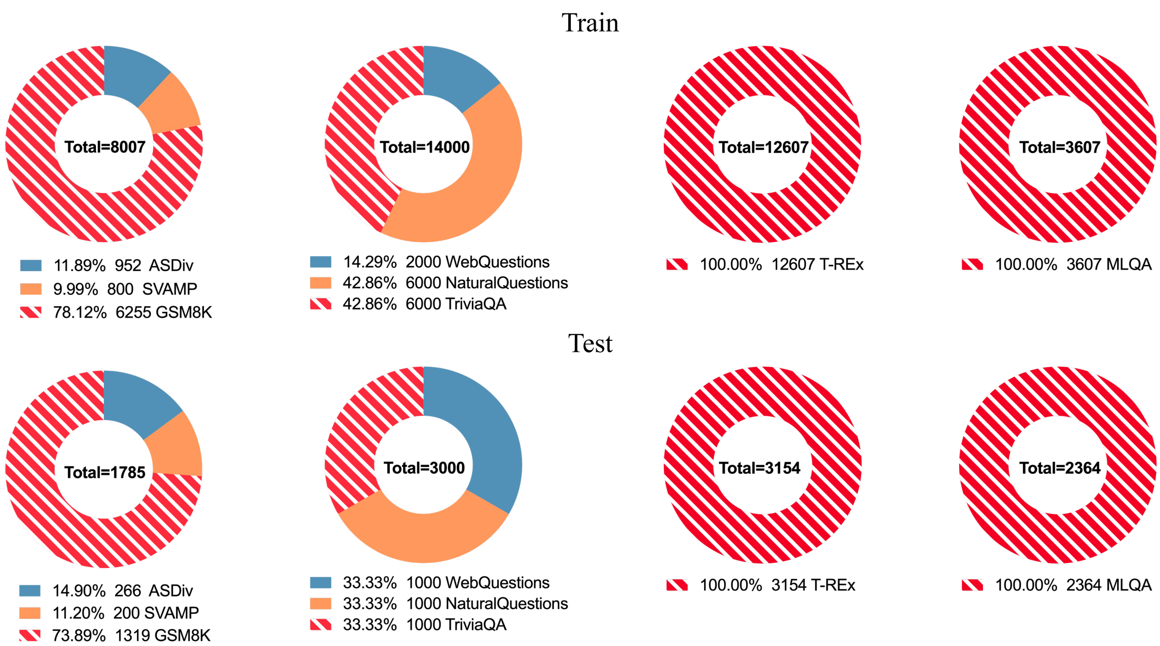


Figure 11: Data distribution for each task.

```
Given a math problem, please solve it and you can use a calculator for help.
Here are some examples:

Input: Mrs. Hilt has 50 cents. A pencil costs 5 cents. How many pencils can she buy with the money she has?
Output: calculator(50/5)

Input: James writes a 3-page letter to 2 different friends twice a week.  How many pages does he write a year?
Output: calculator(3*2*2*52)

Input: Weng earns $12 an hour for babysitting. Yesterday, she just did 50 minutes of babysitting. How much did she earn?
Output: calculator((12/60)*50)

Input: Betty is saving money for a new wallet which costs $100. Betty has only half of the money she needs. Her parents decided to give her $15 for that purpose, and her grandparents twice as much as her parents. How much more money does Betty need to buy the wallet?
Output: calculator(100-100/2-15*2-15)

Input:{question}
Output:
```

Figure 12: Prompt used for Math Reasoning to generate candidate responses.

```
Given a question, please answer it and you can use a WikiSearch for help.
Here are some examples:

Input: Who has scored most runs in test cricket
Output: WikiSearch(most runs scorer in test cricket)

Input: How did Jock die in Dallas?
Output: WikiSearch(Jock Ewing death)

Input: Where are the Netherlands on a world map?
Output: WikiSearch(Location of the Netherlands on world map)

Input: What is Nina Dobrev nationality?
Output: WikiSearch(Nina Dobrev nationality)

Input: {question}
Output:
```

Figure 13: Prompt used for Question Answering to generate candidate responses.

```
Given a question, please answer it and you can use a QA model for help.
Here are some examples:

Input: The City Council divide itself into ?
Output: QA(What did the City Council divide itself into?)

Input: Arcos de Canasí is a small town in the east of the La Habana Province of ?
Output: QA(Which country is Arcos de Canasí located in?)

Input: The steel is named after Damascus, the capital city of ?
Output: QA(What is the capital city of Syria?)

Input: Winners of the festivals «Chervona Ruta» (Ukraine), «Pearls of the Season» (Ukraine), «Boards»
(Moscow), «Woodstock» (?
Output: QA(Where is the Woodstock festival held?)

Input: {question}
Output:
```

Figure 14: Prompt used for LAMA to generate candidate responses.

Given a context, please answer the question in English and you can use a translator for help.
Here are some examples:

Input:
Context: Over the next decade, she went on more than 40 field missions, meeting with refugees and internally displaced persons in over 30 countries. In 2002, when asked what she hoped to accomplish, she stated, “Awareness of the plight of these people. I think they should be commended for what they have survived, not looked down upon.” To that end, her 2001–02 field visits were chronicled in her book Notes from My Travels, which was published in October 2003 in conjunction with the release of her humanitarian drama Beyond Borders.
Question: cô ấy đã thực hiện bao nhiêu nhiệm vụ trong hơn 10 năm?
Output: translator(cô ấy đã thực hiện bao nhiêu nhiệm vụ trong hơn 10 năm?)

Input:
Context: John Canfield Spencer (January 8, 1788 – May 17, 1855) was an American lawyer, politician, judge and United States Cabinet secretary in the administration of President John Tyler.
Question: John Canfield Spencer làm việc với Tổng thống nào?
Output: translator(John Canfield Spencer làm việc với Tổng thống nào? )

Input:
Context: The story follows the adventures of Garde pilot Nagate Tanikaze, who lived in the underground layer of Sidonia since birth and was raised by his grandfather. Never having met anyone else, he trains himself in an old Guardian pilot simulator every day, eventually mastering it. After his grandfather's death, he emerges to the surface and is selected as a Guardian pilot, just as Sidonia is once again threatened by the Gauna.
Question: Ông được xét chọn là gì sau khi qua đời?
Output: translator(Ông được xét chọn là gì sau khi qua đời?)

Input:
Context: Emma Goldman: A Documentary History of the American Years, Volume 1 – Made for America, 1890–1901. Berkeley: University of California Press, 2003. ISBN 0-520-08670-8.
Question: Phim tài liệu dựa trên khoảng thời gian nào?
Output: translator(Phim tài liệu dựa trên khoảng thời gian nào?)

Input:
Context: {context}
Question: {question}
Output:

Figure 15: Prompt used for Multilingual QA to generate candidate responses.