Making Language Models Better Tool Learners with Execution Feedback

Shuofei Qiao♠, Honghao Gui♠, Huajun Chen♠♡, Ningyu Zhang♠*
♠ Zhejiang University ♡ Donghai Laboratory
{shuofei,guihonghao,huajunsir,zhangningyu}@zju.edu.cn

arXiv:2305.13068v1 [cs.CL] 22 May 2023

Abstract

Tools serve as pivotal interfaces that enable humans to understand and reshape the world. With the advent of foundational models, AI systems can utilize tools to expand their capabilities and interact with the world. Existing tool learning methodologies, encompassing supervised fine-tuning and prompt engineering approaches, often induce language models to utilize tools indiscriminately, as complex problems 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 language model to selectively use tools by decreasing the model's dependency on tools while enhancing the performance[1].

1 Introduction

Tools serve as vital interfaces that allow humans to comprehend and reshape the world. A defining characteristic that distinguishes humans from other animals is our remarkable capacity to create and utilize tools (Orban and Caruana, 2014; Osiurak and Reynaud, 2020). The recent rapid advancement of foundation models (Brown et al., 2020; Ouyang et al., 2022; Chowdhery et al., 2022) enables them to possess surprising capabilities of generation (Brown et al., 2020; Touvron et al., 2023), reasoning (Qiao et al., 2022; Huang and Chang, 2022), and decision-making (Yao et al., 2023; Shen et al., 2023), making it practical for AI machines to utilize tools effectively (Paranjape et al., 2023; Lu et al., 2023).

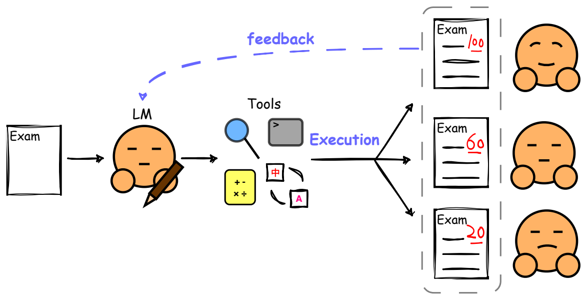


Figure 1: Language model learns to use tools from execution feedback.

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., 2023). Toolformer (Schick et al., 2023) teaches LLMs themselves to use tools by fine-tuning them in a self-supervised way. ART (Paranjape et al., 2023) leverages task-specific demonstrations to prompt frozen LLMs to generate intermediate reasoning steps and tool use. Other works (Shen et al., 2023; Ge et al., 2023; Lu et al., 2023) employ LLMs as a hub for human-tool interaction, responsible for orchestrating the deployment and usage of tools and consolidating the results for user interpretation.

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,

* Corresponding Author.
[1]Code and datasets will be available in https://github.com/zjunlp/trice.

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 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 tool execution, thereby learning when and how to use tools effectively. Specifically, we first build a dataset that helps discern when tool usage is necessary for LLMs and when it is not. We evaluate the untrained model by having it answer questions directly, considering correct responses as instances where tools aren't needed, and incorrect responses as instances where tools are required for assistance. Given the lack of gold labels, we utilize ChatGPT (OpenAI, 2022) to automatically generate tool usage APIs for data requiring tools, thereby effectively eliminating the need for time-consuming manual annotation. For data that does not require tools, we directly use the model's correct answer as the label. Then, we introduce a two-stage training strategy to teach the model when to use tools: 1) **Behavior Cloning**. We conduct supervised fine-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 tool execution feedback, guiding the model to selectively use tools to avoid error propagation. We utilize RRHF (Yuan et al., 2023), a simple and effective reinforcement learning algorithm, as the basic backbone.

We train and evaluate TRICE on two mathematical reasoning datasets. Experimental results and further analyses demonstrate that TRICE successfully instructs the model to judiciously use tools, simultaneously reducing its reliance on tools and enhancing 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 achieve superior performance on two benchmark datasets compared to previous fine-tuning-based tool learning methods.
- Through extensive empirical analysis, we demonstrate that our TRICE framework can guide the model in judicious tool usage, thereby reducing the model's dependency on tools and improving the precision of tool use.

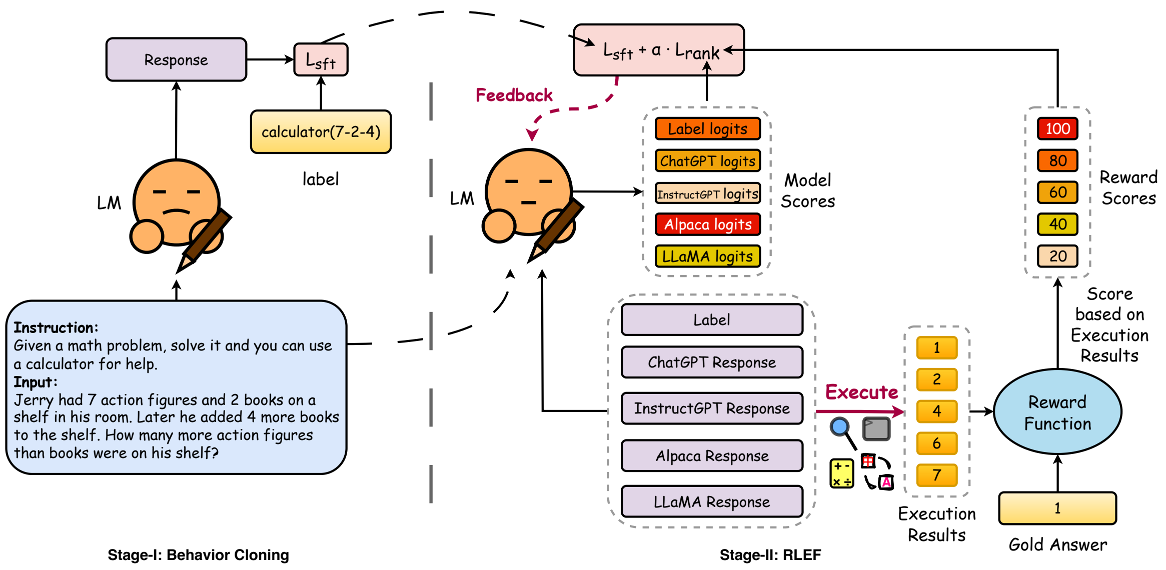
2 Related Work

Tool Learning. Though possessing remarkable capabilities of generation (Brown et al., 2020; Touvron et al., 2023), reasoning (Qiao et al., 2022; Huang and Chang, 2022), and decision-making (Yao et al., 2023; Shen et al., 2023; Lu et al., 2023), LLMs still struggle in many basic aspects such as arithmetic calculation (Patel et al., 2021), knowledge querying (Komeili et al., 2022; Ji et al., 2022), etc. where much smaller and simpler tools may precisely excel. Under this circumstance, a new paradigm, called Tool Learning (Qin et al., 2023), 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., 2023), 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., 2023).

Despite their success, tool-augmented approaches tend to force LMs to use tools mindlessly regardless of whether they actually need to lend 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 focus on the tool-augmented learning paradigm and try to **make LMs better tool learners by teaching them to use tools selectively instead of blindly**.

Learning from Feedback. An intuitive training approach of tool learning is to fit LMs on examples with human-labeled tools directly (Torabi et al., 2018; Li et al., 2022) which is so-called Behavior Cloning (Bain and Sammut, 1995). However, this method is time-consuming and labor-intensive. Moreover, it is impractical to explicitly annotate every possible scenario (Codevilla et al., 2019) and LMs can only imitate human-labeled behavior but are difficult to generalize to new scenarios. It is worth noting that humans generally have the ability to correct and reflect on their own behavior from



Response
L_{sft}
calculator(7-2-4)
label
LM
Instruction:
Given a math problem, solve it and you can use a calculator for help.
Input:
Jerry had 7 action figures and 2 books on a shelf in his room. Later he added 4 more books to the shelf. How many more action figures than books were on his shelf?
Stage-I: Behavior Cloning
$L_{sft} + a \cdot L_{rank}$
Feedback
LM
Label logits
ChatGPT logits
InstructGPT logits
Alpaca logits
LLaMA logits
Model Scores
100
80
60
40
20
Reward Scores
Score based on Execution Results
Label
ChatGPT Response
InstructGPT Response
Alpaca Response
LLaMA Response
Execute
1
2
4
6
7
Execution Results
Reward Function
1
Gold Answer
Stage-II: RLEF

Figure 2: The overview of our proposed framework TRICE. In stage-I (Behavior Cloning), We conduct supervised fine-tuning on the dataset to let the model imitate the tool-using behavior. In stage-II (RLEF), We further reinforce the model based on RRHF with tool execution feedback, guiding the model to selectively use tools.

trial and error (Allen et al., 2019). Intuitively, feedbacks from the environments or humans enable LMs 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). RLHF (Christiano et al., 2017; Ouyang et al., 2022) applies a state-of-the-art RL algorithm, Proximal Policy Optimization (PPO) (Schulman et al., 2017), to align LMs with human feedback. Nevertheless, PPO has shown sensitivity to hyperparameters and its conventional implementation requires a minimum of four models, making it memory-intensive and challenging to train. RAINIER (Liu et al., 2022) reinforces knowledge introspection for commonsense question answering with a fixed QA model providing feedback. OpenAGI (Ge et al., 2023) proposes RL with task feedback for complex task-solving with various external expert models.

Compared to previous feedback strategies, we introduce **RLEF** for tool learning which reinforces the LMs by leveraging the execution result of tools. We further apply a much simpler RL framework in contrast to RLHF, called **RRHF** (Yuan et al., 2023), where the model learns to align with human preferences by scoring the responses produced by various sampling policies and utilizing ranking loss.

3 Methodology

Problem Overview. We mainly focus on the mathematical reasoning task, with each training instance in the format of $x = (s, q, t, a)$, where s is the specialized instruction of each task, q is the question, t is the tool API and a is the gold answer. Following an instruction-following paradigm, the complete input of the LLM is as follows:

$$\text{input} = \{\text{instruction}\}\{\text{question}\} \qquad (1)$$

As for the output, when LLM deems that no tool is necessary, it outputs the answer a. Conversely, if the model identifies the need for a tool, it outputs the tool API t, which encompasses the specific type of tool and its corresponding input:

$$\text{output} = \qquad (2)$$
$$\begin{cases} \text{answer} & \mathit{use\_tool} = \text{false} \\ \text{tool\_name(tool\_input)} & \mathit{use\_tool} = \text{true} \end{cases}$$

Given the problem, the **main challenges** lie in 1) determining the LLM when to or not to harness tools for help as well as 2) how to impart the ability to the model to make selective use of tools. For the **former**, we allow the untrained model to 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

Algorithm 1 Two-Stage Training

```
Input: initial model $\theta$, train dataset $\mathcal{D}_{\text{tool}}$, response generation
    model set $\mathcal{M}$
 1: $\theta_{\text{clone}} \leftarrow$ Optimize $\theta$ with Eq.3 from $\mathcal{D}_{\text{tool}}$.   ▷ Section 3.1
 2: $res$, $scores \leftarrow$ Collect responses and calculate reward
    scores with $\mathcal{M}, \mathcal{D}_{\text{tool}}$.
 3: $\theta_{\text{RLEF}} \leftarrow$ RLEF($\mathcal{D}_{\text{tool}}$, $\theta_{\text{clone}}$, $res$, $scores$)   ▷ Section 3.2
 4:
 5: procedure RLEF($\mathcal{D}_{\text{tool}}$, $\theta$, $res$, $scores$)
 6:     $\theta_{\text{old}} \leftarrow \theta$
 7:     for iterations = 1, 2, ... do
 8:         Sample a minibatch from $\mathcal{D}_{\text{tool}}$.
 9:         for step = 1, 2, ..., |minibatch| do
10:             Calculate $\mathcal{L}_{\text{rank}}$ and $\mathcal{L}_{\text{sft}}$ with $res$, $scores$,
    $\mathcal{D}_{\text{tool}}$ based on Eq.5&6
11:             Calculate $\mathcal{L}_{\text{policy}}$ based on Eq.7.
12:             Optimize $\theta$ with $\mathcal{L}_{\text{policy}}$ for one step.
13:         end for
14:         $\theta_{\text{old}} \leftarrow \theta$
15:     end for
16:     return $\theta$
17: end procedure
Output: $\theta_{\text{RLEF}}$
```

use **Behavior Cloning** to teach the model how to invoke tools. Building upon this foundation, in the second stage, we leverage **RLEF** to train the model in selective tool usage. The overview of our method is illustrated in Figure 2.

Data Construction. The construction of the dataset is based on the assumption that when LLM generates incorrect answers, it indicates the need for assistance from tools. For the initial training set $\mathcal{D}_{\text{init}} = \{(q, a)\}_{i=1}^{|\mathcal{D}_{\text{init}}|}$, we first utilize the pre-trained LLM without fine-tuning to generate final answers. Since we do not have gold labels of tool APIs, we employ ChatGPT (OpenAI, 2022) to generate $t = \text{tool\_name}(\text{tool\_input})$ as pseudo-labels under a few-shot prompting manner for questions where the generated answers are incorrect. As for questions with correct answers, we directly set $t = \text{None}$, indicating that no tool API is required. We devise specific instructions s tailored to the mathematical reasoning task. In the end, we obtain the dataset $\mathcal{D}_{\text{tool}} = \{(s, q, t, a)\}_{i=1}^{|\mathcal{D}_{\text{tool}}|}$ as we desire.

Training. As shown in Figure 2, based on $\mathcal{D}_{tool}$, 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 of ChatGPT by fine-tuning it on $\mathcal{D}_{\text{tool}}$ in a sequence-to-sequence manner. This empowers our model with the preliminary functionality of tool API calls. II) **Reinforcement Learning with Execution Feedback** (§3.2). We continue to train our model obtained in stage I with reinforcement learning in order to enhance the accuracy of tool utilization. Specially, we leverage RRHF (Yuan et al., 2023) as our backbone framework and replace its reward score with the effect of tool execution. The entire training procedure is outlined in Algorithm 1.

3.1 Training Stage I: Behavior Cloning

During the behavior cloning stage, we aim to enable the LM to master the schema of tool API calls and develop preliminary skills in selectively utilizing tools. As the existing dataset for tool learning is limited and the accuracy of tool usage in this stage may not be high, we leverage $\mathcal{D}_{\text{tool}}$ contained pseudo-labels for tool API calls generated by ChatGPT to fine-tune the model.

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

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

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

3.2 Training Stage II: RLEF

In stage II, we continue to optimize θ_{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.

Policy Loss. For each question q, we have k different responses $y_i, 1 \leq i \leq k$ generated by LLMs like ChatGPT, LLaMA, or even provided by human experts. We apply a reward function to score each y_i with $R(a, y_i) = r_i$ where a is the gold answer of question q. Aiming to align with scores $\{r_i\}_k$, we then score each y_i with our model:

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

where p_i is the conditional log probability of y_i and $||y_i||$ is the length-normalized factor.

To facilitate the LM 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 calls structure, we reintroduce the supervised fine-tuning loss:

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

Finally, the overall policy loss is defined as follows:

$$\mathcal{L}_{\text{policy}} = \alpha \cdot \mathcal{L}_{\text{rank}} + \mathcal{L}_{\text{sft}}, \quad (7)$$

where α is a hyperparameter that determines the proportion of the rank loss.

Reward Function. The purpose of the reward function is to give each y_i a r_i and rank them. The main criterion for scoring is derived from the feedback from the execution result of y_i. Concretely, we set $\mathcal{S}$ as the maximum reward score. We execute y_i to obtain the predicted answer a_i^* which is to say that if y_i contains a tool API call, we invoke the tool and use the return result as the predicted answer a_i^*, and if y_i is indeed the answer generated by the LLM (this means y_i does not need a tool for help), then the generated answer is considered as the final prediction a_i^*. The answer to the mathematical reasoning task is exactly a number, so we compare a_i^* and the gold answer a by evaluating their proximity:

$$e_i = |a - a_i^*|. \quad (8)$$

Afterward, we equally divide $\mathcal{S}$ into k values:

$$\{r_i\}_k = \{\frac{\mathcal{S}}{k}, \frac{2\mathcal{S}}{k}, \ldots, \frac{(k-1)\mathcal{S}}{k}, \mathcal{S}\}. \quad (9)$$

We view the output o regulated in $\mathcal{D}_{\text{tool}}$ as the pseudo-human-expert response and assign it with the maximum score $\mathcal{S}$. The remaining responses are scored based on e_i, where a smaller e_i corresponds to a higher r_i, and in the case of equal e_i, random ordering is applied to ensure fairness.

Obviously, the process of obtaining responses and reward scores can be decoupled from the training process which greatly simplifies the training process. Therefore, in this stage, we only need to train a single model.

4 Experiments

4.1 Experimental Settings

Models. We apply Alpaca-7B (Taori et al., 2023) (instruction fine-tuned LLaMA) trained in the LoRA (Hu et al., 2022) manner as the backbone LLM and continue to train the LoRA during both I and II training stages. In the response generation part, we sample responses from 4 different models, e.g. ChatGPT, InstuctGPT, Alpaca-LoRA-7B, LLaMA-LoRA-7B, and the output o regulated in $\mathcal{D}_{\text{tool}}$ as the pseudo-human-expert response. For ChatGPT and InstructGPT, we prompt them with instructions and few-shot examples, and for Alpaca-LoRA-7B and LLaMA-LoRA-7B, we finetune them on $\mathcal{D}_{\text{tool}}$ for a few epochs in order to equip them with initial abilities for answer and tool generation.

Datasets. We mainly train and test our model on two mathematical reasoning datasets, ASDiv (Miao et al., 2020) and SVAMP (Patel et al., 2021). ASDiv covers a diverse set of English math word problem corpus including various arithmetic operations. To facilitate the use of the calculator, we select its basic arithmetic operations part which contains 1,218 instances. We randomly split them into 952 instances of the training set and 266 instances of the test set. SVAMP is another more challenging math word problem dataset containing 1,000 instances. We randomly select 856 as the training set and 144 as the test set. We incorporate the training sets of them as the original dataset $\mathcal{D}_{\text{init}}$ and test our model on their test sets.

Baselines. Throughout the remainder of this section, we mainly compare the following models:

- **Toolformer** (Schick et al., 2023), an approach to teaching the LLM to utilize tools on its own through also fine-tuning, where the model is trained to use tools blindly.

- **Alpaca-LoRA-7B**, Alpaca-7B (Taori et al., 2023) trained in the LoRA (Hu et al., 2022) manner which is indeed the model of ours without training.

- **TRICE-I**, model trained only on the behavior cloning stage to show the imitation ability of our model on tool learning.

- **TRICE-II**, model trained only on the RLEF stage to reflect the results of training with execution feedback alone.

- **TRICE-All**, model trained on both two stages.

Model	ASDiv	SVAMP	Avg.
Toolformer (Schick et al., 2023)	40.4	29.4	34.9
Alpaca-LoRA-7B	32.8	14.4	23.6
TRICE-I	69.2	31.4	50.3
TRICE-II	44.0	29.0	36.5
TRICE-All	**70.6**	**35.9**	**53.3**

Table 1: Performance comparison of different baselines. We show the Accuracy score on ASDiv (Miao et al., 2020) and SVAMP (Patel et al., 2021).

Setups. For the mathematical reasoning task, we choose the Calculator tool to spy on the effectiveness of our method. During both stage I and II, we train LoRA with $\mathrm{lora\_r} = 8$, $\mathrm{lora\_alpha} = 16$, and $\mathrm{lora\_target\_modules} = Q, V$. In stage I, we have a 32 batch size per GPU. In stage II, we have a 4 batch size per GPU. In both stages, we apply gradient accumulation at 8 steps and the model max length at 512 tokens. We first warm up the learning rate to 2e-5 and decay to 0 linearly. Since sampling responses and training are separated, our whole training procedure only needs to load one model, largely reducing the training costs. We train on 3 24GB Nvidia 3090 GPUs, typically costing 0.5-1 hours for stage I and 1-2 hours for stage II.

4.2 Main Results

Table 1 presents the performance of our method compared to various baselines on two mathematical reasoning datasets. We can observe that just training on $\mathcal{D}_{tool}$ with stage I (**TRICE-I**), our method significantly outperforms Toolformer by an average of 15.4% on accuracy, not to mention the untrained Alpaca-LoRA-7B which performs even worse. Building upon this, after further training in stage II (**TRICE-All**), our method achieve an additional 3.0% improvement. This indicates that our approach equips the model with a solid ability to learn and utilize tools. However, the results obtained solely from training in stage II (**TRICE-II**) are not satisfactory, indicating that the initial tool generation ability endowed to the model in stage I is essential to a stabler training with reinforcement learning. Another intriguing observation is that our method exhibits a more significant enhancement on the more challenging SVAMP dataset (4.5%) compared to ASDiv (1.4%) from TRICE-I to TRICE-All. This finding suggests that tool learning with execution feedback may have the potential to generalize the problem-solving capabilities of the model for complex tasks.

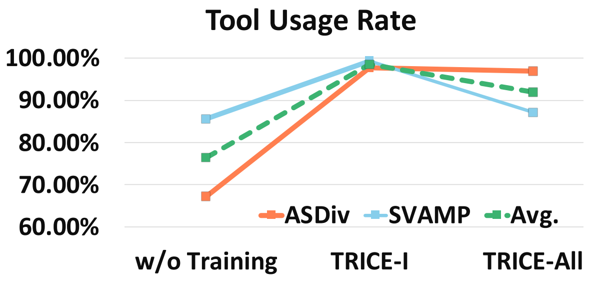


Figure 3: Comparison of tool usage rate among different training stages on the test dataset. In the w/o Training stage, we consider a need for using tools when the model reaches a wrong answer.

4.3 Analysis

Selective Tool Usage. Figure 3 illustrates the proportion of tool usage by our model at each training stage. It is noticed that after the training stage I, the reliance of our model on tools has significantly deepened to an average of 98.5% compared to the original 76.4% without training. This indicates that due to the imbalanced data distribution regarding the presence or absence of tools in the training set, supervised fine-tuning tends to make the model overly dependent on tools. However, after undergoing training in stage II, our model not only shows improvement in performance but also reduces its dependency on tools to 92.0%. This demonstrates that our model can learn selective tool usage through execution feedback.

Case Study. To further analyze the role of each training stage, we provide several cases of responses from different stages in Table 2. As shown in Case 1&2, stage I equips the model with a certain level of tool generation capability, but it may not excel in making optimal decisions about the input of tools. Stage II alleviates this limitation and enhances the effectiveness of our model in learning to utilize tools. Additionally, Case 3 confirms once again that our proposed method, TRICE, enables the model to use tools selectively. Furthermore, as illustrated in Case 4, our model still exhibits certain flaws leading to errors in tool usage. We speculate that this could be attributed to two factors: 1) our backbone model has a scale of 7B, which may limit its performance in mathematical reasoning tasks; 2) training incorporating reinforcement learning introduces certain instabilities that could contribute to the occurrence of errors in tool usage.

Id	Question	Gold Answer	After Stage I	After Stage I&II
1	While playing at the arcade, Kaleb won 8 tickets playing 'whack a mole' and 7 tickets playing 'skee ball'. If he was trying to buy candy that cost 5 tickets a piece, how many could he buy?	3	calculator(15/5) ✔	calculator((8+7)/5) ✔
2	Winter is almost here and most animals are migrating to warmer countries. There were 89 bird families living near the mountain. If 60 bird families flew away for winter. How many more bird families flew away for the winter than those that stayed behind?	31	calculator(89-60) ✗	calculator(60-29) ✔
3	Rachel had to complete 8 pages of math homework. If she had to complete 6 more pages of reading homework than math homework How many pages of reading homework did she have to complete?	14	calculator(8-6) ✗	She had to complete 14 pages of reading homework. ✔
4	Jerry had 7 action figures and 2 books on a shelf in his room. Later he added 4 more books to the shelf. How many more action figures than books were on his shelf?	1	3. ✗	calculator(7-2) ✗

Table 2: Case analysis on the response of after training in stage I and stage I&II.

5 Discussion

In tool learning, LLMs manipulate tools and respond to users conditioned on a variety of knowledge sources. The most crucial aspect is for the model to have its own understanding of when it needs to use tools and when it doesn't need to use them. One particularly challenging issue in this context is the problem of knowledge conflicts (Qin et al., 2023) 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. LMs 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 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.

However, our current method is unable to learn the usage of multiple tools or tool compositions. In the future, more sophisticated trial-and-error processes and feedback mechanisms will be necessary to teach the LMs to better utilize tools and even create new tools.

6 Conclusion

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 language model better tool learners with execution feedback. We also create a dataset to assist the model in learning when and how to use tools properly. Through comprehensive experiments on two mathematical reasoning datasets, we have shown that our method can achieve better performance compared to finetune-based tool learning approaches. Extensive analysis illustrates that our TRICE can selectively use tools by reducing the dependency of the model on tools while improving the accuracy of tool usage.

Limitations

Given our limited computational resources, we only conduct experiments with a single tool. However, in the future, we plan to incorporate additional tasks and tools, and further investigate the use of compositional tools to enhance our approach. To cater to various tasks, we intend to explore more complex reward mechanisms that integrate execution feedback from a range of tasks across various scenarios.

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.

Michael Bain and Claude Sammut. 1995. A framework for behavioural cloning. In *Machine Intelligence 15, Intelligent Agents [St. Catherine's College, Oxford, UK, July 1995]*, pages 103–129. Oxford University Press.

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*.

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.

Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4299–4307.

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.

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.

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.

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.

Jie Huang and Kevin Chen-Chuan Chang. 2022. Towards reasoning in large language models: A survey. *arXiv preprint arXiv:2212.10403*.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. 2022. Survey of hallucination in natural language generation. *CoRR*, abs/2202.03629.

Mojtaba Komeili, Kurt Shuster, and Jason Weston. 2022. Internet-augmented dialogue generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 8460–8478. 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*.

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/`.

Guy A. Orban and Fausto Caruana. 2014. The neural basis of human tool use. *Frontiers in Psychology*, 5.

François Osiurak and Emanuelle Reynaud. 2020. The elephant in the room: What matters cognitively in cumulative technological culture. *Behavioral and Brain Sciences*, 43:e156.

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*.

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

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.

Shuofei Qiao, Yixin Ou, Ningyu Zhang, Xiang Chen, Yunzhi Yao, Shumin Deng, Chuanqi Tan, Fei Huang, and Huajun Chen. 2022. Reasoning with language model prompting: A survey. *arXiv preprint arXiv:2212.09597*.

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. *CoRR*, abs/2304.08354.

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. *Nat.*, 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.

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.

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.