# EfficientTool: A Cost-Effective Aligning Framework for Tool-Conditioned Agents in SME Scenarios

**Yuanqi Mu[1,2], Bingfeng Pi[2], Defei Xia[2], Lei Zuo[2], Yongqi Zhang[1] *.**

[1]Hong Kong University of Science and Technology (Guangzhou)
[2]TianjuDihe (Suzhou) Technology Co., Ltd.

## Abstract

Large language models (LLMs) are increasingly adopted in downstream industries, yet aligning proprietary agents remains challenging due to limited high-quality data and hardware constraints in small and medium-sized enterprises (SMEs). We propose **EfficientTool**, a cost-effective, tool-conditioned alignment framework forming a closed loop over data collection, iterative training, and deployment-oriented evaluation. EfficientTool adopts a self-evolving bootstrapping-based Trajectory Collection Pipeline for high-quality trajectory generation, followed by iterative Model Training Pipeline using tool-conditioned parameter-efficient fine-tuning (PEFT) (Houlsby et al., 2019). We evaluate the model with Interaction and Evaluation Pipeline in public and private benchmarks, and deploy for an internal enterprise agent. Results show that EfficientTool effectively aligns model in SME scenarios while preserving general tool-calling capability.

## 1 Introduction

Recent advances in LLMs accelerate their adoption across downstream industrial applications (Wang et al., 2024; Xu et al., 2025), yet aligning proprietary agent systems remains challenging. SMEs typically operate under limited GPU budgets, small engineering teams, and highly heterogeneous tool ecosystems, while still requiring reliable task-level correctness and controllability. The primary bottleneck lies in the acquisition of high-quality interaction data, the cost of iterative alignment, and the risk of catastrophic forgetting during scenario adaptation (Luo et al., 2025). Most existing works (von Werra et al., 2020; Sheng et al., 2024) focus on model training itself, with less attention paid to the challenge of acquiring customized data.

Mainstream alignment approaches, including Supervised Fine-Tuning (SFT) (Devlin et al., 2019), reinforcement learning from human feedback (RLHF) (Ouyang et al., 2022), and pair-wise preference learning, typically assume access to special datasets or additional reward models. However, these assumptions rarely hold in highly fragmented and personalized SME scenarios, where private data is costly to collect manually (Kandpal and Raffel, 2025) and additional models incur extra costs and risks.

To address these, we propose **EfficientTool**, a cost-effective, tool-conditioned agent alignment framework for resource-constrained SMEs. EfficientTool prioritizes two core objectives: **high-quality, labor-efficient data acquisition** and **effective alignment with generalization preservation**. We apply it to real-world enterprise support scenarios involving heterogeneous SME tasks, integrating over 700 APIs and 100 domain-specific datasets for dataset recommendation, API assistance, and customer support(details in 5). The main contributions are as follows:

- We present **EfficientTool**, a cost-effective alignment framework designed for tool-conditioned agents. EfficientTool addresses challenges of SMEs through the design of three pipelines.
- We introduce a self-evolving **Trajectory Collection Pipeline** based on bootstrapping and experience replay. By sampling based on the model's performance and annotating data with the teacher model, the framework ensures high data quality while effectively reducing human labor costs.
- We introduce an iterative **Model Training Pipeline** utilizing parameter-efficient fine-tuning (PEFT) (Houlsby et al., 2019). This minimizes GPU memory constraints and time consumption, enabling efficient agent alignment and domain adaptation without sacrificing the model's general tool-calling competency.

*Corresponding Author, yongqizhang@hkust-gz.edu.cn

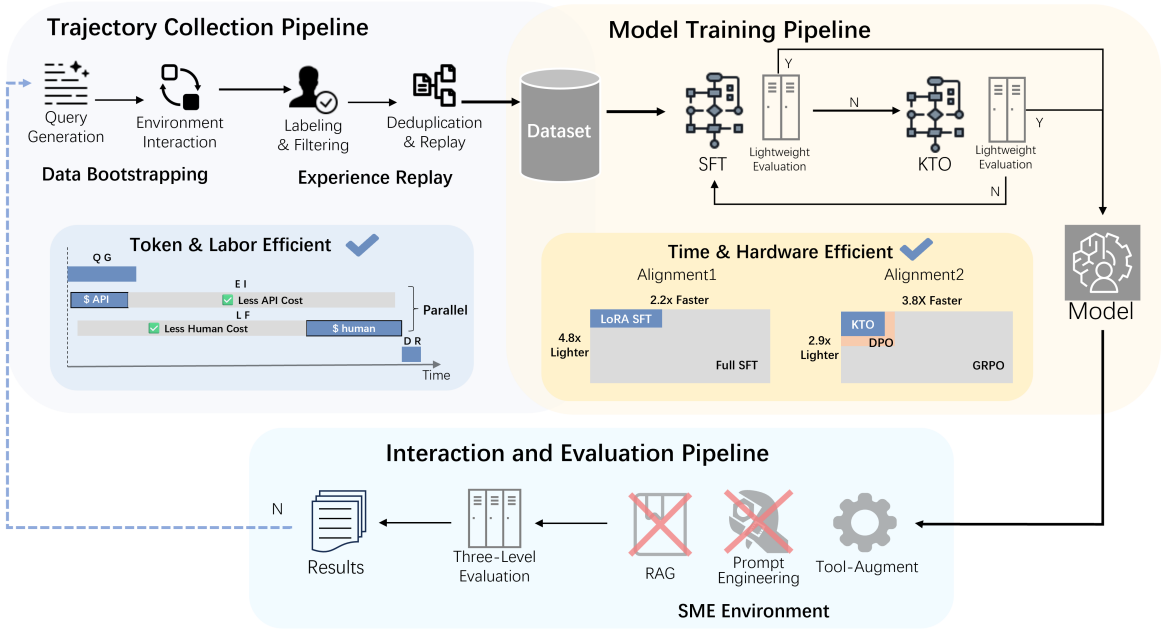


Figure 1: Architectural overview of EfficientTool. Trajectory Collection Pipeline first leverages data bootstrapping and experience replay to collect high-quality data, which are utilized in Model Training Pipeline. Our methods (blue) are more efficient than others (gray, orange) in four domains. A lightweight evaluation determines the transition to the next alignment module, while Interaction and Evaluation outputs real-world result to trigger the next global iteration. This iterative loop ensures the framework continuously refines its tool-use capabilities.

- We conduct extensive evaluation using **Interaction and Evaluation Pipeline** on interaction-based private benchmark and static public benchmarks. The results demonstrate strong alignment performance and validate the framework's cost-effectiveness in real-world deployment.

## 2 Related Work

In the context of agent alignment, Acikgoz et al. (2025) explore a variant of SFT to improve generalization ability. However, SFT-based approaches often struggle to handle ambiguous or underspecified scenarios, where deeper alignment is required.

Traditional RLHF (Schulman et al., 2017) provides a more expressive alignment paradigm but typically incurs substantial overhead in reward and value models. Recent method GRPO (Shao et al., 2024) and its variants (Yu et al., 2025; Zheng et al., 2025; Zhao et al., 2025; Ding et al., 2025) partially alleviate these costs, and have been applied to tool-use training like NVIDIA Nemotron (Zhang et al., 2025b). Nevertheless, they rely on heuristic or rule-based reward signals, which provide limited guidance for prompt-driven tool usage (details in appendix A.2).

Preference learning methods (Rafailov et al., 2023; Amini et al., 2024; Meng et al., 2024; Hong et al., 2024) have gained attention due to their reduced reliance on reward models and lower resource requirements. Despite these advantages, such methods depend on pair-wise preference data, the collection of which is costly and difficult in highly customized and resource-constrained SMEs.

## 3 The EfficientTool Framework

In this section, we introduce the **EfficientTool** framework, as illustrated in Figure 1.

### 3.1 Trajectory Collection Pipeline

This is the first step of each iteration, which collects interaction trajectory (the formatted and simplified example can be found in the appendix A.1) for later alignment.

#### 3.1.1 Data Bootstrapping Module

We use data bootstrapping-based on query seed to obtain trajectories, including the following steps:

**Query Generation.** Basic queries are crafted manually from functional descriptions, followed by the generation of equally challenging variants by a teacher model utilizing tool documentation, basic queries and existing historical experience. This step can be executed several times until the query seeds achieve adequate coverage of domains,

querying styles, and tool-use patterns, such that the seed set is not dominated by a narrow background or a single query template.

**Environment Interaction**. We use the previous query set to generate interaction trajectories. Initially, Qwen3-Max (Yang et al., 2025) is used to generate first 20% interaction trajectories, which is later replaced by the trained model after light SFT to balance data acquisition efficiency and data quality.

#### 3.1.2 Experience Replay Module

Inspired by RLHF, we adopt a replay-like module to clean the data, including the following steps:

**Labeling and Filtering**. All generated trajectories undergo triple filtering: (1) **Rule Layer**: Verify trajectory integrity (including valid tool calls), tool call legality (parameter type/range compliance). (2) **Teacher Model Layer**: The teacher model with strict review prompt outputs binary labels and structured reasons for each trajectory with low false positive rate. (3) **Manual Review Layer**: Manually inspect negative trajectories for preference alignment and conduct proportional sampling of positive cases based on the model's performance, which can save 81% labor costs (details in Appendix A.4, compare to LLM + Human method).

**Experience Deduplication and Replay**. We use user queries as the primary key and 15 n-grams strategy (Shao et al., 2024) to deduplicate new data before entering the dataset.

### 3.2 Model Training Pipeline

This constitutes the second stage of each iteration, where it leverages the collected data and two aligning algorithms to refine the behavior of LLM.

#### 3.2.1 SFT Module

We use SFT as the basic training paradigm to ensure stable tool calling ability and multi round dialogue coherence. To better simulate real-world interaction environments, we dynamically bind actual retrieval tools within each conversational turn. Furthermore, Low-Rank Adaptation (LoRA) algorithm (Hu et al., 2022) is employed to mitigate significant distribution shifts from the base model while preserving its inherent generalization and reasoning capabilities. These architectural adjustments are consistently integrated across all training strategies.

#### 3.2.2 KTO Module

To address the challenges of aligning algorithms mentioned in Section 2, We adopt the Kahneman-Tversky Optimization (KTO) algorithm (Ethayarajh et al., 2024), whose loss function is:

$$\mathcal{L}_{\text{KTO}} = \begin{cases} \lambda_D \left( \beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)} - KL \right) \\ \lambda_U \left( KL - \beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)} \right) \end{cases}, \quad (1)$$

where $\lambda_D/\lambda_U$ denotes desirable/undesirable factor, KL denotes KL divergence between new policy and reference policy, and $\beta > 0$ has a similar effect in the DPO (Rafailov et al., 2023) loss. KTO directly optimizes absolute utility rather than relative preference probability, eliminating the need for pair-wise data construction. Although supervision at the trajectory level, KTO yields more localized optimization signals for tool selection decisions, leading to improved tool-use training efficiency.

To optimize this process, we implement a **Lightweight Evaluation** after each alignment, using the automation and human level metrics detailed in Section 3.3.1. This dynamic strategy transitions are illustrated in Figure 1: (1) If tool-call biases persist, the training strategy transitions to KTO; (2) If fundamental errors occur, the pipeline iteration restarts. LoRA weights of the passed model are merged into the backbone for subsequent phases.

### 3.3 Interaction and Evaluation Pipeline

This pipeline provides feedback for the training transitions in the second stage and conducting the final performance audit. To ensure a fair evaluation, we disabled adaptive prompt generation, tool format correction and other auxiliary modules in the enterprise environment. The performance of the model determines the commencement of the next global iteration.

#### 3.3.1 Three-Level Evaluation Module

To assess the model, we establish a three-level evaluation module. We construct several test cases tailored to the functional complexity of each scenario. Notably, each global iteration only necessitates the addition of new test cases, rather than reconstruct the entire benchmark from scratch. The three-level evaluation are detailed as follows:

**Automation Level**. We employ the 4T metric (Fu et al., 2025; Xia et al., 2026), including task

completion rate (TCR), task failure rate (TFR), task incomplete rate (TIR), and task performance score (TPS), where TCR + TFR + TIR = 1 (details in appendix C.1, proven in appendix C.2).

**LLM Level**. Inspired by (Wang et al., 2025), we use LLM as an auxiliary evaluator to score trajectory quality along five dimensions: Accuracy, Appropriateness, Coverage, Coherence, and Faithfulness. In the experiment, we used a closed-source model for evaluation, while in actual deployment, we will use a teacher model to save costs.

**Expert Level**. To mitigate multi-turn randomness and the variability of tool invocation paths, we conduct binary (accept/reject) evaluations by 10 senior software architects with 8+ years experience. Final judgments are determined by majority voting.

#### 3.3.2 Tool-Augment Module

We observe that when a single-turn query is long or includes many intents, the target tool is frequently drowned out by irrelevant candidates. To address this, we introduce two context-aware retrieval enhancements (evaluations in Appendix C.4):

**Intent Refinement.** We design a prompt to instruct the LLM to perform explicit intent identification if the query requires external tools beyond the model's base capability.

**Multi-Path Vector Retrieval**. (1) **Cosine-Max**: Cosine similarity is computed both against the original query and against the refined intent vector. The higher of the two scores is selected and the final tools are returned using the Kneedlle algorithm (Satopaa et al., 2011). (2) **RRF-CCLI**: The top-k tools are returned after multi-path vectors are fused using Reciprocal Rank Fusion (RRF) (Cormack et al., 2009). It ensures low computational overhead and full interpretation ability without introducing additional trainable parameters.

New requirements are first validated here, and the evaluation outcomes determine whether the next global iteration is initiated. This design connects the end of the previous loop and the beginning of a new loop.

## 4 Experiments

This section presents the experimental setup, the performances of aligned model and the evidences of efficiency.

### 4.1 Experimental Setup

#### 4.1.1 Implementation Details

All experiments are conducted on a compute node equipped with 8 NVIDIA A100 GPUs (40GB), connected via PCIe. The base model was Qwen3-32B (Yang et al., 2025), the teacher model was DeepSeek-R1-Distill-Llama-70B (DeepSeek-AI, 2025), and the embedding model was Qwen3-Embedding-0.6B (Zhang et al., 2025c). Training was carried out on existing open-source infrastructure, LLaMA-Factory (Zheng et al., 2024), with lightweight task-specific modifications. Interaction-base evaluations are conducted within the Jenius agent System (Xia et al., 2026) without adaptive prompt generation, tool format correction and other auxiliary modules.

#### 4.1.2 Datasets and Benchmarks

The SFT dataset contains 20K samples, with sequence lengths ranging from 4K to 24K tokens. It consists: (1) 4000 desensitized enterprise log data; (2) 6000 samples generated by the Trajectory Collection Pipeline; (3) approximately 10,000 open source samples, including Chinese Markdown[1], GSM8K(Yu et al., 2023), ShareGPT-GPT4[2], and Leetcode1000[3].

The KTO dataset contains 4000 samples ranging from 4K to 8K tokens, with the same proportional design as (Shao et al., 2024). It combines SFT data with curated samples derived from failure patterns observed after SFT (see Appendix B.2).

For tool retrieval, we use XLAM (Zhang et al., 2025a) as the public benchmark and a private benchmark consisting of 3K transformed SFT samples. For multi-round samples, we use short-term memory prompts to summarize the history and prepend the summary to each query, thereby transforming an $n$-turn interaction into $n$ single turns.

For model performance evaluation, we adopt MCPToolBench++ (Fan et al., 2025), BFCL-v3 (Patil et al., 2025), and ToolACE (Liu et al., 2024) as public benchmarks (details in appendix B.3). In addition, we construct a private benchmark of 210 challenging multi-round conversations, covering a wide range of tool invocation scenarios (tool distribution in Appendix B.5).

[1] https://huggingface.co/datasets/rojasdiego/chinese-markdown

[2] https://huggingface.co/datasets/shibing624/sharegpt_gpt4

[3] https://huggingface.co/datasets/RayBernard/leetcode1000

| | Automation Level | | | | LLM Level | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Model | TCR | TFR | TIR | TPS | Accuracy | Appropriateness | Coverage | Coherence | Faithfulness |
| deepseek-v3.2 | 0.534 | 0.232 | 0.232 | 0.571 | 0.703 | 0.884 | 0.723 | 0.953 | 0.761 |
| glm-4.6 | 0.481 | 0.349 | 0.169 | 0.498 | 0.683 | 0.778 | 0.688 | 0.814 | 0.707 |
| kimi-k2 | 0.719 | 0.084 | 0.195 | **0.744** | 0.776 | 0.914 | 0.784 | 0.949 | 0.813 |
| Llama-3.3-70B-Instruct | 0.439 | 0.201 | 0.359 | 0.448 | 0.388 | 0.528 | 0.365 | 0.644 | 0.420 |
| Nemotron-3-Nano-30B | 0.481 | 0.370 | 0.148 | 0.502 | 0.623 | 0.801 | 0.637 | 0.903 | 0.670 |
| Qwen2.5-32B-Instruct | 0.597 | 0.343 | **0.058** | 0.602 | 0.713 | 0.889 | 0.685 | 0.966 | 0.765 |
| Qwen3-32B | 0.656 | 0.164 | 0.179 | 0.668 | 0.700 | 0.870 | 0.685 | 0.952 | 0.745 |
| ETool-sft | **0.724** | 0.111 | 0.164 | 0.739 | 0.799 | 0.941 | **0.823** | **0.968** | 0.832 |
| ETool-kto | 0.714 | **0.089** | 0.195 | 0.729 | **0.812** | **0.951** | 0.812 | 0.966 | **0.839** |

Table 1: Performance on Private Benchmark in Automation and LLM Level. **ETool** represents the models trained with EfficientTool framework. The best results are highlighted in **bold**, while the second-best results are underlined.

| | BFCL | TOOLACE | | MCPToolBench++ | |
|---|---|---|---|---|---|
| Model | w/o p | w/o p | w/ p | w/o p | w/ p |
| Qwen3-32B | 0.707 | 0.873 | 0.965 | 0.460 | 0.925 |
| ETool-sft | 0.693 | 0.858 | 0.921 | 0.567 | 0.915 |
| | ↓1.98% | ↓1.70% | ↓4.54% | ↑23.31% | ↓1.12% |
| ETool-kto | 0.700 | 0.854 | 0.949 | 0.533 | 0.919 |
| | ↓1.10% | ↓2.34% | ↓1.71% | ↑15.86% | ↓0.66% |

Table 2: Performance on Public Benchmark. The contents with arrow represent the difference of general ability compared to the original model. *w/o p* means without prompt while *w/ p* means with prompt.

| | Expert Level | | |
|---|---|---|---|
| Model | Accept | Reject | Ratio |
| deepseek-v3.2 | 160 | 50 | 76.19% |
| glm-4.6 | 166 | 44 | 79.04% |
| kimi-k2 | 184 | 26 | 87.62% |
| Llama-3.3-70B-Instruct | 108 | 102 | 51.43% |
| Nemotron-3-Nano-30B | 124 | 86 | 59.04% |
| Qwen2.5-32B-Instruct | 166 | 44 | 79.04% |
| Qwen3-32B | 164 | 46 | 78.09% |
| ETool-sft | 193 | 17 | 91.90% |
| ETool-kto | 200 | 10 | **95.24%** |

Table 3: Performance on private benchmark in Expert Level. The best results are highlighted in **bold**, while the second-best results are underlined

.

#### 4.1.3 Evaluation Metrics

We use the three-level evaluation method in the **Interaction and Evaluation Pipeline** as the evaluation metrics. we use Qwen3-Plus as the judge LLM and the prompt is shown in appendix C.3.

### 4.2 Evaluation on Tool Calling Tasks

#### 4.2.1 Performance on Public Benchmarks

We compare our models against nine baselines with comparable or larger parameter scales, including closed-source models (Liu et al., 2025; Team et al., 2025b,a), open-source models explicitly trained for tool usage (Grattafiori et al., 2024; NVIDIA, 2025; Yang et al., 2024), and our two trained variants, ETool-sft and ETool-kto. ETool-sft represents the model after SFT alignment, while ETool-kto is aligned using KTO based on ETool-sft.

When using the original benchmark interactions, Llama-3.3 frequently responded with "no suitable tools for the task" on TOOLACE and MCPTooLBench++. To mitigate this issue, we report results under both the original and the prompted setting, while BFCL is evaluated using its original interaction (details in appendix B.4 and appendix C.5).

As shown in Table 2, the trained models exhibit only a modest performance change—about 1–2% in *w/o p*—compared to the base model. They also achieve a notable improvement on MCPToolBench++, which likely comes from the original interaction being more ambiguous and more tool-selection-heavy, which is closer to our target failure modes. From an industrial perspective, even minor regressions in general capability can result in significant maintenance overhead and operational risk. The observed performance stability therefore represents a critical success of catastrophic forgetting (Luo et al., 2025) avoidance.

| Method | Avg ($) | Total ($) | Quality | Time |
|---|---|---|---|---|
| H + H | 8.605 | 86K | Medium | Slow |
| LLM + H | 4.305 | 43K | **High** | Medium |
| LLM + LLM | 0.010057 | 100 | Low | **Fast** |
| Ours | 0.818 | 8180 | **High** | **Fast** |

Table 4: Cost comparison of different annotation methods. H represents human, and the former one represents interaction way while the latter represents filtering.

#### 4.2.2 Performance on Private Benchmark

Table 1 reports the results at the Automation Level and LLM Level. In terms of overall performance, ETool-sft and ETool-kto exhibit complementary strengths: The SFT model performs more prominently at the Automation Level, achieving higher TCR and stronger expression-related scores; whereas the KTO model performs better overall at the LLM Level, and also achieves a reduction in the TFR metric.

Table 3 presents the results at the Expert-Level. To ensure comparability, experts assessed complete trajectories from prior interactions rather than isolated responses. ETool-kto achieves the highest acceptance rate, indicating that KTO substantially mitigates portion of the failure patterns, where SFT alignment alone was insufficient to fully meet user requirements.

This reflects the difference between rule-based metrics and human judgment. Although we have taken into account as many tool invocation schemes as possible in the 4T metric, supplemented by LLM to assist in the judgment of model final responses and tool parameters, it still cannot fully bridge the gap between the two methods.

### 4.3 Evaluation of Efficiency

We provide a conservative estimation of the operational costs associated with the **EfficientTool** framework. Table 4 quantifies the expenditures for 10K samples, substantiating cost-effective without compromising quality (complete calculation process and conditions to in Appendix A.4). Seed-design cost is excluded from the table because it is a one-off insider coordination cost, not per-sample annotation cost. Our solution strikes a balance between high labor costs and data quality, achieving high data quality with reduced latency and lower costs. Despite the extremely low cost of utilizing LLMs for the entire process of data generation, they fail to yield data suitable for subsequent training in customized SME scenarios. Although manual prompt tuning can alleviate this issue, it implicitly introduces unquantified human costs and leads to a vicious cycle of periodic prompt adjustments.

| Method | GPU Mem (GB) | Time (h) | Entries |
|---|---|---|---|
| Full SFT* | 120 | 110 | 20K |
| LoRA SFT | **25** | **50** | 20k |
| GRPO* | 100 | 25 | 4k |
| DPO | 40 | 8 | 4k |
| KTO | **35** | **6.6** | 4k |

Table 5: Comparison of resource consumption for different alignment methods. *GRPO* and *Full SFT* values are extrapolated from initial training logs and theoretical calculation. **GPU Mem** refers to the memory usage of each GPU under pipeline parallelism conditions.

The results in Table 5 incorporate DeepSpeed ZeRO-3 memory optimization (Rajbhandari et al., 2020), Liger Kernel (Hsu et al., 2025), pipeline parallelism(Harlap et al., 2018), and PCIe interconnects under same compute node and same data scales each stage in the experiment before. For RLHF and preference alignment, LoRA algorithm was employed equally. **GPU Mem** refers to the memory usage of each GPU under pipeline parallelism conditions. The results validate that the integrated SFT+KTO strategy via LoRA demonstrably improves resource efficiency, thereby minimizing both training latency and GPU memory consumption. These results between time, cost, and hardware efficiency are visually showed in Figure 1.

## 5 Deployment Scenario

We have integrated EfficientTool into DataSteward, an agent designed for enterprise-level customer service scenarios. The system currently incorporates over 700 APIs and 100 domain-specific datasets spanning diverse fields such as mechanical engineering, medicine, humanities, and natural language processing.

We leverage desensitized historical customer service interactions, API and dataset documents, and the Trajectory Collection Pipeline to generate QA data and tool-augmented interaction data. We first utilize SFT with tool-augmented data for initial alignment, followed by KTO tailored to API debugging and dataset recommendation scenarios for further refinement. The aligned agent supports functionalities including dataset recommendations, API usage assistance, and conventional customer support responses. The figure of deployment process is in the appendix D.1.

Traditional human-mediated support lacks the

necessary technical depth and dataset familiarity, yielding a proactive satisfaction rate of only **21.6%** and incurring queuing latencies.

Conversely, agent perfectly resolves **39.6%** of inquiries and generally addresses another **27.7%**, with only 32.6% requiring human escalation caused by inherent permission constraints.

Facing a **30.5%** response gap caused by surging business volume and constrained service staffing in 2025, we evaluated our system under simulated maximum peak concurrency requests. The results indicate that the model maintains robust performance, delivering timely responses within predefined tolerance limits.

## 6 Conclusion

We present EfficientTool, a comprehensive, cost-effective framework specifically designed to bridge the gap between LLM alignment and the practical constraints of SME environments. We establish a closed loop between self-evolving trajectory collection, iterative PEFT-based training, and interaction-centric evaluation. Through teacher model bootstrapping, sampling labeling, and PEFT algorithms, we effectively reduce the burden on time, cost, and hardware. Experiments on both public benchmarks and real-world private benchmark demonstrate that EfficientTool achieves effective alignment while preserving general tool-calling capabilities, supporting practical and iterative optimization of LLM agents in SME settings. Deployment results demonstrate strong service efficiency and user satisfaction in real-world scenarios. Moving forward, we aim to extend EfficientTool to more complex, cross-domain industrial applications.

## Limitations

There are still several expandable aspects to this work. Firstly, the current system focuses on the design of SME tool calling scenarios, emphasizing high-precision single tool retrieval and reliable execution; Due to limitations in the research and development cycle and existing training frameworks, we have not yet conducted systematic modeling, policy optimization, and scale validation for concurrent calling of multiple tools. Secondly, the SME industry categories covered by private datasets are not yet extensive enough, and the cross industry generalization performance needs further evaluation. The above directions are the focus of our subsequent research.

## Ethics Statement

This study strictly adheres to ethical guidelines for the application of artificial intelligence and does not involve human subject experiments, personal privacy data collection, or interventions in sensitive areas. All experiments are based on publicly available benchmark datasets and private business logs authorized by authorized enterprise partners for desensitization (PII information such as user identity, contact information, financial details, etc. have been removed and approved in writing by the enterprise for technical verification). We explicitly declare that this work does not encourage, support, or implement any fully automated proxy deployment that bypasses manual review; All technical designs are based on the premise of enhancing human efficiency and ensuring controllability and traceability.

## Acknowledgments

This work was jointly supported by The Hong Kong University of Science and Technology and Tianju Dihe (Suzhou) Technology Co., Ltd. This work was also partially supported by Guangdong Provincial Natural Science Foundation 2025A1515010304, Guangdong Province Project 2024QN11X088, Guangzhou Science and Technology Planning Project 2025A03J4491. The authors sincerely thank the units for providing substantial support in terms of computing resources, industrial scenario data docking, and engineering implementation guidance.

## References

Emre Can Acikgoz, Jeremiah Greer, Akul Datta, Ze Yang, William Zeng, Oussama Elachqar, Emmanouil Koukoumidis, Dilek Hakkani-Tur, and Gokhan Tur. 2025. Can a single model master both multi-turn conversations and tool use? coalm: A unified conversational agentic language model. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12370–12390.

Afra Amini, Tim Vieira, and Ryan Cotterell. 2024. Direct preference optimization with an offset. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 9954–9972.

Xiaoyin Chen and Sam Wiseman. 2023. Bm25 query augmentation learned end-to-end. *arXiv preprint arXiv:2305.14087*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan

Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90 https://lmsys.org/blog/2023-03-30-vicuna/. LMSYS Org.

Gordon V Cormack, Charles LA Clarke, and Stefan Buettcher. 2009. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 758–759.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

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.

Yuyang Ding, Chi Zhang, Juntao Li, Haibin Lin, Xin Liu, and Min Zhang. 2025. Fapo: Flawed-aware policy optimization for efficient and reliable reasoning. *arXiv preprint arXiv:2510.22543*.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*.

Shiqing Fan, Xichen Ding, Liang Zhang, and Linjian Mo. 2025. Mcptoolbench++: A large scale ai agent model context protocol mcp tool use benchmark. *arXiv preprint arXiv:2508.07575*.

Yuchuan Fu, Xiaohan Yuan, and Dongxia Wang. 2025. Ras-eval: A comprehensive benchmark for security evaluation of llm agents in real-world environments. *arXiv preprint arXiv:2506.15253*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Aaron Harlap, Deepak Narayanan, Amar Phanishayee, Vivek Seshadri, Nikhil Devanur, Greg Ganger, and Phil Gibbons. 2018. Pipedream: Fast and efficient pipeline parallel dnn training. *arXiv preprint arXiv:1806.03377*.

Jiwoo Hong, Noah Lee, and James Thorne. 2024. Orpo: Monolithic preference optimization without reference model. In *2024 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR.

Pin-Lun Hsu, Yun Dai, Vignesh Kothapalli, Qingquan Song, Shao Tang, Siyu Zhu, Steven Shimizu, Shivam Sahni, Haowen Ning, Yanning Chen, and Zhipeng Wang. 2025. Liger-kernel: Efficient triton kernels for LLM training. In *Championing Open-source DEvelopment in ML Workshop @ ICML25*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Nikhil Kandpal and Colin Raffel. 2025. Position: The most expensive part of an llm should be its training data. *arXiv preprint arXiv:2504.12427*.

Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, and 1 others. 2025. Deepseek-v3.2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*.

Weiwen Liu, Xu Huang, Xingshan Zeng, 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, Xinzhi Wang, Yong Liu, Yasheng Wang, and 8 others. 2024. Toolace: Winning the points of llm function calling. *Preprint*, arXiv:2409.00920.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. 2025. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *IEEE Transactions on Audio, Speech and Language Processing*.

Shiho Matta, Yin Jou Huang, Fei Cheng, Hirokazu Kiyomaru, and Yugo Murawaki. 2025. Optimizing cost-efficiency with llm-generated training data for conversational semantic frame analysis. In *Proceedings of the 9th Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature (LaTeCH-CLfL 2025)*, pages 238–251.

Yu Meng, Mengzhou Xia, and Danqi Chen. 2024. Simpo: Simple preference optimization with a reference-free reward. *Advances in Neural Information Processing Systems*, 37:124198–124235.

NVIDIA. 2025. Nemotron 3 Nano: Open, efficient mixture-of-experts hybrid Mamba-Transformer model for Agentic reasoning. Technical report.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

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*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE.

Clemens Rawert, Marc Klingen, and Maximilian Deichmann. 2023. Langfuse — open-source llm engineering platform. Software available from https://langfuse.com.

Ville Satopaa, Jeannie Albrecht, David Irwin, and Barath Raghavan. 2011. Finding a" kneedle" in a haystack: Detecting knee points in system behavior. In *2011 31st international conference on distributed computing systems workshops*, pages 166–171. IEEE.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2024. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*.

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, Zhuofu Chen, Jialei Cui, Hao Ding, Mengnan Dong, Angang Du, Chenzhuang Du, Dikang Du, Yulun Du, Yu Fan, and 150 others. 2025a. Kimi k2: Open agentic intelligence. *Preprint*, arXiv:2507.20534.

V Team, Wenyi Hong, Wenmeng Yu, Xiaotao Gu, Guo Wang, Guobing Gan, Haomiao Tang, Jiale Cheng, Ji Qi, Junhui Ji, Lihang Pan, Shuaiqi Duan, Weihan Wang, Yan Wang, Yean Cheng, Zehai He, Zhe Su, Zhen Yang, Ziyang Pan, and 69 others. 2025b. Glm-4.5v and glm-4.1v-thinking: Towards versatile multimodal reasoning with scalable reinforcement learning. *Preprint*, arXiv:2507.01006.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. 2020. TRL: Transformers Reinforcement Learning.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, and 1 others. 2024. Openhands: An open platform for ai software developers as generalist agents. *arXiv preprint arXiv:2407.16741*.

Yubo Wang, Xueguang Ma, Ping Nie, Huaye Zeng, Zhiheng Lyu, Yuxuan Zhang, Benjamin Schneider, Yi Lu, Xiang Yue, and Wenhu Chen. 2025. Scholarcopilot: Training large language models for academic writing with accurate citations. *CoRR*.

Defei Xia, Bingfeng Pi, Shenbin Zhang, Song Hua, Yunfei Wei, and Lei Zuo. 2026. Jenius agent: Towards experience-driven accuracy optimization in real-world scenarios. *arXiv preprint arXiv:2601.01857*.

Weikai Xu, Chengrui Huang, Shen Gao, and Shuo Shang. 2025. Llm-based agents for tool learning: A survey. *Data Science and Engineering*, pages 1–31.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 40 others. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *CoRR*.

Jianguo Zhang, Tian Lan, Ming Zhu, Zuxin Liu, Thai Quoc Hoang, Shirley Kokane, Weiran Yao, Juntao Tan, Akshara Prabhakar, Haolin Chen, and 1 others. 2025a. 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.

Shaokun Zhang, Yi Dong, Jieyu Zhang, Jan Kautz, Bryan Catanzaro, Andrew Tao, Qingyun Wu, Zhiding Yu, and Guilin Liu. 2025b. Nemotron-research-tool-n1: Tool-using language models with reinforced reasoning. *arXiv preprint arXiv:2505.00024*.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, and 1 others. 2025c. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.

Yuzhong Zhao, Yue Liu, Junpeng Liu, Jingye Chen, Xun Wu, Yaru Hao, Tengchao Lv, Shaohan Huang, Lei Cui, Qixiang Ye, and 1 others. 2025. Geometric-mean policy optimization. *arXiv preprint arXiv:2507.20673*.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, and 1 others. 2025. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.

# A Details on Framework Design

## A.1 Example of Trajectory

In order to visually demonstrate the **trajectory** mentioned in our work, we provide the following example. Due to the length of the original sample, we have omitted some unimportant and sensitive content.

- **Human:**

  The culture of integrity is the sum of knowledge, concepts, norms, beliefs, and corresponding lifestyles and social evaluations about integrity. Its core is to advocate the values of innocence, fairness, frugality, and self-discipline. It is not only a basic requirement for public officials, but also an excellent cultural gene that integrates into various aspects of social life. Please express your poster creativity around the following directions: ... Generate several posters.

- **Tools:**

  [

  {"name": "poster_generation",...},

  {"name": "text2image",...},

  {"name": "edit_image",...},

  {"name": "understand_image",...},

  {"name": "juhewin_marketing_activity",...},

  {"name": "create_pptx_task",...},

  ...

  ]

- **Function_call:**

  {"name": "poster_generation", "arguments": {"prompt": "...", "wh_ratios": "vertical version"}}

- **Observation:** {"success": true, "message": "Image generation successful, link valid within 24 hours, please download and save in a timely manner!", "image_url": "https://example.com", ïmage_info̊: {}}

- **Function_call:**

  {"name": "poster_generation", "arguments": {"prompt": "...", "wh_ratios": "vertical version"}}

- **Observation:**

  {"success": true, ...}

- **Function_call:**

  {"name": "poster_generation", "arguments": {"prompt": "...", "wh_ratios": "vertical version"}}

- **Observation:**

  {"success": true, ...}

- **Assistant:**

  Here are three themed posters generated for you to promote a culture of integrity, focusing on three directions: family ethics and education, diligence and frugality, and traditional culture. ...

- **Human:**

  Generate a few simple ones again.

- **Tools:** [

  {"name": "poster_generation",...},

  {"name": "text2image",...},

  {"name": "edit_image",...},

{"name": "generate_word_cloud_chart",...},

{"name": "generate_liquid_chart",...},

{"name": "generate_pie_chart",...},

...

]

- **Function_call:**

  {"name": "poster_generation", "arguments": {"prompt": "...", "wh_ratios": "vertical version"}}

- **Observation:**

  {"success": true, "message": "Image generation successful, link valid within 24 hours, please download and save in a timely manner!", "image_url": "https://example.com", ïmage_infő: {}}

- **Function_call:**

  {"name": "poster_generation", "arguments": {"prompt": "...", "wh_ratios": "vertical version"}}

- **Observation:**

  {"success": true, ...}

- **Function_call:**

  {"name": "poster_generation", "arguments": {"prompt": "...", "wh_ratios": "vertical version"}}

- **Observation:**

  {"success": false, "message": "Task creation failed: Invalid task, task create failed: {"status_code": ..., "request_id": ..., "code": ..., "message": "Requests rate limit exceeded, please try again later.", "output": null, "usage": null}", "image_url": null, "image_info": {}}}

- **Function_call:**

  {"name": "poster_generation", "arguments": {"prompt": "..., "wh_ratios": "vertical version"}}

- **Observation:**

  {"success": true, ...}

- **Assistant:**

  Here are three newly generated minimalist style "Promote Clean Culture" themed posters, designed around rule of law spirit, social trends, and red culture. The overall design adopts minimalist composition, restrained colors, and highly condensed visual language, highlighting conceptual expression and aesthetic whitespace. ...

We first batch-download the data from Langfuse (Rawert et al., 2023) platform. We then concatenate single-turn records into multi-turn interactions based on thread_id, remove data that is irrelevant to training, and retain dynamically matched tool information to obtain an initially cleaned dataset. Subsequently, we apply the filtering procedures described in the main text and convert the data into a ShareGPT (Chiang et al., 2023) like format for storage in the database.

### A.2 Tool Call Classification

After careful observation and summary, tool calling tasks can be divided into two typical paradigms:

1. **Prompt Driven**: Agents need to understand semantic intent and generate descriptive prompts to call the corresponding tools. Appendix A.1 is a good example of this.

2. **Schema Filling**: Agent must accurately parse the structural requirements and generate compliant format. The parameter values generated by the model usually have strict format and value requirements, and the core parameters even have only one correct value. Below is a simplified example:

   - **human:**
     Generate a five dimensional radar chart with dimensions of communication ability, teamwork, professional skills, time management, and innovation ability, corresponding scores of 85, 78, 92, 88, and 80, respectively.
   - **Tools:**
     [
     {"name": "generate_radar_chart",...},
     {"name": "generate_venn_chart",...},
     {"name": "generate_funnel_chart",...},
     {"name": "generate_fishbone_diagram",...},
     {"name": "generate_pie_chart",...},
     {"name": "generate_violin_chart",...},
     ...
     ]

- **Function_call:**
  {"name": "generate_radar_chart", "arguments": {"data": [{"name": "communication ability", "value": 85}, {"name": "teamwork", "value": 78}, {"name": "professional skills". "value": 92}, {"name": "time management", "value": 88}, {"name": "innovation ability", "value": 80}]}}
- **Observation:**
  {"success": true, ...}
- **Assistant:**
  I have generated a five dimensional ability assessment radar chart for you! It can be clearly seen from the figure that:...

RLHF faces distinct challenges depending on the tool interface. While rule-based rewards are preferable for prompt driven tools, schema filling tools require learned reward models. These requirements reveal a critical bottleneck: the heavy reliance on extensive training, parameter tuning, and the delicate task of balancing rigid rule-based logic with scoring-model outputs, all of which pose substantial barriers to cost-effective deployment.

### A.3 Query Length Distribution

Figure 2 shows the length distribution map of all the multi turn valid queries in our private benchmark. This serves as a basis for choosing the appropriate threshold length to trigger the intent recognition in Tool-Augment module.

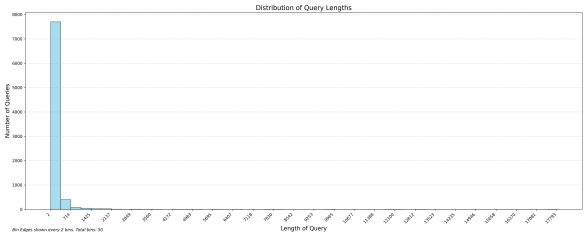


Figure 2: Query Length Distribution Map.

### A.4 Price Calculation

From Ali Bailian platform and (Matta et al., 2025), we can obtain and assume that:

- The labor cost for each annotation or generation is $4.3 per item.
- The input cost of the LLM is 0.0032 Yuan per 1000 tokens.
- The output cost of the LLM is 0.0128 Yuan per 1000 tokens.
- The average number of input tokens during interaction is 7000.
- The average number of output tokens during interaction is 1000.
- Due to query seeds are generated by insiders and involve the coordination between the business department and the technical department, these costs are difficult to calculate. So they are ignored here.
- Our method needs 2000 data generated from closed-source LLM among 10000 data.
- Assume the accuracy of light-sft model is 0.9 (evaluation results in table 6), then 1000 (teacher's judge) + 900 (proportional sampling) = 1900 data need human refinement.

| Model | Metric | |
|---|---|---|
| ETool-sft-light | TCR | 0.719 |
| | TFR | 0.111 |
| | TIR | 0.169 |
| | TPS | 0.730 |
| | Accuracy | 0.789 |
| | Appropriateness | 0.940 |
| | Coverage | 0.807 |
| | Coherence | 0.973 |
| | Faithfulness | 829 |
| | Accept Ratio | 89.5% |

Table 6: Evaluation results of ETool-sft-light on three level.

| Method | Formula ($) |
|---|---|
| Human + Human | 7000/1000*0.0032/7 + 1000/1000*0.0128/7 + 4.3*2 = 8.605 |
| LLM + Human | 8.605 - 4.3 = 4.305 |
| LLM + LLM | (7000/1000*0.0032/7 + 1000/1000*0.0128/7) *2 = 0.010057 |
| Ours | 2000/10000*0.0050285 + 1900/10000*4.3 = 0.818057 |

Table 7: Cost Comparison of Different Annotation Methods. *0.0050285* is the result of interaction API cost for one sample.

- Seed-design cost is excluded from the table because it is a one-off insider coordination cost, not per-sample annotation cost.

Then we can get the process in Table 7.

### A.5 Human Effort Breakdown

We provide a detailed breakdown of human effort involved in the data construction and evaluation process.

- **Query-seed design:** Conducted by 2 domain insiders, each contributing approximately 3 hours.
- **Manual refinement:** A total of 1,900 trajectories were reviewed by 3 annotators, each spending around 13 hours.
- **Final evaluation:** 210 trajectories were independently assessed by 10 senior software architects, with approximately 8 hours per expert.

Overall, human effort is moderate and primarily concentrated on seed design and hard-case inspection. The seed-design cost is excluded from Table 4, as it is a one-off coordination effort among domain experts rather than a per-sample annotation cost.

### A.6 Dynamic Matching Algorithm Adaptability

| Method | Accuary |
|---|---|
| Cosine-max | 0.998 |
| RRF-CCLI | 0.117 |

Table 8: The adaptability of the proposed algorithm to dynamic matching algorithms. Here we take Kneedle as an example and test it on the same private benchmark.

Table 8 shows the results of applying Kneedle algorithm to Cosine-Max and RRF-CCLI in private tool retrieval benchmark. Both algorithms use the optimal parameter configuration and apply the original Kneedle algorithm. It indicates that RRF-CCLI is not well compatible with dynamic matching algorithms due to inherent issues with the RRF algorithm.

### A.7 Application of Algorithms with Different Lengths

Table 9 shows the results of applying proposed method (Cosine-Max as an example) in private tool retrieval benchmark with different token thresholds.

| Lengths | Top-5 | Top-10 |
|---|---|---|
| 0 | 0.971 | 0.992 |
| 50 | 0.965 | 0.989 |
| 100 | 0.956 | 0.985 |
| 200 | 0.950 | 0.985 |
| 500 | 0.947 | 0.983 |
| 1000 | 0.943 | 0.982 |
| 1500 | 0.940 | 0.980 |
| 2000 | 0.933 | 0.966 |

Table 9: Application of Algorithms with Different Lengths. The proposed algorithm with intent recognition is employed when the length exceeds the threshold; otherwise, the traditional cosine algorithm is used. The length distribution of the benchmark refers to the appendix A.3

It can be seen that as more queries use the original cosine method instead of our method, the accuracy of Top-5 and Top-10 shows a downward trend. But this trend is relatively flat, which creates space for us to find a balance between response time and retrieval accuracy.

For scenarios that only require a few seconds or tens of seconds, spending an extra 3 seconds (according to Table 10) may affect the user experience. However, for complex scenarios that require several minutes or even longer, or where large models get stuck in a dead loop, 3 seconds seems negligible.

### A.8 Response Time Accounting

| Operation | Statistical | Time/Speed |
|---|---|---|
| Prompt Throughput | Avg | 1000 tokens/s |
| Generation Throughput | Avg | 12 tokens/s |
| One-turn Chat | Med | 40.79 s |
| One-turn Chat | Min | 12.18 s |
| One-turn Chat | Max | 269 s |
| Embedding Tools | Avg | 2 s |
| Embedding Query | Avg | 50 ms |
| Vector Similarity | Avg | 0.4 ms |
| Cosine | Avg | 53 ms |
| Cosine-Max | Avg | 107 ms<br>+54 ms |
| RRF-CCLI | Avg | 112 ms<br>+59 ms |

Table 10: Time consumption of different operations and algorithms.

Table 10 shows complete results in one compute node with 8 A100 (40G) GPUs with PCIe. We use Qwen3-32B as LLM and use Qwen3-Embedding-

0.6B. The data used for accounting time is the private benchmark of tool calling in the main body.

Our intent refinement prompt is 600 tokens while output is 20 tokens (for simplicity round up to 24 tokens). Thus our methods require an additional 2.605 seconds plus query length divided by Prompt Throughput.

We assume that the query length is **1k** and the median duration of the original conversation is 40.79 seconds. So we need an additional 3.605 seconds, which is 8.84% longer than the median one chat time. However, this ratio is likely to be overestimated, as we can see a maximum of 269 seconds, indicating that agent may require more time when dealing with complex requirements.

Based on observations in this sector and appendix A.7, we adopt 1K tokens as a threshold to decide whether to invoke computationally intensive strategies.

## B Details on Datasets and Benchmarks

### B.1 SFT Training Dataset

The following introduces the use of open-source datasets in the SFT training set:

- The **chinese-markdown** datasetis a curated collection of Markdown documents primarily in Chinese, extracted from diverse web pages.

  When a user query has long markdown content with instructions, the model is difficult to recognize the boundaries of the true markdown content and construct the correct prompt and parameters for tool-calling. Therefore, we used this dataset to construct some Q&A questions about the content of markdown to enhance the model's ability in this regard.

- The **MetaMathQA-GSM8K-zh** dataset is the Chinese-translated version of the MetaMathQA dataset, generated using GPT-3.5-Turbo with few-shot prompting. It contains approximately 231,685 augmented question-answer pairs derived from the GSM8K training set , focusing on mathematical reasoning tasks such as arithmetic word problems.

  When constructing prompts and parameters involving mathematical operations, the model sometimes makes mistakes. Therefore, we add some data to relieve this problem.

- The **leetcode1000** dataset is a structured, multi-dimensionally annotated corpus of algorithmic problems, designed to systematically organize and navigate over 1,000 programming challenges from LeetCode.

  Many of our users like to practice on coding platforms such as Leetcode, but the model cannot provide answers that pass most of the test points. Therefore, we added these data to meet the needs of users.

- The **sharegpt-gpt4** dataset is a high-quality, multi-turn dialogue corpus generated by GPT-4 and meticulously filtered. Its core academic value lies in providing premium samples for the SFT phase of large language models, aiming to enhance model capabilities in intent understanding and complex conversational reasoning through high-quality human-preference-aligned data.

  We add this data to ensure that the model does not have catastrophic forgetting in terms of its ability to call general tools.

- Through security testing, we find that the model is unable to recognize some dangerous and inductive instructions. Therefore, we collected different types of data that the model should **refuse to answer**, including prejudice discrimination, crimes, swearing insults, moral ethics, negative induction, target hijacking, prompt leakage, and property privacy.

### B.2 SFT Failure Scenario

The following introduces the four scenarios where performance is still disappointing after SFT:

- The model may encounter language confusion when faced with vague language requirements in user queries.

- The model faces the situation where there is existing historical information but the model needs to repeatedly call the tools to obtain the latest data.

- The model ignores tool descriptions and has a limited scope of use for small scope tools rather than large scope tools.

- The model does not call widely used tools frequently enough. These tools can enhance the vividness of model responses.

After KTO alignment, the first three issues can fully meet the demands of users, and the fourth issue has also been greatly improved.

### B.3 Open-source Benchmarks

The following introduces the use of open-source benchmarks in the tool retrieval and calling evaluation:

- The **XLAM** is a high-quality function-calling instruction dataset created and publicly released by Salesforce AI Research, specifically designed to enhance the tool-using capabilities of AI agents. To ensure data quality, the dataset was generated via the APIGen automated pipeline and subjected to a stringent three-tier validation process encompassing format checking, actual function execution, and semantic verification, which guarantee data reliability and diversity.

- **MCPToolBench** is a comprehensive benchmark developed based on MCP for evaluating the tool calling and planning capabilities of large language model agents in real complex tasks. Unlike previous benchmarks, it builds an ecosystem covering finance, science, travel, and other fields, and requires the model to handle fuzzy natural language tasks that require multi tool, multi-step collaboration, in order to more comprehensively test the model's higher-order capabilities, such as long-term planning, cross domain coordination, and understanding of task dependencies.

- **bfcl-v3** we use is a subset of datasets extracted from the widely used Berkeley Function Jialing Leaderboard v3 benchmark, whose core is specifically designed to evaluate the function calling capabilities of large language models. The model needs to accurately select and call one or more suitable functions from a given function list based on user queries, generating the correct parameters without the need for multiple conversations back and forth.

- **ToolACE** is a benchmark jointly proposed by Huawei Noah's Ark Laboratory, University of Science and Technology of China, and other institutions. Through its unique tool of self evolutionary synthesis, multi-agent interactive dialogue generation, and double-layer verification mechanism, it generates training and testing data that is accurate, complex, and diverse.

### B.4 Additional Prompt

Since any modification to the original benchmark should be explicitly specified, we distinguish between the w/o p and w/ p settings in our experiments. We use a simple example to illustrate the difference between w/o p and w/p, which is used in TOOLACE and MCPToolBench++ benchmark.

- **w/o p**

  **System:**

  [

  {name:start_codegen_session,...},

  {name:end_codegen_session,...},

  {name:get_codegen_session,...},

  {name:clear_codegen_session,...},

  ...

  ]

  **Human:**

  Click the button within the iframe identified by #embedFrame using the selector #search-input.

- **w/ p**

  **System:**

  You are a helpful assistant.

  You have access to the following tools.

  When you need to use a tool, you MUST respond ONLY with a JSON object in the following format, and nothing else:

  {{"name": "tool_name", "arguments": {{"arg_name": "value"}}}}

  Tools available:

  {tools}"

  **Human:**

  Click the button within the iframe identified by #embedFrame using the selector #search-input.

### B.5 Distribution of Tools

Figure 3 shows the distribution and classification of our MCP tools. In the figure, the area of the outermost ring represents the number of tools within each subcategory. The entire toolkit is divided into five core areas, covering a total of 29 sub functional nodes.

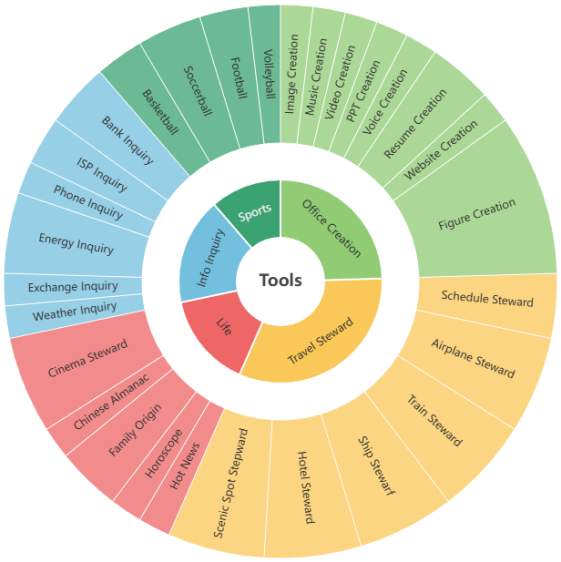


Figure 3: Distribution and classification of MCP tools. In the figure, the area of the outermost ring represents the number of tools within each subcategory.

- Office Creation - Green Zone

  This field is the largest part of the toolkit, mainly focusing on the generation of multimedia content and the improvement of office efficiency. Specifically, it includes:

  Media generation: Image Creation, Music Creation, Video Creation, Voice Creation.

  Documentation and Presentation: PPT Plugin Creation, Resume Creation, Website Creation, and Chart/Portrait Creation.

- Travel Steward - Orange Zone

  This field focuses on closed-loop services for itinerary planning and transportation accommodation, including:

  Transportation ticketing: Airplane Steward, Train Steward, Ship Steward.

  Travel support: Schedule Steward, Hotel Steward, and Scenic Spot Steward.

- Information Inquiry - Blue Area

  This field provides basic data interface support for retrieving real-time or industry-specific information, including:

  Finance and Livelihood: Bank Inquiry, Exchange Inquiry, Energy Inquiry.

  Life services: ISP Inquiry, Phone Inquiry, and Weather Inquiry.

- Life Services - Red Zone

  This field covers cultural entertainment and daily information acquisition:

  Entertainment and Information: Cinema Steward, Real time Hot News.

  Tradition and Individuals: Chinese Almanac, Family Origin, and Horoscope.

- Sports - Deep Green Zone

  This field categorizes rules, data, or event information for specific sports events and currently includes four core ball games:

  Categories: Basketball, Soccer, American Football, and Volleyball.

# C Details on Experiments

## C.1 4T Metric

Below are the evaluation details of the 4T metrics. We make some improvements to conditional discrimination based on the original author (Fu et al., 2025).

- **Task Completion Rate (TCR)**.

  Actual tool set and expected tool set are perfectly matched and results are appropriate.

  There is a situation where tools are called repeatedly, but the expected tool set is exactly the same as the actual tool set used and the returned result is valid

- **Task Failure Rate (TFR)**.

  Due to LLM issues, exceptions or errors occur during the interaction process, leading to interruptions in the interaction process.

  LLM did not call any tools and this task requires LLM to call tools. LLM itself does not have the ability to implement relevant functions.

- **Task Incomplete Rate (TIR)**.

  Tools outside the expected tool set are called.

  The actual tool set is a subset of the expected tool set.

  Actual tool set and expected tool set are perfectly matched but results are not appropriate.

- **Task Performance Score (TPS).**

  The calculation formula for TPS is shown below, where $C_c$ is the size of intersection of expected tools and actual tools, $W_c$ is the number of tools actually called but not expected, $M_c$ is the number of tools which are expected but not invoked. In the evaluation experiment, we set $\lambda_w = \lambda_m = 1.0$.

$$TPS(i) = \frac{C_c}{C_c + \lambda_w \times W_c + \lambda_m \times M_c} \quad (2)$$

$$TPS_{avg} = (1/N) \times \sum TPS(i) \quad (3)$$

Appendix C.2 proves that $TCR + TIR + TFR = 1$

## C.2 Proof of Formula

Let $S$ be the universal set of all task execution instances. We define three orthogonal dimensions for any task $s \in S$:

- **Execution State ($E$):** $E_c$ (Completed) and $E_i$ (Interrupted/Failed to initiate).
- **Tool Matching ($M$):** $M_p$ (Perfect match) and $M_m$ (Mismatch, including subsets and supersets).
- **Result Validity ($V$):** $V_a$ (Appropriate/Valid) and $V_n$ (Inappropriate/Invalid).

Based on the appendix C.1, we can define the metrics as subsets of $S$:

$$TCR = E_c \cap M_p \cap V_a \quad (4)$$

$$TFR = E_i \quad (5)$$

$$TIR = (E_c \cap M_m) \cup (E_c \cap M_p \cap V_n) \quad (6)$$

To prove $TCR + TFR + TIR = 1$, we must show that their union equals $S$ and they are mutually exclusive.

*Proof.* Consider the sum of the three rates:

$$\begin{aligned} Sum &= TCR + TIR + TFR \\ &= (E_c \cap M_p \cap V_a) \\ &\quad + [(E_c \cap M_m) \cup (E_c \cap M_p \cap V_n)] \\ &\quad + E_i \end{aligned}$$

```
# Role
You are a senior reviewer with years of experience in natural language processing
(NLP), specializing in evaluating multimodal or text generation tasks. You have
extremely high logical rigor and can accurately capture subtle semantic
differences between reference answers and generated answers.

# Task
Please compare the quality performance of "reference answers" and "agent generated
answers" in answering "user questions".

# Input Data
- **User Query**: {query}
- **Reference Answer**: {gt_answer}
- **Answers Generated by Agent**:
{response}

# Evaluation Dimensions & Rules
Please strictly adhere to the definitions of the following five dimensions for
scoring(0.0000 - 1.0000):

1. **Accuracy**: Accuracy of facts.
Evaluate whether all the facts mentioned in the generated answer (such as person,
time, place, causal relationship, etc.) are consistent with the reference answer.
If there is a factual error, the score should significantly decrease.
2. **Appropriateness**: Scene relevance.
Evaluate whether the voice, tone, and content of the generated answer are
accurately aligned with the user's intention of the question. If the answer is
correct but does not answer the question or contains a lot of irrelevant nonsense,
this deduction will be made.
3. **Coverage**: Key information coverage.
Break down the reference answer into several core points and evaluate the
percentage covered by the generated answer. The coverage ratio is the core basis
for this rating.
4. **Coherence**: Language coherence.
Evaluate whether the grammar of the generated answer is correct, whether the
logical connection is natural, and whether there are any repetitive or interrupted
sentences. 1.0000 represents fluent expression at the native language level.
5. **Faithfulness**: Loyalty/Compliance.
Evaluate whether the generated answer strictly follows the constraints in the
problem, and whether the logical deduction process is consistent with the input
context, without any hallucination phenomenon.

# Scoring Guidelines
- **0.8001-1.0000**: Good match/good performance.
- **0.6001-0.8000**: There are minor flaws (such as improper rhetoric, omission of
non core information).
- **0.4001-0.6000**: There are obvious defects (such as some factual errors and
missing key points).
- **0.0000-0.4000**: Serious errors or complete unavailability (such as severe
hallucinations, answering irrelevant questions).

utOutput Constraint Constraint
- Must output in a valid JSON format.
- Prohibited from including any opening remarks, explanatory notes, Markdown code
block identifiers (such as ```json ).
- Only output pure JSON strings.

# Output Format
{{
  "Accuracy": 0.0000,
  "Appropriateness": 0.0000,
  "Coverage": 0.0000,
  "Coherence": 0.0000,
  "Faithfulness": 0.0000
}}
```

Figure 4: The prompt for LLM level evaluation.

Factor out the completion state $E_c$ from the first two terms:

$$\begin{aligned} Sum = E_c \cap [(M_p \cap V_a) \cup M_m \cup (M_p \cap V_n)] \\ + E_i \end{aligned}$$

Group the terms involving $M_p$:

$$(M_p \cap V_a) \cup (M_p \cap V_n) = M_p \cap (V_a \cup V_n)$$

Since $V_a \cup V_n = U$ (the results are either valid or invalid), we have $M_p \cap U = M_p$.

Substituting back into the summation:

$$Sum = E_c \cap [M_p \cup M_m] + E_i$$

Since $M_p \cup M_m = U$ (the tool set either matches

perfectly or it does not), we have:

$$Sum = (E_c \cap U) + E_i$$
$$Sum = E_c \cup E_i$$

By definition, a task is either successfully completed through the workflow or it is interrupted/not initiated. Thus, $E_c \cup E_i = S$.

Therefore, the probabilities satisfy:

$$P(TCR) + P(TIR) + P(TFR) = 1 \quad (7)$$

□

### C.3 LLM Evaluation Prompt

The prompt template we used for LLM level evaluation is shown in figure 4.

### C.4 Performance of Tool-Augment Module

We conducted separate experiments on the Tool-Augment Module. Recognizing that the private benchmark are substantially more challenging and ambiguous, we report Top-1 to Top-5 accuracy on public benchmarks, and Top-5, Top-10, Top-20 accuracy on private benchmarks.

Table 11 reports the tool retrieval performance on XLAM. The results show that both fusion strategies consistently achieve strong performance across Top-$k$ settings, demonstrating their effectiveness in general tool retrieval scenarios.

| | XLAM | | | | |
|---|---|---|---|---|---|
| Method | Top-1 | Top-2 | Top-3 | Top-4 | top-5 |
| cosine | 0.852 | 0.977 | 0.996 | 1 | 1 |
| L2 | 0.291 | 0.501 | 0.708 | 0.892 | 0.923 |
| IP | 0.863 | **0.990** | **0.999** | 1 | 1 |
| W-BC | 0.822 | 0.969 | 0.994 | 0.999 | 1 |
| Cosine-Max | 0.836 | 0.977 | 0.997 | 1 | 1 |
| RRF-CCLI | **0.864** | 0.987 | 0.998 | 1 | 1 |

Table 11: Test results of tool retrieval on open source benchmark. The best results are highlighted in **bold**, while the second-best results are underlined.

Table 12 presents the results on the private benchmark. Both fusion strategies yield clear improvements to the baselines, indicating the effective of both fusion methods for private tool retrieval.

### C.5 Public Tool Calling Evaluation

Table 13 shows the rest evaluation results of other model on public benchmarks. It can be seen that if there is no prompt, Llama-3.3-70B-Instrust will frequently answer that "no suitable tools for the task". Therefore, it is necessary to design prompts to ensure the accuracy and effectiveness of the evaluation.

| Method | Top-5 | Top-10 | Top-20 |
|---|---|---|---|
| Cosine | 0.903 | 0.950 | 0.985 |
| L2 | 0.859 | 0.890 | 0.942 |
| IP | 0.904 | 0.945 | 0.972 |
| W-BC | 0.881 | 0.915 | 0.973 |
| C-CB | 0.858 | 0.916 | 0.953 |
| Cosine-Max | 0.971 | 0.992 | 0.998 |
| RRF-CCLI | **0.987** | **0.993** | **0.998** |

Table 12: Test results of tool retrieval on private benchmark. W-BC denotes Weighted Hybrid Retrieval of BM25 (Chen and Wiseman, 2023) (B) and cosine similarity (C) with $\alpha = 0.8$ assigned to the cosine. C-BC denotes Cascade Hybrid Retrieval with dense vector first. The best results are highlighted in **bold**, while the second-best results are underlined.

| | BFCL | TOOLACE | | MCPToolBench++ | |
|---|---|---|---|---|---|
| Model | w/o p | w/o p | w/ p | w/o p | w/ p |
| deepseek-v3.2 | 0.706 | 0.810 | 0.873 | 0.359 | 0.828 |
| glm-4.6 | 0.632 | 0.568 | 0.928 | 0.378 | 0.875 |
| kimi-k2 | 0.692 | 0.740 | 0.938 | 0.379 | 0.825 |
| Llama-3.3-70B-Instruct | 0.696 | 0.026 | 0.940 | 0.0344 | 0.898 |
| Nemotron-3-Nano-30B | 0.502 | 0.728 | 0.731 | 0.564 | 0.754 |
| Qwen2.5-32B-Instruct | 0.699 | 0.841 | 0.965 | 0.273 | 0.934 |

Table 13: The rest evaluation results of other model on public benchmarks. The abnormal results are underlined.

### C.6 Experiments on 8B-Scale Models

To further evaluate the applicability of our framework under more constrained computational settings, we conduct additional experiments on 8B-scale base models. We also report results from a quick ablation study conducted during the rebuttal phase, using a reduced configuration with half the data, 4× gradient accumulation, and 0.6× training epochs. The results are shown in Table 14.

During the SFT stage, we adopt the **same configuration** as used for training the 32B model. The SFT-8B model achieves significant improvements over the baseline on the private benchmark. However, on the open-source benchmark, its general tool-calling capability degrades more substantially than that of the 32B model. We attribute this to the relatively limited model capacity. More fine-grained tuning of training hyperparameters—such as the learning rate and its scheduling strategy, as

| | Qwen3-8B | Abl-8B | SFT-8B | KTO-8B |
|---|---|---|---|---|
| TCR | 0.630 | 0.650 | **0.714** | 0.714 |
| TFR | 0.185 | 0.143 | 0.126 | **0.116** |
| TIR | 0.185 | 0.206 | **0.158** | 0.169 |
| TPS | 0.645 | 0.675 | 0.735 | **0.736** |
| Accuracy | 0.553 | 0.675 | 0.712 | **0.722** |
| Appropriateness | 0.791 | 0.860 | 0.880 | **0.902** |
| Coverage | 0.502 | 0.666 | 0.683 | **0.692** |
| Coherence | 0.964 | 0.937 | 0.959 | **0.961** |
| Faithfulness | 0.612 | 0.719 | 0.754 | **0.763** |
| Expert | 68.57% | 84.76% | 87.1% | **89.05%** |
| BFCL | **0.644** | 0.626 | 0.605 | 0.591 |
| TOOLACE w/o p | **0.768** | 0.762 | 0.484 | 0.485 |
| TOOLACE w/ p | **0.943** | 0.941 | 0.815 | 0.806 |
| MCPToolBench++ w/o p | **0.434** | 0.397 | 0.389 | 0.358 |
| MCPToolBench++ w/ p | **0.886** | 0.805 | 0.856 | 0.854 |

Table 14: The evaluation results of on 8B-Scale Models. Abl-8B represents the model after SFT based on Qwen-8B with qiuck ablation cofigs. SFT-8B represents the model after SFT based on Qwen-8B. KTO-8B represents the model after KTO based on SFT-8B. The best results are highlighted in **bold**, while the second-best results are underlined.

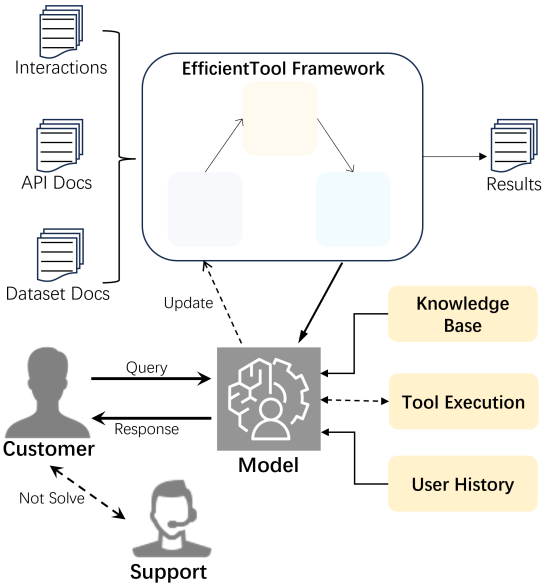


Figure 5: The deployment of the internal enterprise agent. The EfficientTool framework leverages historical customer service interactions, API and dataset documents to train the model.

well as the number of training epochs—may help alleviate this issue.

During KTO training, we observe more pronounced policy shifts in the model. To mitigate this issue, we reduce both the learning rate and the number of training epochs, while keeping other hyperparameters and preference settings unchanged. Compared to the SFT-8B model, the KTO-8B model achieves further improvements. Moreover, on the open-source benchmark, its general tool-calling capability exhibits a smaller decline relative to the SFT-8B model, except for MCPToolBench++ w/o p. These results further support the claim made in the SFT paragraph regarding the effectiveness of such adjustments.

Overall, the above results demonstrate that using an 8B model as the base model can still yield improvements on the private benchmark, validating the effectiveness of our framework. However, the trained 8B model still lags behind the trained 32B model and suffers from a more pronounced degradation in general tool-calling capability. Therefore, we recommend selecting larger models whenever hardware resources permit, in order to ensure stronger performance and better generalization ability. Due to limited time, an exhaustive investigation into the optimal training configurations for the 8B model was beyond the scope of this study and is reserved for future research.

# D Details of Deployment

## D.1 Figure of Deployment Process

Figure 5 shows the deployment process of our internal enterprise agent. The EfficientTool framework first leverages historical customer service interactions, API and dataset documents to train the model. By default, customers interact with the agent, which operates within an environment integrated with a knowledge base, tool execution capabilities, and user interaction history. If the model fails to resolve a user query, due to technical limitations or scope constraints, the session is escalated to human support. We conduct periodic audits of service logs and customer feedback to monitor system performance. Any detectable degradation in service quality triggers a subsequent iteration of the EfficientTool framework to ensure continuous model refinement and customer service quality.