Confucius: Iterative Tool Learning from Introspection Feedback by Easy-to-Difficult Curriculum

Shen Gao†, Zhengliang Shi†, Minghang Zhu†, Bowen Fang†, Xin Xin†, Pengjie Ren†, Zhumin Chen†, Jun Ma†, Zhaochun Ren†

1Shandong University, Qingdao, China
2Leiden University, Leiden, The Netherlands
{shengao, xinxin, renpengjie, chenzhumin, majun}@sdu.edu.cn, bwn.fang@gmail.com, z.ren@liacs.leidenuniv.nl

Abstract
Augmenting large language models (LLMs) with external tools has emerged as a promising approach to extending the capability of LLMs. Although some works employ open-source LLMs for the tool learning task, most of them are trained in a controlled environment in which LLMs only learn to execute the human-provided tools. However, selecting proper tools from the large toolset is also a crucial ability for the tool learning model to be applied in real-world applications. Existing methods usually directly employ self-instruction methods to train the model, which ignores differences in tool complexity. In this paper, we propose the Confucius, a novel tool learning framework to train LLM to use complicated tools in real-world scenarios, which contains two main phases: (1) We first propose a multi-stage learning method to teach the LLM to use various tools from an easy-to-difficult curriculum; (2) thenceforth, we propose the Iterative Self-Instruct from Introspective Feedback (ISIF) to dynamically construct the dataset to improve the ability to use the complicated tool. Extensive experiments conducted on both controlled and real-world settings demonstrate the superiority of our tool learning framework in real-world application scenarios compared to both tuning-free (e.g., ChatGPT, Claude) and tuning-based baselines (e.g., GPT4Tools).

Introduction
The task of tool learning aims to unleash the power of large language models (LLMs) to effectively interact with various tools to accomplish complex tasks (Qin et al. 2023b). By integrating LLM with APIs, we can greatly expand their utility and empower LLM to serve as an efficient intermediary between users and the vast ecosystem of applications (Qin et al. 2023a; Jin et al. 2023; Park et al. 2023). Existing tool learning approaches can be divided into two categories: tuning-free and tuning-based methods. The former ones leverage the proprietary LLMs, such as ChatGPT or GPT-4, to interact with various tools to solve complex tasks. These methods prompt the proprietary LLMs with demonstrations of tool usage. However, the only way for the proprietary LLMs to access the user-defined tools is the prompt (Li et al. 2023). Thus, the limited context length of LLMs restricts the application of massive tools. In contrast, the tuning-based methods fine-tune open-source LLMs to memorize and understand external tools by explicitly training on elaborate datasets (Li et al. 2023; Schick et al. 2023). The majority of these methods (Qin et al. 2023b) first use Self-Instruct technique to collect tool-use data from proprietary LLMs and then fine-tune an open-source model. Since the training data only contains a limited range of tools, most turn-based methods lack the capability to generalize to unseen tools (tools outside the training data). In Table 1, we list several cutting-edge tool-use LLMs.

As shown in Figure 1, most existing methods directly provide a minimal essential toolset to LLMs without redundant tools. However, when adapting to real-world applications, LLMs typically face a large toolset that contains various tools across different tasks. Thus, how to teach LLMs to select an appropriate tool from the candidates becomes the first challenge.

Intuitively, the difficulty of using different tools is not the same. Some tools are used in different ways in

---

†Equal contribution.
†Corresponding author.
Copyright © 2024, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Figure 1: Comparison between the existing tuning-based tool learning methods and our proposed Confucius. Instead of using a pre-constructed dataset, we propose an iterative data construction framework with multi-stage learning to train the tool-use model effectively.
different scenarios, so more attention should be paid to using such complicated tools during model training. For example, the Google Map tool for exploring the surrounding places requires only the current coordinates when traveling. However, when planning a commute route to work, more additional information, such as the starting and ending points, as well as the preference, should be specified to execute this tool. To better interact with such complicated tools, it is necessary to train to use the tool in many different scenarios. Thus, the second challenge is knowing which tool is more complicated and how to improve the ability to use these tools.

In this paper, we propose the Confucius, a tool-learning framework to train LLM to use complicated tools in real-world scenarios. Confucius contains two main phases: (1) To tackle the first challenge, we first propose a multi-stage learning method to teach the LLM to use various tools from an easy-to-difficult curriculum; (2) We propose an Iterative Self-instruct from Introspective Feedback (ISIF) technique to dynamically construct the dataset to improve the ability to use the complicated tool.

Specifically, the multi-stage learning method involves three training stages: (1) warm-up training, (2) in-category training, and (3) cross-category training. In the warm-up training stage, we feed the model with the required minimal toolset and aim to teach the model to schedule and execute the tool correctly. Next, in the in-category training stage, we aim at teaching the model to learn to select the proper tools among related candidates. Finally, we employ the cross-category training stage, which trains the model in the real-world application setting, where the candidate toolset is constructed by a tool retriever that conducts semantics matching between the user query and tool demonstrations. After being trained under our multi-stage training, an LLM becomes more straightforward and applied to real application scenarios.

Since the usage of some tools varies significantly in different scenarios, more extensive training should be conducted to fully master them. Hence, we introduce the Iterative Self-instruct from Introspective Feedback (ISIF), to customize the tool-use training dataset iteratively, which includes two phases: instance generation and updates with introspective feedback. In the instance generation phase, we start with a diverse toolset and an initial set of tool-use instance data. Then the demonstration of tools is taken as prompts to ChatGPT to generate diverse queries and then answer these queries through compositional reasoning with various tools. Since intricate tools require more training data for LLM to fully master, the pre-created dataset is out of sync with the up-to-date LLM.

Therefore, we take the introspection of the LLM for using tools as the feedback and use this feedback to guide the dataset update phase. Specifically, in this phase, we aim to generate more tool-use instances related to the intricate tools that are usually misused by the current LLM. Compared to previous works, ISIF facilitates the LLM to master more intricate tools and prevents it from overfitting to a subset of simple tools. To verify the effectiveness of Confucius, we conduct extensive experiments on controlled and real-world settings using a large-scale tool-use dataset. Experimental results show that our proposed Confucius outperforms the tuning-free (e.g., ChatGPT and Claude) and tuning-based baselines (e.g., GPT4Tools) in terms of four aspects, which demonstrates the effectiveness of our tool-learning framework in the real-world application scenario.

To sum up, our contributions can be summarized as follows: (i) We propose the Confucius, a tool-learning framework, teaching the LLM to use complicated tools in real-world scenarios. (ii) We propose a multi-stage learning method to improve the ability of multiple tool selection from a large-scale toolset. (iii) We propose an iterative training strategy ISIF to improve the performance of using intricate tools by dynamically updating the dataset according to the model introspection. (iv) Experiments on both seen and unseen toolsets show that the Confucius effectively accesses various tools and achieves comparable and even better performance to proprietary LLMs (e.g., ChatGPT).

### Related Work

#### Tuning-free Tool Learning

The tuning-free methods leverage the inherent in-context learning capability of LLMs, where the demonstrations of tools are taken as input to prompt LLMs to use various
tools (Paranjape et al. 2023; Yao et al. 2023; Kim, Baldi, and McAleer 2023). For example, Shen et al. and Wu et al. integrate existing models hosted by Huggingface as the toolset to handle various downstream tasks, such as object detection and question answering. The other studies, such as Chameleon (Lu et al. 2023), utilize the GPT-4 as the base model to devise long-term plans and automatically execute different tools, which further demonstrates the potential ability to tackle more complex tasks including table-based reasoning. However, there are two main drawbacks of tuning-free methods: (1) For data security reasons, not all the applications (Gao et al. 2019) can transmit tool and user data to LLM service providers (Gudibande et al. 2023). And it restricts the use of proprietary LLMs in such applications. (2) Due to the limitation of the input length, the prompt cannot accommodate massive tools, thus constraining the model to utilize only a few tools to tackle the task.

Tuning-based Tool Learning

The tuning-based tool learning methods directly fine-tune the parameter of language models on the tool-use dataset (Wang et al. 2023), typically constructed by prompting proprietary LLMs to use specific tools, e.g., search (Qin et al. 2023a; Nakano et al. 2023; Shi et al. 2023a), calculation (Hao et al. 2023; Gao et al. 2023) and translation (Schick et al. 2023). The advantage of these methods is that they can be easily deployed in a self-host environment. However, fine-tuning language models on the constructed datasets typically introduces generalization problems (Tang et al. 2023), where performance degradation is usually observed when dealing with new tools which have not been seen during training. To improve the generalization of tool-learning models for new tools, some works (Qin et al. 2023b; Xu et al. 2023b; Patil et al. 2023) devote to constructing datasets across diverse toosets and increasing the diversity of training datasets, which present a promising solution to enhancing the performance of unseen tools. However, they ignore the complexity distinctions between various tools which potentially leads to some complex tools with intricate usage are not well-learned, hurting the generalization of the model.

Task Formulation

We formulate the Confucius as a tool learning framework to train an open-source large language model $M$ to master various tools in real-world scenarios. In detail, we start off a large tooset $T^*$ with various tools and construct a tool-use dataset $D$. Following (Li et al. 2023), we divide the tools into different categories (ten in our work), such as navigation and smart home. Each instance $d$ in the dataset consists of the query $q = \{q_1, q_2, \ldots, q_{|q|}\}$, response $y = \{y_1, y_2, \ldots, y_{|y|}\}$ and ground truth tools $T = \{\tau_1, \tau_2, \ldots, \tau_{|\tau|}\}$ for answering the query $q$. Meanwhile, we denote the relevant toolset as $T_r$, which derives from the same category as $T$, and has no overlap with the $T$. We then train the target model $M$ to decompose the original query $q$ into sub-tasks via compositional reasoning and schedule the appropriate tools step by step to generate the response $y$. During inference, we first retrieve a subset $\tilde{T} = \{\tau_1, \tau_2, \ldots, \tau_{|\tilde{T}|}\}$ from tooset $T^*$ for the given query $q$ which contains the candidate tools to generate the response.

Figure 2 shows the overall architecture of our proposed Confucius operates the two main phrases iteratively: (1) Given a tool-use dataset, we propose a multi-stage learning method to finetune the LLM in an easy-to-difficult curriculum paradigm; (2) After tuning the LLM on the dataset, we dynamically update such dataset according to the
confused set caused by the finetuned LLM. Continuously, we employ the updated dataset to finetune the LLM and conduct the training paradigm in an iterative manner.

Multi-stage Learning

In real-world applications, the tool-use model should select appropriate tools from the retrieved tools and schedule them correctly (a.k.a., difficult mode), instead of directly using human given candidate toolset (a.k.a., easy mode). Similar to human learning procedures, tool learning models can benefit from an easy-to-difficult curriculum during model training (Xu et al. 2020). Therefore, we propose a multi-stage learning method that consists of warm-up training, in-category training, and cross-category training, teaching the LLM to master various tools in the real-world setup.

Warm-Up Training Stage

In the initial warm-up stage, for each query $q$, we provide the LLM $M$ with the ground truth toolset $T$ to generate the response $y$, which can be formulated as:

$$P(y|q, T) = \prod_{t=1}^{|y|} P_M(y_t|y_{(t)}<1), q, T).$$

(1)

Then we employ the log-likelihood objective $L_{\text{warm-up}}$ to train the $M$ to decompose the query into tool-use sub-tasks and generate the response $y$ by scheduling multiple tools:

$$L_{\text{warm-up}} = -\log P(y|q, T).$$

(2)

In-Category Training Stage

To gradually adapt the model to the real-world setting, for each query, we integrate a mixture of the ground truth toolset $T$ and the relevant toolset $T_r$ which are randomly selected from the same category as the $T$. The category of a tool indicates the using scenario, for example, planning a route and searching for a place are tools of the map navigation category. In this setting, in addition to arranging the appropriate tools for the model, it is also necessary to first select the proper tools from candidates $T_r \cup T$. And the LLM generates the response on the condition of the query $q$ and mixed toolset $T_r \cup T$, which can be formulated as:

$$L_{\text{in}} = -\sum_{t=1}^{|y|} P_M(y_t|y_{(t)}<1), q, T, T_r).$$

(3)

Cross-Category Training Stage

Since the tools used to answer the query should be retrieved automatically rather than manually provided in real-world applications, we introduce the cross-category training method, which explicitly empowers LLM to select appropriate tools in the realistic setting. Specifically, we first construct a tool retriever model based on the dual-encoder framework (Reimers and Gurevych 2019) to retrieve the candidate toolset $\tilde{T}$, which encodes the user query $q$ and the tool demonstrations into dense representations and computes the cosine similarity as relevance. Intuitively, the retrieved toolset $\tilde{T}$ contains the hard negative (redundant)

### Here are some usage examples:

{Tool list}

### You can use the following APIs:

{in-context learning examples}

Each query involves four tools at least.

Table 2: Prompt used for generating new tool-use instances.

In order to conduct more targeted training for intricate tools, we propose the Iterative Self-instruct from Introspection (ISIF), a dynamic method for constructing training data, which updates the training dataset continuously based on model knowledge of tools. As shown in Figure 2, ISIF iterates the two phases, i.e., instance generation and update with introspective feedback.

Initial Dataset Construction

We start off building a tool store which contains 110 common-used tools and usage instances, which are constructed manually as the seed instance pool. Specifically, each instance consists of a concrete query, and the answer follows the chain-of-thought format, where at least four tools are involved to encourage the complexity of our dataset. As shown in Figure 2, for each step, we first sample 5–7 tools from the tool store, denoted as $T^*$. Then, the demonstrations of sampled tools paired with corresponding instances are taken as input, prompting ChatGPT to reason the potential compositional relationship of tools and generate diverse instances. In Table 2, we show an example of the prompt, which consists of three main parts: (1) task instruction; (2) candidate tools list; (3) tool-use instance demonstrations, which consist of a user query and a ground truth response. More details for statistics and comparison with other related datasets are given in Table 3.

Updates with Introspective Feedback

Since the instances generated via self-instruct may be uncontrolled without any training targeted guidance (Xu et al. 2023a; Bian et al. 2023), we propose to construct a prompt to guide the instance generation phase according to the training procedure. Given a query containing $n$ tokens $q = \{q_1, ...q_n\}$, we first retrieve a toolset $T^*$, and then
provide the LLM $\mathcal{M}$ with $T^*$ to generate the response. The generation perplexity $h$ of the target response which contains $m$ tokens $y = \{y_1, ..., y_m\}$ conditioned on $q$ and $T^*$ can be factorized as follows:

$$h = \frac{1}{\sum_{i=1}^{\|y\|} P_M(y_i | y_{<i}, q, T^*)}.$$  \hspace{1cm} (5)

where the $P_M(y_i | q, T^*)$ is the generation probability, formulated as:

$$P_M(y_i | q, T^*) = \prod_{i=1}^{\|y\|} P_M(y_i | y_{<i}, q, T^*).$$  \hspace{1cm} (6)

Since perplexity $h$ represents the degree of generation uncertainty, samples with higher perplexity $h$ requires further training in subsequent training.

And next, we filter the generated instances $D = \{d_1, d_2, ..., d_{|D|}\}$ with high perplexity instances $D^{\sigma}$ which should be trained more. These filtered instances $D^{\sigma}$ are then utilized in the self-instruct prompt to generate more similar tool-use instances for further training. The instance generate method is the same as the initial dataset construction, only the tool-use demonstration in the prompt is replaced by the filtered instance $d^{\sigma} \in D^{\sigma}$. Specifically, for each update, we generate $\sigma$ percent new instances of the original dataset, which is guided by the filtered instances, and we append these instances to the original dataset $D$. The updated dataset will be used to train the model in the next epoch, and this process is conducted iteratively for each epoch.

### Experimental Setup

#### Dataset

To verify the effectiveness of Confucius, we employ two test sets: Seen and Unseen toolset, and each of them consists of 2,000 instances with ten tools. All the tools in the Seen toolset have been used in the training set, while the tools in the Unseen toolset have not been used when training.

#### Evaluation Metrics

Following Li et al. (2023) and Tang et al. (2023), we evaluate from four aspects: tool selection, parameter correctness, compositional reasoning, and interaction fluency. Tool Selection evaluates the capability to select correct tools from the candidate toolset. Since multiple tools are involved for each task, we employ the listwise metric, which calculates the NDCG (Järvelin and Kekäläinen 2002) score between the tools in the generated response and the ground-truth response. Parameter Correctness measures the correctness of the input parameter type for the tools, which validates whether the LLM response conforms to the schema of the tool’s interface. Compositional Reasoning first identifies the topological order of tools in generated and ground-truth response and calculates the ROUGE-L score of two sequences of tools. Interaction Fluency employs the average of ROUGE-1, ROUGE-2, and ROUGE-L scores as the similarity between the generated and ground-truth responses, which indicates whether the model comprehends tools execution results and delivers fluent responses. We also employ human evaluation where three master students are invited to evaluate 50 randomly sampled cases with a three-scale score in the following aspects: (1) Executability: whether multiple tools are invoked in the correct order to generate the response (2) Fluency: whether the generated response is human-like and fluent (Shi et al. 2023b).

### Baselines

We compare our Confucius with tuning-based baselines, including ToolFormer-6B (Schick et al. 2023), ToolLLaMA-7B (Qin et al. 2023b) and GPT4Tools (Yang et al. 2023a). We also compare with tuning-free methods, including the proprietary LLMs (e.g., ChatGPT and GPT-3) and open-source models, which interact with various tools by context learning. For a fair comparison, all the tuning-based methods use the dataset as ours, and all the baselines are provided with the same candidate toolset, which is retrieved by our dense tool retriever. We use the top-10 tools with the highest cosine similarity as the candidate toolset.

### Implementation Details

In our work, we take the LLaMA-7B\footnote{https://huggingface.co/huggingface/llama-7B} as our base model. We vary the percent $\sigma$ in $\{10, 15, 20, 25, 30\}$ and find that the $\sigma = 20$ achieves the best performance. We optimize the model using deepspeed ZeRO strategy (Rasley et al. 2020) with the learning rate of $5e^{-5}$ and the weight decay coefficient of 0.01. The training of our model can be done within 20 hours with 4 NVIDIA A100-PCIE-80GB GPUs.

### Experimental Results

#### Overall Performance

Table 4 shows the experimental results of all baselines. We can find that our proposed Confucius achieves the best performance in seen and unseen toolsets in terms of all metrics. Compared with ChatGPT, Confucius gets 88.61 (4.99 absolute improvement) in terms of tool selection in the seen test set, which suggests Confucius shows great potential for selecting proper tools correctly. We observe that the Confucius reaches 87.99 and 63.65 in the compositional reasoning aspect with the seen and unseen toolset, which has a significant improvement compared with the tuning-based baseline, and it also outperforms the advanced proprietary LLM, i.e., ChatGPT. This result highlights the Confucius
The potential reason for the performance drop when generalizing from seen to unseen toolset is that the LLM can acquire robust tool-use skills from our iterative training strategy ISIF.

Since the candidate toolset of all the baselines is retrieved automatically, we also verify the effectiveness of our tool retriever. The recall@10 of our tool retriever achieves 93.49 and 91.20 in seen and unseen toolsets. It shows that the tool retriever with dual-encoder architecture is qualified to find proper tools closely aligned with the ground truth.

**Generalization for Different Base Models**

To further explore the robustness of our proposed Confucius, we finetune the other two open-source LLMs (LLaMA-7B and Vicuna-7B) using Confucius with the same setting. As Table 4 shows, compared with the corresponding tuning-free versions, both two models trained using Confucius outperform their base model by a large margin, which demonstrates the generalization of our framework.

**Human Evaluation**

We conduct human evaluation, and Table 5 summarizes the results. We find that the Confucius consistently outperforms the best tuning-based baselines in two aspects, such as pushing Executability to 2.73 (0.90 absolute improvement) with seen toolset. Moreover, we also observe that the Confucius achieves comparable or even better results with ChatGPT, indicating the effectiveness of our framework. The average Kappa statistics for two evaluation metrics are 0.762 and 0.732, illustrating agreement among the human evaluators.

### Table 4: Comparing with baselines on seen and unseen test datasets.

<table>
<thead>
<tr>
<th>Method</th>
<th>Seen Toolset</th>
<th></th>
<th>Unseen Toolset</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tool</td>
<td>Selection</td>
<td>Parameter</td>
<td>Correctness</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Claude</td>
<td>75.30</td>
<td>56.00</td>
<td>74.18</td>
<td>55.81</td>
</tr>
<tr>
<td>ChatGPT</td>
<td>83.62</td>
<td>67.31</td>
<td>82.59</td>
<td>65.65</td>
</tr>
<tr>
<td>Text-davinci-003</td>
<td>79.13</td>
<td>59.71</td>
<td>78.66</td>
<td>60.57</td>
</tr>
<tr>
<td>ChatGLM-6B (Du et al. 2022)</td>
<td>41.74</td>
<td>30.81</td>
<td>41.11</td>
<td>43.62</td>
</tr>
<tr>
<td>ChatGLM2-6B (Du et al. 2022)</td>
<td>24.34</td>
<td>18.33</td>
<td>24.43</td>
<td>39.32</td>
</tr>
<tr>
<td>Llama-7B (Pati et al. 2023)</td>
<td>67.52</td>
<td>53.81</td>
<td>65.33</td>
<td>47.71</td>
</tr>
<tr>
<td>Llama2-7B (Pati et al. 2023)</td>
<td>70.93</td>
<td>54.52</td>
<td>67.84</td>
<td>58.49</td>
</tr>
<tr>
<td>Vicuna-7B (Chiang et al. 2023)</td>
<td>66.79</td>
<td>51.19</td>
<td>65.60</td>
<td>58.72</td>
</tr>
<tr>
<td>Vicuna-13B (Chiang et al. 2023)</td>
<td>72.26</td>
<td>57.51</td>
<td>71.17</td>
<td>61.75</td>
</tr>
<tr>
<td>GPT4Tools (Yang et al. 2023a)</td>
<td>75.20</td>
<td>58.52</td>
<td>74.07</td>
<td>64.99</td>
</tr>
<tr>
<td>ToolLLaMA (Qin et al. 2023b)</td>
<td>62.92</td>
<td>44.92</td>
<td>62.99</td>
<td>62.26</td>
</tr>
<tr>
<td>Toolformer (Schick et al. 2023)</td>
<td>30.81</td>
<td>20.48</td>
<td>29.65</td>
<td>38.75</td>
</tr>
<tr>
<td>Ours (LLaMA-7B)</td>
<td><strong>88.61</strong></td>
<td><strong>77.72</strong></td>
<td><strong>89.99</strong></td>
<td><strong>79.09</strong></td>
</tr>
</tbody>
</table>

**Ablation Study**

<table>
<thead>
<tr>
<th>Method</th>
<th>Tool Selection</th>
<th>Parameter Correctness</th>
<th>Compositional Reasoning</th>
<th>Interaction Fluency</th>
</tr>
</thead>
<tbody>
<tr>
<td>- w/o $L_{warm-up}$</td>
<td>86.11</td>
<td>70.16</td>
<td>74.86</td>
<td>57.30</td>
</tr>
<tr>
<td>- w/o $L_{ln}$</td>
<td>85.73</td>
<td>70.21</td>
<td>75.21</td>
<td>57.23</td>
</tr>
<tr>
<td>- w/o $L_{cross}$</td>
<td>83.49</td>
<td>67.73</td>
<td>70.73</td>
<td>53.51</td>
</tr>
<tr>
<td>- w/o ISIF</td>
<td>83.52</td>
<td>67.40</td>
<td>73.05</td>
<td>54.72</td>
</tr>
</tbody>
</table>

**Effectiveness Analysis**

<table>
<thead>
<tr>
<th>Method</th>
<th>Tool Selection</th>
<th>Parameter Correctness</th>
<th>Compositional Reasoning</th>
<th>Interaction Fluency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours (LLaMA2-7B)</td>
<td>89.40</td>
<td>77.81</td>
<td>84.38</td>
<td>75.22</td>
</tr>
<tr>
<td>Ours (Vicuna-7B)</td>
<td>87.30</td>
<td>73.50</td>
<td>83.02</td>
<td>76.04</td>
</tr>
</tbody>
</table>

**Table 5: Human evaluation on seen and unseen test datasets.**

The potential reason for the performance drop when generalizing from seen to unseen toolset is that the LLM can acquire robust tool-use skills from our iterative training strategy ISIF.

Since the candidate toolset of all the baselines is retrieved automatically, we also verify the effectiveness of our tool retriever. The recall@10 of our tool retriever achieves 93.49 and 91.20 in seen and unseen toolsets. It shows that the tool retriever with dual-encoder architecture is qualified to find proper tools closely aligned with the ground truth.

**Generalization for Different Base Models**

To further explore the robustness of our proposed Confucius, we finetune the other two open-source LLMs (LLaMA-7B and Vicuna-7B) using Confucius with the same setting. As Table 4 shows, compared with the corresponding tuning-free versions, both two models trained using Confucius outperform their base model by a large margin, which demonstrates the generalization of our framework.

**Human Evaluation**

We conduct human evaluation, and Table 5 summarizes the results. We find that the Confucius consistently outperforms the best tuning-based baselines in two aspects, such as pushing Executability to 2.73 (0.90 absolute improvement) with seen toolset. Moreover, we also observe that the Confucius achieves comparable or even better results with ChatGPT, indicating the effectiveness of our framework. The average Kappa statistics for two evaluation metrics are 0.762 and 0.732, illustrating agreement among the human evaluators.
Analysis of Multi-stage Training

In Table 4, we compare Confucius with several ablation variants, including the model (w/o $\mathcal{L}_{\text{warm-up}}$, $\mathcal{L}_{\text{in}}$, and $\mathcal{L}_{\text{cross}}$) which removes each training stage in the multi-stage training method. We can find that all the variant models suffer performance degradation, which demonstrates the effectiveness of our proposed multi-stage training methods in Confucius. We observe that the model w/o $\mathcal{L}_{\text{cross}}$ has the largest performance drop compared with the other two variant models in terms of the tool selection score. This phenomenon demonstrates the necessity of constructing a candidate toolset similar to the real-world setting to improve the tool selection ability of LLM.

Analysis of ISIF

We explore whether the performance improvement is simply caused by the expansion of the training set so as to further verify the necessity of the introspective feedback in ISIF. For a fair comparison, different from ISIF, which updates the dataset according to the perplexity instance, we random sample some instances as the prompt of self-instruct to generate new instances. And then, the updated dataset is used to train the LLaMA, which is the same base model as our Confucius. Figure 3 shows the performance of the models trained on different sizes of initial datasets. We find that our proposed ISIF performs constantly better than the model directly trained by vanilla self-instruct in each size of the dataset, which verifies the effectiveness of dynamically updating the dataset guided by the introspective feedback.

Conclusion

In this paper, we propose the Confucius, a novel tool learning framework to teach LLM to master various tools, which consists of two main steps: (1) multi-stage learning and (2) iterative self-instruct from introspective feedback (ISIF). Concretely, we fine-tune the LLM with three learning stages from an easy-to-difficult curriculum, i.e., warm-up, in-category, and cross-category stages. Since the usage of some tools varies in different scenarios, which requires more training to fully understand the usage, we introduce the ISIF to iteratively update the tool-use training dataset based on the model introspection. Extensive experiments on seen and unseen toolsets demonstrate that Confucius can boost the tool-learning performance of LLM compared with both tuning-based and tuning-free baselines, including ChatGPT.
Acknowledgements
This work was supported by the National Natural Science Foundation of China (T2293773).

References


 Wang, Z.; Cai, S.; Liu, A.; Ma, X.; and Liang, Y. 2023. Describe, Explain, Plan and Select: Interactive Planning


