Enhancing Tool Retrieval with Iterative Feedback from Large Language Models

Anonymous ACL submission

Abstract

Tool learning aims to enhance and expand large language models' (LLMs) capabilities with external tools, which has gained significant attention recently. Current methods have shown that LLMs can effectively handle a certain amount of tools through in-context learning or fine-tuning. However, in real-world scenarios, the number of tools is typically extensive and irregularly updated, emphasizing the necessity for a dedicated tool retrieval component. Tool retrieval is nontrivial due to the following challenges: 1) complex user instructions and tool descriptions; 2) misalignment between tool retrieval and tool usage models. To address the above issues, we propose to enhance tool retrieval with iterative feedback from the large language model. Specifically, we prompt the tool usage model, i.e., the LLM, to provide feedback for the tool retriever model in multi-round, which could progressively improve the tool retriever's understanding of instructions and tools and reduce the gap between the two standalone components. We build a unified and comprehensive benchmark to evaluate tool retrieval models. The extensive experiments indicate that our proposed approach achieves advanced performance in both in-domain evaluation and out-of-domain evaluation.

1 Introduction

Large language models (LLMs) have demonstrated remarkable success in language-related tasks and are considered a potential pathway to achieving artificial general intelligence (Zhao et al., 2023). However, despite their powerful capabilities, LLMs are still limited in many aspects, such as knowledge update and mathematical reasoning. A promising way to overcome these limitations is to empower LLMs with external tools, known as tool learning. Tool learning not only enhances LLMs' performance on existing tasks but also allows them to tackle tasks that were previously beyond their reach. Besides,

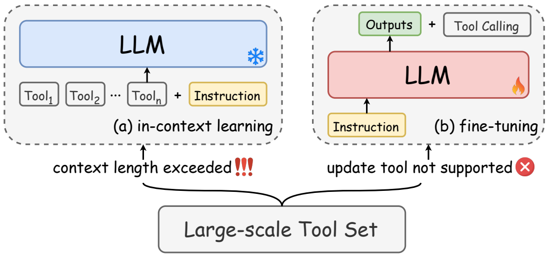

Figure 1: Illustration of two tool-learning approaches in LLMs: (a) in-context learning and (b) fine-tuning. The challenges posed by the extensive and frequently updated tools require the external tool retrieval component.

the ability to use tools is a crucial hallmark on the path to advanced intelligence.

Existing tool learning methods have preliminarily demonstrated that LLMs could effectively utilize specific tools to complete corresponding tasks. They either leverage LLMs' in-context learning ability to facilitate tool usage with tool descriptions (Shen et al., 2023) or fine-tune LLMs to integrate tool learning capabilities into parameters, e.g., Toolformer (Schick et al., 2023). However, as illustrated in Figure 1, existing methods still face significant challenges in real-world scenarios due to the following reasons. 1) The number of tools is usually vast, making it impossible for LLMs to handle them all with the limited input length of in-context learning. 2) Tools would frequently and irregularly update, rendering finetuning-based approaches costly and impractical. Therefore, a tool retrieval component, which aims to select appropriate tools from a large-scale tool set, is essential for LLMs.

Despite the practicality and necessity, tool retrieval has been inadequately studied. Some approaches have adopted traditional document retrieval methods to retrieve tools for LLMs (Li et al., 2023; Patil et al., 2023; Qin et al., 2023b). However, we argue that they overlook the unique challenges of tool retrieval for LLMs: 1) Complex user in-

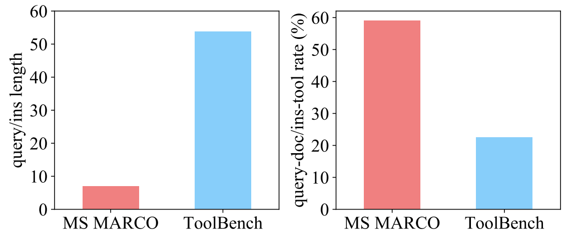

Figure 2: Comparison between the document retrieval and tool retrieval datasets. Tool retrieval presents more challenges due to the complex instructions (in the left figure) and the lower reputation rate (in the right figure).

structions and tool descriptions. As illustrated in Figure 2, compared with document retrieval, user instructions are usually ambiguous and complex, and the reputation rate between instructions and corresponding tool descriptions is much lower. Unfortunately, the retriever model is typically limited in its capacities because of the efficiency requirements, which makes tool retrieval more difficult and challenging. 2) Misalignment between tool retrieval and tool usage models. Previous approaches deploy the tool retriever separately from the downstream tool-usage model, which hinders the LLM from knowing which tools are really useful from the tool-usage perspective. Thus, it will result in a tool recognition gap between the tool retriever and tool usage model, degrading the tool-use performance further.

To address the above issues, we propose to enhance tool retrieval with iterative feedback. Our motivation is to utilize the LLM to enhance the comprehension ability of the tool retriever and bridge the gap between the two independent models. At each iteration, we conduct a feedback generation process by asking the LLM to provide feedback step-by-step, conditioned on the user instruction and retrieved tools from the retriever. The LLM will first comprehend the instruction and tool functionalities thoroughly, and then assess the effectiveness of those retrieved tools. According to the assessment, the LLM will refine the user instruction to improve the tool retrieval process. The refined instruction will substitute previous user instruction and be used to retrieve a new list of tools from the tool set. In the next iteration, the new candidate tool list will be fed into the LLM for a new round of LLMs' feedback. During this iterative process, the tool retriever is expected to provide more appropriate tools for the tool-usage model. In this manner, the comprehension capability and tool preference of LLMs could be progressively incorporated into the retriever, and thus the tool retriever's performance could be continuously enhanced. We build a comprehensive tool retrieval benchmark, named TR-bench. The benchmark takes into account real-world practices with updated tools, and therefore encompasses both in-domain and out-of-domain settings. The experimental results show our approach achieves the best performance among the current methods with both in-domain and out-of-domain settings.

The key contributions are summarized:

- We identify the importance of tool retrieval in tool learning and present the distinct challenges of tool retrieval.
- We propose to enhance tool retrieval with iterative feedback from the LLM. By leveraging iterative feedback, the tool retriever model gets continual improvements, ultimately reducing the misalignment between them.
- We build a comprehensive tool retrieval benchmark with in-domain and out-of-domain settings, which will also aid future tool retrieval research. The extensive experiments illustrate our approach has demonstrated superior performance.

2 Related Work

2.1 Tool Learning in LLMs

Tool learning aims to equip LLMs with external tools to enhance and expand their capabilities (Schick et al., 2023; Tang et al., 2023; Yao et al., 2023; Qin et al., 2023a; Shen et al., 2023). Generally, existing tool learning methods could be categorized into in-context learning and fine-tuning approaches. The former approach encourages LLMs to use tools with descriptions, documentation, or demonstrations, while the latter one trains the parameters of LLMs using specially created tool-use datasets. However, no matter whether the in-context learning or fine-tuning approach encounters severe challenges in real-world scenarios, where the candidate tools are extensive and frequently updated. Therefore, it is crucial to equip LLMs with a tool retrieval component to select appropriate tools from a large-scale tool set. Recent works have proposed a stopgap measure through traditional document retrieval methods (Li et al., 2023; Patil et al., 2023; Qin et al., 2023b). In this work, we aim to develop a specialized method for retrieving tools.

2.2 Document Retrieval

Early popular document retrieval methods rely on sparse retrieval that calculates the relevance of documents to a query based on the frequency of query terms in each document, e.g., BM25 (Robertson and Zaragoza, 2009). With the development of language models (Devlin et al., 2019), the dense retrieval (Zhao et al., 2024; Mitra and Craswell, 2017) paradigm has gained considerable attention in the research community. By encoding queries and documents into high-dimensional vector representations and computing their relevance scores through inner product calculations, the paradigm can capture semantic relationships between queries and documents, thereby enhancing retrieval performance (Karpukhin et al., 2020). However, tool retrieval presents unique challenges, rendering traditional document retrieval methods suboptimal. We address these challenges by harnessing LLMs' feedback to iteratively refine the tool retrieval process.

3 Preliminaries

3.1 Task Definition

Given a user's instruction, tool retrieval aims to select a small number of tools, which could aid the LLM in answering the instruction, from a large-scale tool set. Formally, we define the user instruction as q and the tool set as $D = \{d_1, d_2, ..., d_N\}$, where d_i represents the description of each tool and N is the total number of tools. The retriever model R needs to measure the relevance $R(q, d_i)$ between the instruction q and each tool description d_i, and return K tools, denoted as $D = \{d_1, d_2, ..., d_K\}$.

3.2 Dense Retriever

Dense retriever usually leverages the encoder-based LLM to encode the user instruction q and a tool description d into dense embeddings $E(q)$ and $E(d)$, respectively. Then, it could measure the relevance between q and d by calculating the similarity score between these two embeddings, denoted as $R(q, d) = sim(E(q), E(d))$.

Dense retriever is trained via the contrast learning objective, which is designed to minimize the distance between the instruction embedding and embeddings of positive tools (the instruction's ground-truth tools) while maximizing the distance between the instruction embedding and embeddings of negative tools. The objective can be formulated as follows,

$$\mathcal{L} = -\frac{1}{B}\sum_{i=1}^{B} log \frac{e^{R(q_i,d_i^+)}}{e^{R(q_i,d_i^+)} + \sum_j e^{R(q_i,d_{ij}^-))}}, \quad (1)$$

where B denotes the batch size, d_i^+ denotes the positive tool, and d_{ij}^- represents the j-th negative tool to the instruction q_i.

However, due to the efficiency requirements, dense retrieval utilizes a dual-encoder architecture, which has limited ability to understand instructions. In this study, our goal is to improve the tool retrieval process with the feedback from the tool-usage model, i.e., the LLM.

4 Methodology

4.1 Overview

Recent studies have found that LLMs show a great capability in acting as a critic (Zheng et al., 2023) and could provide comprehensive feedback to improve performance across a range of tasks (Madaan et al., 2023; Asai et al., 2023). Inspired by those observations, we propose an innovative framework that leverages the LLM's feedback to improve the tool retrieval process iteratively.

As illustrated in Figure 3, at each iteration, the LLM will provide feedback on the current-turn retrieval results. Specifically, the LLM will first comprehend the user instruction and tool functionalities thoroughly. Then, it will assess the effectiveness of those retrieved tools for handling the instruction. Based on the assessment, the LLM could provide a refinement to the retrieval model, refining the user instruction if necessary. To ensure that the retriever model is aware of the iteration round, we conduct an iteration-aware feedback training process to adapt the retriever model with continuously refined user instructions.

4.2 Feedback Generation

Assuming at the iteration step t, given the refined instruction q^t, we could utilize retriever model R to retrieve a list of top-K tools $\{d_1^t, ..., d_K^t\}$. We then conduct a three-step feedback generation process by feeding those retrieved tools and associated tool descriptions into the LLM as follows.

Comprehension. Firstly, the LLM is prompted to give comprehension on both the given instruction and retrieved tools. The prompt provided to LLM includes two parts: (1) summarize the abstract user goals by ignoring detailed entity information in the

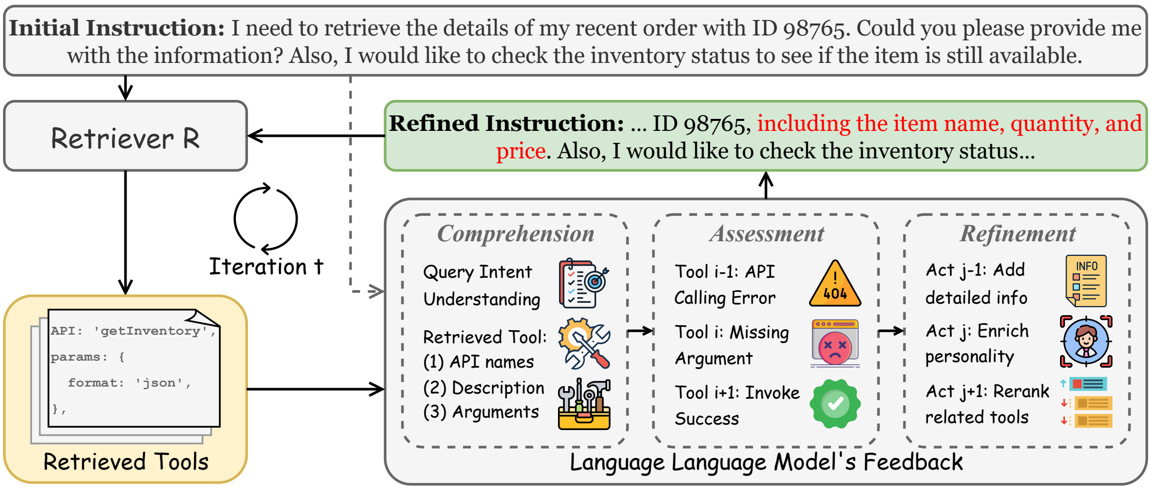

Figure 3: Illustration of our proposed iterative tool retrieval method. At each iteration, the LLM follows a three-step feedback generation process, which includes comprehension, assessment, and refinement, to improve the instruction.

given instruction; (2) understand the functionalities of retrieved tools, focusing on the category, name, description, input and output parameters of given tools. This step can be formulated as,

$$F_C = LLM(P_C, q^t, \{d_1^t, ..., d_K^t\}), \quad (2)$$

where F_C denotes LLM's comprehension output and P_C denotes the prompt provided to LLM.

Assessment. The LLM will assess the effectiveness of retrieved tools for handling the instruction based on its comprehension of the user's itent and tool functionalities. The assessment is conducted from two perspectives: 1) identify which of the user's goals could and could not be solved by the retrieved tools with corresponding reasons; and 2) analyze whether the ranked order of retrieved tools corresponds with their significance in addressing the user's intent with specific reasons. The step can be formulated as,

$$F_A = LLM(P_A, q^t, \{d_1^t, ..., d_K^t\}, F_C), \quad (3)$$

where F_A denotes the LLM's assessment output.

Refinement. Lastly, the LLM will refine user instruction based on its assessment. Specifically, we ask the LLM to determine whether the refinement is necessary based on the two following questions: 1) Whether all the user's goals have been solved by currently retrieved tools, 2) and whether all existing appropriate tools are given the highest ranking priorities by the retriever. If one of the answers is not "yes", we prompt the LLM to provide a potential refinement for retrieval improvement. Otherwise, the LLM will directly return a special token "N/A" without conducting any refinement.

The feedback from the LLM is finalized made on the current user instruction q^t Specifically, we prompt the LLM to generate refined instruction with enriched information in two dimensions: 1) more detailed and personalized content about those user's intent which have not been solved by current tools, helping the retriever explore other relevant tools; (2) more scenario-specific tool-usage information about existing appropriate tools, helping the retriever give higher ranking priority to those tools. This step can be formulated as,

$$F_R = LLM(P_R, q^{t-1}, \{d_1^{t-1}, ..., d_K^{t-1}\}, F_A), \quad (4)$$

where P_R is the corresponding prompt and F_R denotes LLM's refinement output, i.e., the new refined instruction q^{t+1}.

4.3 Iteration-Aware Feedback Training

We concatenate a special token "Iteration t" in front of the instruction, where t is the instruction's iteration step (e.g., "Iteration $t-1$" for q^{t-1} and "Iteration t" for q^t). The final training loss is formulated as the sum of losses in each iteration as follows,

$$\mathcal{L}_{feedback} = \sum_{t=1}^{T} \alpha^t \mathcal{L}(q^t), \quad (5)$$

where α^t is a balancing factor. In this way, the LLM's comprehensive knowledge of the user requirements could be injected into the retriever through those refined instructions. Besides, with the aid of iteration-aware tokens and joint-training manner, the retriever could maintain a balance between newly learned knowledge and previously acquired knowledge.

We also employ the hard negative sampling in training. Concretely, for each given instruction, we randomly sample an incorrect tool from the retrieved top-K tool list. The high similarity scores of those tools indicate that they are prone to be mistaken as correct tools by the retriever. In feedback training, we utilize those tool-instruction pairs as hard negative samples.

Then, the loss function for each iteration could be calculated as,

$$\mathcal{L} = -\frac{1}{B}\sum_{i=1}^{B} log \frac{e^{R(q_i,d_i^+)}}{e^{R(q_i,d_i^+)} + \sum_{j\neq i} e^{R(q_i,d_{ij}^-))} + \sum e^{M(q_i,d_{ij}^H)}}, \tag{6}$$

where d_{ij}^H denotes the hard negative sample. By distinguishing the subtle differences in the tool descriptions, the retriever could achieve a deeper understanding of the tool functionalities and their relation with user instructions.

4.4 Inference

At the time of inference, the feedback generation process keeps working while the feedback training process ceased. The retriever will update the candidate tool list based on the refined user instruction from LLM's feedback iteratively, until output the final retrieved tools.

Concretely, assume that we have obtained a retriever R after the feedback training. For each initial test instruction q_{test}^0, we add a special token "Iteration 0" in front of the instruction. Then we use the trained retriever R to retrieve an initial tool list D_{test}^0, containing K candidate tools $\{d_1, d_2, ..., d_K\}$. The retrieved D_{test}^0 and q_{test}^0 will be fed to the LLM for feedback generation, including instruction refinement, as discussed in Section 4.2. After obtaining the refined instruction q_{test}^1, we add a token "Iteration 1" to it and then input it to R for the next-round tool retrieval. Then, we can get an updated tool list D_{test}^1 for a new round of feedback generation. As such, we could obtain a final tool list D_{test}^T after T iterations.

5 Experiments

5.1 Setup

Datasets and evaluation. To assess the tool retrieval performance of models, we conduct a tool retrieval benchmark, referred to as **TR-bench**, based on three datasets, including ToolBench (Qin et al., 2023b), T-Eval (Chen et al., 2023), and UltraTools (Huang et al., 2024). To address real-world requirements, we conduct evaluations in both *in-domain* and *out-of-domain* settings. Specifically, the training set is from ToolBench, while the test set of ToolBench is employed for in-domain evaluation, and the test sets from T-Eval and UltraTools are used for out-of-domain evaluation. The statistics of TR-bench are summarized in Table 1.

	scenarios	# instructions	# tool set
Training Set	ToolBench-I1	86,643	-
	ToolBench-I2	84,270	-
	ToolBench-I3	25,044	-
	ToolBench-All	195,937	-
In-domain Evaluation	ToolBench-I1	796	10,439
	ToolBench-I2	573	13,142
	ToolBench-I3	218	1,605
	ToolBench-All	1,587	13,954
Out-of-domain Evaluation	Teval	553	50
	UltraTools	1,000	498

Table 1: Statistics of the TR-bench, which is conducted from ToolBench (Qin et al., 2023b), T-Eval (Chen et al., 2023), and UltraTools (Huang et al., 2024).

Following previous work (Qin et al., 2023b), we adopt the Normalized Discounted Cumulative Gain (NDCG) (Järvelin and Kekäläinen, 2002), an ideal ranking metric for tool retrieval since it could evaluate the relevance and quality of retrieved tool candidates according to their ranked orders. We report NDCG@m ($m = 1, 3, 5, 10$), where m refers to top-m ranked search results for evaluation.

Baselines. We compare our method against representative retrieval methods. 1) BM25 (Robertson and Zaragoza, 2009): the classical sparse retrieval method; 2) Ada Embedding: the closed-sourced OpenAI's text-embedding-ada-002 model[1]; 3) ToolRetriever (Qin et al., 2023b): a dense retrieval approach specifically finetuned on tool retrieval datasets.

Implementation details. We employ Sentence-BERT (Reimers and Gurevych, 2019) to train our retriever model based on BERT-base (Devlin et al., 2019). We set the learning rate to $2e{-}5$ with 500 warm-up steps. The batch size in training is set to 64. We utilize ChatGPT (gpt-3.5-turbo-0125)[2] as the LLM for giving feedback. The number of tool candidates K, the balancing factor α, and the

[1] https://platform.openai.com/docs/guides/embeddings/embedding-models.

[2] https://openai.com/index/introducing-chatgpt-and-whisper-apis/.

Methods	Single-tool (I1)			Category (I2)			Collection (I3)			All		
	N@1	N@3	N@5	N@1	N@3	N@5	N@1	N@3	N@5	N@1	N@3	N@5
BM25	18.37	17.97	19.65	11.97	9.85	10.95	25.23	18.95	20.37	15.84	13.98	15.63
Ada Embedding	57.52	54.90	58.83	36.82	28.83	30.68	54.59	42.55	46.83	46.59	41.06	43.95
ToolRetriever	84.20	89.59	89.65	68.24	77.43	77.90	81.65	87.24	87.13	75.73	83.19	83.06
Ours	**90.70**	**90.95**	**92.47**	**89.01**	**85.46**	**87.10**	**91.74**	**87.94**	**90.20**	**88.53**	**87.00**	**88.83**
% improve	7.72%	1.52%	3.15%	30.44%	10.37%	11.81%	12.36%	0.80%	3.52%	16.90%	4.58%	6.95%

Table 2: In-domain evaluation on TR-bench in terms of NDCG@m under scenarios including single-tool (*I1*), intra-category multi-tool (*I2*), intra-collection multi-tool (*I3*), and the whole data (*All*). % improve represents the relative improvement achieved by our method over the previously best tool retrieval method.

Methods	T-Eval				UltraTools			
	N@1	N@3	N@5	N@10	N@1	N@3	N@5	N@10
BM25	52.12	43.19	45.23	52.91	15.10	14.13	16.03	18.34
Ada Embedding	80.11	69.11	71.95	79.62	31.46	33.75	39.91	46.40
ToolRetriever	82.10	72.03	74.15	**80.76**	48.20	**47.73**	53.01	58.93
Ours	**84.45**	**73.31**	**74.45**	80.25	**49.30**	47.50	**54.30**	**59.92**
% improve	2.86%	1.78%	0.40%	-0.06%	2.28%	-0.48%	2.43%	1.68%

Table 3: Out-of-domain evaluation on TR-bench in terms of NDCG@m under two scenarios, T-Eval (Chen et al., 2023) and UltraTools (Huang et al., 2024). % improve represents the relative improvement achieved by our method over the previously best tool retrieval method.

iteration round T are set to 10, 1, and 3, respectively. We have trained the model several times to confirm that the improvement is not a result of random chance and present the mid one. Our experiments were conducted on four NVIDIA A6000 GPUs with 48 GB of memory

5.2 Main Results

In-domain evaluation. The results of the in-domain evaluation are reported in Table 2. It is observed that non-finetuned retrieval methods, i.e., BM25 and Ada Embedding, perform much worse than other finetuned methods. This is reasonable since non-finetuned methods have not been specifically adopted for tool retrieval. While Tool Retriever outperforms non-finetuned methods, the performance is still not satisfying. In comparison, our proposed method consistently outperforms all finetuned and non-finetuned baselines. Significantly, our method maintains strong performance in the intra-category multi-tool (I2) scenario, even as other methods' performance declines, demonstrating the robustness of our proposed method across different scenarios. The above results prove the effectiveness of our method in enhancing tool retrieval accuracy, particularly in challenging scenarios with multi-tools.

Out-of-domain evaluation. Since the tools are usually frequently updated in real-world, we further test all methods in the out-of-domain setting, where the training data from ToolBench and the test data from T-Eval and UltraTools are used. The experimental results are shown in Table 3. We could observe that our method significantly outperforms other baselines across both scenarios. This demonstrates that our method not only excels in in-domain benchmarks but also maintains robust performance across varied scenarios, revealing its generalization ability of tool retrieval.

Methods	N@1	N@3	N@5	N@10
Ours	89.01	85.46	87.10	88.41
w/o warm-up	85.51	81.36	84.47	86.92
w/o hard-negative	86.04	80.41	84.00	85.98
w/o joint & hard-neg	83.77	77.67	81.21	83.69

Table 4: Ablation study of our method under the intra-category multi-tool (I2) scenario.

5.3 Ablation Study

We conduct ablation studies to investigate the efficacy of different components in our methods. First, we remove the warm-up training by directly conducting our method on an retriever based on Sentence-BERT. Then, we analyze the contribution of hard negative sampling in our method by removing the hard-to-distinguish samples from the training. In addition, we assess the efficacy of joint training in our method, by substituting it with a loss $\mathcal{L}_{feedback} = \mathcal{L}(q^t)$, with respect to

Iteration	N@1	N@3	N@5	N@10	Efficiency
1	85.69	80.48	83.94	86.27	6.12s
2	87.78	83.48	86.31	88.26	8.59s
3	89.01	85.46	87.10	88.41	10.30s

Table 5: Analysis on iteration round under the intra-category multi-tool instructions (I2) scenario. The efficiency is measured by the time consumption to complete one user instruction.

Methods	N@1	N@3	N@5
ToolRetriever (BERT-based)	68.24	77.43	77.90
Ours (BERT-based)	89.01	85.46	87.10
ToolRetriever (RoBERTa-based)	76.61	69.81	74.99
Ours (RoBERTa-based)	85.40	78.72	81.50

Table 6: Analysis on different base models under the intra-category multi-tool instructions (I2) scenario.

only the refined instructions q^t at current iteration t. Table 4 reports the ablation test performance (i.e., NDCG@m ($m = 1, 3, 5, 10$)) under the intra-category multi-tool instructions (I2) scenario on ToolBench.

From the results, we can observe that our method achieves comparably high NDCG scores even without warm-up training, indicating that it does not heavily rely on prior tool-use knowledge. When hard negative sampling is removed, the performance degradation illustrates that hard negative sampling could enable the model to discriminate between similar tool functionalities. Besides, the model's performance further declines when joint training is removed, demonstrating that the model could balance new and previous knowledge in this joint-training manner.

5.4 In-depth Analysis

Analysis on iteration round. The iteration round is an important factor in our method. We conduct experiments to investigate changes in effectiveness and efficiency with different iteration round T. The results are presented in Table 5, and the efficiency is measured by the cost of time to complete one user instruction on average.

By analyzing the results in Table 5, we gain two findings. 1) We could observe a continuous improvement as the iteration round increases. This shows that the tool retriever progressively enhances its performance with the aid of LLMs' feedback. 2) In terms of time efficiency, we find that adding one additional round of refinement takes an average of 6.12s/instruction, primarily resulting from the time waiting for LLM's feedback when calling the OpenAI API. As the number of iterations increases, we can see that the extra inference time required for each instruction decreases. This is due to the fact that there will be fewer instructions requiring refinement as retrieval performance improves.

Embedding Size	N@1	N@3	N@5	N@10
300	87.61	83.49	85.20	86.50
512	87.61	82.85	84.67	85.81
768	89.01	85.46	87.10	88.41
1024	88.66	83.91	85.94	87.04
2048	88.74	83.95	85.98	87.43

Table 7: Analysis on embedding sizes under the intra-category multi-tool instructions (I2) scenario.

Analysis on base models. We further analyze the impact of different base models on the performance. Specifically, we replace the base model BERT in our method with another classic language model, RoBERTa (Liu et al., 2019). The results are shown in Table 6. As we can see, our method still achieves significant improvement over the baseline with the same RoBERTa model. Another observation is that RoBERTa is more effective in serving as a base model for the retrieval application, which benefits from its effective training strategies. The improvements demonstrate the robustness of our method with different base models.

Analysis on embedding sizes. Since the retriever model R encodes the textual instruction and tool description into dense vectors, we explore the impact of the embedding size on retrieval performance. as shown in Table 7. From the table, we can find that larger embedding sizes result in greater performance improvements compared to smaller embedding sizes. This is probably due to the fact that embeddings with larger sizes could accommodate more knowledge. However, when the embedding size increases from 768 to 2048, there is a slight decrease in performance. This suggests that a specific embedding size is sufficient, and larger embedding sizes may pose challenges to training. It is worth noting that larger embedding sizes necessitate higher training costs and increased inference memory. Therefore, we recommend an optimal embedding size of 768.

5.5 Case Study

As shown in Figure 4, we conduct case study by using an example of instruction refinement to take a closer look at the effect of our method.

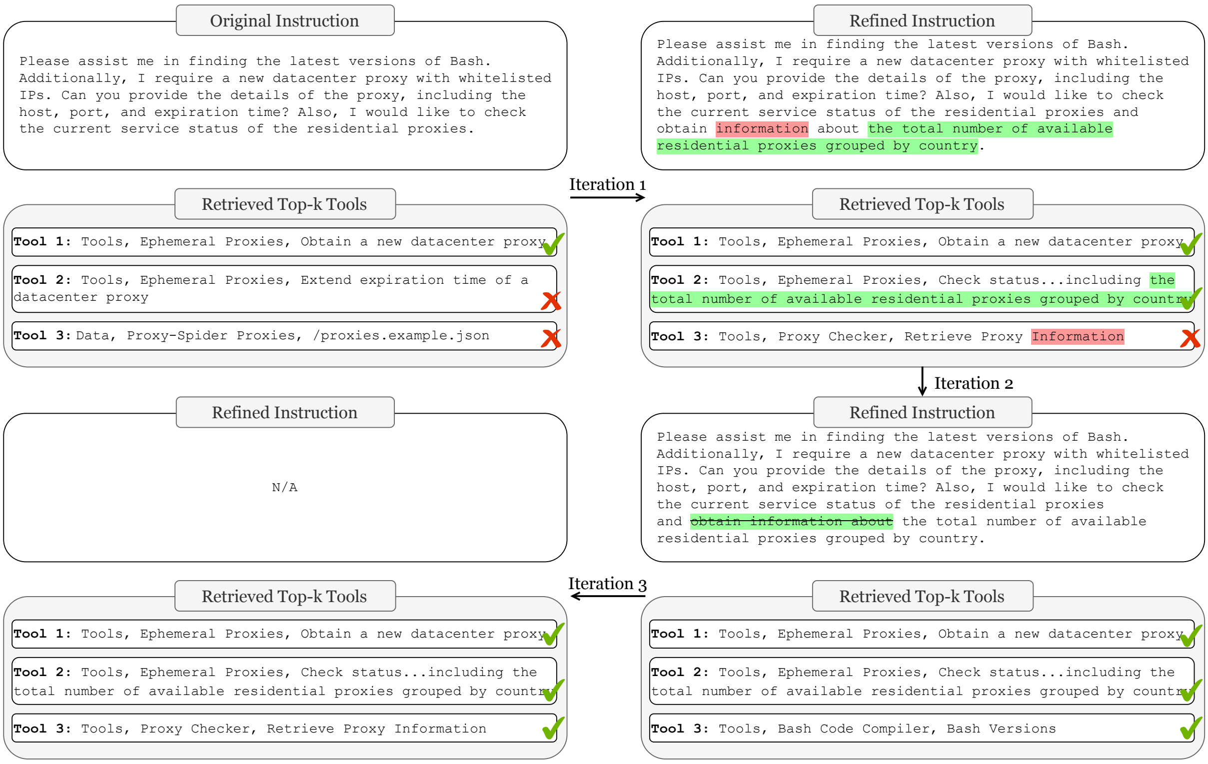

Figure 4: Case study on the effect of user instruction refinement through 3 iterations. The original instruction is revised step-by-step, leading to improved retrieval results.

In the $1st$ iteration, we can observe that the refined instruction has included more detailed information (i.e., "total number") about the user's requirements than the original instruction, enabling the retriever to identify more appropriate tools (e.g., Check residential proxies service status). This reveals that the comprehension capabilities of LLMs could be instilled into the retrieval process through feedback. In the $2nd$ iteration, our method further refines the instruction by omitting irrelevant content (i.e., "information") which may mislead the retriever into retrieving incorrect tools (e.g., Retrieve Proxy Information). Another benefit of the refinement is that some correct tools (e.g., Bash Code Compiler) will move up in positions of the top-K rankings, improving the overall retrieval performance. In the $3rd$ iteration, our method showcases great decision-aware capabilities, where the iterative process could be terminated if no further refinement is deemed necessary.

6 Conclusion and Future Work

In this study, we concentrate on the crucial tool retrieval in the tool learning of LLMs. We have identified the bottleneck in the tool retrieval-usage pipeline as the limited tool retrieval model. We propose the unique challenges of the tool retrieval compared with document retrieval. To improve the current tool retrieval process, we propose leveraging the LLM's feedback to assess the retrieval results and provide detailed suggestions for refining user instructions. In order to integrate the retriever model into this iterative process, we implement iteration-aware feedback training. This will improve the tool retriever's capabilities and close the gap between tool retrieval and usage models. We conduct the TR-benchmark to comprehensively evaluate the models' ability in real-world tool retrieval scenarios. Our method demonstrates the best performance in both in-domain and out-of-domain settings.

In the future, we aim to improve this work from the following aspects. 1) Limited by the training speed, we have applied the offline feedback generation, where feedback is generated before training the tool retriever. We will also assess whether online feedback generation yields further improvements in the future. 2) Furthermore, as the tool retriever serves the subsequent tool usage model in tool learning, we intend to conduct further evaluations of the tool retriever models based on the subsequent tool usage results.

Limitations

1) Undoubtedly, our iterative refinement will reduce the inference speed of the tool retrieval. We have evaluated the efficiency as the number of iterative rounds increases. Fortunately, we observed that just one additional round of refinement could yield significant improvements. Furthermore, the performance enhancement of the tool retrieval is crucial for the subsequent tool usage model. 2) Similar to document retrieval, the used datasets in our work also contain "false negative" samples. For instance, some tools may be capable of handling the user's instruction but are not labeled as positive. This can disrupt the training and evaluation of tool retrieval and is a common limitation in many retrieval scenarios.

Ethics Statement

The datasets used in our experiment are publicly released and labeled through interaction with humans in English. In this process, user privacy is protected, and no personal information is contained in the dataset. The scientific artifacts that we used are available for research with permissive licenses. And the use of these artifacts in this paper is consistent with their intended use. Therefore, we believe that our research work meets the ethics of the conference.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. arXiv preprint arXiv:2310.11511.

Zehui Chen, Weihua Du, Wenwei Zhang, Kuikun Liu, Jiangning Liu, Miao Zheng, Jingming Zhuo, Songyang Zhang, Dahua Lin, Kai Chen, et al. 2023. T-eval: Evaluating the tool utilization capability step by step. arXiv preprint arXiv:2312.14033.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186. Association for Computational Linguistics.

Shijue Huang, Wanjun Zhong, Jianqiao Lu, Qi Zhu, Jiahui Gao, Weiwen Liu, Yutai Hou, Xingshan Zeng, Yasheng Wang, Lifeng Shang, et al. 2024. Planning, creation, usage: Benchmarking llms for comprehensive tool utilization in real-world complex scenarios. arXiv preprint arXiv:2401.17167.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of ir techniques. ACM Transactions on Information Systems, 20(4):422–446.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 6769–6781. Association for Computational Linguistics.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023. API-bank: A comprehensive benchmark for tool-augmented LLMs. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pages 3102–3116. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In Advances in Neural Information Processing Systems, volume 36, pages 46534–46594. Curran Associates, Inc.

Bhaskar Mitra and Nick Craswell. 2017. Neural models for information retrieval. arXiv preprint arXiv:1705.01509.

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

Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, et al. 2023a. Tool learning with foundation models. arXiv preprint arXiv:2304.08354.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2023b. Toolllm: Facilitating large language models to master 16000+ real-world apis. arXiv preprint arXiv:2307.16789.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In Proceedings of the 2019 Conference on

Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 3982–3992. Association for Computational Linguistics.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. Foundations and Trends in Information Retrieval, 3(4):333–389.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In Advances in Neural Information Processing Systems, volume 36, pages 68539–68551. Curran Associates, Inc.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. In Advances in Neural Information Processing Systems, volume 36, pages 38154–38180. Curran Associates, Inc.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In International Conference on Learning Representations (ICLR).

Wayne Xin Zhao, Jing Liu, Ruiyang Ren, and Ji-Rong Wen. 2024. Dense text retrieval based on pretrained language models: A survey. ACM Transactions on Information Systems, 42(4):1–60.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. arXiv preprint arXiv:2303.18223.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In Advances in Neural Information Processing Systems, volume 36, pages 46595–46623. Curran Associates, Inc.