

ToolForge: A Data Synthesis Pipeline for Multi-Hop, Multi-Turn, and Self-Reflective Tool-Use Data

Hao Chen^{1,2*}, Shunyu Zhou^{2*}, Zhixin Hu², Jiajun Chai²,
Hang He^{2,3}, Haocheng Yang^{2,4}, Xiaohan Wang², Wei Lin²,
Tianyu Wang¹, Zhuofeng Zhao^{1†}, Guojun Yin^{2†},

¹Beijing Key Laboratory of Key Technologies for AI+ Domain Applications, North China University of Technology, Beijing, China

²Meituan, Beijing, China ³East China Normal University, Shanghai, China

⁴National University of Singapore, Singapore

edzhao@ncut.edu.cn gjyin@outlook.com

Abstract

Training LLMs for tool use requires more than final-answer supervision: models need trajectories that specify how to decompose a question, call tools across turns, use retrieved evidence, and recover from earlier errors. Existing data construction pipelines often rely on large-scale real-world tool API calls or simple templates, making them costly or limited in interaction coverage. We introduce ToolForge, a data synthesis pipeline for constructing tool-use datasets with three targeted abilities: multi-hop reasoning, multi-turn tool use, and self-reflection. Starting from QA seed data, ToolForge builds an extensible tool space from 19 base virtual tools and automatically generated variants, selects among four tool-use paradigms, simulates both successful and misleading retrieval outcomes, and generates validated tool-use dialogues without large-scale real-world tool API execution. A Multi-Layer Validation module filters invalid traces and is audited with hard negatives mined by Monte Carlo Tree Search. Fine-tuning Qwen3-8B on only 4,250 ToolForge samples produces a model named ToolForge-8B, which improves over the Tool-Integrated Reasoning baseline by 21.6% on five out-of-domain QA benchmarks and remains competitive on BFCL and API-Bank. We further quantify the pipeline construction cost, including tool initialization, dialogue generation, validation, and SFT, and compare ToolForge with existing synthetic tool-data sources. Our code is publicly available at <https://github.com/Buycar-arb/ToolForge>.

1 Introduction

Tool-augmented LLMs extend static language models by enabling calls to external tools, retrieval of

up-to-date information (Zhou et al., 2025), and interaction with task environments (Schick et al., 2023; Yuan et al., 2024; Qu et al., 2025; Chai et al., 2025; Xia et al., 2026). This capability is particularly important for complex search tasks, where a model must select appropriate tools, construct valid arguments, interpret retrieved evidence, and revise earlier decisions when necessary (Hu et al., 2026; Lin et al., 2026). Training such behavior therefore requires supervision over complete tool-use trajectories rather than final answers alone.

High-quality tool-use data remains difficult to construct. Existing pipelines often collect traces by executing large numbers of real-world APIs or search calls (Liu et al., 2024; Team et al., 2025; Wang et al., 2024). Such traces provide realistic observations, but they are costly and difficult to scale across domains, especially because robust tool generalization benefits from diverse tools and schemas (Shi et al., 2025; Mekala et al., 2024). Template- or prompt-based approaches reduce this cost (Yao et al., 2023; Tang et al., 2023), but they often emphasize single-step tool use and provide limited supervision for complex interaction patterns.

This supervision gap is most pronounced in multi-hop search. Many questions require evidence to be gathered and integrated across multiple entities, documents, or reasoning steps (Yang et al., 2018; Ho et al., 2020; Trivedi et al., 2022; Press et al., 2022). Moreover, when retrieval returns ambiguous or misleading evidence, an effective tool-using model should inspect the result, reformulate the query or tool choice, and continue reasoning. Prior work shows that reflection can help models revise their own reasoning (Shinn et al., 2023), but this ability is rarely incorporated into synthetic tool-use data. We therefore target datasets that jointly contain *multi-hop reasoning*, *multi-turn tool use*,

*Equal contribution.

†Corresponding authors.

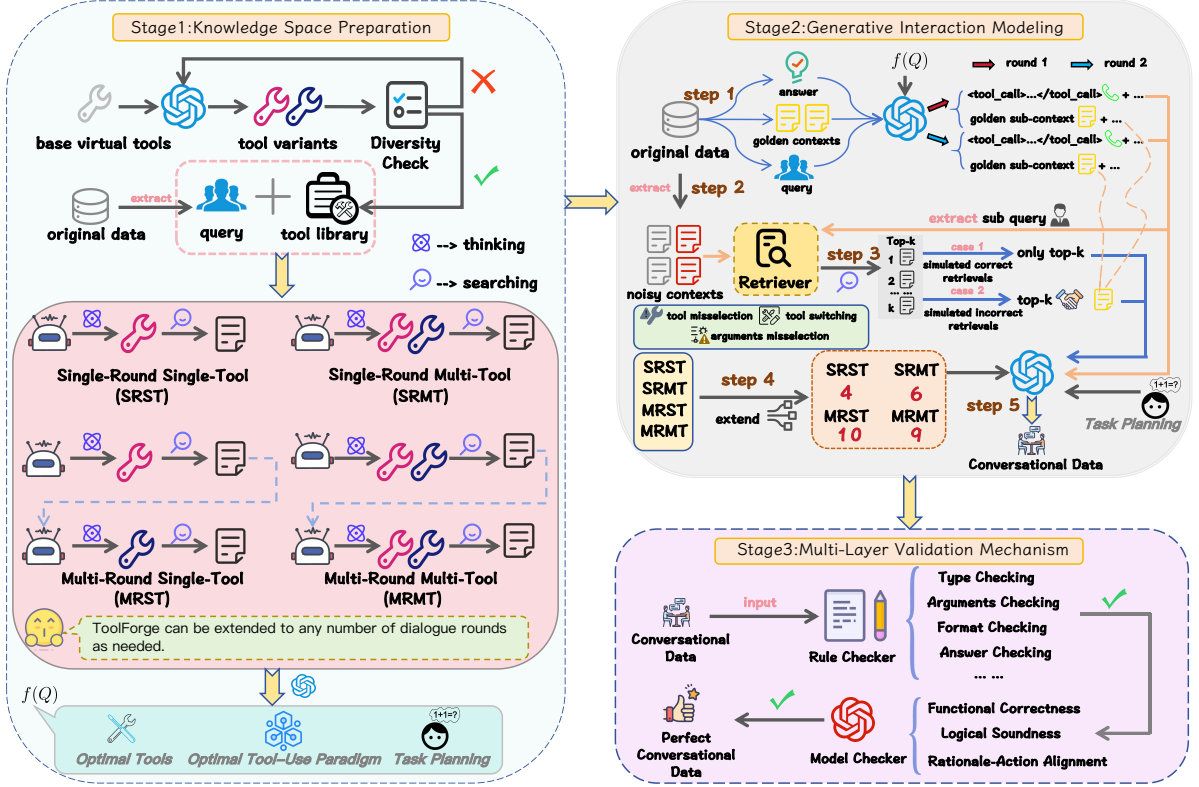


Figure 1: Overview of ToolForge. KSP prepares the virtual tool space and tool-use paradigms; GIM generates tool-use dialogues under successful and noisy retrieval outcomes; MLV filters structurally or semantically invalid samples.

and *self-reflection* under noisy retrieval.

We propose ToolForge, a data synthesis pipeline for constructing such trajectories from QA seed data. A virtual tool is a schema-level retrieval tool whose observation is simulated from controlled evidence pools rather than obtained through live API execution. ToolForge first builds an extensible tool space from 19 manually designed base tools and automatically generated variants, and then uses four tool-use paradigms and three error perturbation types to synthesize dialogues with multi-hop evidence use, cross-turn dependencies, and reflective correction. This design reduces dependence on large-scale live tool execution during data construction while preserving realistic tool-use structure.

Because automatically generated tool-use trajectories may contain subtle errors, ToolForge also includes a Multi-Layer Validation (MLV) module. Existing validation in synthetic tool-data pipelines often focuses on format, tool-call syntax, or final-answer consistency (Basu et al., 2024; Wang et al., 2024; ?), leaving semantic inconsistencies in intermediate reasoning less directly checked (Liu et al., 2024). MLV combines rule-based checks with model-based semantic evaluation, and we further audit it with MLV-Bench, whose rule-level

hard negatives are mined using Monte Carlo Tree Search (MCTS) (Silver et al., 2016). Figure 1 illustrates the overall pipeline.

In summary, our key contributions are threefold:

- We introduce ToolForge, a data synthesis pipeline that converts QA seed data into tool-use trajectories with multi-hop reasoning, multi-turn tool calls, and self-reflection, while reducing reliance on large-scale live tool execution.
- We design an extensible virtual-tool space with 19 base tools, dual-gated tool variants, four tool-use paradigms, and plug-and-play support for additional tools, noise types, and interaction patterns.
- We introduce a multi-layer validation mechanism that combines structural checks with semantic verification and is audited using hard negatives mined with Monte Carlo Tree Search, improving the reliability of synthetic tool-use trajectories.

2 Related Work

Tool Learning. Tool learning has been shown to substantially improve LLM performance on complex tasks (Qu et al., 2025; Chai et al., 2025). Fine-

tuning methods collect tool-use traces for supervised or reinforcement learning (Liu et al., 2024; Team et al., 2025; Qin et al., 2023), while prompt-based methods elicit tool use without parameter updates (Yuan et al., 2024; Yao et al., 2023). Recent tool-data and benchmark efforts further clarify the design space. Toucan (Xu et al., 2025) emphasizes scale and environmental realism by synthesizing tool-agentic data from real-world MCP environments. TaskBench (Shen et al., 2024) focuses on evaluating task decomposition, tool selection, and parameter prediction. SPIRAL (Zhang et al., 2026b) improves inference-time planning through a structured search loop. ToolForge is complementary to these directions: it is closest to fine-tuning-based data construction, but uses virtual tools and controlled evidence simulation to synthesize multi-hop, multi-turn, and self-reflective tool-use trajectories without relying on large-scale live tool calls.

Multi-Hop Data. Multi-hop reasoning requires systems to integrate dispersed evidence into coherent reasoning chains (Yang et al., 2018). Existing multi-hop datasets are mainly constructed through manual annotation (Ho et al., 2020; Trivedi et al., 2022) or template-based synthesis (Talmor and Berant, 2018; Press et al., 2022; He et al., 2025), both of which typically follow static interaction paradigms. Recent synthesis methods use LLMs to reduce annotation cost and improve diversity (Liu et al., 2024; Team et al., 2025; Wang et al., 2024), but they often provide limited coverage of dynamic tool interactions and reflective correction. ToolForge targets this gap by generating multi-hop tool-use trajectories with controlled retrieval evidence and validation.

Data Synthesis. The scarcity of high-quality data remains a critical bottleneck for LLMs (Bauer et al., 2024). Existing data synthesis methods rely primarily on prompt templates (Xu et al., 2024; Yu et al., 2023), API-based generation (Basu et al., 2024; Liu et al., 2024), or dialogue construction (Chen et al., 2025; Patil et al., 2025; Wang et al., 2024). Recent pipelines further improve multi-turn conversational realism (Wang et al., 2025; Shim et al., 2025), function and environment coverage (Sullivan et al., 2025), or trajectory repair through self-refinement, label correction, and explicit tool feedback (Zeng et al., 2026c; Zhang et al., 2026a). ToolForge instead synthesizes multi-hop, multi-turn, and self-reflective trajectories from QA seeds using paired golden and noisy retrieval observations, followed by multi-layer validation. This design trains mod-

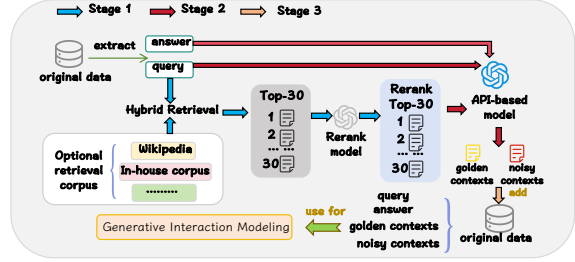


Figure 2: Preprocessing original QA data into query, answer, golden contexts, and noisy contexts before ToolForge synthesis.

els to recover when successful tool calls return plausible but insufficient evidence.

3 Methods

ToolForge converts a QA instance into a verified tool-use dialogue without large-scale live tool execution. Given an original sample with a query Q and answer A , we first attach supporting evidence (*golden contexts*) and retrieval distractors (*noisy contexts*) when they are not already provided, as illustrated in Figure 2. The pipeline then proceeds in three stages (Figure 1): **Knowledge Space Preparation** (KSP) builds the tool library and chooses the tool-use plan, **Generative Interaction Modeling** (GIM) turns this plan into a dialogue with successful or noisy retrieval observations, and **Multi-Layer Validation** (MLV) filters structurally or semantically invalid samples before training.

3.1 Knowledge Space Preparation

Knowledge Space Preparation (KSP) defines the action space before dialogue generation. It first constructs the tool library and accepted tool variants, then selects the tool sequence, interaction paradigm, and task plan used for a given query.

Virtual tool library. We manually design 19 retrieval-oriented base tools covering common knowledge domains such as people, works, events, locations, science, law, economics, medicine, culture, and politics; the full list is provided in Appendix C. This manual step is a one-time tool-space initialization; per-sample synthesis is automatic once the tool library is specified. To increase schema diversity while preserving tool functionality, an LLM generates variants for each base tool. A candidate variant $\tilde{T}$ is accepted only if it remains semantically related to the existing tool set while being textually non-redundant. We implement this diversity check with a dual gate:

$$\mathcal{J} = \mathbf{1}\left\{\delta_{\mathcal{V}} \vee \text{Agg}_{\text{cos}}(\tilde{T}, \mathcal{V}) > \theta_c\right\} \cdot \mathbf{1}\left\{\text{Agg}_{\text{text}}(\tilde{T}, \mathcal{V}) < \theta_b\right\} \quad (1)$$

$$\delta_{\mathcal{V}} = \begin{cases} 1, & \text{if } |\mathcal{V}| < 2, \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

$$\text{Agg}_{\text{cos}}(\tilde{T}, \mathcal{V}) = \frac{1}{|\mathcal{V}|} \sum_{T_i \in \mathcal{V}} \text{Sim}_{\text{cos}}(\tilde{T}, T_i), \quad (3)$$

$$\text{Agg}_{\text{text}}(\tilde{T}, \mathcal{V}) = \frac{1}{|\mathcal{V}|} \sum_{T_i \in \mathcal{V}} \text{Sim}_{\text{BM25}}(\tilde{T}, T_i). \quad (4)$$

Here $\mathcal{V}$ is the set of accepted variants. θ_c is the minimum semantic similarity required to preserve tool functionality, and θ_b is the maximum textual similarity allowed to prevent near-duplicate schemas. The cold-start term $\delta_{\mathcal{V}}$ relaxes the semantic gate when fewer than two variants have been accepted. This criterion keeps generated variants functionally aligned with the base tool while increasing schema and argument diversity.

Tool-use paradigms. To cover progressively more complex search behaviors, we organize tool use into four paradigms: **SRST**, a single tool call in one round; **SRMT**, multiple tool calls in one round; **MRST**, one tool used across multiple rounds; and **MRMT**, multiple tools used across multiple dependent rounds. Examples of the four paradigms are provided in Appendix J. For each query $\mathcal{Q}$, a planner chooses from the prepared tool library and paradigm set, producing three variables:

$$\text{KSP}(\mathcal{Q}) = \langle \mathcal{S}^*, \mathcal{C}^*, \mathcal{R}^* \rangle. \quad (5)$$

Here $\mathcal{S}^*$ is the selected tool sequence, $\mathcal{C}^*$ is the selected tool-use paradigm, and $\mathcal{R}^*$ is the task plan or reasoning rationale passed to GIM. This tuple provides the control signal for dialogue generation. Ablation studies of KSP are provided in Section 4.8.1.

3.2 Generative Interaction Modeling

Generative Interaction Modeling (GIM) converts the KSP control signal into executable conversational data. It synthesizes both successful and noisy retrieval situations, so the final dialogue teaches not only which tool to call, but also when to revise an inappropriate tool choice, argument, or retrieved evidence.

Evidence preparation. For each sample, GIM uses $\mathcal{Q}$, $\mathcal{A}$, and the available golden contexts. If golden contexts are missing, we obtain them by retrieving and reranking candidate passages, followed by

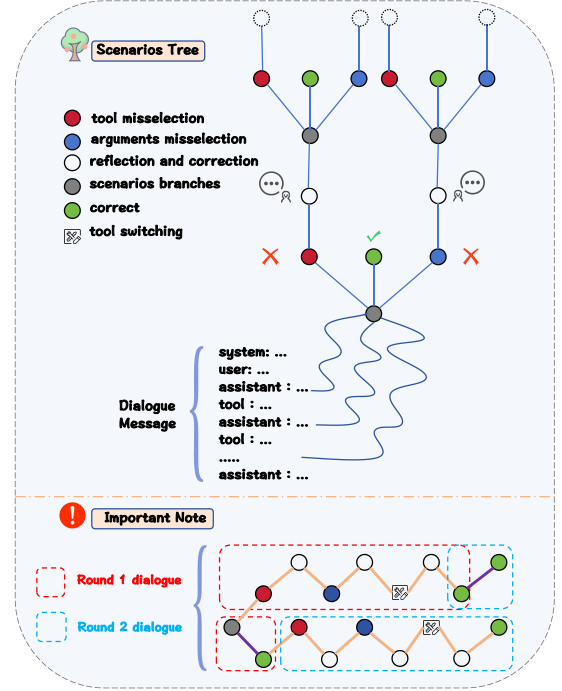


Figure 3: Scenario tree used to expand tool-use paradigms with correct actions, tool misselection, argument misselection, and tool switching.

LLM-based filtering. We also collect noisy contexts that are topically related to the query but insufficient for deriving the answer. This yields two evidence pools: one supporting correct reasoning and one simulating realistic retrieval noise.

Retrieval outcome simulation. Conditioned on $\mathcal{R}^*$, the generator decomposes the original query into sub-queries and planned tool calls. For each sub-query q_i , we retrieve top- k passages K_i from the noisy pool. We then form two possible observations: $I_i^{\text{good}} = K_i \cup g_i$, where g_i is the golden sub-context required by the planned step, and $I_i^{\text{bad}} = K_i$, which contains only plausible but incomplete or misleading retrieval results. This paired simulation prevents the synthesized dialogues from assuming overly clean retrieval conditions.

Dialogue synthesis. Starting from the four paradigms in KSP, we introduce three perturbation types: tool misselection, argument misselection, and tool switching. Their combinations yield 29 representative reasoning-tool interaction patterns $\mathcal{M} = \{\text{flow}_i\}_{i=1}^{29}$, as illustrated in Figure 3. A generator model receives $\mathcal{Q}$, $\mathcal{A}$, $\text{KSP}(\mathcal{Q})$, one interaction pattern, and the simulated observations, then produces a dialogue containing tool calls, retrieved evidence, intermediate reasoning, optional reflection, and the final answer. The output is parsed into the final training format $\mathcal{D}_{\text{final}}$. Further implemen-

tation details are provided in Appendix A.

3.3 Multi-Layer Validation

Synthetic tool-use dialogues are suitable for training only when their structure, evidence, and reasoning are mutually consistent. We therefore apply Multi-Layer Validation (MLV) before any sample is used for training. MLV is designed to remove two distinct failure modes.

Rule Verification Layer. The rule layer uses nine deterministic checks for executable-form errors, including invalid tool names, malformed arguments, missing observations, inconsistent answer fields, tool-protocol violations, and sample traceability failures. For example, the checker extracts every assistant-side `<tool_call>` tag and verifies that the tool name and arguments conform to the pre-defined schema. This layer filters non-executable samples without requiring real tool execution.

Model Verification Layer. A sample can satisfy all formal rules while remaining semantically invalid. The model layer therefore targets semantic errors such as choosing a plausible but inappropriate tool, reasoning over unsupported evidence, or producing a rationale that does not justify the next action. Each criterion is judged by an LLM-based expert agent, and only samples passing both validation layers are retained.

To audit MLV itself, we construct MLV-Bench with both positive samples and hard negative cases. For rule-level negatives, we use Monte Carlo Tree Search (MCTS) (Silver et al., 2016) to explore combinations of protocol, schema, argument, and consistency violations. We further include semantic negatives targeting the three model-level criteria. MLV-Bench is used only to evaluate validator reliability and is not part of data generation or model training. Appendix D details its construction and evaluation, on which MLV achieves 98.47% accuracy and 97.39% F1. Appendix E reports the post-validation pass rates across the four tool-use paradigms, which range from 88.83% to 91.70%. The corresponding MLV ablation is reported in Section 4.8.3.

4 Experiments

4.1 Datasets and Setup

We train ToolForge-8B by fine-tuning Qwen3-8B (Yang et al., 2025) on 4,250 synthetic samples generated from HotpotQA (Yang et al., 2018) and 2WikiMultiHopQA (Ho et al., 2020). The de-

fault training mixture is dominated by SRST samples, with smaller portions of SRMT, MRST, and MRMT samples; Section 4.8.2 studies how this composition affects performance. For retrieval-based evaluation, all methods use the same 2018 Wikipedia dump (Karpukhin et al., 2020), the same BM25s retriever (Lù, 2024), and top-5 retrieved passages.

For synthesis, KSP uses $\theta_c = 0.8$ as the semantic-similarity lower bound and $\theta_b = 0.5$ as the lexical-similarity upper bound, and generates 20 variants for each base virtual tool. We train for two epochs with full-parameter SFT on four A100 80GB GPUs, using DeepSpeed ZeRO-2, bfloat16, FlashAttention2, a learning rate of 1×10^{-5} , an effective batch size of 8, and a maximum sequence length of 12k tokens. Training takes approximately two hours. At inference time, ToolForge emits virtual tool calls; because the Wikipedia dump is not a live API backend, each call is converted into a retrieval query by concatenating its argument names and values. A top- k sensitivity study (Section 4.8.4) shows that top-5 provides the best trade-off, while larger k values introduce more irrelevant evidence.

We evaluate QA performance on three general QA datasets, NQ (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), and PopQA (Mallen et al., 2022), and two multi-hop QA datasets, MusiQue (Trivedi et al., 2022) and Bamboogle (Press et al., 2022). We report Exact Match (EM), following Search-R1 (Jin et al., 2025). We additionally evaluate tool-calling generalization on BFCL (Patil et al., 2025) and API-Bank (Li et al., 2023).

4.2 Baselines

We compare with four groups of baselines. **Inference-only baselines** include Direct Inference, Chain-of-Thought (CoT) (Wei et al., 2022) and Tool-Integrated Reasoning (TIR) (Chai et al., 2025). **Retrieval baselines** include RAG (Lewis et al., 2020), IRCOT (Trivedi et al., 2023), and Search-o1 (Li et al., 2025a). **RL-training baselines** include R1 (Guo et al., 2025), Search-R1 (Jin et al., 2025), and ZEROSEARCH (Sun et al., 2025). **Synthetic-data baselines** include ToolACE (Liu et al., 2024), ToolLLM (Qin et al., 2023), and xLAM (Zhang et al., 2025). This grouping separates inference strategies, retrieval strategies, RL training, and data synthesis, thereby avoiding direct comparison between methods with different training objectives. Retrieve-rerank-generate systems such

Table 1: Performance comparison of ToolForge-8B and different baseline methods across five QA datasets. All retrieval-based methods use the same Wikipedia retriever. * denotes out-of-domain datasets. Percentages in the Avg. column denote relative changes over TIR. Best in bold.

Model	Avg. ($\uparrow$ / $\downarrow$ vs. TIR)	General QA			Multi-Hop QA	
		NQ*	TriviaQA*	PopQA*	Musique*	Bamboogle*
<i>Qwen3-8B + Direct Inference</i>						
TIR (Baseline)	0.342 (–)	0.301	0.593	0.358	0.094	0.365
Direct Inference	0.260 ($\downarrow$ 24.0%)	0.200	0.519	0.191	0.058	0.334
CoT	0.258 ($\downarrow$ 24.6%)	0.209	0.512	0.182	0.055	0.332
<i>Qwen3-8B + Inference with Retrieval</i>						
IRCoT	0.318 ($\downarrow$ 7.0%)	0.264	0.542	0.331	0.113	0.342
RAG	0.317 ($\downarrow$ 7.3%)	0.299	0.572	0.346	0.096	0.274
Search-o1	0.284 ($\downarrow$ 17.0%)	0.223	0.531	0.235	0.073	0.356
<i>Qwen3-8B + RL Training</i>						
R1-base	0.265 ($\downarrow$ 22.5%)	0.297	0.423	0.202	0.085	0.320
R1-instruct	0.276 ($\downarrow$ 19.3%)	0.301	0.447	0.199	0.124	0.307
Search-R1	0.364 ($\uparrow$ 6.4%)	0.305	0.600	0.394	0.143	0.380
ZEROSEARCH-base	0.374 ($\uparrow$ 9.3%)	0.334	0.597	0.402	0.138	0.398
ZEROSEARCH-instruct	0.361 ($\uparrow$ 5.6%)	0.325	0.599	0.404	0.121	0.357
<i>Qwen3-8B + Data Synthesis + SFT</i>						
ToolForge-8B	0.416 ($\uparrow$ 21.6%)	0.309	0.602	0.410	0.278	0.480

Table 2: Comparison with synthetic tool-data sources on BFCL using Qwen3-8B as the backbone. Best in bold.

Training data	Avg.	Single Turn		Multi Turn	Hallucination	
		Non-live	Live	Multi-turn	Relevance	Irrelevance
ToolACE (25k)	0.746	0.941	0.765	0.386	0.931	0.707
ToolLLM (25k)	0.423	0.421	0.561	0.000	1.000	0.135
xLAM (25k)	0.557	0.911	0.576	0.156	0.805	0.338
ToolForge (4.25k)	0.778	0.899	0.827	0.437	0.954	0.775

as RetroLLM (Li et al., 2025b) are relevant retrieval pipelines, but they optimize a different system component and are therefore not treated as direct synthetic-data baselines.

4.3 Cost and Construction Effort

Table 3 decomposes the operational cost in our setup. Search-R1 relies on live search/tool API requests during data construction, whereas ZEROSEARCH avoids API fees but still uses a longer GPU-based training pipeline. ToolForge does not issue live tool API requests during synthesis; its cost comes from LLM-based dialogue generation and validation plus a short SFT run. The 19 base virtual tools were created as a one-time setup by analyzing user intents from eight Wikipedia-based datasets. This process took three annotators approximately 12 hours, with most time spent on dataset analysis and the remainder on formatting tool schemas. Once the virtual tool library is specified, the synthesis process runs automatically.

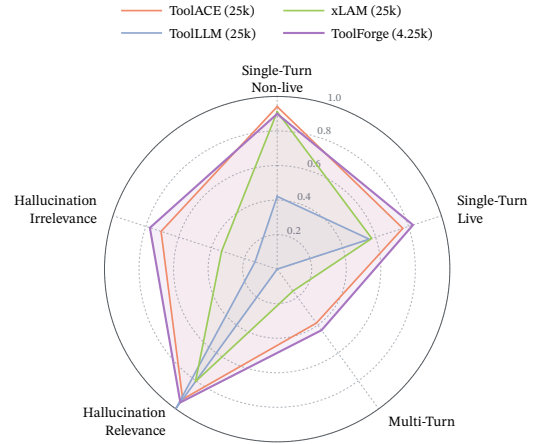


Figure 4: BFCL performance of models trained on different synthetic tool-use datasets using Qwen3-8B as the shared backbone. ToolForge uses 4.25k training samples, whereas the compared datasets use 25k samples. Higher is better.

4.4 Main QA Results

Table 1 reports the main QA results. ToolForge-8B achieves the best average score among the compared 8B-scale settings, improving over TIR by 21.6%. The gains are particularly pronounced on

Table 3: Estimated training-data construction cost in our setup. Tool API requests count live tool/search executions during data construction. Synthesis cost denotes the LLM generation and validation cost used to construct ToolForge data.

Method	Tool API req.	Training time	GPU used	API cost	GPU cost	Synthesis cost	Total cost
Search-R1	~64k	~12h	4 × A100	\$586.7	\$70.3	\$0.0	\$657.0
ZEROSEARCH	~64k	~12h	8 × A100	\$0.0	\$140.6	\$0.0	\$140.6
ToolForge	0	~2h	4 × A100	\$0.0	\$12.1	\$80.6	\$92.7

Table 4: Ablation study of KSP on BFCL. Avg. changes are relative to Qwen3-8B; gray indicates the default setting. Best in bold.

Setting	Avg.	Single Turn		Multi Turn	Hallucination	
		Non-live	Live	Multi-turn	Relevance	Irrelevance
<i>Baseline</i>						
Qwen3-8B	0.751 (–)	0.876	0.805	0.418	0.938	0.719
<i>Varying number of variants (19 types)</i>						
19 types & 5 variants	0.606 (↓ 19.3%)	0.682	0.628	0.392	0.718	0.612
19 types & 10 variants	0.688 (↓ 8.4%)	0.782	0.717	0.409	0.842	0.692
19 types & 15 variants	0.744 (↓ 0.9%)	0.852	0.778	0.425	0.918	0.738
19 types & 20 variants	0.778 (↑ 3.6%)	0.899	0.827	0.437	0.954	0.775
19 types & 25 variants	0.787 (↑ 4.8%)	0.911	0.839	0.442	0.962	0.782
19 types & 30 variants	0.791 (↑ 5.3%)	0.918	0.842	0.445	0.964	0.787
<i>Varying number of tool types (20 variants)</i>						
5 types & 20 variants	0.626 (↓ 16.6%)	0.716	0.654	0.383	0.761	0.618
10 types & 20 variants	0.696 (↓ 7.3%)	0.798	0.732	0.402	0.858	0.692
15 types & 20 variants	0.746 (↓ 0.6%)	0.859	0.791	0.422	0.920	0.740
19 types & 20 variants	0.778 (↑ 3.6%)	0.899	0.827	0.437	0.954	0.775
25 types & 20 variants	0.802 (↑ 6.8%)	0.928	0.858	0.452	0.967	0.799
30 types & 20 variants	0.821 (↑ 9.3%)	0.948	0.881	0.469	0.978	0.818

multi-hop QA: ToolForge-8B improves MusiQue from 0.094 to 0.278 and Bamboogle from 0.365 to 0.480. On general QA, ZEROSEARCH-base obtains the highest NQ score, whereas ToolForge-8B achieves the highest TriviaQA and PopQA scores among the compared settings. Thus, although ToolForge-8B does not lead on every dataset, it delivers the strongest average performance and substantial gains in multi-hop QA using a compact SFT dataset.

4.5 Comparison with Synthetic Tool Data

We further compare ToolForge with three existing synthetic tool-data sources. Table 2 evaluates models trained on ToolACE, ToolLLM, xLAM, and ToolForge using BFCL, while Figure 4 visualizes their performance profiles. Despite using only 4.25k samples, ToolForge achieves the highest average score of 0.778, outperforming models trained on 25k samples. ToolACE leads on non-live function calling but is weaker on multi-turn interactions, whereas ToolLLM obtains a perfect relevance score but scores 0.000 on multi-turn function calling and 0.135 on irrelevance. ToolForge instead exhibits the most balanced performance across the

five BFCL dimensions. These results suggest that targeted multi-hop, multi-turn, and self-reflective trajectories can provide more effective supervision than substantially larger but less structured tool-use corpora. Appendix I reports both pure-data and mixed-data comparisons.

4.6 Tool-Calling Generalization

Although ToolForge is trained on synthetic retrieval-style tool traces, its gains transfer to external function-calling benchmarks. ToolForge-8B improves over the Qwen3-8B base model on both BFCL (0.751 to 0.778) and API-Bank (0.622 to 0.647), while remaining competitive among the models listed in the full benchmark comparisons. These improvements are obtained without direct training on the corresponding benchmark test distributions. Full benchmark tables are provided in Appendix G.

4.7 Plug-and-Play Extensibility

We further test whether ToolForge can incorporate new tool-error types without redesigning the virtual-tool library. Starting from the three default perturbations, we extend the noise spectrum to

Table 5: Effect of the ratio between SRST and complex trajectories on ToolForge-8B. Complex trajectories comprise SRMT, MRST, and MRMT. * denotes out-of-domain datasets. Avg. changes are relative to Qwen3-8B; gray indicates the default setting. Best in bold.

Setting	Avg.	NQ*	TriviaQA*	PopQA*	MusiQue*	Bamboogle*
SRST:Complex						
Qwen3-8B (Baseline)	0.322 (–)	0.208	0.523	0.301	0.196	0.380
ToolForge-8B (10:0)	0.397 (↑ 23.3%)	0.334	0.623	0.486	0.183	0.359
ToolForge-8B (9:1)	0.416 (↑ 29.2%)	0.309	0.602	0.410	0.278	0.480
ToolForge-8B (8:2)	0.426 (↑ 32.3%)	0.310	0.607	0.413	0.289	0.511
ToolForge-8B (7:3)	0.432 (↑ 34.2%)	0.311	0.605	0.434	0.295	0.515
ToolForge-8B (6:4)	0.444 (↑ 37.9%)	0.308	0.611	0.430	0.307	0.564
ToolForge-8B (5:5)	0.421 (↑ 30.7%)	0.297	0.621	0.407	0.281	0.499
ToolForge-8B (4:6)	0.451 (↑ 40.1%)	0.318	0.637	0.442	0.301	0.557
ToolForge-8B (3:7)	0.450 (↑ 39.8%)	0.316	0.636	0.442	0.301	0.555
ToolForge-8B (2:8)	0.410 (↑ 27.3%)	0.286	0.541	0.409	0.303	0.511
ToolForge-8B (1:9)	0.405 (↑ 25.8%)	0.274	0.531	0.372	0.304	0.544
ToolForge-8B (0:10)	0.401 (↑ 24.8%)	0.263	0.519	0.334	0.328	0.561

Table 6: Ablation study of the MLV module. * denotes out-of-domain datasets. Avg. changes are relative to the full MLV setting; best results are in bold.

Validation setting	Avg.	General QA			Multi-Hop QA	
		NQ*	TriviaQA*	PopQA*	MusiQue*	Bamboogle*
<i>ToolForge-8B: Data Synthesis + SFT</i>						
Full MLV (RVL & MVL)	0.416 (–)	0.309	0.602	0.410	0.278	0.480
RVL only	0.347 (↓ 16.6%)	0.244	0.582	0.323	0.261	0.327
MVL only	0.357 (↓ 14.2%)	0.254	0.561	0.334	0.253	0.382
Without RVL/MVL	0.279 (↓ 32.9%)	0.181	0.432	0.287	0.201	0.293

eight ToolScan-style (Kokane et al., 2025) categories by adding insufficient API calls, incorrect argument type, incorrect argument name, repeated API calls, and invalid format error. This extension keeps the same 19 base virtual tools and the same 4,250-sample training budget. As shown in Appendix H, ToolForge-8B improves from 0.77 average score under the original three noise types to 0.92 under the expanded eight-type setting, matching xLAM-8x7B and ranking second behind GPT-4-0125-preview. This result supports the plug-and-play extensibility of the proposed data synthesis pipeline.

4.8 Ablation Study

4.8.1 KSP Module

We ablate KSP along two design axes: *schema diversity* and *tool-space coverage*. Schema diversity is controlled by the number of accepted variants for each base tool, while tool-space coverage is controlled by the number of virtual tool types. These axes capture distinct sources of generalization. Variants mainly change the surface form and argument schema of a tool, and should therefore improve robustness to function-schema variation. Tool types, in contrast, change the semantic domain of the available actions, and should improve intent

coverage and tool-selection boundaries.

Table 4 reports the KSP ablation results, with the corresponding trends visualized in Figures 6 and 7. The results reveal two trends across the two design axes. First, with 19 tool types fixed, performance rises quickly from 5 to 20 variants and then saturates. This suggests that variants help prevent overfitting to a small set of function names and argument templates, but additional paraphrases give diminishing returns once basic schema robustness is learned. Second, expanding tool types has a larger and more persistent effect: moving from 5 to 30 types improves the BFCL average from 0.626 to 0.821. This indicates that KSP contributes more by broadening the semantic action space than by merely renaming tools. Thus, the model benefits most when training exposes it to distinct retrieval intents, not only to alternative formulations of the same intent.

4.8.2 Data Composition

The second KSP-related choice is the proportion of simple versus complex tool-use paradigms in the training data. SRST examples train the model to map a question to a relevant tool and evidence span in a stable single-step setting. SRMT, MRST, and MRMT examples introduce parallel evidence col-

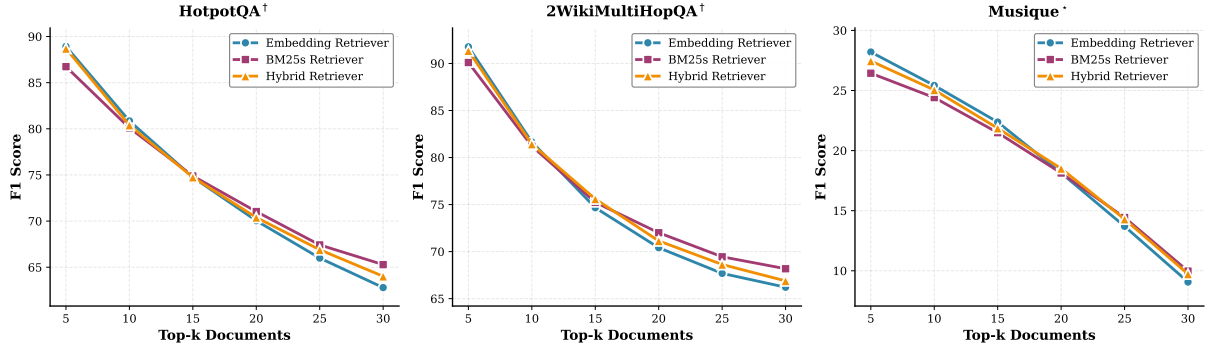


Figure 5: Performance of ToolForge-8B inference under different retrieval methods and various numbers of retrieved passages (top-k). [†] denotes in-domain datasets, while * denotes out-of-domain datasets.

lection, cross-round dependency, and correction after imperfect retrieval. We therefore expect SRST-heavy data to support general QA stability, while complex trajectories should benefit multi-hop reasoning.

Table 5 supports this interpretation. Pure SRST training gives the strongest NQ and PopQA results, while the 4:6 mixture achieves the highest TriviaQA score. However, pure SRST does not improve multi-hop QA over the base model because the model rarely observes dependent evidence gathering. Adding complex paradigms substantially improves MusiQue and Bamboogle, with the best overall score obtained at a 4:6 balance. Using only complex paradigms lowers general-QA performance, although it yields the highest MusiQue score. These results indicate that paradigm complexity should be balanced rather than maximized: SRST supports reliable first-step tool selection, while complex paradigms teach decomposition and recovery. The default 9:1 mixture is used for the main controlled comparison, while this ablation identifies data composition as an independent tuning dimension.

4.8.3 MLV Module

We ablate the two validation layers to test whether structural and semantic checks contribute complementary supervision.

Table 6 explains why both layers are necessary. Using only RVL removes non-executable samples but leaves semantically inconsistent traces, causing a 16.6% average drop relative to the full validator. Using only MVL is slightly stronger on multi-hop QA, because semantic checking is more directly aligned with the reasoning failures that arise in multi-step trajectories, but it cannot reliably remove protocol and schema errors. Removing both layers yields the worst result, indicating that noisy synthetic traces can actively degrade the

model rather than merely add weak supervision. This complementary pattern supports MLV as a two-layer filter rather than a single generic quality scorer.

4.8.4 Additional Retrieval and Setup Details

We investigate the impact of three retrieval methods on the inference of ToolForge-8B under various numbers of retrieved passages (top- k), using three datasets and the 2018 Wikipedia dump as the knowledge source. To better understand this, we conduct additional studies with top- k set to 10, 15, 20, 25, and 30. Through the top- k sensitivity study (Figure 5), we observe a clear trade-off: smaller top- k values generally yield higher F_1 scores and better overall performance, whereas larger values introduce noise and degrade performance, discouraging the model from leveraging retrieved content when additional context is unhelpful or misleading. Consequently, the main paper uses top-5 because it provides the best trade-off between evidence coverage and retrieval noise.

5 Conclusion

We presented ToolForge, a data synthesis pipeline for training information-seeking, tool-using agents. Starting from QA seeds, ToolForge constructs validated trajectories that cover single- and multi-tool use and recovery from tool-selection, argument-selection, and retrieval failures. By combining controlled golden and noisy retrieval observations with multi-layer validation, the pipeline synthesizes multi-hop, multi-turn, and self-reflective tool-use data without large-scale live API execution. Experiments show that ToolForge data improves multi-hop QA, generalizes to external tool-calling benchmarks, and compares favorably with substantially larger synthetic tool-use datasets.

Limitations

ToolForge has four limitations. First, the 19 base virtual tools are manually initialized and may not cover all real-world tool domains. Second, the seed data mainly comes from HotpotQA and 2WikiMultiHopQA, which may bias queries toward Wikipedia-style evidence. Third, BFCL and API-Bank approximate function calling but are not substitutes for live API deployment. Fourth, the default mixture is SRST-heavy; ratio ablations suggest that further tuning the mixture could improve performance.

Ethics Statement

ToolForge uses LLMs for tool-schema variation, trajectory generation, and model-based validation. Experiments rely on publicly available QA datasets and Wikipedia-based retrieval corpora; we do not collect private user data or execute live real-world APIs during synthesis. The released resources are intended for research on tool-use data synthesis and evaluation. Because synthetic data may contain model errors or corpus biases, users should inspect it before downstream or deployed use.

Acknowledgements

This work was supported by the Special Research Project for Provincial-Ministerial-Level Research Platforms (Project No. 110051360026XN086-12).

References

- Kinjal Basu, Ibrahim Abdelaziz, Subhajit Chaudhury, Soham Dan, Maxwell Crouse, Asim Munawar, Sadhana Kumaravel, Vinod Muthusamy, Pavan Kapaniathi, and Luis A Lastras. 2024. Api-blend: A comprehensive corpora for training and benchmarking api llms. *arXiv preprint arXiv:2402.15491*.
- André Bauer, Simon Trapp, Michael Stenger, Robert Leppich, Samuel Kounev, Mark Leznik, Kyle Chard, and Ian Foster. 2024. Comprehensive exploration of synthetic data generation: A survey. *arXiv preprint arXiv:2401.02524*.
- Jiajun Chai, Guojun Yin, Zekun Xu, Chuhuai Yue, Yi Jia, Siyu Xia, Xiaohan Wang, Jiwen Jiang, Xiaoguang Li, Chengqi Dong, et al. 2025. Rlfactory: A plug-and-play reinforcement learning post-training framework for llm multi-turn tool-use. *arXiv preprint arXiv:2509.06980*.
- Chen Chen, Xinlong Hao, Weiwen Liu, Xu Huang, Xingshan Zeng, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Yuefeng Huang, et al. 2025. Acebench: Who wins the match point in tool usage? *arXiv preprint arXiv:2501.12851*.
- 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*.
- Hang He, Chuhuai Yue, Chengqi Dong, Mingxue Tian, Zhenfeng Liu, Jiajun Chai, Xiaohan Wang, Yufei Zhang, Qun Liao, Guojun Yin, Wei Lin, Chengcheng Wan, Haiying Sun, and Ting Su. 2025. [Localsearch-bench: Benchmarking agentic search in real-world local life services](#). *Preprint*, arXiv:2512.07436.
- Hang He, Chuhuai Yue, Chengqi Dong, Chengcheng Wan, Ting Su, Haiying Sun, Jiajun Chai, Xiaohan Wang, and Guojun Yin. 2026. [Vistahop: Benchmarking long-horizon visual deepsearch](#). *Preprint*, arXiv:2606.03273.
- Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060*.
- Zhexin Hu, Li Wang, Xiaohan Wang, Jiajun Chai, Xiaojun Guo, Wei Lin, and Guojun Yin. 2026. Ziprl: Adaptive multi-turn context compression with hindsight response replay. *arXiv preprint arXiv:2605.28069*.
- 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. *arXiv preprint arXiv:2503.09516*.
- Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.
- Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pages 6769–6781.
- Shirley Kokane, Ming Zhu, Tulika Manoj Awalganekar, Jianguo Zhang, Akshara Prabhakar, Thai Quoc Hoang, Zuxin Liu, Rithesh R N, Liangwei Yang, Weiran Yao, Juntao Tan, Zhiwei Liu, Shelby Heinicke, Huan Wang, Juan Carlos Niebles, Caiming Xiong, and Silvio Savarese. 2025. [Toolscan: A benchmark for characterizing errors in tool-use LLMs](#). In *ICLR 2025 Workshop on Building Trust in Language Models and Applications*.
- Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob

- Uszkoreit, Quoc Le, and Slav Petrov. 2019. [Natural questions: A benchmark for question answering research](#). *Transactions of the Association for Computational Linguistics*, 7:452–466.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.
- Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023. Api-bank: A comprehensive benchmark for tool-augmented llms. *arXiv preprint arXiv:2304.08244*.
- Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. 2025a. Search-o1: Agentic search-enhanced large reasoning models. *arXiv preprint arXiv:2501.05366*.
- Xiaoxi Li, Jiajie Jin, Yujia Zhou, Yongkang Wu, Zhonghua Li, Ye Qi, and Zhicheng Dou. 2025b. Retrollm: Empowering large language models to retrieve fine-grained evidence within generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16754–16779.
- Zihan Lin, Xiaohan Wang, Jie Cao, Jiajun Chai, Guojun Yin, Wei Lin, and Ran He. 2026. Rest: Reshaping token-level policy gradients for tool-use large language models. In *International Conference on Learning Representations*, volume 2026, pages 83901–83945.
- Weiwen Liu, Xu Huang, Xingshan Zeng, Xinlong Hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, et al. 2024. Toolace: Winning the points of llm function calling. *arXiv preprint arXiv:2409.00920*.
- Yinan Liu, Zihan Zhou, Zichun Jin, Xinyu Wang, Bin Wang, and Xiaochun Yang. 2026. [Enhancing llms in predictive political qa with semi-structured data](#). *Preprint*, arXiv:2608.21218.
- Xing Han Lù. 2024. Bm25s: Orders of magnitude faster lexical search via eager sparse scoring. *arXiv preprint arXiv:2407.03618*.
- Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khoshabi, and Hannaneh Hajishirzi. 2022. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. *arXiv preprint arXiv:2212.10511*.
- Dheeraj Mekala, Jason Weston, Jack Lanchantin, Roberta Raileanu, Maria Lomeli, Jingbo Shang, and Jane Dwivedi-Yu. 2024. Toolverifier: Generalization to new tools via self-verification. *arXiv preprint arXiv:2402.14158*.
- 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 (bfc1): from tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning, ICML’25*. JMLR.org.
- Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. 2022. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*.
- Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*.
- Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. 2025. Tool learning with large language models: A survey. *Frontiers of Computer Science*, 19(8):198343.
- Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551.
- Yongliang Shen, Kaitao Song, Xu Tan, Wenqi Zhang, Kan Ren, Siyu Yuan, Weiming Lu, Dongsheng Li, and Yueting Zhuang. 2024. Taskbench: Benchmarking large language models for task automation. *Advances in Neural Information Processing Systems*, 37:4540–4574.
- Zhengliang Shi, Shen Gao, Lingyong Yan, Yue Feng, Xiuyi Chen, Zhumin Chen, Dawei Yin, Suzan Verberne, and Zhaochun Ren. 2025. Tool learning in the wild: Empowering language models as automatic tool agents. In *Proceedings of the ACM on Web Conference 2025*, pages 2222–2237.
- Jeonghoon Shim, Gyuhyeon Seo, Cheongsu Lim, and Yohan Jo. 2025. Tooldial: Multi-turn dialogue generation method for tool-augmented language models. In *International Conference on Learning Representations*, volume 2025, pages 70414–70439.
- Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.
- David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489.

- Jin Su, Zhuofeng Zhao, Huanhuan Wang, and Hao Chen. 2026. Coal-rag: A complexity-aware legal retrieval-augmented generation method. *arXiv preprint arXiv:2608.17536*.
- Michael Sullivan, Mareike Hartmann, and Alexander Koller. 2025. [Procedural environment generation for tool-use agents](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 18544–18562, Suzhou, China. Association for Computational Linguistics.
- Hao Sun, Zile Qiao, Jiayan Guo, Xuanbo Fan, Yingyan Hou, Yong Jiang, Pengjun Xie, Yan Zhang, Fei Huang, and Jingren Zhou. 2025. Zerossearch: Incentivize the search capability of llms without searching. *arXiv preprint arXiv:2505.04588*.
- Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. *arXiv preprint arXiv:1803.06643*.
- Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, Boxi Cao, and Le Sun. 2023. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases. *arXiv preprint arXiv:2306.05301*.
- Kimi Team, Yifan Bai, Yiping Bao, Y Charles, Cheng Chen, Guanduo Chen, Haiting Chen, Huarong Chen, Jiahao Chen, Ningxin Chen, et al. 2025. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*.
- Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Musique: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.
- Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 10014–10037.
- Hongru Wang, Rui Wang, Boyang Xue, Heming Xia, Jingtao Cao, Zeming Liu, Jeff Z Pan, and Kam-Fai Wong. 2024. Appbench: Planning of multiple apis from various apps for complex user instruction. *arXiv preprint arXiv:2410.19743*.
- Qingsong Wang, Chang Yao, and Jingyuan Chen. 2026. Causal-steer: Disentangled continuous style control without parallel corpora. In *International Conference on Learning Representations*, volume 2026, pages 32767–32788.
- Ze Zhong Wang, Xingshan Zeng, Weiwen Liu, Liangyou Li, Yasheng Wang, Lifeng Shang, Xin Jiang, Qun Liu, and Kam-Fai Wong. 2025. Toolflow: Boosting llm tool-calling through natural and coherent dialogue synthesis. 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 4246–4263.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.
- Siyu Xia, Chenheng Zhang, Yanting Wu, Haoxuan Li, Jiajun Chai, Xiaohan Wang, Guojun Yin, Wei Lin, Zhouchen Lin, Haifeng Zhang, et al. 2026. Unimem: Complementary episodic-to-parametric memory for boundary-agnostic task streams. *arXiv preprint arXiv:2607.26017*.
- Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. 2024. Wizardlm: Empowering large pre-trained language models to follow complex instructions. In *International Conference on Learning Representations*, volume 2024, pages 30745–30766.
- Zhangchen Xu, Adriana Meza Soria, Shawn Tan, Anurag Roy, Ashish Sunil Agrawal, Radha Pooven-dran, and Rameswar Panda. 2025. Toucan: Synthesizing 1.5 m tool-agentic data from real-world mcp environments. *arXiv preprint arXiv:2510.01179*.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.
- Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.
- Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhengguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.
- Siyu Yuan, Kaitao Song, Jiangjie Chen, Xu Tan, Yongliang Shen, Ren Kan, Dongsheng Li, and Deqing Yang. 2024. Easytool: Enhancing llm-based agents with concise tool instruction. *arXiv preprint arXiv:2401.06201*.
- Lihang Zeng, Shaoting Zhang, and Xiaofan Zhang. 2026a. [Evidx: Evidence-aware active diagnosis with scaffolded llm agents](#). *Preprint*, arXiv:2608.24570.

Wenwen Zeng, Jinhui Zhang, Hao Chen, Zhaoyu Hu, Yongqi Liang, Jiajun Chai, Dengcan Liu, Zhenfeng Liu, Shurui Yan, Minglong Xue, et al. 2026b. Recrm-bench: Benchmarking multidimensional reward modeling for agentic recommender systems. *arXiv preprint arXiv:2605.11874*.

Xingshan Zeng, Weiwen Liu, Xu Huang, Zezhong Wang, Lingzhi Wang, Liangyou Li, Yasheng Wang, Lifeng Shang, Xin Jiang, Ruiming Tang, et al. 2026c. Toolace-r: Model-aware iterative training and adaptive refinement for tool learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 34593–34601.

Jianguo Zhang, Tian Lan, Ming Zhu, Zuxin Liu, Thai Hoang, Shirley Kokane, Weiran Yao, Juntao Tan, Zhiwei Liu, Yihao Feng, 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, Albuquerque, New Mexico. Association for Computational Linguistics.

Kangning Zhang, Weiwen Liu, Wenxiang Jiao, Kounianhua Du, Yuan Lu, Weinan Zhang, and Yong Yu. 2026a. Looptool: Closing the data-training loop for robust llm tool calls. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 21144–21164.

Yifan Zhang, Giridhar Ganapavarapu, Srideepika Jayaraman, Bhavna Agrawal, Dhaval Patel, and Achille Fokoue. 2026b. Spiral: Symbolic llm planning via grounded and reflective search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 36527–36535.

Ziliang Zhang, Yubo Zhu, Wei Tong, Jingyu Hua, Zijian Wang, Yuan Zhang, and Sheng Zhong. 2026c. [Mitigating database leakage in rag systems with keyword-grounded fact substitution](#). *Preprint*, arXiv:2608.21656.

Heng Zhou, Ao Yu, Yuchen Fan, Jianing Shi, Li Kang, Hejia Geng, Yongting Zhang, Yutao Fan, Yuhao Wu, Tiancheng He, et al. 2025. Livesearchbench: An automatically constructed benchmark for retrieval and reasoning over dynamic knowledge. *arXiv preprint arXiv:2511.01409*.

A Additional GIM Details

Algorithm 1 summarizes the GIM data-generation loop. The generator first plans tool calls and evidence references, constructs paired correct and noisy retrieval observations, generates a dialogue under the specified interaction pattern, and assembles the parsed output for MLV filtering.

Algorithm 1 Synthetic Dialogue Generation

Require: $\mathcal{Q}, \mathcal{A}, \mathcal{R}^*, \mathcal{G}^*, \mathcal{T}, \mathcal{D}, \mathcal{L}$
Ensure: Final dialogue data $\mathcal{D}_{\text{final}}$

► **Phase 1: Planning**

- 1: $\mathcal{P}_{\text{plan}} \leftarrow \text{CONSTRUCTPROMPT}(\mathcal{Q}, \mathcal{A}, \mathcal{R}^*, \dots)$
- 2: $\tau \leftarrow \mathcal{L}(\mathcal{P}_{\text{plan}})$
- 3: $(\text{calls}_1, \text{refs}_1) \leftarrow \text{PARSETRACE}(\tau)$

► **Phase 2: Augmentation**

- 4: $\mathcal{I}_1 \leftarrow []$
- 5: **for each** $(\text{call}_i, \text{ref}_i) \in (\text{calls}_1, \text{refs}_1)$ **do**
- 6: $\text{query}_i \leftarrow \text{EXTRACTQUERY}(\text{call}_i)$
- 7: $K_i \leftarrow \text{BM25}(\mathcal{D}, \text{query}_i)$
- 8: $I_i^{\text{good}} \leftarrow K_i \cup \{\text{ref}_i\}$
- 9: $I_i^{\text{bad}} \leftarrow K_i$
- 10: $\text{APPEND}(\mathcal{I}_1, (I_i^{\text{good}}, I_i^{\text{bad}}))$
- 11: **end for**

► **Phase 3: Generation**

- 12: $\mathcal{P}_{\text{dialogue}} \leftarrow \text{CONSTRUCTPROMPT}(\mathcal{Q}, \text{calls}_1, \mathcal{I}_1, \dots)$
- 13: $\mathcal{D}_{\text{raw}} \leftarrow \mathcal{L}(\mathcal{P}_{\text{dialogue}})$
- 14: $\mathcal{D}_{\text{parsed}} \leftarrow \text{PARSEDIALOGUE}(\mathcal{D}_{\text{raw}})$

► **Phase 4: Assembly**

- 15: $\mathcal{D}_{\text{final}} \leftarrow \text{ASSEMBLEOUTPUT}(\mathcal{D}_{\text{parsed}}, \mathcal{I}_1, \dots)$
- 16: **return** $\mathcal{D}_{\text{final}}$

B Prompt Templates

Figure 11 shows the system prompt used to enforce structured reasoning, tool calls, and answer formatting. Figure 12 shows the corresponding user prompt wrapper.

C Virtual Tool Library

Table 18 lists the 19 manually designed base virtual tools used by ToolForge before variant generation. The library covers 18 specialized knowledge domains, including people, creative works, historical events, geography, science, academic research, law, economics, medicine, traditional culture, the environment, sports, transportation, education, lifestyle services, military affairs, religion, and politics. A general-purpose tool is additionally included as a fallback for ambiguous, emerging, or cross-domain queries that cannot be assigned to a specialized domain.

Each base tool represents a distinct retrieval intent and defines the expected information scope for its domain. Tool variants modify tool names, descriptions, and argument schemas while preserving the underlying retrieval intent, thereby increasing schema diversity without changing the semantic coverage of the base library.

D MLV-Bench Construction and Evaluation

MLV-Bench is designed to test whether MLV rejects difficult invalid trajectories rather than only obvious formatting errors. It contains 1,972 sam-

ples: 580 human-verified positive samples drawn from the 29 interaction patterns, 1,305 rule-level hard negatives, and 87 semantic negatives. To construct the rule-level negatives, we start from nine static validation rules and use Monte Carlo Tree Search (MCTS) to explore combinations of protocol, schema, argument, and answer-consistency violations. For each rule and interaction pattern, we retain the five most challenging invalid trajectories. The semantic negatives cover the three model-level criteria: tool-choice correctness, logical soundness over evidence, and rationale-action alignment. MCTS is used only to construct this diagnostic benchmark, not during ToolForge data generation or model training.

Table 7: MLV performance on MLV-Bench.

Metric	Score (%)
Accuracy	98.47
Precision	96.55
Recall	98.25
F1 Score	97.39

Table 7 reports MLV performance on MLV-Bench. MLV achieves 98.47% accuracy and 97.39% F1. The relatively balanced precision and recall indicate that the result is not driven solely by rejecting most inputs or accepting superficially valid trajectories. This behavior is consistent with the complementary roles of the two validation layers: RVL detects protocol, schema, argument, and consistency errors, whereas MVL evaluates tool choice, evidence-grounded reasoning, and rationale-action alignment. Moreover, the MCTS-generated negatives combine multiple violations across interaction patterns, making them less dependent on isolated surface-level errors. Because MLV-Bench follows the error taxonomy defined in this work, these results support MLV reliability within this taxonomy rather than unrestricted generalization to all real-world tool errors.

E ToolForge Synthesis Pass Rates

Table 8 reports the proportion of synthesized trajectories retained by MLV after KSP and GIM. Pass rates remain above 88% across all four tool-use paradigms, with a modest decrease as the interaction structure becomes more complex.

These rates measure pipeline yield rather than downstream model quality. Their consistently high values indicate that KSP and GIM produce mostly admissible trajectories, while the modest decrease

Table 8: Pass rates of synthesized trajectories across the four tool-use paradigms after Multi-Layer Validation.

Paradigm	Pass Rate (%)
SRST	91.70
SRMT	91.25
MRST	90.30
MRMT	88.83

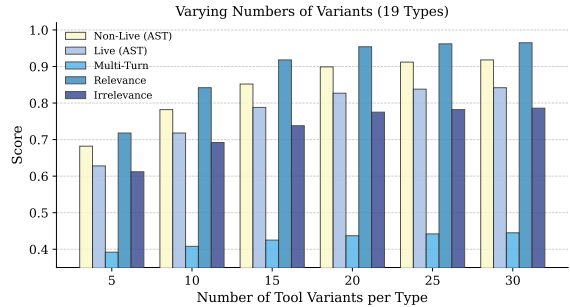


Figure 6: BFCL performance when varying the number of variants with 19 tool types fixed.

from SRST to MRMT is consistent with the additional cross-turn dependencies and consistency constraints introduced by more complex interactions.

F Additional QA and Robustness Results

This appendix section collects QA and robustness analyses that support, but are not central to, the main experimental narrative. Table 9 examines the general non-retrieval capabilities of ToolForge-8B. Table 10 reports the same QA evaluation at the 4B scale. Table 11 contextualizes ToolForge-8B against larger open-source and API models under the same prompting and retrieval setup. Table 12 evaluates whether models can solve the internal ToolForge benchmark splits by selecting the correct tools, tool-use paradigm, and final answer.

Recent evidence-seeking systems span predictive political QA (Liu et al., 2026), legal RAG (Su et al., 2026), leakage-aware RAG (Zhang et al., 2026c), active diagnosis (Zeng et al., 2026a), and multimodal deep search (He et al., 2026). Agent evaluation has also expanded to domain-specific settings such as recommendation (Zeng et al., 2026b). ToolForge does not evaluate these applications directly; we include them here to delimit broader settings for future extensions of tool-use data synthesis.

Retrieval-unrelated robustness. Beyond retrieval, targeted post-training has also been explored for controllable generation, including disentangled style control (Wang et al., 2026). We

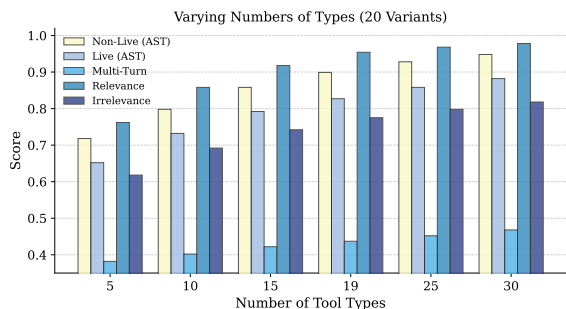


Figure 7: BFCL performance when varying the number of tool types with 20 variants fixed.

therefore evaluate whether ToolForge-8B preserves general capabilities outside its primary tool-use training objective. Table 9 covers mathematical reasoning, general reasoning, instruction following, alignment evaluation, and code generation. ToolForge-8B remains comparable to the Qwen3-8B base model and improves six of the eight reported metrics, indicating that its gains in tool use do not come at the expense of broad non-retrieval capabilities.

4B-scale QA results. Table 10 repeats the five-dataset QA comparison with Qwen3-4B as the backbone. ToolForge-4B obtains the best average score in this setting, improving over the TIR baseline by 23.4%. This result indicates that the synthetic data is not only effective for the 8B model used in the main paper, but also transfers to a smaller backbone.

Comparison with larger open-source and API models. Table 11 compares ToolForge-8B with larger open-source models and API models on the same five QA datasets. This comparison is not intended as a parameter-matched baseline; rather, it contextualizes how far a compact model fine-tuned on 4,250 ToolForge samples can go under the same prompting setup and Wikipedia retrieval environment. ToolForge-8B achieves the best average score in this table and is competitive with substantially larger models.

Internal ToolForge benchmark splits. We additionally reserve 200 test samples for each of the four ToolForge paradigms (SRST, SRMT, MRST, and MRMT), separate from the training data. Each split is evaluated with three metrics: Optimal-Tool EM checks whether the model selects the required tool set, Optimal Tool-calling Paradigm EM checks whether it selects the intended tool-use structure, and Answer EM checks whether the final answer is correct. The benchmark uses the same controlled

golden/noisy context construction as ToolForge synthesis, with BM25s as the retriever. Table 12 reports the average across the four splits.

G Full Tool-Use Benchmark Results

Tables 13 and 14 report BFCL and API-Bank results, complementing the main-paper summary.

On BFCL, ToolForge-8B achieves an average score of 0.778, a 3.6% relative improvement over Qwen3-8B, and ranks joint fourth among 21 models. The largest gain occurs in irrelevance detection, which increases from 0.719 to 0.775. ToolForge-8B improves Non-live AST, Live AST, and relevance detection. Its multi-turn score rises from 0.418 to 0.437 but remains below larger or specialized systems. The benefit lies in reliable tool use and evidence discrimination rather than uniform gains across dimensions.

On API-Bank, ToolForge-8B improves the Qwen3-8B average from 0.622 to 0.647. It improves Call from 0.707 to 0.737 and Retrieval + Call from 0.537 to 0.556. ToolForge-8B ranks sixth among 18 models and outperforms Qwen3-14B on average, although larger proprietary models remain stronger.

These gains support the design of ToolForge. Tool variants diversify schemas, while paired golden and noisy observations encourage evidence verification and rejection of plausible but insufficient results. Multi-layer validation removes structurally invalid and semantically inconsistent trajectories. These components improve generalization to unseen tool interfaces and interaction settings.

H ToolScan Extension Study

Table 15 reports the ToolScan comparison used to evaluate whether ToolForge can incorporate additional tool-error categories without redesigning the virtual-tool library. The expanded setting adds five error categories to the three default perturbations: insufficient API calls, incorrect argument type, incorrect argument name, repeated API calls, and invalid format error. Metric abbreviations follow the ToolScan benchmark.

I Full Synthetic-Data Comparisons

Tables 16 and 17 report complete BFCL results for models trained on ToolForge, ToolACE, Tool-LLM, and xLAM data. Figures 9 and 10 visualize performance across the five BFCL dimensions.

Table 9: Performance evaluation of ToolForge-8B across eight retrieval-unrelated downstream tasks. All metrics are calculated using the official scripts. Best in bold.

Model	Mathematical Reasoning		General Reasoning		Alignment Evaluation		Code Generation	
	MATH-500	AIME-2025	MMLU-Redux	GPQA-Diamond	IFEval	Arena-Hard	HumanEval	MBPP
	Acc.	Pass@64	Acc.	Pass@10	Acc.	Win rate	Pass@1	Pass@1
Qwen3-8B (Base)	97.4	67.3	87.5	62.0	85.0	85.8	57.9	72.1
ToolForge-8B (Ours)	96.3	71.2	88.4	61.3	86.8	87.1	58.3	73.4

Table 10: Performance comparison of ToolForge-4B and baseline methods across five QA datasets. All retrieval-based methods use the same Wikipedia retriever. * denotes out-of-domain datasets. Percentages in the Avg. column denote relative improvement over TIR. Best in bold.

Model	Avg. (↑ / ↓ vs. TIR)	General QA			Multi-Hop QA	
		NQ*	TriviaQA*	PopQA*	Musique*	Bamboogle*
Qwen3-4B + Direct Inference						
TIR (Baseline)	0.321 (−)	0.308	0.581	0.351	0.077	0.290
Direct Inference	0.230 (↓ 28.3%)	0.158	0.513	0.157	0.033	0.290
CoT	0.184 (↓ 42.7%)	0.149	0.375	0.146	0.022	0.228
Qwen3-4B + Inference with Retrieval						
IRCoT	0.308 (↓ 4.0%)	0.244	0.516	0.343	0.095	0.340
RAG	0.304 (↓ 5.2%)	0.289	0.559	0.337	0.075	0.261
Search-o1	0.272 (↓ 15.3%)	0.203	0.498	0.231	0.069	0.357
Qwen3-4B + RL Training						
R1-base	0.251 (↓ 21.7%)	0.277	0.401	0.200	0.077	0.299
R1-instruct	0.271 (↓ 15.6%)	0.297	0.434	0.193	0.126	0.302
Search-R1	0.354 (↑ 10.2%)	0.283	0.595	0.387	0.133	0.372
ZEROSEARCH-base	0.357 (↑ 11.2%)	0.291	0.577	0.396	0.118	0.402
ZEROSEARCH-instruct	0.344 (↑ 7.3%)	0.296	0.590	0.383	0.120	0.333
Qwen3-4B + Data Synthesis + SFT						
ToolForge-4B (Ours)	0.396 (↑ 23.4%)	0.299	0.581	0.399	0.254	0.447

Under pure synthetic-data training, ToolForge achieves the highest average score with both Qwen3-8B and LLaMA-3.1-8B-Instruct, using 4.25k examples rather than the 25k used by the other datasets. It improves over ToolACE by 4.3% with Qwen3-8B and 1.7% with LLaMA-3.1-8B-Instruct. Figure 9 shows that the gains vary across dimensions. ToolACE remains stronger on non-live function calling, while ToolLLM achieves the highest relevance score but performs poorly on multi-turn tool use and irrelevance detection. ToolForge instead presents a more balanced profile, with consistent strength in live function calling, multi-turn interactions, and irrelevance detection. This pattern aligns with its emphasis on multi-turn trajectories and recovery from plausible but insufficient retrieval observations.

Adding 0.2k BFCL in-domain examples improves the average performance of all dataset and backbone combinations. ToolForge retains the highest average score, reaching 0.790 with Qwen3-8B and 0.698 with LLaMA-3.1-8B-Instruct, rep-

resenting improvements of 2.5% and 1.5% over ToolACE, respectively. Figure 10 shows that the relative performance profiles remain consistent under in-domain supervision. ToolForge remains strong on multi-turn tool use and irrelevance detection, indicating that its benefits transfer across backbones and complement limited task-specific training data.

J Qualitative Examples

Tool-use paradigms. Figure 8 illustrates the four tool-use paradigms instantiated in ToolForge. Each row pairs a question, its golden contexts, and reference answer with the corresponding trajectory structure. Gray, cyan, and pink nodes denote the start, end, and tool calls, respectively, while red bridge arrows indicate dependencies in which earlier outputs condition subsequent calls.

The paradigms vary in the number of interaction rounds and tool calls per round. Single-Round Single-Tool (SRST) performs one call in one round, whereas Single-Round Multi-Tool (SRMT) per-

Table 11: Performance comparison of ToolForge-8B and open-source/API models across five QA datasets using the Wikipedia dump. * denotes out-of-domain datasets. Percentages in the Avg. column denote relative improvement over Qwen3-8B. Best in bold.

Model	Avg. (↑ / ↓ vs. Qwen3-8B)	General QA			Multi-Hop QA	
		NQ*	TriviaQA*	PopQA*	Musique*	Bamboogle*
Open-Source						
Qwen3-8B (Baseline)	0.322 (–)	0.208	0.523	0.301	0.196	0.380
Qwen3-4B	0.272 (↓ 15.6%)	0.193	0.501	0.237	0.078	0.351
Qwen3-14B	0.334 (↑ 4.0%)	0.239	0.531	0.307	0.207	0.386
Qwen3-32B	0.357 (↑ 11.0%)	0.257	0.567	0.321	0.234	0.408
Qwen3-235B	0.399 (↑ 24.0%)	0.302	0.601	0.342	0.272	0.480
Qwen2.5-72B-Instruct	0.345 (↑ 7.1%)	0.362	0.571	0.336	0.127	0.329
GPT-4o-2024-08-06	0.353 (↑ 9.6%)	0.168	0.593	0.236	0.293	0.477
Grok-3	0.406 (↑ 26.1%)	0.296	0.573	0.396	0.295	0.471
Deepseek-R1	0.313 (↓ 2.7%)	0.196	0.511	0.298	0.185	0.377
Claude-Sonnet-3.5	0.322 (≈ 0.0%)	0.252	0.465	0.330	0.216	0.351
Claude-Sonnet-3.7	0.373 (↑ 15.7%)	0.293	0.487	0.399	0.291	0.393
Claude-Sonnet-4	0.411 (↑ 27.8%)	0.308	0.600	0.405	0.336	0.408
Qwen3-4B + Data Synthesis + SFT						
ToolForge-4B	0.396 (↑ 23.0%)	0.299	0.581	0.399	0.254	0.447
Qwen3-8B + Data Synthesis + SFT						
ToolForge-8B	0.416 (↑ 29.2%)	0.309	0.602	0.410	0.278	0.480

Table 12: Performance comparison on the four internal ToolForge benchmark splits (SRST, SRMT, MRST, and MRMT). Percentages in the Avg. column denote relative improvement over ToolForge-8B. Best results are in bold.

Model	Avg. (↑ / ↓ vs. ToolForge-8B)	Metric		
		Optimal-Tool EM	Optimal Tool-calling Paradigm EM	Answer EM
Other Tool-Augmented Models				
ToolForge-8B (Baseline)	0.917 (–)	0.904	0.923	0.925
xLAM-2-8b-fc-r	0.556 (↓ 39.4%)	0.453	0.673	0.543
ToolACE-8B	0.701 (↓ 23.6%)	0.661	0.731	0.712
Qwen3-235B-A3B-Instruct-2507	0.890 (↓ 2.9%)	0.887	0.911	0.873
GPT-4o-2024-08-06	0.570 (↓ 37.8%)	0.531	0.621	0.557
GPT-5.2-2025-12-11	0.943 (↑ 2.9%)	0.934	0.946	0.950
o3-2025-04-16	0.925 (↑ 0.9%)	0.921	0.931	0.922
Gemini-3-Pro-Preview	0.914 (↓ 0.3%)	0.933	0.910	0.899
GLM-4.6	0.914 (↓ 0.3%)	0.899	0.941	0.902

forms multiple calls in the same round to gather complementary evidence. Multi-Round Single-Tool (MRST) uses one call per round and passes information across rounds. Multi-Round Multi-Tool (MRMT) combines multiple calls per round with cross-round dependencies for more complex evidence acquisition and composition. The *Round* and *Tool* annotations specify the number of interaction rounds and tool calls per round, respectively.

Recovery from plausible retrieval noise. Table 19 presents a two-hop question derived from PopQA: “Which country was the director of the 1978 film ‘The Band’ born in?” Answering it requires first identifying the film’s director and then determining that person’s country of birth.

Both traces initially receive the same plausible

but insufficient observation: Daniel Roher directed *Once Were Brothers: Robbie Robertson and the Band*. Although the tool call succeeds and the result is topically related, it concerns a different film. GPT-5.2 accepts the lexical overlap as sufficient evidence, queries Daniel Roher’s nationality, and incorrectly answers Canada. ToolForge-8B instead detects that the retrieved title does not match the target entity, reformulates the query, and identifies Avi Nesher as the director of the 1978 film. It then completes the second hop by retrieving his birthplace and correctly answers Israel.

This case illustrates the intended role of ToolForge supervision: training models to verify whether retrieved evidence supports the target relation and to revise successful but misleading tool calls without an explicit error signal.

Table 13: BFCL test results, updated April 12, 2026. “FC” denotes function-calling mode, whereas “Prompt” denotes function-call extraction using a customized prompt. All metrics are calculated using the official scripts. Changes in Avg. are relative to Qwen3-8B; best results are in bold.

Rank Model	Avg. (↑ / ↓ vs. Qwen3-8B)	Single-Turn		Multi-Turn	Hallucination	
		Non-live	Live	Multi-turn	Relevance	Irrelevance
Baseline						
(-) Qwen3-8B (FC)	0.751 (-)	0.876	0.805	0.418	0.938	0.719
Other Tool-Augmented Models						
1 BitAgent-Bounty-8B	0.807 (↑ 7.4%)	0.816	0.931	0.624	0.687	0.975
2 GLM-4.6 (FC)	0.793 (↑ 5.6%)	0.876	0.809	0.680	0.750	0.850
3 o3-2025-04-16 (Prompt)	0.790 (↑ 5.2%)	0.819	0.732	0.623	0.938	0.840
4 ToolForge-8B (FC)(ours)	0.778 (↑ 3.6%)	0.899	0.827	0.437	0.954	0.775
4 Gemini-3-Pro-Preview (Prompt)	0.778 (↑ 3.6%)	0.906	0.831	0.608	0.688	0.856
6 Qwen3-32B (FC)	0.778 (↑ 3.6%)	0.888	0.820	0.479	0.938	0.764
7 Qwen3-235B-A3B-Instruct-2507 (Prompt)	0.773 (↑ 3.0%)	0.903	0.787	0.446	0.938	0.789
7 Grok-4-1-fast-reasoning (FC)	0.773 (↑ 3.0%)	0.883	0.785	0.589	0.813	0.794
9 Claude-Opus-4.5-20251101 (FC)	0.768 (↑ 2.3%)	0.886	0.798	0.684	0.625	0.847
10 xLAM-2-8b-fc-r (FC)	0.747 (↓ 0.5%)	0.846	0.680	0.700	0.875	0.633
11 Moonshotai-Kimi-K2-Instruct (FC)	0.746 (↓ 0.6%)	0.816	0.787	0.506	0.750	0.873
12 Nanbeige4-3B-Thinking-2511 (FC)	0.740 (↓ 1.5%)	0.816	0.794	0.511	0.750	0.831
13 ToolACE-2-8B (FC)	0.737 (↓ 1.9%)	0.871	0.774	0.384	0.750	0.908
14 Qwen3-14B (FC)	0.737 (↓ 1.9%)	0.845	0.800	0.348	0.875	0.819
15 GPT-4.1-2025-04-14 (FC)	0.732 (↓ 2.5%)	0.830	0.700	0.389	0.875	0.865
16 Qwen3-30B-A3B-Instruct-2507 (FC)	0.710 (↓ 5.4%)	0.858	0.780	0.300	0.813	0.800
17 ToolACE-8B (FC)	0.692 (↓ 8.0%)	0.893	0.732	0.144	0.854	0.839
18 Llama-3.3-70B-Instruct (FC)	0.679 (↓ 9.6%)	0.880	0.766	0.215	1.000	0.535
19 GPT-5.2-2025-12-11 (FC)	0.670 (↓ 10.8%)	0.819	0.704	0.281	0.750	0.794
20 GPT-5-nano-2025-08-07 (FC)	0.652 (↓ 13.2%)	0.680	0.594	0.345	0.750	0.891
21 Qwen3-235B-A3B-Instruct-2507 (FC)	0.642 (↓ 14.5%)	0.374	0.689	0.454	0.875	0.817

Table 14: API-Bank test results. All metrics are calculated using the official scripts. Percentages in the Avg. column denote relative improvement over Qwen3-8B. Best results are in bold.

Rank	Model	Avg. (↑ / ↓ vs. Qwen3-8B)	Metric Call Retrieval + Call
Baseline			
(-)	Qwen3-8B	0.622 (-)	0.707 0.537
Other Tool-Augmented Models			
1	Claude-Opus-4-5-20251101	0.901(↑ 44.8%)	0.917 0.884
2	GPT-5.2-2025-12-11	0.851 (↑ 36.7%)	0.931 0.770
3	Gemini-3-Pro-Preview	0.825 (↑ 32.6%)	0.899 0.751
4	GLM-4.6	0.712 (↑ 14.5%)	0.793 0.631
5	Qwen3-32B	0.680 (↑ 9.3%)	0.773 0.586
6	ToolForge-8B	0.647 (↑ 4.0%)	0.737 0.556
7	Qwen3-14B	0.642 (↑ 3.2%)	0.717 0.567
8	GPT-4-0613	0.624 (↑ 0.3%)	0.759 0.489
9	ToolACE-8B	0.617 (↓ 0.8%)	0.759 0.474
10	GPT-3.5-turbo-0125	0.613 (↓ 1.4%)	0.704 0.523
11	GPT-4o-mini-2024-07-18	0.603 (↓ 3.1%)	0.747 0.459
12	GPT-4o-2024-04-09	0.596 (↓ 4.2%)	0.762 0.430
13	GPT-4-turbo-2024-04-09	0.559 (↓ 10.1%)	0.724 0.393
14	LLaMA-3.1-8B-Instruct	0.541 (↓ 13.0%)	0.712 0.370
15	Lynx-7B	0.402 (↓ 35.4%)	0.499 0.304
16	xLAM-7b-fc-r	0.272 (↓ 56.3%)	0.328 0.215
17	ChatGLM-6B	0.185 (↓ 70.3%)	0.236 0.133
18	Alpaca-7B	0.149 (↓ 76.1%)	0.246 0.052

Data Type	Example	Start End Tool Bridge	Structure
SRST	query: Who is the paternal grandfather of Henry Stopford? answer: James Stopford, 2nd Earl of Courtown golden contexts: (1) Courtown was the eldest son of James Stopford, 2nd Earl of Courtown, and his wife Mary (née Powys). (2) The fifth son of James Stopford, 3rd Earl of Courtown and his wife Mary, daughter of Henry Scott, 3rd Duke of Buccleuch, he was born at Dalkeith. explain: Round 1 (Direct Information Retrieval): The system executes a single tool invocation to directly look up and retrieve the requested fact—the name of Henry Stopford's paternal grandfather. The final answer is the direct output of this lone tool call.		Round = 1 Tool = 1
SRMT	query: Which documentary came first, Mandela or Grand Canyon? answer: Grand Canyon golden contexts: (1) Grand Canyon is a 1958 American short documentary film directed by James Algar and produced by Walt Disney Productions. (2) Mandela (also called Mandela: Son of Africa, Father of a Nation) is a 1996 documentary film directed by Angus Gibson and Jo Menell. explain: Round 1 (Parallel Information Gathering): The system executes two independent tool invocations in parallel to acquire the necessary data points: One invocation to retrieve the release year of the documentary Mandela; A second invocation to retrieve the release year of the documentary Grand Canyon. The final answer is derived by performing a comparison on the results of these parallel tool calls.		Round = 1 Tool > 1
MRST	query: The creator of von Neumann's inequality, was what nationality? answer: Hungarian-American golden contexts: (1) John von Neumann (; Hungarian: "Neumann János Lajos" ,] ; December 28, 1903 – February 8, 1957) was a Hungarian-American mathematician, physicist, and computer scientist. (2) In operator theory, von Neumann's inequality, due to John von Neumann, states that, for a fixed contraction "T", the polynomial functional calculus map is itself a contraction. explain: Round 1 (Creator Identification): The system first invokes a specialized tool to retrieve information about the scientific concept, "von Neumann's inequality," and identify its creator; Round 2 (Nationality Retrieval): The output from Round 1 ("John von Neumann") serves as the direct input for a subsequent, single tool invocation to find the nationality of that individual. The final answer is derived from the result of the second round.		Round = 2 Tool = 1
MRMT	query: Do both directors of films Turbulence (1997 Film) and Nagamadathu Thampuratti have the same nationality? answer: No golden contexts: (1) Nambiathusseril Varkey John(14 October 1927 – 17 July 2014), better known by his screen name Sasikumar, was an Indian film director who worked in Malayalam cinema. (2) Robert Butler(born November 16, 1927) is an American film and Emmy Award- winning television director. (3) Turbulence is a 1997 American disaster crime thriller film directed by Robert Butler and starring Ray Liotta and Lauren Holly. (4) Nagamadathu Thampuratti is a 1982 Indian Malayalam film, directed by J. Sasikumar and produced by E. K. Thyagarajan. explain: Round 1 (Director Identification): The system must execute two parallel tool invocations to retrieve the director's name for Turbulence and Nagamadathu Thampuratti respectively; Round 2 (Nationality Retrieval): The outputs from Round 1 (the two directors' names) serve as inputs for a second set of parallel tool invocations to find the nationality of each director. The final answer is derived by comparing the results from the second round.		Round = 2 Tool > 1

Figure 8: Examples of four tool-use paradigms in ToolForge. SRST/SRMT vary the number of tools within a single round, while MRST/MRMT introduce cross-round dependencies.

Table 15: ToolScan results for plug-and-play tool-error extension. The expanded ToolForge setting uses eight tool-error categories while keeping the same 19 base virtual tools and 4,250-sample training budget. Best results are in bold.

Rank	Model	Success Rate	Tool-Error Category							Avg.
			IAC	RAC	IAV	IFE	IAT	IAN	IFN	
1	GPT-4-0125-preview	0.73	0.84	1.00	0.94	1.00	0.93	0.94	0.94	
2	ToolForge-8B (8 types)	0.72	0.80	1.00	0.90	0.96	0.93	0.94	0.92	
2	xLAM-8x7B	0.72	0.79	0.95	0.92	1.00	0.95	0.93	0.91	
4	xLAM-7B	0.64	0.78	0.98	0.93	0.98	0.95	0.86	0.91	
5	GPT-3.5-turbo-1106	0.67	0.70	0.94	0.86	0.95	0.94	0.93	0.91	
6	GPT-4o-turbo-2024-05-13	0.70	0.59	0.93	0.88	0.94	0.92	0.88	0.92	
7	ToolForge-8B (3 types)	0.62	0.66	0.77	0.90	0.80	0.62	0.75	0.91	
7	Mixtral-8x22B-Instruct-v0.1	0.40	0.70	0.92	0.62	0.99	0.60	0.81	0.78	
9	Code-Llama-13B	0.41	0.43	0.69	0.70	0.66	0.72	0.70	0.68	
10	DeepSeek-R1-Distill-Qwen-14B	0.32	0.47	0.66	0.63	0.43	0.73	0.72	0.73	
11	Meta-Llama3-8B	0.27	0.58	0.40	0.42	0.71	0.49	0.42	0.64	
12	Vicuna-13B-16k	0.16	0.27	0.26	0.61	0.69	0.39	0.40	0.50	
13	Mistral-7B-Instruct-v0.1	0.24	0.49	0.54	0.21	0.23	0.52	0.47	0.60	
14	DeepSeek-R1-Distill-Qwen-7B	0.11	0.19	0.31	0.35	0.11	0.23	0.37	0.37	
15	Mixtral-8x7B-Instruct-v0.1	0.10	0.11	0.12	0.11	0.10	0.15	0.12	0.20	

Table 16: BFCL performance of models trained on different synthetic datasets. All metrics are calculated using the official evaluation scripts. Relative changes in Avg. are computed against ToolACE for each backbone; best results within each backbone are in bold.

Training data	Avg.	Single Turn		Multi Turn	Hallucination	
		Non-live	Live	Multi-turn	Relevance	Irrelevance
<i>Qwen3-8B</i>						
ToolACE (25k; baseline)	0.746 (–)	0.941	0.765	0.386	0.931	0.707
ToolLLM (25k)	0.423 (↓ 43.2%)	0.421	0.561	0.000	1.000	0.135
xLAM (25k)	0.557 (↓ 25.3%)	0.911	0.576	0.156	0.805	0.338
ToolForge (4.25k)	0.778 (↑ 4.3%)	0.899	0.827	0.437	0.954	0.775
<i>LLaMA-3.1-8B-Instruct</i>						
ToolACE (25k; baseline)	0.679 (–)	0.896	0.714	0.165	0.756	0.864
ToolLLM (25k)	0.373 (↓ 45.1%)	0.425	0.396	0.000	1.000	0.044
xLAM (25k)	0.429 (↓ 36.8%)	0.819	0.432	0.044	0.732	0.119
ToolForge (4.25k)	0.690 (↑ 1.7%)	0.933	0.701	0.227	0.721	0.866

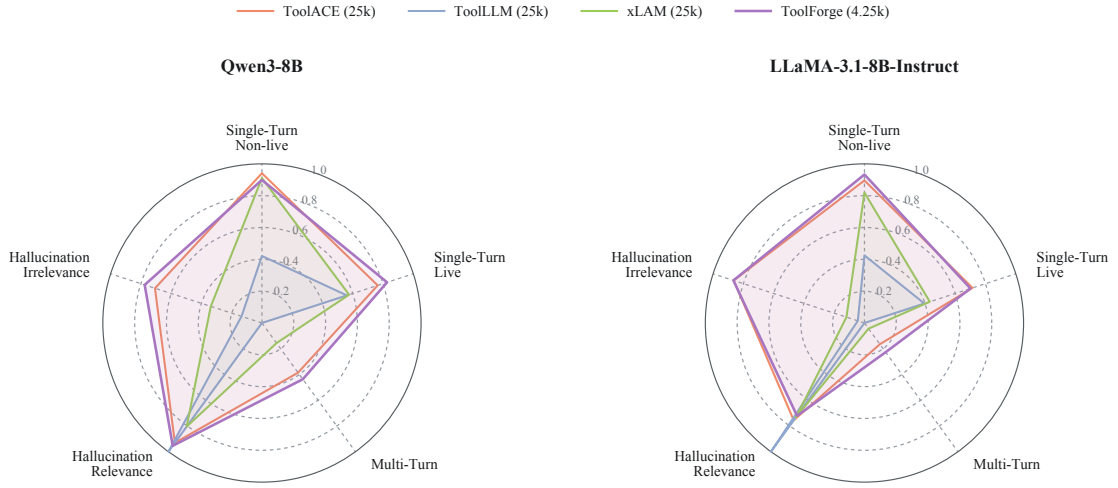


Figure 9: BFCL performance of models trained on different synthetic datasets using Qwen3-8B (left) and LLaMA-3.1-8B-Instruct (right) as backbones. ToolForge uses 4.25k training samples, whereas the compared datasets use 25k samples. Higher values indicate better performance.

Table 17: BFCL performance of models trained on different synthetic datasets mixed with 0.2k in-domain examples. All metrics are calculated using the official evaluation scripts. Relative changes in Avg. are computed against ToolACE for each backbone; best results within each backbone are in bold.

Training data	Avg.	Single Turn		Multi Turn	Hallucination	
		Non-live	Live	Multi-turn	Relevance	Irrelevance
<i>Qwen3-8B</i>						
ToolACE (25k + 0.2k; baseline)	0.771 (–)	0.947	0.811	0.432	0.944	0.721
ToolLLM (25k + 0.2k)	0.469 (↓ 39.1%)	0.464	0.621	0.103	0.982	0.176
xLAM (25k + 0.2k)	0.592 (↓ 23.2%)	0.923	0.578	0.245	0.842	0.371
ToolForge (4.25k + 0.2k)	0.790 (↑ 2.5%)	0.921	0.833	0.455	0.956	0.783
<i>LLaMA-3.1-8B-Instruct</i>						
ToolACE (25k + 0.2k; baseline)	0.688 (–)	0.901	0.723	0.162	0.789	0.864
ToolLLM (25k + 0.2k)	0.399 (↓ 42.0%)	0.453	0.427	0.081	0.983	0.051
xLAM (25k + 0.2k)	0.512 (↓ 25.5%)	0.878	0.534	0.152	0.730	0.267
ToolForge (4.25k + 0.2k)	0.698 (↑ 1.5%)	0.941	0.723	0.232	0.726	0.867

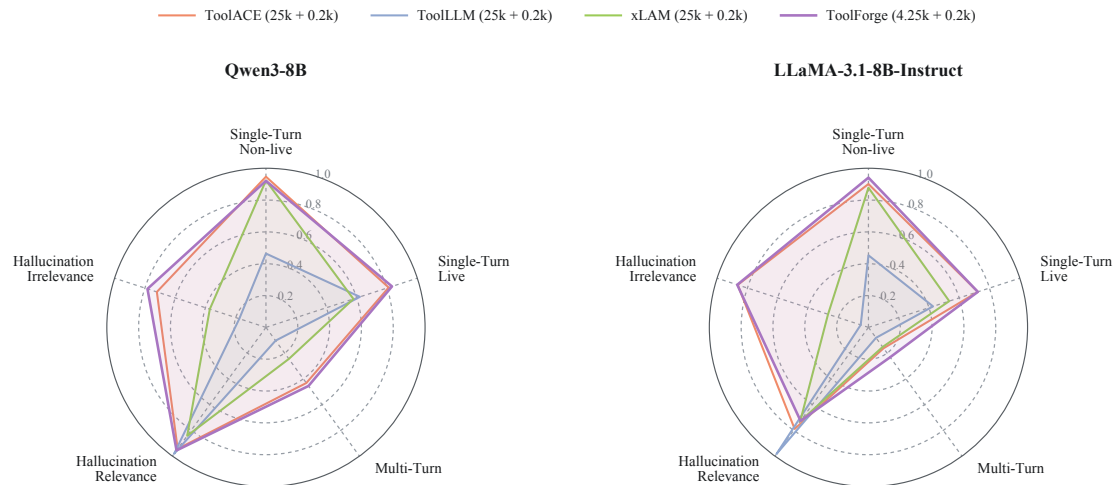


Figure 10: BFCL performance after mixing each synthetic dataset with 0.2k in-domain examples, using Qwen3-8B (left) and LLaMA-3.1-8B-Instruct (right) as backbones. ToolForge uses 4.25k synthetic samples, whereas the compared datasets use 25k samples. Higher values indicate better performance.

Table 18: The 19 base virtual tools used in ToolForge.

Tool name	Description
person_information_search	Searches biographical information about people, including identity, life history, family relations, education, and career records.
creation_information_search	Searches information about creative works, including plots, themes, technical specifications, artistic features, and content for literature, films, music, art, games, magazines, and related works.
historical_event_information_search	Searches factual information about historical events, including time, location, process, participants, outcomes, and influence for political events, conflicts, social movements, disasters, and major meetings.
geographic_information_search	Searches spatial information about geographic entities, including locations, coordinates, area, terrain, climate, landmarks, cities, countries, regions, buildings, organizations, companies, and demographic attributes.
scientific_information_search	Searches foundational scientific knowledge, including theories, concepts, laws, principles, mechanisms, definitions, and applications in physics, chemistry, biology, mathematics, and related sciences.
academic_research_search	Searches academic research outputs, including papers, reports, conferences, journals, methods, experimental results, scholarly claims, citation information, and publication metadata.
legal_information_search	Searches legal and policy information, including statutes, regulations, judicial interpretations, administrative rules, scope of application, legal effect, and implementation details.
economic_information_search	Searches quantitative economic and financial information, including GDP, inflation, exchange rates, stock prices, interest rates, market data, monetary policy, and investment statistics.
medical_information_search	Searches medical and health information, including diseases, symptoms, diagnosis, treatment, drugs, medical procedures, clinical guidance, prevention, and health data.
tradition_cultural_information_search	Searches traditional culture information, including customs, festivals, ethnic cultures, symbols, rituals, cultural meanings, and traditional crafts.
environmental_information_search	Searches environmental and ecological information, including environmental conditions, ecosystems, climate data, monitoring records, biodiversity, pollution indicators, and conservation status.
sports_information_search	Searches sports information, including competitions, results, records, statistics, schedules, rankings, rules, and event organization.
transportation_information_search	Searches transportation information, including transportation networks, systems, infrastructure, technologies, routes, capacity, planning, and technical standards.
education_information_search	Searches education information, including education systems, curricula, teaching methods, education policy, learning methods, assessment, standards, and institutional structures.
lifestyle_service_information_search	Searches everyday service information, including dining, shopping, travel, accommodation, consumer guidance, service reviews, and practical life information.
military_information_search	Searches public military technical information, including equipment, weapon systems, defense technology, specifications, operational characteristics, and development history.
religious_information_search	Searches religious and philosophical information, including doctrines, scriptures, philosophical concepts, schools of thought, spiritual culture, and belief systems.
politics_information_search	Searches political information, including political systems, government structures, parties, elections, administrative divisions, international relations, and power structures.
general_information_search	A fallback search tool for exploratory, ambiguous, emerging, real-time, or cross-domain queries that do not clearly fit any specialized tool description.

Table 19: Case study comparing the reasoning process of GPT-5.2 and ToolForge-8B on the PopQA question.

GPT-5.2	ToolForge-8B
Question: Which country was the director of the 1978 film “The Band” born in?	
Turn 1: Initial retrieval <pre><think> The question requires two reasoning steps. I first need to identify the director of the 1978 film “The Band,” and then determine the country in which that director was born. I will begin by searching for the film’s director. </think> <tool_call> {"name": "creation_information_search", "arguments": {"query": "The Band film director"}} </tool_call> <tool_response> “Once Were Brothers: Robbie Robertson and the Band” is a documentary film directed by Daniel Roher. </tool_response> <i>Plausible but insufficient evidence.</i></pre>	Turn 1: Initial retrieval <pre><think> The question requires two reasoning steps. I first need to identify the director of the 1978 film “The Band,” and then determine the country in which that director was born. I will begin by searching for the film’s director. </think> <tool_call> {"name": "creation_information_search", "arguments": {"query": "The Band film director"}} </tool_call> <tool_response> “Once Were Brothers: Robbie Robertson and the Band” is a documentary film directed by Daniel Roher. </tool_response> <i>Plausible but insufficient evidence.</i></pre>
Turn 2: Evidence acceptance and second hop <pre><think> The result concerns a film about The Band and identifies Daniel Roher as its director. I will use Daniel Roher to complete the second reasoning step and determine his country of origin. </think> <tool_call> {"name": "person_information_search", "arguments": {"query": "Daniel Roher nationality"}} </tool_call> <tool_response> Daniel Roher is a Canadian documentary filmmaker. </tool_response> <answer>Canada</answer></pre>	Turn 2: Reflection and corrective retrieval <pre><think> The tool call succeeded, but the evidence refers to “Once Were Brothers,” not the 1978 film “The Band.” It therefore does not identify the director of the target film. I should disambiguate the title and refine the query. </think> <tool_call> {"name": "creation_information_search", "arguments": {"query": "1978 Israeli film The Band director"}} </tool_call> <tool_response> “The Band,” also known as “The Troupe,” is a 1978 Israeli musical film directed by Avi Nesher. </tool_response></pre>
	Turn 3: Second reasoning hop <pre><think> The corrected evidence identifies Avi Nesher as the director. I now need to determine his birthplace and the corresponding country. </think> <tool_call> {"name": "person_information_search", "arguments": {"query": "Avi Nesher birthplace"}} </tool_call> <tool_response> Avi Nesher was born in Ramat Gan, Israel. </tool_response> <answer>Israel</answer></pre>
Final Answer: Canada ✗ <i>The model accepts topical overlap as sufficient evidence.</i>	Final Answer: Israel ✓ <i>The model detects the entity mismatch, revises the query, and completes the second hop.</i>

System Prompt

Role

You are a helpful assistant responsible for answering the user's original question. You are adept at designing complex tool-calling sequences and multi-step reasoning chains. During reasoning, you must integrate a reflection mechanism to support self-monitoring, error correction, and dynamic optimization.

Task Description

You have three response modes:

1. Based on the user's original question, formulate a plan and select the appropriate tool or tools.
2. Based on a tool result, analyze the retrieved information and decide whether another tool call is required.
3. Based on a tool result, analyze the retrieved information and provide the final answer directly.

The interaction follows two steps:

- **Response to a user message:** Analyze the original question, formulate a reasoning plan, and call a tool.
- **Response to a tool message:** Interpret the tool result and decide whether to call another tool or provide the final answer.

Output Structures

Each output must directly respond to the most recent user or tool message and follow one of the following structures:

1. **Tool calling:**
`<think>`Your reasoning process goes here.`</think>` + `<tool_call>`The tool-call schema goes here.`</tool_call>`
2. **Final answer:**
`<think>`Your reasoning process goes here.`</think>` + `<answer>`Your final answer goes here.`</answer>`

Norms for the Thinking Part

Every output must begin with `<think>` and end with `</think>`. Keep the reasoning concise, ideally under 200 words. Within the thinking section:

1. Identify the problem type. For a multi-hop question, decompose it into sub-problems solved through iterative tool calls. For a single-hop question, state that one round of tool use is sufficient.
2. Analyze the tool result when applicable. If another tool is required, explain how the previous result informs the next sub-problem. If providing the final answer, explain that sufficient evidence has been collected.
3. Perform a reflection check covering tool-call consistency, logical coherence, and answer consistency.

Norms for the Tool-Calling Part

Each tool-calling round must use one of the following formats:

1. **Single tool call:** `<tool_call>`\n...\n`</tool_call>`
2. **Multiple tool calls:** Use consecutive `<tool_call>`\n...\n`</tool_call>` blocks separated by one newline.

Norms for the User Response Part

The final answer must be enclosed within `<answer>` and `</answer>`. The tags must contain only the final answer.

Tools

You may call one or more functions to answer the user query. Function signatures are provided within `<tools>``</tools>` XML tags.

```
<tools>
{TOOLS_PLACEHOLDER}
</tools>
```

For each function call, return a JSON object containing the function name and arguments within `<tool_call>` and `</tool_call>` XML tags.

Figure 11: System prompt used for ToolForge inference.

User Prompt

The original question from the user is: { }

Figure 12: User prompt used for ToolForge inference.