# TInR: Exploring Tool-Internalized Reasoning in Large Language Models

**Qiancheng Xu**[1], **Yongqi Li**[1†], **Fan Liu**[2], **Hongru Wang**[3], **Min Yang**[4], **Wenjie Li**[1]
[1] The Hong Kong Polytechnic University [2] Southeast University [3] University of Edinburgh
[4] Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
qiancheng.xu@connect.polyu.hk liyongqi0@gmail.com cswjli@comp.polyu.edu.hk

## Abstract

Tool-Integrated Reasoning (TIR) has emerged as a promising direction by extending Large Language Models' (LLMs) capabilities with external tools during reasoning. Existing TIR methods typically rely on external tool documentation during reasoning. However, this leads to tool mastery difficulty, tool size constraints, and inference inefficiency. To mitigate these issues, we explore Tool-Internalized Reasoning (TInR), aiming at facilitating reasoning with tool knowledge internalized into LLMs. Achieving this goal presents notable requirements, including tool internalization and tool-reasoning coordination. To address them, we propose TInR-U, a tool-internalized reasoning framework for unified reasoning and tool usage. TInR-U is trained through a three-phase pipeline: 1) tool internalization with a bidirectional knowledge alignment strategy; 2) supervised fine-tuning warm-up using high-quality reasoning annotations, and 3) reinforcement learning with TInR-specific rewards. We comprehensively evaluate our method across in-domain and out-of-domain settings. Experiment results show that TInR-U achieves superior performance in both settings, highlighting its effectiveness and efficiency. Codes are available at https://github.com/travis-xu/TInR.

## 1 Introduction

Large language models (LLMs), such as Deepseek-R1 (Guo et al., 2025), have demonstrated remarkable reasoning and problem-solving capabilities in complex tasks such as code generation, logical deduction, and workflow planning. However, they often struggle in scenarios beyond their capabilities such as knowledge updates, weather inquiries, or restaurant reservations. To address this issue, Tool-Integrated Reasoning (TIR) has been proposed, enabling LLMs to leverage external tools during the reasoning process and thus extend their capabilities beyond purely language-based reasoning to a broader range of practical applications.

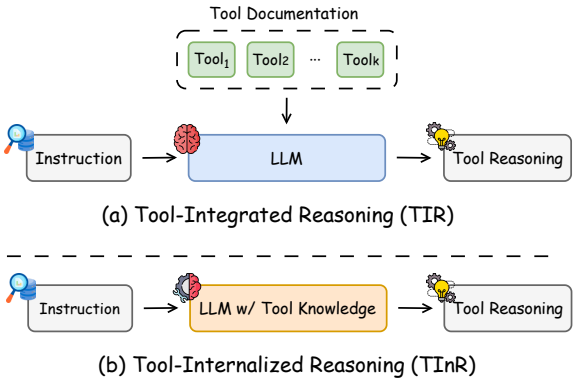


Figure 1: Comparison between (a) Tool-Integrated Reasoning (TIR) and (b) Tool-Internalized Reasoning (TInR). TInR internalizes tool knowledge into LLMs to facilitate reasoning.

A typical TIR process, as illustrated in Figure 1(a), begins with a user instruction and a set of available tools accompanied by their documentation; the LLM is then expected to reason among the candidate tools to address the given instruction. Under this setting, early TIR methods often rely on supervised fine-tuning (SFT) to teach LLMs with annotated reasoning paths, but suffer from limited generalization and adaptability (Chu et al., 2025). Thereafter, reinforcement learning (RL) approaches have been proposed to allow LLMs to integrate reasoning with tool learning through outcome-based feedback (Qian et al., 2025; Wang et al., 2025a), thus fostering more strategic and exploratory reasoning abilities.

Despite recent progress, existing TIR methods still rely on prompt-based tool documentation to inform LLMs about available tools. In other words, tool knowledge remains external to LLM and must be explicitly provided for reasoning. This brings

†Corresponding author.



*Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 44851–44865

several limitations: 1) **Tool mastery difficulty**. Tool documentation is often heterogeneous and inconsistent, making it difficult for LLMs to quickly grasp tool knowledge on the fly (Yuan et al., 2025; Qu et al., 2025). This brings a gap between external tool knowledge and LLM's internal understanding, which hinders effective tool mastery during reasoning. 2) **Tool size constraints**. As the number of tools increases, it becomes infeasible to include all tool documentation within the context window. While retrieval strategies can partially alleviate this, they introduce additional pipeline complexity and cause a potential misalignment between retrieval and tool usage (Xu et al., 2024). 3) **Inference inefficiency**. Including all tool documentation significantly increases prompt length, leading to higher inference latency and computational overhead. This makes real-time applications or large-scale deployments more costly and less efficient.

Humans, by contrast, are capable of internalizing tool knowledge into their brains and applying it continuously to problem solving without consulting external tool manuals. Inspired by this, we explore **T**ool-**In**ternalized **R**easoning (**TInR**) in LLMs. As shown in Figure 1(b), TInR enables reasoning with internalized tool knowledge rather than relying on external tool documentation, which could harmonize heterogeneous tool knowledge and facilitate effective and efficient tool utilization. To realize TInR, the LLM must satisfy the following requirements: 1) **Tool internalization**. The LLM should internalize tool knowledge into its parameters, encompassing both diverse tool functionalities and strict usage rules (e.g., parameter constraints, tool call formats). 2) **Tool-reasoning coordination**. Building on internalized tool knowledge, the LLM should seamlessly integrate them into its reasoning process for adaptive tool use and strategic problem solving.

In this work, we propose **TInR-U**, a **T**ool-**In**ternalized **R**easoning framework with **U**nified tool usage and reasoning. To progressively endow the LLM with TInR capability, TInR-U adopts a three-phase training pipeline: 1) **Tool internalization**. The LLM is trained through a bidirectional knowledge alignment strategy, where it learns to map tool documentation to unique tool tokens and, conversely, to recall the original documentation from each token. We also conduct tool usage training for practical knowledge application. This design aims at both fine-grained preservation and a holistic understanding of the tool knowledge. 2) **TInR SFT warm-up**. We construct annotated reasoning data via rejection sampling and data formatting, ensuring both high reliability and close alignment with the TInR task. We then employ supervised fine-tuning to equip the LLM with a foundational ability to leverage internalized tool knowledge during reasoning. 3) **TInR RL**. We employ reinforcement learning with specially designed rewards on tool tokens to further encourage exploratory tool reasoning, thereby promoting a deeper integration of tool usage competence with inherent reasoning ability.

To comprehensively evaluate TInR capabilities, we conduct experiments in both in-domain and out-of-domain settings. Experimental results demonstrate that our approach achieves state-of-the-art performance across both domains, showing relative improvement of $18.13\%$ in out-of-domain tool calling.

In summary, our contributions are as follows:

- We explore tool-internalized reasoning (TInR) in LLMs, aiming to facilitate reasoning with internalized tool knowledge instead of external tool documentation.
- We introduce TInR-U, a framework that achieves tool internalization through dedicated tool tokens, and further unifies tool usage with reasoning through a carefully designed three-phase training process.
- Experiments demonstrate that TInR-U outperforms baselines in both in-domain and out-of-domain settings, achieving effective and efficient TInR capabilities.

## 2 Related Work

### 2.1 Tool-Integrated Reasoning

Tool-Integrated Reasoning (TIR) has recently been recognized as an effective way to enhance the reasoning capabilities of LLMs by enabling interaction with external tools. Early efforts in this direction have mainly depended on either in-context demonstrations, which guide LLMs to perform tool reasoning directly through carefully designed prompts without training (Li et al., 2025b; Lu et al., 2025; Wu et al., 2025), or supervised fine-tuning (SFT), which transfers tool-usage ability to smaller LLMs by distilling trajectories from stronger ones (Gou et al., 2024; Li et al., 2025a;

Chen et al., 2025). However, these approaches struggle to generalize to novel tasks or unfamiliar tool settings. To overcome this limitation, more recent studies employ reinforcement learning (RL), encouraging more flexible and exploratory tool usage behaviors (Li et al., 2025c; Jin et al., 2025). Despite recent progress, existing TIR methods still rely on external tool knowledge with limited tool understanding and efficiency. In this work, we explore TInR which internalizes external tool knowledge into the LLM parameters to enhance tool reasoning.

### 2.2 Tool Learning in LLMs

Tool learning aims to augment LLMs with external tools to extend their ability beyond text generation. To achieve this, prior work has explored in-context learning, where LLMs interact directly with tool documentation in context (Lumer et al., 2024; Li et al., 2024b; Liu et al., 2025b), as well as fine-tuning approaches that specialize LLMs on curated tool-use datasets (Hao et al., 2023; Tang et al., 2023; Chen et al., 2024; Xu et al., 2025; Chen et al., 2025). To address the tool size constraints in context, tool retrieval has been adopted as an upstream component to narrow down tool candidates before usage (Qin et al., 2024; Xu et al., 2024; Shi et al., 2025). Recently, several studies (Hao et al., 2023; Wang et al., 2025b; Su et al., 2025) have explored internalizing tools into LLMs. However, these efforts are limited by small toolsets, simple reasoning strategies or unstable LLM-based evaluation (Iskander et al., 2024). In contrast, our work provides a comprehensive investigation and rigorous evaluation of TInR in complex tool-use environments with large toolsets, and deliberately designs a three-phase training pipeline that enables both effective tool knowledge internalization and strategic tool-coordinated reasoning.

## 3 Methodology

### 3.1 Task formulation

Given a user instruction $q$, the goal of TInR is to solve the task through a sequence of reasoning steps interleaved with tool invocations, but without relying on any external tool documentation. Formally, consider a tool set $\mathcal{T} = \{t_1, t_2, ..., t_N\}$, where each tool $t_i$ is associated with documentation $D(t_i)$. At step $j$, the LLM first performs natural language reasoning $r_j$. If a tool usage is required, the LLM executes a tool-use action $a_j$, which specifies a set of tool calls, each defined as a pair $(t, p)$ consisting of a selected tool $t \in \mathcal{T}$ and its associated parameters $p$. The tool's output $o_j$ is then incorporated into the next reasoning step. The overall reasoning trajectory can thus be described as:

$$\tau = (r_1, a_1, o_1), (r_2, a_2, o_2), \ldots, (r_T, a_T, o_T).$$

Note that the underlying decision mechanism of TInR fundamentally differs from that of TIR: tool actions in TInR are generated from internalized knowledge within the LLM rather than from external documentation.

### 3.2 Overview

TInR-U is a unified framework that internalizes external tool knowledge into the LLM parameters and integrates it into the reasoning process, as illustrated in Figure 2. It addresses two central requirements: 1) encoding tool functionalities and usage constraints into the LLM parameters, and 2) coordinating this internalized knowledge with multi-step reasoning to guide tool selection and invocation.

The framework is composed of three training phases: 1) **Phase 1** - Tool knowledge internalization. Tool functionalities and usage semantics are embedded into the LLM parameters to support tool understanding and invocation. 2) **Phase 2** - TInR SFT warm-up. The LLM is trained with curated reasoning trajectories to align generation behavior with expected tool usage. 3) **Phase 3** - TInR RL. Reward-driven optimization refines the robustness and accuracy of tool reasoning.

### 3.3 Tool Internalization

The first phase embeds tool knowledge into the LLM's parameters. It consists of two steps: expanding the LLM vocabulary with dedicated tool tokens to unify reasoning and tool invocation, and aligning tool semantics and usage knowledge through a bidirectional learning objective.

**Tool tokenization.** To support seamless integration of tool usage within the language modeling process, the vocabulary is extended with tool-specific tokens. Each tool $\{t_i\}$ is assigned a unique token $\{I(t_i)\}$, enabling the LLM to reference and invoke tools through the next-token generation.

To reduce the action space and improve reasoning reliability, two control tags are introduced: <tool_token> and <tool_call>. This two-step generation first predicts a set of tool tokens $\{I(t_i)\}_{i=1}^{K}$ within the <tool_token> scope, corresponding to

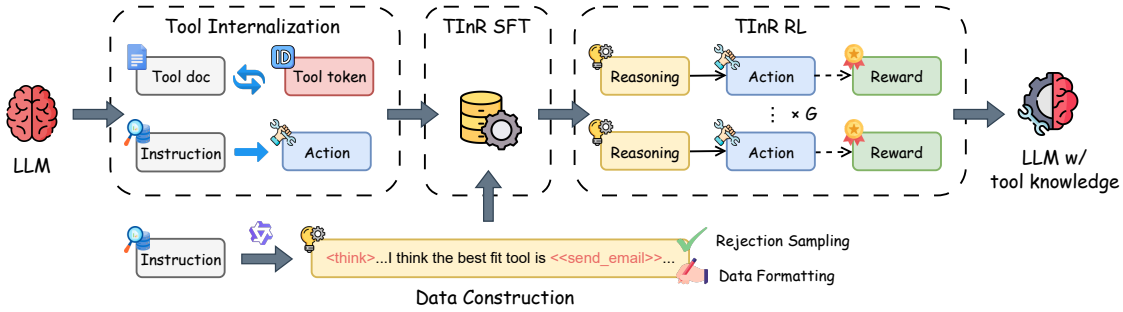


Figure 2: Illustration of our proposed TInR-U, with a three-phase training pipeline including tool internalization, TInR SFT warm-up, and TInR RL. TInR-U facilitates tool knowledge internalization and tool usage during reasoning.

tools $\{t_i\}_{i=1}^{K} \subseteq \mathcal{T}$. Based on the associated documentation $\{D(t_i)\}_{i=1}^{K}$, the LLM then pairs each identified tool token with its parameters, thereby generating complete tool calls $(I(t), p)$ within the <tool_call> scope. We empirically demonstrate that this two-step design can benefit the TInR performance in Section 4.3.

**Bidirectional knowledge alignment.** After tokenization, we internalize tool knowledge within the LLM. Specifically, we design a bidirectional knowledge alignment strategy consisting of three objectives: tool memorization, tool recall, and tool usage grounding.

The first objective is to develop semantic mappings from the documentation to tokens. Specifically, the LLM is trained to predict each tool token $I(t)$ based on its tool documentation $D(t)$. Formally, the loss is defined as:

$$\mathcal{L}_{\text{memorization}} = -\sum_{t \in \mathcal{T}} \log P(I(t) \mid D(t)). \quad (1)$$

The tool recall objective is to encourage fine-grained preservation of tool documentation in the internalized representation. Specifically, the LLM is enforced to reconstruct the original documentation $D(t)$ based on each tool token $I(t)$, which can be formulated as:

$$\mathcal{L}_{\text{recall}} = -\sum_{t \in \mathcal{T}} \sum_{s=1}^{|D(t)|} \log P(D(t)_s \mid I(t), D(t)_{<s}), \quad (2)$$

where $D(t)_s$ denotes the $s$-th token of the documentation.

To better align tool tokens with real usage scenarios, we conduct direct tool usage training. Given each user instruction $q$ in the training dataset $\mathcal{D}$, the LLM is trained to directly generate the correct tool-use action $a$. Note that $a$ may consist of one or multiple tool actions, allowing the LLM to learn complex task–tool associations. The training loss is defined as:

$$\mathcal{L}_{\text{usage}} = -\sum_{q,a \in \mathcal{D}} \sum_{s=1}^{|a|} \log P(a_s \mid q, a_{<s}). \quad (3)$$

The overall training objective is:

$$\mathcal{L}_{\text{Phase 1}} = \mathcal{L}_{\text{memorization}} + \alpha \mathcal{L}_{\text{recall}} + \beta \mathcal{L}_{\text{usage}}, \quad (4)$$

where $\alpha$ and $\beta$ are weighting factors. This design enables the LLM to capture both fine-grained details and high-level semantic understanding of tool knowledge, thereby laying a solid foundation for the subsequent reasoning-oriented phases.

### 3.4 TInR SFT Warm-up

The second phase aligns the LLM's reasoning behavior with expected tool usage through supervised fine-tuning.

**Data Construction.** We construct high-quality TInR trajectories via rejection sampling. Specifically, for each user instruction $q$, we collect 10 candidates tools from $\mathcal{T}$, including ground-truth, retrieved and randomly sampled tools. Based on $q$ and the documentation of candidate tools, we prompt a large reasoning model (LRM) to synthesize multiple reasoning trajectories and only keep the correct ones validated by ground-truth tool-use actions. To enhance tool-reasoning coordination, we further conduct data formatting by replacing each tool name field appearing in the reasoning content with its corresponding tool token, thus enabling LLMs to explicitly incorporate internalized tool knowledge into reasoning steps. In this manner, we obtain an SFT dataset $\mathcal{D}_{\text{SFT}} = \{(q, \tau)\}$ that

is highly reliable and well-aligned with the TInR goal.

We then optimize the LLM under the SFT objective:

$$\mathcal{L}_{\text{Phase 2}} = -\sum_{q,\tau \in \mathcal{D}_{\text{SFT}}} \sum_{s=1}^{|\tau|} \log P(\tau_s \mid q, \tau_{<s}), \quad (5)$$

where $\tau_s$ denotes the $s$-th token in the reasoning trajectory.

### 3.5 TInR RL

The final phase improves the robustness and adaptability of tool reasoning using reinforcement learning. A composite reward function encourages both structural correctness and accurate tool usage.

**Reward Design.** Rule-based reward mechanisms have demonstrated strong empirical performance and are commonly adopted in TIR methods (Li et al., 2025c; Jin et al., 2025). Following this line, we design a composite reward that combines format reward and correctness reward to ensure both structural validity and correctness of TInR trajectories. The format reward $R_{\text{format}}$ verifies whether the predicted trajectory $\tau$ conforms to the required structure, i.e., contains all special tags in the correct order:

$$R_{\text{format}} = \begin{cases} 1, & \text{if the format of } \tau \text{ is correct} \\ 0, & \text{otherwise} \end{cases} \quad (6)$$

The correctness reward measures both tool identification and parameter specification accuracy of tool calls. Both accuracy is measured by the Jaccard similarity. Let $\mathcal{C}$ and $\hat{\mathcal{C}}$ denote the ground-truth and predicted tool calls. The tool reward $r_{\text{tool}}$, parameter reward $r_{\text{param}}$ and correctness reward $R_{\text{correct}}$ can be defined as:

$$r_{\text{tool}} = \left| \frac{\mathcal{I} \cap \hat{\mathcal{I}}}{\mathcal{I} \cup \hat{\mathcal{I}}} \right| \in [0, 1], \quad (7)$$

$$r_{\text{param}} = \frac{1}{|\mathcal{C}|} \sum_{\mathcal{P}_i \in \mathcal{C}} \left| \frac{\mathcal{P}_i \cap \hat{\mathcal{P}}_i}{\mathcal{P}_i \cup \hat{\mathcal{P}}_i} \right| \in [0, 1], \quad (8)$$

$$R_{\text{correct}} = r_{\text{tool}} + r_{\text{param}}, \quad (9)$$

where $\mathcal{I}$ and $\hat{\mathcal{I}}$ are the sets of tool tokens extracted from $\mathcal{C}$ and $\hat{\mathcal{C}}$, while $\mathcal{P}_i$ and $\hat{\mathcal{P}}_i$ denote the parameter sets of the $i$-th tool call in $\mathcal{C}$ and $\hat{\mathcal{C}}$, respectively. The final reward is then calculated as:

$$R = R_{\text{format}} + R_{\text{correct}}. \quad (10)$$

| Settings | Split | # Instructions | # Tools |
|---|---|---|---|
| In-domain | Train | 2552 | 2467 |
| | Test(Seen) | 185 | 307 |
| | Test(Unseen) | 580 | 831 |
| Out-of-domain | | 1996 | 2025 |

Table 1: Statistics of the experiment datasets conducted from ToolACE (Liu et al., 2025a), xLAM (Zhang et al., 2025), and BFCL (Patil et al., 2025).

**Training objective.** We employ Group Relative Policy Optimization (GRPO) to optimize LLM under $R$. Specifically, for each user instruction $q$, the LLM samples a group of $G$ trajectories $\{\tau_i\}_{i=1}^{G}$, where each $\tau_i$ is assigned a reward $R_i$. By normalizing the rewards within the group, the advantage function for $\tau_i$ is calculated as $A_i = \frac{R_i - \text{mean}(\{R_j\}_{j=1}^{G})}{\text{std}(\{R_j\}_{j=1}^{G})}$. Then the training objective can be defined as:

$$\mathcal{L}_{\text{Phase 3}} = \mathbb{E}_{q\sim\mathcal{D},\tau\sim\pi_\theta} \left[ \frac{1}{G} \sum_{i=1}^{G} \min \left( \frac{\pi_\theta(\tau_i \mid q)}{\pi_{\theta_{\text{old}}}(\tau_i \mid q)} A_i, \right. \right. \\ \left. \left. \text{clip} \left( \frac{\pi_\theta(\tau_i \mid q)}{\pi_{\theta_{\text{old}}}(\tau_i \mid q)}, 1-\epsilon, 1+\epsilon \right) A_i \right) \right], \quad (11)$$

where $\pi_\theta$ is the updated policy and $\pi_{\theta_{\text{old}}}$ is the reference policy. Following Qian et al. (2025), we remove the KL penalty term for fast adaptation to the task-specific reward.

### 3.6 Inference

During inference, the LLM follows the prescribed format with special tags to conduct reasoning step by step, until ultimately resolving the user instruction. As tool knowledge is fully embedded within the LLM parameters, no external documentation or retrieval is required, allowing efficient and scalable deployment of tool-augmented reasoning in real-world applications.

## 4 Experiments

### 4.1 Setup

**Datasets.** To reflect the diversity and complexity of real-world tool environments, we conduct our experiments on three datasets, ToolACE (Liu et al., 2025a), xLAM (Zhang et al., 2025), and BFCL (Patil et al., 2025), covering 1) multiple domains with large toolsets and verifiable answers, 2) both single-turn and multi-turn tasks, and 3) both in-domain and out-of-domain settings. For the in-domain setting, we adopt ToolACE and xLAM,

| Methods | SEEN | | | | | UNSEEN | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Tool Identification | | Tool Calling | | | Tool Identification | | Tool Calling | | |
| | EM | F1 | EM | T. Acc | P. Acc | EM | F1 | EM | T. Acc | P. Acc |
| BM25+ToolRL | 48.11 | 54.68 | 54.59 | 65.41 | 61.62 | 42.76 | 51.24 | 46.38 | 59.48 | 51.90 |
| Ada-embedding+ToolRL | 56.22 | 61.71 | 58.38 | 69.19 | 67.57 | 51.38 | 60.19 | 45.69 | 60.00 | 51.90 |
| TR-Feedback+ToolRL | 65.41 | 72.61 | 62.70 | 73.51 | 71.35 | 59.31 | 67.80 | 51.38 | 67.41 | 57.76 |
| ToolRetriever+Hammer2.1-7b | 63.78 | 69.46 | 36.22 | 44.32 | 41.08 | 59.66 | 68.47 | 33.28 | 43.45 | 34.83 |
| ToolRetriever+xLAM-7b-r | 63.78 | 69.46 | 48.65 | 60.00 | 54.05 | 59.66 | 68.47 | 36.38 | 51.21 | 40.00 |
| ToolRetriever+Qwen3-8B | 63.78 | 69.46 | 58.38 | 71.89 | 64.32 | 59.66 | 68.47 | 48.45 | 68.79 | 52.93 |
| ToolRetriever+ToolRL | 63.78 | 69.46 | 61.08 | 75.14 | 70.27 | 59.66 | 68.47 | 51.72 | 67.41 | 59.83 |
| ATU | – | – | 61.08 | 74.59 | 71.35 | – | – | 26.38 | 44.14 | 41.38 |
| ToolGen | 83.78 | 86.76 | 71.89 | 83.78 | **77.30** | 73.79 | 80.55 | 55.86 | 72.59 | 60.52 |
| **TInR-U** | **85.95** | **88.38** | **74.05** | **84.86** | **77.30** | **75.86** | **81.86** | **57.24** | **74.83** | **62.76** |
| % improve | 2.59% | 1.87% | 3.00% | 1.29% | 0.00% | 2.81% | 1.63% | 2.47% | 3.09% | 3.70% |

Table 2: In-domain evaluation results of baselines and TInR-U. EM, T.Acc, and P. Acc stand for Exact Match, Tool Accuracy, Parameter Accuracy, respectively. % improve represents the relative improvement achieved by our method over the previously best-performing method.

| Methods | Tool Identification | | Tool Calling | | |
|---|---|---|---|---|---|
| | EM | F1 | EM | T. Acc | P. Acc |
| BM25+ToolRL | 24.94 | 25.80 | 16.10 | 32.55 | 26.46 |
| Ada-embedding+ToolRL | 32.32 | 32.38 | 16.04 | 36.82 | 28.92 |
| TR-Feedback+ToolRL | 30.33 | 30.43 | 16.39 | 35.71 | 29.63 |
| ToolRetriever+xLAM-7b-r | 30.56 | 30.66 | 10.13 | 27.17 | 21.55 |
| ToolRetriever+Hammer2.1-7b | 30.56 | 30.66 | 12.06 | 28.57 | 22.72 |
| ToolRetriever+Qwen3-8B | 30.56 | 30.66 | 17.45 | 36.42 | 29.27 |
| ToolRetriever+ToolRL | 30.56 | 30.66 | 16.63 | **37.35** | 28.45 |
| ATU | – | – | 11.59 | 25.29 | 33.37 |
| ToolGen | 34.89 | 34.97 | 22.01 | 30.91 | 48.24 |
| **TInR-U** | **38.06** | **38.06** | **26.00** | 35.83 | **50.12** |
| % improve | 9.09% | 8.84% | 18.13% | -4.07% | 3.90% |

Table 3: Out-of-domain evaluation results of baselines and TInR-U.

where each dataset is sampled and split into training and test sets. The test set is further partitioned into seen and unseen subsets, depending on whether the ground-truth test tools appear in the training data. For the out-of-domain setting, we use BFCL as the test set. The statistics of datasets are summarized in Table 1.

**Metrics.** We evaluate the TInR capability from two complementary dimensions: 1) Tool identification. This dimension measures the ability to correctly identify which tools should be used at each step. We adopt Exact Match (EM) and F1 score as evaluation metrics. EM captures whether the predicted set of tool tokens exactly matches the ground truth, while F1 provides a more nuanced measure by balancing precision and recall in partial matches. 2) Tool calling. To measure the ability to generate accurate tool-use actions with appropriate parameters, we use three evaluation metrics: Exact Match, which checks whether the predicted tool calls are entirely matched; Tool Accuracy, which assesses whether all tool tokens in tool calls are correct; and Parameter Accuracy, which evaluates the correctness of the predicted parameters in tool calls.

**Baselines.** To provide a thorough comparison, we evaluate tool retrieval, tool reasoning and end-to-end methods. For **tool retrieval**, we include four representative methods: 1) BM25 (Robertson and Zaragoza, 2009): the classical sparse retrieval method; 2) Ada Embedding: the OpenAI's text-embedding-ada-002 model; 3) ToolRetriever (Qin et al., 2024): a dense retrieval method finetuned on tool retrieval tasks. 4) TR-Feedback (Xu et al., 2024): a dense retrieval method leveraging LLMs' iterative feedback. For **tool reasoning** methods, we include: 1) Hammer-2.1-7B (Lin et al., 2025): a model fine-tuned with robust function-calling op-

timization; 2) xLAM-7B-r (Zhang et al., 2025): a model tailored for tool usage with reasoning and action decomposition; 3) Qwen3-8B (Yang et al., 2025): an LRM with strong reasoning ability and built-in tool calling support. 4) ToolRL (Qian et al., 2025): an RL-based tool usage model optimized with structured rewards and Group Relative Policy Optimization (GRPO). For **end-to-end** methods, we include: 1) ATU (Li et al., 2024c): an end-to-end method for direct tool usage without tool documentation. 2) ToolGen (Wang et al., 2025b): a unified generation framework for both tool retrieval and tool calling using virtual tokens. To thoroughly evaluate models across the entire pipeline, we employ ToolRL as the downstream tool usage model for each retrieval model, and ToolRetriever as the upstream tool retriever for each tool usage model.

### 4.2 Main Results

**In-domain evaluation.** The in-domain evaluation results are shown in Table 2. From the results, we summarize the following key findings: 1) All baselines consistently perform worse on the unseen test set than on the seen set. This highlights the intrinsic difficulty of generalizing tool knowledge to novel tools. 2) Both tool retrieval and tool usage methods perform substantially worse than tool-internalized approaches (e.g., ToolGen). This supports our claim that it is difficult for LLMs to master tool knowledge solely from external tool documentation. 3) Tool usage models that demonstrate strong reasoning ability (e.g., ToolRL) suffer from performance ceilings due to the upstream retrieval quality. This confirms our claim that addressing the context-length limitation of TIR through retrieval strategies is suboptimal. 4) Our proposed TInR-U achieves the best performance on both seen and unseen test sets with slightly larger improvements on unseen tools, demonstrating its effectiveness and generalization ability.

**Out-of-domain evaluation.** We further test all methods in the out-of-domain setting, and the experimental results are shown in Table 3. We could observe that the performance across all methods is worse than in-domain settings, indicating that the task in this setting is more challenging. In contrast, our method continues to achieve the best results across most metrics, with larger relative improvements on several key metrics (e.g., +18.13% on EM for tool calling), further confirming its generalization ability.

| Methods | Tool Identification | | Tool Calling | | |
|---|---|---|---|---|---|
| | EM | F1 | EM | T. Acc | P. Acc |
| **TInR-U** | **78.30** | **83.44** | **61.31** | **77.25** | **66.27** |
| *w/o BKA* | 49.67 | 56.81 | 40.39 | 48.63 | 49.15 |
| *w/o RL* | 76.47 | 81.46 | 59.61 | 74.90 | 64.18 |
| *w/o recall* | 76.21 | 82.05 | 59.74 | 75.29 | 64.58 |
| *w/o memorization* | 58.43 | 65.45 | 45.49 | 57.25 | 54.64 |
| *w/o usage* | 59.35 | 66.76 | 43.79 | 58.56 | 51.76 |
| *w/o two-step* | – | – | 43.40 | 72.94 | 50.72 |

Table 4: Ablation study on key components of TInR-U. BKA stands for bidirectional knowledge alignment.

### 4.3 Ablation Study

We conducted an ablation study to assess the contribution of different components in our framework. First, we remove the bidirectional knowledge alignment strategy and RL training to evaluate their effect. Then, we separately remove the three objectives in tool knowledge internalization, i.e., tool memorization, tool recall, and tool usage, to measure their individual impact. We also ablate the two-step design in the tool-use action to assess its importance by enforcing single-step tool calling, where LLM generates the complete tool calls directly without associating intermediate tool tokens to documentation. Since our experiments reveal that LLMs trained without SFT warm-up are incapable of performing effective reasoning, we do not include this ablation. Table 4 reports the test results in the in-domain setting. We observe that removing bidirectional knowledge alignment strategy and RL leads to consistent performance drops, indicating their necessity for tool knowledge acquisition and strategic tool-reasoning. All three objectives in the tool internalization phase (memorization, recall, usage) prove beneficial to performance, showing the importance of jointly preserving fine-grained tool details and grounding them in usage. Finally, eliminating the two-step design results in substantial performance degradation, validating its efficacy in reducing the LLM's burden to enhance reasoning.

### 4.4 In-depth Analysis

**Analysis on inference efficiency.** To investigate inference efficiency, we compare ToolRL, a representative TIR method, with our proposed TInR-U. We measure the number of user instructions each model can process per minute under varying tool set sizes, as shown in Figure 3. We can observe that ToolRL's inference speed consistently declines as the number of tools increases, which is due to the longer prompts required to append tool documentation and the resulting computational over-

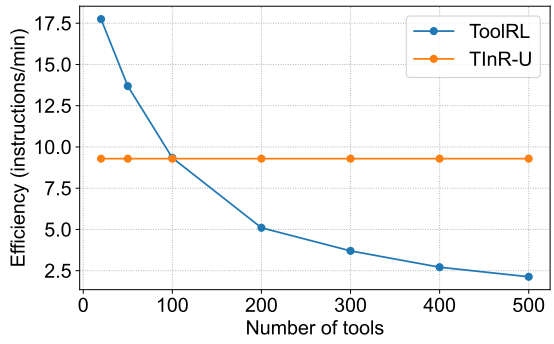


Figure 3: Comparison of inference efficiency of ToolRL and TInR-U under varying tool set sizes, measured in terms of instructions processed per minute. As the tool size increases, TInR-U demonstrates superior efficiency.

| Methods | Tool Identification | | Tool Calling | | |
|---|---|---|---|---|---|
| | EM | F1 | EM | T. Acc | P. Acc |
| ToolGen (Qwen) | 76.21 | 82.05 | 59.74 | 75.29 | 64.58 |
| **TInR-U (Qwen)** | **78.30** | **83.44** | **61.31** | **77.25** | **66.27** |
| ToolGen (LLaMA) | 70.59 | 76.44 | 51.37 | 67.58 | 57.12 |
| **TInR-U (LLaMA)** | **75.03** | **80.32** | **56.60** | **73.33** | **61.57** |
| ToolGen (Mistral) | 70.46 | 77.97 | 51.90 | 69.14 | 57.07 |
| **TInR-U (Mistral)** | **73.20** | **79.18** | **55.56** | **72.68** | **60.78** |

Table 5: Analysis of TInR-U on different base models.

head. In contrast, TInR-U maintains constant efficiency, since tool knowledge is already internalized in the model parameters without the need for extra prompt expansion. Notably, once the tool set size exceeds 100, TInR-U surpasses ToolRL and the efficiency gap widens with larger tool sizes. This demonstrates that our approach is well-suited for real-world scenarios with numerous tools.

**Analysis on base models.** We further examine the robustness of TInR-U across different backbone LLMs. Specifically, we substitute our base model Qwen-2.5B-Instruct with LLaMA-3.1-8B-Instruct (Dubey et al., 2024) and Mistral-7B-Instruct-v0.3 (Jiang et al., 2023). As shown in Table 5, while absolute performance varies with model capacity and architecture, TInR-U consistently outperforms ToolGen under all base models, demonstrating that our method is model-agnostic and can generalize effectively across different LLM backbones. Besides, Qwen-2.5B-Instruct achieves the strongest overall results compared to alternatives, suggesting that Qwen is more suitable as the base model for tool-internalized reasoning, likely due to its stronger built-in support for instruction following and tool usage.

| Methods | Tool Identification | | Tool Calling | | |
|---|---|---|---|---|---|
| | EM | F1 | EM | T. Acc | P. Acc |
| Numeric | 42.59 | 49.83 | 16.38 | 41.03 | 30.00 |
| Hierarchical | 46.72 | 52.24 | 19.14 | 43.14 | 32.07 |
| Semantic | 50.07 | 56.32 | 20.78 | 48.10 | 33.20 |
| **TInR-U** | **78.30** | **83.44** | **61.31** | **77.25** | **66.27** |

Table 6: Analysis on different internalization methods.

| Methods | Tool Identification | Tool Calling |
|---|---|---|
| DeepAgent | 62.35 | 55.82 |
| ToolRetriever+ToolRL | 60.78 | 53.99 |
| **TInR-U** | **78.30** | **61.31** |

Table 7: Comparison with DeepAgent.

**Analysis on tool internalization methods.** Our tool internalization method is built upon the atomic indexing strategy (Hao et al., 2023; Li et al., 2024a; Wang et al., 2025b), where each tool is assigned a dedicated token for internalization. To better understand its effectiveness, we compare it against three alternative internalization approaches: 1) Semantic indexing, which directly uses the tool name as its identifier, thereby relying on surface-level lexical semantics; 2) Numeric indexing, which assigns each tool a unique number, introducing a simple but semantically uninformative mapping; and 3) Hierarchical indexing, which clusters tools into a tree structure based on the semantic similarity of their documentation, and then assigns each tool a numerical path string from the root to its leaf. As shown in Table 6, our method achieves a large margin over all alternatives. These results demonstrate that our method provides unique and unambiguous representations that are easier for LLMs to memorize and recall, thereby greatly reducing confusion in tool identification and improving downstream tool calling accuracy.

**Comparison with agent-style framework.** We added an additional comparison with a recent agent-style framework DeepAgent (Li et al., 2026), which equips an LLM with iterative reasoning and a scalable tool-search mechanism to select appropriate tools from large toolsets. Results are shown in Table 7. These results suggest that DeepAgent improves over a traditional separated retrieval baseline (ToolRetriever+ToolRL), but still falls short of TInR-U. Moreover, agent-style tool retrieval typically incurs additional latency, e.g., "thinking time" before searching tools, which further reduces efficiency compared to both our approach and conventional separated retrieval pipelines.

| Methods | Tool Identification | | Tool Calling | |
|---|---|---|---|---|
| | ToolACE | BFCL | ToolACE | BFCL |
| ToolGen | 73.97 | 31.03 | 58.90 | 22.98 |
| **TInR-U** | **75.34** | **34.48** | **61.64** | **24.13** |

Table 8: Multi-turn tool-use results on ToolACE and BFCL datasets.

| Methods | Task 1 | Task 2 |
|---|---|---|
| | EM | EM |
| $\theta_1$ | 60.55 | 15.53 |
| $\theta_2$ | 58.29 | 60.52 |

Table 9: Experimental results in continual learning of TInR-U.

**Analysis on multi-turn tool-use.** To explicitly demonstrate our method's performance on multi-step and multi-turn tool use scenarios, we separately report results on the multi-step/multi-turn subset of ToolACE and the multi-turn category of BFCL. The results are reported in Table 8. From the results, we can see that our TInR-U consistently outperforms the strongest baseline ToolGen, indicating that our gains are not limited to single-turn tool calling.

**Analysis on continual learning.** We further assess our method's ability to handle continual learning. We first randomly split our test set into two subsets treated as Task 1 and Task 2, containing 398 and 367 samples respectively. We first train our base model on Task 1 to obtain $\theta_1$. Then, we extend the vocabulary to accommodate the new tools in Task 2 and continue training on Task 2 to obtain $\theta_2$. To mitigate forgetting, we adopted a rehearsal strategy by mixing a small subset of 50 Task 1 samples during Task 2 training. After training, we evaluated the tool calling accuracy of $\theta_1$ and $\theta_2$ on both tasks. From the results shown in Table 9, we observe a mild performance decrease on Task 1 from 60.55% to 58.29%, but a large improvement on Task 2 from 15.53% to 59.12%, indicating our methods' adaptability to new tasks.

We further conducted additional experiments to assess our model in tool-update scenarios. We simulate tool updates by leveraging GPT-5 to modify names or descriptions of tools and parameters in the test set. We found that the tool calling performance remains stable from 61.31% to 58.25%. We attribute this robustness to our two-step TInR design, where the LLM can refer to intermediate documentation before parameter filling.

For more substantial changes, we acknowledge that strict zero-shot deployment may not be feasible. However, the continual learning results suggest that modest adaptation training is sufficient to recover strong performance. Overall, this reflects an explicit trade-off: while our approach may bring some maintenance costs, it delivers substantial gains in both accuracy and inference efficiency, which we believe is worthwhile in many real deployment settings.

We conducted additional experiments including case study, deeper ablation and efficiency analysis in Appendix B.

## 5 Conclusion and Future Work

In this paper, we explore tool-internalized reasoning (TInR), aiming at reasoning with internalized tool knowledge without relying on tool documentation. We propose a novel framework TInR-U to endow LLMs with TInR capabilities under a three-phase training process. Extensive experiments in both in-domain and out-of-domain settings demonstrate that TInR-U consistently surpasses existing baselines, demonstrating both effectiveness and efficiency. Looking ahead, we intend to explore multi-modal scenarios involving tools for vision, speech, or robotics, which could further broaden the applicability of TInR.

## Limitations

1) The evaluation datasets may not fully capture the breadth of real-world tools. However, TInR demonstrates consistent improvements across both in-domain and out-of-domain settings, suggesting strong potential to generalize beyond the evaluated scenarios. 2) The tools in our datasets may include false negatives; for example, functionally similar tools that could in principle satisfy a user's instruction are not labeled as valid, potentially biasing tool accuracy evaluation. However, this issue is inherent to many tool datasets and, given that such cases are infrequent, their effect on our results is likely negligible.

## Ethics Statement

The dataset used in our work is derived from publicly available sources and generated through interactions with LLMs in English. Since the SFT reasoning data in our study are entirely simulated, user privacy is fully protected, and no real personal

information is included in the dataset. Furthermore, all scientific artifacts used in this research are publicly accessible for academic purposes under permissive licenses, and their use in this paper complies with their intended purposes. Given these considerations, we believe our research adheres to the ethical standards of the conference.

## Acknowledgments

The work described in this paper was supported by Research Grants Council of Hong Kong (PolyU/15207122, PolyU/15209724, PolyU/15213323, PolyU/15205325), PolyU internal grants (BDWP) and the National Natural Science Foundation of China under Grants 62476071.

## References

Guoxin Chen, Zhong Zhang, Xin Cong, Fangda Guo, Yesai Wu, Yankai Lin, Wenzheng Feng, and Yasheng Wang. 2025. Learning evolving tools for large language models. In *The Thirteenth International Conference on Learning Representations*.

Sijia Chen, Yibo Wang, Yi-Feng Wu, Qing-Guo Chen, Zhao Xu, Weihua Luo, Kaifu Zhang, and Lijun Zhang. 2024. Advancing tool-augmented large language models: Integrating insights from errors in inference trees. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Sergey Levine, and Yi Ma. 2025. SFT memorizes, RL generalizes: A comparative study of foundation model post-training. In *The Second Conference on Parsimony and Learning (Recent Spotlight Track)*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Zhibin Gou, Zhihong Shao, Yeyun Gong, yelong shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. 2024. ToRA: A tool-integrated reasoning agent for mathematical problem solving. In *The Twelfth International Conference on Learning Representations*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. 2023. Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings. In *Advances in Neural Information Processing Systems*, volume 36, pages 45870–45894. Curran Associates, Inc.

Shadi Iskander, Sofia Tolmach, Ori Shapira, Nachshon Cohen, and Zohar Karnin. 2024. Quality matters: Evaluating synthetic data for tool-using LLMs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 4958–4976. Association for Computational Linguistics.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning.

Chengpeng Li, Mingfeng Xue, Zhenru Zhang, Jiaxi Yang, Beichen Zhang, Xiang Wang, Bowen Yu, Binyuan Hui, Junyang Lin, and Dayiheng Liu. 2025a. Start: Self-taught reasoner with tools.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. 2025b. Search-o1: Agentic search-enhanced large reasoning models.

Xiaoxi Li, Wenxiang Jiao, Jiarui Jin, Guanting Dong, Jiajie Jin, Yinuo Wang, Hao Wang, Yutao Zhu, Ji-Rong Wen, Yuan Lu, and Zhicheng Dou. 2026. Deepagent: A general reasoning agent with scalable toolsets. In *Proceedings of the ACM Web Conference 2026*, WWW '26, page 2219–2230. Association for Computing Machinery.

Xuefeng Li, Haoyang Zou, and Pengfei Liu. 2025c. Torl: Scaling tool-integrated rl.

Yongqi Li, Wenjie Wang, Leigang Qu, Liqiang Nie, Wenjie Li, and Tat-Seng Chua. 2024a. Generative cross-modal retrieval: Memorizing images in multimodal language models for retrieval and beyond. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11851–11861. Association for Computational Linguistics.

Zekun Li, Zhiyu Chen, Mike Ross, Patrick Huber, Seungwhan Moon, Zhaojiang Lin, Xin Dong, Adithya Sagar, Xifeng Yan, and Paul Crook. 2024b. Large language models as zero-shot dialogue state tracker through function calling. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8688–8704. Association for Computational Linguistics.

Zhi Li, Yicheng Li, Hequan Ye, and Yin Zhang. 2024c. Towards autonomous tool utilization in language

models: A unified, efficient and scalable framework. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 16422–16432. ELRA and ICCL.

Qiqiang Lin, Muning Wen, Qiuying Peng, Guanyu Nie, Junwei Liao, Xiaoyun Mo, Jiamu Zhou, Cheng Cheng, Yin Zhao, Jun Wang, et al. 2025. Robust function-calling for on-device language model via function masking. In *The Thirteenth International Conference on Learning Representations*.

Weiwen Liu, Xingshan Zeng, Xu Huang, xinlong hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, Zezhong WANG, Yuxian Wang, Wu Ning, Yutai Hou, Bin Wang, Chuhan Wu, Wang Xinzhi, Yong Liu, Yasheng Wang, Duyu Tang, Dandan Tu, Lifeng Shang, Xin Jiang, Ruiming Tang, Defu Lian, Qun Liu, and Enhong Chen. 2025a. ToolACE: Enhancing function calling with accuracy, complexity, and diversity. In *The Thirteenth International Conference on Learning Representations*.

Yanming Liu, Xinyue Peng, Jiannan Cao, Shi Bo, Yuwei Zhang, Xuhong Zhang, Sheng Cheng, Xun Wang, Jianwei Yin, and Tianyu Du. 2025b. Tool-planner: Task planning with clusters across multiple tools. In *The Thirteenth International Conference on Learning Representations*.

Pan Lu, Bowen Chen, Sheng Liu, Rahul Thapa, Joseph Boen, and James Zou. 2025. Octotools: An agentic framework with extensible tools for complex reasoning. In *Workshop on Reasoning and Planning for Large Language Models*.

Elias Lumer, Vamse Kumar Subbiah, James A. Burke, Pradeep Honaganahalli Basavaraju, and Austin Huber. 2024. Toolshed: Scale tool-equipped agents with advanced rag-tool fusion and tool knowledge bases. *Preprint*, arXiv:2410.14594.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. 2025. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru WANG, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. 2025. ToolRL: Reward is all tool learning needs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

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

Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. 2025. From exploration to mastery: Enabling LLMs to master tools via self-driven interactions. In *The Thirteenth International Conference on Learning Representations*.

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

Zhengliang Shi, Yuhan Wang, Lingyong Yan, Pengjie Ren, Shuaiqiang Wang, Dawei Yin, and Zhaochun Ren. 2025. Retrieval models aren't tool-savvy: Benchmarking tool retrieval for large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 24497–24524. Association for Computational Linguistics.

Yunyue Su, Zhang Jinshuai, Bowen Fang, Wen Ye, Jinghao Zhang, Bowen Song, Weiqiang Wang, Qiang Liu, and Liang Wang. 2025. Toolscaler: Scalable generative tool calling via structure-aware semantic tokenization. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 556–578. Association for Computational Linguistics.

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*.

Hongru Wang, Cheng Qian, Wanjun Zhong, Xiusi Chen, Jiahao Qiu, Shijue Huang, Bowen Jin, Mengdi Wang, Kam-Fai Wong, and Heng Ji. 2025a. Acting less is reasoning more! teaching model to act efficiently.

Renxi Wang, Xudong Han, Lei Ji, Shu Wang, Timothy Baldwin, and Haonan Li. 2025b. Toolgen: Unified tool retrieval and calling via generation. In *The Thirteenth International Conference on Learning Representations*.

Junde Wu, Jiayuan Zhu, Yuyuan Liu, Min Xu, and Yueming Jin. 2025. Agentic reasoning: A streamlined framework for enhancing LLM reasoning with agentic tools. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 28489–28503. Association for Computational Linguistics.

Qiancheng Xu, Yongqi Li, Heming Xia, and Wenjie Li. 2024. Enhancing tool retrieval with iterative feedback from large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 9609–9619. Association for Computational Linguistics.

Qiancheng Xu, Yongqi Li, Heming Xia, Fan Liu, Min Yang, and Wenjie Li. 2025. PEToolLLM: Towards personalized tool learning in large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 21488–21503. Association for Computational Linguistics.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. 2025. Qwen3 technical report.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Siyu Yuan, Kaitao Song, Jiangjie Chen, Xu Tan, Yongliang Shen, Kan Ren, Dongsheng Li, and Deqing Yang. 2025. EASYTOOL: Enhancing LLM-based agents with concise tool instruction. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 951–972. Association for Computational Linguistics.

Jianguo Zhang, Tian Lan, Ming Zhu, Zuxin Liu, Thai Quoc Hoang, Shirley Kokane, Weiran Yao, Juntao Tan, Akshara Prabhakar, Haolin Chen, Zhiwei Liu, Yihao Feng, Tulika Manoj Awalgaonkar, Rithesh R N, Zeyuan Chen, Ran Xu, Juan Carlos Niebles, Shelby Heinecke, Huan Wang, Silvio Savarese, and Caiming Xiong. 2025. xLAM: A family of large action models to empower AI agent systems. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 11583–11597. Association for Computational Linguistics.

# A Implementation Details

In data construction, we employ Qwen3-14B as the LRM to generate reasoning data. Our candidate tool set consists of three parts: the ground-truth tools along with 5 tools retrieved using ToolRetriever, and the remaining tools randomly sampled. We train TInR-U based on Qwen2.5-7B-Instruct (Yang et al., 2024). In Phase 1 training of TInR-U, we fine-tune the Qwen2.5-7B-Instruct model with a learning rate set to $5e{-}5$, a batch size of $64$ and a warm-up ratio of $0.1$, for $8$ epochs. The weighting factor $\alpha$ and $\gamma$ are set to 1. In Phase 2 training of TInR-U, we fine-tune with a learning rate set to $5e{-}6$ and a batch size of $64$ for $4$ epochs. For reinforcement learning in Phase 3, we use GRPO with a learning rate set to $2e{-}6$ and a batch size of $128$ for $20$ epochs. We have trained the model several times to ensure the improvement is not randomly achieved and present the mid one. To accelerate the memorization of tools that without instructions, we generate pseudo-instructions for these tools. Since the maximum context length varies in different LLMs, we constrain the context window to 4096 tokens. The experiments are conducted on NVIDIA 5880 GPUs with 48 GB of memory.

# B Additional Experiments and Analysis

## B.1 Case Study

To further demonstrate how internalized tool knowledge manifests in reasoning behavior, we provide more qualitative evidence and deepen the ablation analysis by conducting a concrete case study comparing TInR-U against its variant without bidirectional knowledge alignment. As shown in the Figure 4, when the user requests information about the ‘Fnatic‘ League of Legends team and the full friends list of a Steam user, TInR-U correctly grounds its reasoning on the available tool tokens (e.g., ‘«get_teams_and_players»‘ and ‘«user_friends_list»‘ ). In contrast, the ablated model tends to hallucinate tool tokens and parameter names (e.g., inventing ‘_esports‘ and ‘_friends‘ tools and ‘teamname‘parameters). This highlights that the bidirectional knowledge alignment in our framework not only improves scores but also leads to more faithful, schema-consistent reasoning traces.

## B.2 More Ablation Analysis

The ablation results in Table 4 shows that the effect of removing recall is minor compared with memorization and usage in our current two-step TInR design. To further probe the impact of recall, we conducted an additional ablation in the single-step setting, where the LLMs generate the complete tool calls directly without associating intermediate tool tokens to documentation. The results are summarized in Table 10. We observe that in the single-step setting, removing recall leads to substantially larger drops, especially on parameter accuracy. This indicates that when the reasoning and parameter-filling burden is higher, recall contributes more strongly to model performance. Thus, our findings suggest that the two-step architecture itself already simpli-

Case Study

**User:**
Could you check for any giveaways on the Xbox One platform and also on the Android platform?

**Assistant (TInR-U):**
<think>
Okay, let's see. The user wants information about the 'Fnatic' team in League of Legends Esports and their players. Also, they need the full friends list of a Steam user with ID '77788899900011122'.
First, for the League of Legends Esports team, there's the «get_teams_and_players» tool. The parameter is the team name, which is 'Fnatic'. So I'll use that tool with team_name set to 'Fnatic'.
Next, the Steam friends list. The «user_friends_list» tool requires the steamid, which is given as '77788899900011122'...
</think>
...
<tool_call>
{"token": "«get_teams_and_players»", "parameters": {"name": "Fnatic"}}
{"token": "«user_friends_list»", "parameters": {"is_id": "77788899900011122"}}
</tool_call>

**Assistant (TInR-U wo/ bidirectional knowledge alignment):**
<think>
Okay, let's tackle the user's request step by step. First, they want information about the 'Fnatic' team in League of Legends Esports and their players.
Looking at the available tools, there's the _esports function which takes a team name. The default is 'lng-esports', but the user specified 'Fnatic', so I'll need to set the parameter to 'Fnatic'.
Next, the user also wants the full friends list of a Steam user with ID '77788899900011122'. The _friends tool requires a user_id as an integer. The provided ID is a string, but the tool accepts integers, so I'll convert it to 77788899900011122...
</think>
...
<tool_call>
{"token": "_esports", "parameters": {"teamname": "Fnatic"}}
{"token": "_friends", "parameters": {"is_id": "77788899900011122"}}
</tool_call>

Figure 4: A case study comparing TInR-U against its variant without bidirectional knowledge alignment.

fies the recall burden, and recall becomes more crucial when the model must internally manage more complex parameter inference.

### B.3 More Efficiency Analysis

To broaden the scalability evidence in efficiency, we conducted additional experiments on (i) different hardware and (ii) a larger model scale. We first replace the NVIDIA 5880 GPUs used in our original experiments with NVIDIA A6000 GPUs and measure efficiency across tool sets ranging from 200 to 500 tools. The results are summarized in Table 11. As shown, both models run slightly slower on A6000 GPUs, likely due to architectural and memory bandwidth differences. However, TInR-U consistently maintains around 9 instructions/min on both hardware platforms and remains substantially faster than ToolRL. This confirms that the scalability advantage of our approach is robust to hardware variation. Due to computational resource constraints to fully finetune a larger 14B model, we apply LoRA-based training on Qwen2.5-14B-Instruct and measure inference efficiency. We observe that TInR-U achieves 7.41 instructions/min, which still outperforms ToolRL. These results show that the TInR-U scales reliably across both hard-

| Methods | Tool Calling | | |
|---|---|---|---|
| | EM | T. Acc | P. Acc |
| two-step | 61.31 | 77.25 | 66.27 |
| *w/o recall* | 59.74(-2.6%) | 75.29(-2.5%) | 64.58(-2.6%) |
| single-step | 43.40 | 72.94 | 50.72 |
| *w/o recall* | 40.13(-7.5%) | 70.20(-3.8%) | 46.67(-8.0%) |

Table 10: Ablation of recall in two-step and single-step design of TInR-U on tool calling performance.

| Methods | NVIDIA 5880 | NVIDIA A6000 |
|---|---|---|
| ToolRL | [2.13, 5.10] | [2.07, 4.86] |
| TInR-U | 9.29 | 8.97 |

Table 11: Comparison of inference efficiency of ToolRL and TInR-U with toolsets ranging from 200 to 500 tools across different hardware.

| Methods | Tool Identification | Tool Calling |
|---|---|---|
| | EM | EM |
| $\alpha$=0.5, $\beta$ =1.0 | 76.73 | 59.74 |
| $\alpha$=2.0, $\beta$ =1.0 | 77.51 | 60.52 |
| $\alpha$=1.0, $\beta$ =0.5 | 77.12 | 58.69 |
| $\alpha$=1.0, $\beta$ =2.0 | 78.56 | 61.05 |
| $\alpha$=1.0, $\beta$ =1.0 | 78.30 | 61.31 |

Table 12: Sensitivity analysis of hyperparameters in TInR-U.

ware platforms and model sizes. We further evaluate the inference efficiency of ToolGen for comparison under the same setting, and observe that ToolGen achieves 9.47 instructions/min, which is comparable to our TInR-U with 9.29 instructions/min. The slight improvement is likely because our model is trained with GRPO which is known to increase reasoning length; however, this is acceptable given our performance gains. Importantly, both ToolGen and TInR-U are substantially faster than ToolRL when the tool set exceeds 100 tools, indicating that TInR is essential for maintaining fast inference on large toolsets.

### B.4 Hyperparameter Sensitivity Analysis

We conducted additional sensitivity analysis on the weighting factors $\alpha$ and $\beta$ with varying values. As shown in the Table 12, the performance of TInR-U remains stable across a broad range of hyperparameter settings, with fluctuations within only 1–2 points for both tool identification and tool calling accuracy. This indicates that our method does not rely on delicate tuning and that the default setting used in the paper is already near-optimal and robust.

### B.5 Theoretical Discussion

To further elucidate the effect of bidirectional knowledge alignment, we provide a theoretical discussion of its underlying principles. Similar to many works in tool learning and representation alignment, our bidirectional knowledge alignment is motivated by its effect on representation alignment across three levels: (i) tool documentation $\rightarrow$ (ii) tool-token embeddings $\rightarrow$ (iii) the LLM's internal language space. By enforcing alignment in both directions (i.e., memorization and recall), the LLM learns to ground tool semantics in a way that is consistent for both discrimination (i.e., identifying the appropriate tool) and generation (i.e., producing accurate parameters), thereby facilitating both fine-grained preservation of tool details and holistic understandings of tool functionalities.

## C Prompt Details

The prompt template for inference are shown in Figure 5.

Prompt for TInR inference

**Prompt:**
You are a helpful assistant capable of leveraging your tool knowledge to address the user's query given inside <user> and </user>. First, you should conduct reasoning inside <think> and </think>. During reasoning, you can generate special tool tokens from your vocabulary to recall your tool knowledge. If one or more tools are needed, identify all the corresponding tool tokens inside <tool_token> and </tool_token>. Next, you will be given the corresponding tool documentation of all valid tool tokens. You should specify the tool token and corresponding parameters inside <tool_call> and </tool_call> to invoke the tool call.

Output Format for Tool Identification:
<think> Your reasoning process containing special tool tokens (if possible) </think>
<tool_token>
Tool token ...
</tool_token>

Output Format for Tool Calling:
<tool_call>
"token": "Tool token", "parameters": "Parameter name": "Parameter value", "... ...": "... ..."
"token": "... ...", "parameters": "... ...": "... ...", "... ...": "... ..." ...
</tool_call>

Remember:
1. You must always include the <think>, <tool_token> or <tool_call> fields to outline your reasoning and then specify tool tokens or tool calls. 2. You can invoke multiple tool calls in the <tool_call> field, where each should be specified in JSON format with a "token" field and an "parameters" field containing a dictionary of parameters. If no parameters are needed, leave the "parameters" field an empty dictionary. 3. Refer to the Dialogue Records History, including the user's previous queries, previous tool calls or responses noted as <tool_call> or <response>, and any tool feedback noted as <obs> (if exists).

Figure 5: The prompt for TInR inference.