ToolScope: Enhancing LLM Agent Tool Use through Tool Merging and Context-Aware Filtering

Anonymous ACL submission

Abstract

Large language model (LLM) agents rely on external tools to solve complex tasks. However, real-world toolsets often contain semantically redundant tools with overlapping names and descriptions, introducing ambiguity and degrading tool selection performance. In addition, LLMs face strict input context limits, which prevent the agent from efficiently considering a large number of tools per query. To address these challenges, we propose ToolScope, a novel approach which contains: (1) ToolScopeMerger with Auto-Correction: automatically audits and fixes tool merges, reducing semantic redundancy in large toolsets. (2) ToolScopeRetriever, which ranks and selects only the top-k relevant tools for a given query, effectively compressing the toolset to fit within the LLM's input window without sacrificing selection accuracy. This selective filtering directly mitigates context length constraints by ensuring that only the most relevant tools are passed to the model. We evaluate ToolScope using 3 state-of-the-art LLMs across 3 open-source tool-use benchmarks covering both single-tool and multi-tool scenarios in diverse real-world domains. Experimental results show a substantial increase of 8.38% to 38.6% in tool selection accuracy, demonstrating ToolScope's effectiveness in enhancing LLM tool-use capabilities.

1 Introduction

Tool learning (Qu et al., 2025), which refers to LLMs becoming proficient in using external tools to solve complex problems (Qin et al., 2023), has emerged as a key research topic. It is divided into four stages: task planning, tool selection, tool calling, and response generation (Qu et al., 2025). Among these, tool selection remains a critical yet underdeveloped sub-task (Yang et al., 2023; Qin et al., 2023), as LLMs often fail to choose the most appropriate tools for a given query. Two major factors that contribute to this are: (1) overlapping tool descriptions, which introduce ambiguity that reduces both retrieval and selection accuracy (OpenAI, 2024; Lumer et al., 2024; Huang et al., 2023) and (2) the limited input context of LLMs, which constrain the model's ability to reason effectively over large toolsets (Huang et al., 2023). As toolsets continue to grow in size and complexity, these issues become more pronounced. Therefore, it is essential to address both tool overlap and context limitations for improved tool selection performance.

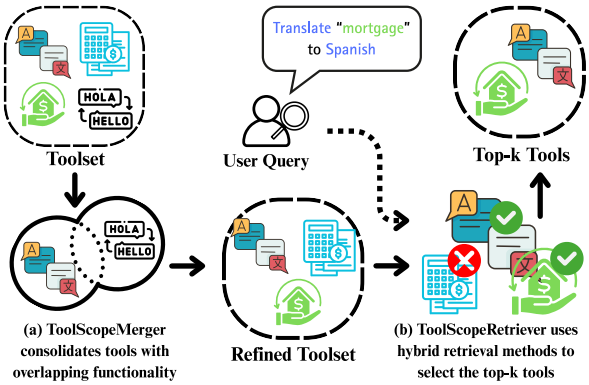

Figure 1: A simplified overview of using ToolScope for a tool selection task.

Previous research has explored several directions for improving tool selection. Tuning-free methods (Qu et al., 2025) have improved tool documentation with frameworks such as DRAFT (Qu et al., 2024) and EASYTOOL (Yuan et al., 2024). Retrieval-based methods have investigated "term-based" approaches such as BM25 (Robertson et al., 2009), "semantic-based" methods such as CRAFT (Yuan et al., 2023), and off-the-shelf embeddings (Qu et al., 2025) as the retrieval method to improve tool selection.

However, these approaches face several key limitations: (1) They only improve individual tool documentations, failing to address semantic overlap across the toolset. Methods that do address this overlap rely on manual tool merging. (2) Lim-

ited studies have evaluated the impact of hybrid retrieval methods on tool selection. (3) No prior work has combined automated tool merging with hybrid retrieval to simultaneously address tool overlap and context limitations.

We propose ToolScope, a two-part solution that automatically merges tools through ToolScope-Merger and retrieves the most relevant tools for a query using ToolScopeRetriever. This addresses the two main challenges presented: overlapping tool descriptions and context length limitations.

ToolScopeMerger automatically cleans raw toolsets in three stages. First, the system identifies candidate overlaps through semantic similarity. Then, it validates relationships with an LLM to ensure precise merges. Finally, it consolidates clusters with automated correction to produce a merged toolset that preserves tool capabilities while improving downstream retrieval by offering a smaller selection of tools.

ToolScopeRetriever addresses single and multi-tool queries through a hybrid retrieval strategy. This is followed by reranking system to ensure a fair, contextually relevant selection. By combining semantic and exact matches, it improves results for retrieval and tool selection accuracy.

In summary, our contributions are: (1) **ToolScopeMerger**, a graph-based framework with Auto-Correction that automatically merges semantically similar tools to reduce tool overlap. (2) **ToolScopeRetriever**, a hybrid retrieval system that combines sparse and dense scores to reduce context length. (3) A joint framework combining (1) and (2) significantly improves retrieval and selection accuracy across open-source benchmarks and models, boosting tool selection accuracy by 34.6% on Seal-Tools, 38.6% on UltraTool, and 8.8% on BFCL (Table 2).

2 Related Work

2.1 Tool Learning

Tool learning (Qu et al., 2025), which refers to LLM as being proficient in using tools to solve complex problems as humans (Qin et al., 2023), can be broken down into four stages of learning: task planning, tool selection, tool calling, and response generation (Qu et al., 2025).

Advancements in tool learning can be divided into two categories: tuning-based and tuning-free methods (Qu et al., 2025). Tuning-based approaches encompass supervised fine-tuning (Yang et al., 2023; Liu et al., 2024a; Shen, 2024; Acikgoz et al., 2025), contrastive learning (Wu et al., 2024b), reinforcement learning (Wang et al., 2025; Qian et al., 2025), and extending the modeling head with trainable tool token embeddings (Alazraki and Rei, 2025). However, these approaches are computationally expensive, requiring substantial model fine-tuning. Tuning-free approaches include chain-of-thought prompting (Inaba et al., 2023; Liu et al., 2024a), data augmentation (Liu et al., 2024a), and response-reasoning strategies (Liu et al., 2024a). Tuning-free approaches present advantages that can be explored, such as working well on both open- and closed-source models (Qu et al., 2025). While recent work has improved tool use by enhancing individual tool descriptions (Huang et al., 2023; Qu et al., 2024), it overlooks cross-tool semantic relationships, limiting its ability to resolve redundancy and ambiguity in large toolsets.

2.2 Tool Overlap

Several prior works have acknowledged the issue of tool overlap, which is defined as a query that can be solved by multiple tools (Huang et al., 2023), but fail to provide a concrete, autonomous solution. OpenAI's guide to LLM Agents highlights tool overlap as one of the possible causes of LLM agents consistently select incorrect tools and suggests user keep tools distinct (OpenAI, 2024). ToolShed (Lumer et al., 2024) identifies this issue of tool overlap in several notable tool selection benchmarks such as ToolBench (Qin et al., 2023) and ToolAlpaca (Tang et al., 2023). Prior work mitigates tool overlap via clustering (Huang et al., 2023) and manual curation (Huang et al., 2023); (Huang et al., 2024), yet no automated, scalable solution has been proposed.

2.3 Retrieval-Based Tool Selection

As tool libraries expand, selecting the appropriate tool increasingly resembles a retrieval task. Prior work has examined lexical methods like BM25 (Robertson et al., 2009) and semantic methods such as CRAFT (Yuan et al., 2023). Recent retrieval approaches, including RAG-Tool Fusion (Lumer et al., 2024), ScaleMCP (Lumer et al., 2025) incorporate Retrieval Augmented Generation (RAG) strategies through pre-retrieval, intra-retrieval, and post-retrieval phases. Those approaches leverage query reformulation, re-ranking, and retrieval-based planning to improve retrieval accuracy. However, limited studies have evaluated hybrid retrieval

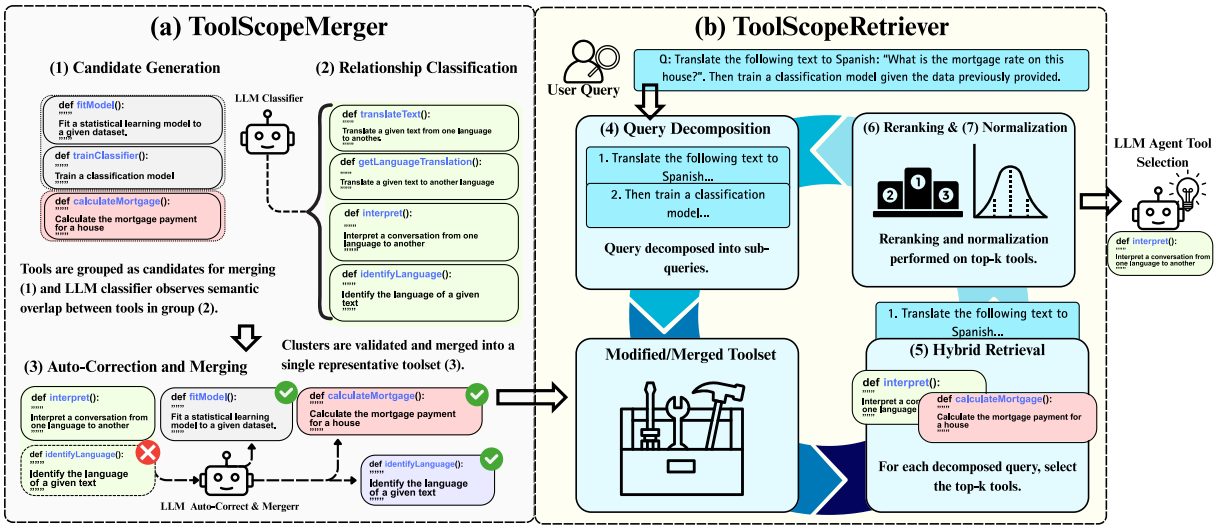

Figure 2: An overview of ToolScope which consists: (a) ToolScopeMerger, steps including: (1) Candidate Generation, (2) Relationship Classification, (3) Auto-Correction and Merging), (b) ToolScopeRetriever, steps including: (4) Query Decomposition, (5) Retrieval, (6) Reranking and (7) Normalization. Retrieved tools are then presented to LLM Agent for tool selection.

(Lumer et al., 2025). Finally, no study has combined automatic tool merging and hybrid tool retrieval to improve LLM agent tool selection.

3 Methods

3.1 Overview of ToolScope

ToolScope is a framework designed to optimize the performance of LLM agents by addressing the two key challenges identified in tool selection: tool overlap and context length limitation. Figure 2 presents an overview of ToolScope.

3.2 ToolScopeMerger

ToolScopeMerger is a graph-based framework that consolidates overlapping tools in large agent toolsets. An optional Auto-Correction stage uses an LLM validator to audit clusters, confirm equivalence, and split or remove members when errors in merging are detected.

We organize ToolScopeMerger into three stages: S1: Candidate Generation (P1: Tool Indexing), S2: Relationship Classification and Graph Formation (P2: Relationship Classification, P3: Graph Construction), and S3: Consolidation and Auto-Correction (P4: Tool Pruning, P5: Auto-Correction, P6: Toolset and Dataset Update).

Tool Indexing. Let the toolset be defined as $T = \{t_1, t_2, \ldots, t_n\}$. For each tool t_i, its signature and natural language description d_i are encoded using an embedding model f to produce a dense vector representation: $v_i = f(d_i), \forall i \in \{1, \ldots, n\}$. Let $V = [v_1, \ldots, v_n] \in \mathbb{R}^{n \times d}$ denote the matrix of all embedded tools, where d is the embedding dimension. For each t_i, we retrieve its top-k most similar tools based on cosine similarity, forming the candidate set: $T_i^{(k)} = \{t_{i_1}, \ldots, t_{i_k}\}$, where $t_{i_j} \in T \setminus \{t_i\}$. This set $T_i^{(k)}$ contains the candidates for potential merging with t_i.

Tool Relationship Classification. We define an LLM-based binary classifier $M_C : T \times T \to \{0, 1\}$ to detect semantic overlap between tools. Given a tool pair, the model determines whether the two serve sufficiently similar functions to justify merging. The input format and prompting strategy for this classification are described in Appendix H.1. For each pair (t_i, t_j) where $t_j \in T_i^{(k)}$, the classifier outputs a binary label indicating whether the tools are semantically equivalent, as defined in Equation 1.

$$M_C(t_a, t_b) = \begin{cases} 1 & \text{if } t_a \sim t_b \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where $\sim$ indicates semantic equivalence.

Graph Construction. Inspired from the Tool Graph which utilizes graph to represent relationships and dependencies between tools (Shen et al., 2024; Liu et al., 2024b), we define an undirected pruning graph $G = (T, E)$, where each node represents a tool $t_i \in T$, and an edge $(t_a, t_b) \in E$ exists if $M_C(t_a, t_b) = 1$. We then extract the set of connected components in G, $C = \{C_1, C_2, ..., C_a\}$, where each component $C_b \subseteq T$ contains tools that

are considered semantically equivalent and therefore overlapping.

Tool Pruning. For each connected component C_b, one representative tool $t_b^* \in C_b$ is selected to serve as the canonical tool for the component. Currently, our implementation selects the tool that minimizes function name string length. We define the pruned toolset as: $T' = \{t_1^*, t_2^*, \ldots, t_k^*\}$, where $k \leq n$. We also maintain a mapping: $\phi : T \to T'$, where $\phi(t) = t_j^*, \forall t \in C_b$.

Auto-Correction. To ensure high-precision merges, we validate and correct tool pruning automatically using an LLM validator V that can be easily turned on and off. For each proposed cluster merge, $V(C_b)$ determines whether all functions satisfy semantic equivalence criteria (see Appendix H.2). If the cluster is valid, the merge proceeds. If the cluster is invalid, $V(C_b)$ proposes a refined splitting of the original cluster C_b into more granular sub-clusters and removes members that are not semantically equivalent.

$$V(C_b) = \{C_1', \ldots, C_m'\}, \quad \text{where each } C_j' \subseteq C_b \tag{2}$$

This iterative correction removes the need for manual review. Final integrity checks ensure no tool appears in multiple prune dictionaries and none is simultaneously kept and pruned. This guarantees a consistent one-to-one mapping $\phi : T \to T'$ for merging and dataset relabeling.

Toolset and Dataset Update. For each cluster C_b, an LLM M_D is prompted to synthesize a new tool signature and description d^* such that $M_D(C_b) = d^*$. The new signature and tool description d^* unifies the functionality of the tools in C_b. See Appendix H.3 for prompt details. This forms a new merged tool based around the representative tool $t_j^{*'} = d^*$. Given our mapping ϕ, we update our original benchmark dataset β by relabeling our gold responses, as shown by the Equation 3:

$$\forall (q, l) \in \beta,\ t \Longrightarrow \phi(t) \tag{3}$$

where (q, l) is a query-response pair, and l can be multiple tools. This final step ensures the new Toolset T' is still compatible with the evaluation benchmark and leads to fair and accurate testing.

3.3 ToolScopeRetriever

Effective tool selection is essential for enhancing the capabilities of large language models (LLMs) when solving complex tasks involving external tools. We adopt a hybrid approach to tool selection that integrates both local and global reranking strategies, supporting both single-tool and multi-tool use cases.

Dataset	# Queries	# Tools	Multi-Tool?
BFCL	400	400	No
Seal-Tools	654	4076	**Yes**
UltraTool	4814	1885	**Yes**

Table 1: Benchmark datasets used in our evaluation, including scale and presence of multi-tool scenarios.

Single-tool selection. Given a query q and a set of candidate tools T, we compute a hybrid retrieval score for each tool $t \in T$ by combining sparse and dense similarity scores through a weighted average, denoted as:

$$s(q, t) = \alpha \cdot s_{\text{dense}}(q, t) + (1 - \alpha) \cdot s_{\text{sparse}}(q, t) \tag{4}$$

The top M candidates based on $s(q, t)$ are then reranked using a cross-encoder that scores each pair (q, t). The tool with the highest reranker score is selected as the final output.

Multi-tool selection. In the multi-tool setting, each query q is associated with multiple subqueries. For each subquery, we apply the same hybrid retrieval and reranking procedure. We then select the top-1 tool $t^{(1)}$ with the highest reranker score from each subquery. The remaining tools $t^{(j)}$ for $j = 2, \ldots, M$ are normalized using min-max normalization:

$$s_{\text{norm}}(t^{(j)}) = \frac{s(t^{(j)}) - s_{\min}}{s_{\max} - s_{\min} + \varepsilon} \tag{5}$$

This normalization rescales scores across subqueries, since raw reranker scores may not be directly comparable due to varying query-to-tool similarity distributions. It enables a fair global ranking of tools from different subqueries. The top-k toolset is assembled by first including all top-1 selections, then iteratively adding tools with the highest normalized scores until k tools are selected.

4 Experiments

4.1 Experiment Setup

Datasets. Prior work has introduced datasets such as ToolACE (Liu et al., 2024a), ToolE (Huang et al., 2023), Berkeley Function Calling Leaderboard (BFCL) (Patil et al., 2024), NexusRaven (Nexusflow, 2023), ToolBench (Qin et al., 2023),

LLM	Method	Seal-Tools			BFCL			UltraTool
		CSR@5	CSR@10	CSR@15	CSR@5	CSR@10	CSR@15	CSR@30
GPT-4o	BM25	0.544	0.593	0.646	0.850	0.875	0.870	0.366
	Dense	0.548	0.603	0.631	0.888	0.900	0.915	0.673
	ToolScope	0.875	0.896	0.905	0.912	0.915	0.915	0.723
	ToolScope+AutoCorrect	**0.890**	**0.921**	**0.924**	**0.938**	**0.935**	**0.940**	**0.752**
LLaMA-3.3-70B	BM25	0.527	0.574	0.618	0.832	0.848	0.865	0.364
	Dense	0.531	0.584	0.611	0.840	0.845	0.862	0.664
	ToolScope	0.883	0.900	0.910	0.888	0.892	0.882	0.729
	ToolScope+AutoCorrect	**0.889**	**0.916**	**0.922**	**0.935**	**0.942**	**0.942**	**0.759**
Command-R-08-2024	BM25	0.463	0.512	0.551	0.785	0.805	0.807	0.368
	Dense	0.523	0.569	0.603	0.802	0.825	0.822	0.669
	ToolScope	0.873	0.888	0.889	0.902	**0.912**	**0.920**	0.702
	ToolScope+AutoCorrect	**0.878**	**0.900**	**0.902**	**0.908**	0.900	0.900	**0.718**

Table 2: Correct Selection Rate (CSR@k) for each method and LLM. BFCL is evaluated at $k \in \{5, 10, 15\}$, Seal-Tools at $k \in \{5, 10, 15\}$, and UltraTool at $k = 30$. k refers to the top k tools retrieved and passed to the LLM agent for tool selection. Full results are shown in Table 4 in Appendix C.

RestBench (Song et al., 2023), Seal-Tools (Wu et al., 2024a), and UltraTool (Huang et al., 2024). Despite covering a range of tool-use tasks, existing benchmarks have key limitations: (1) tool descriptions lack high-quality tool documentation; (2) missing parameters or type hints in ground truth; (3) insufficient count of tools in toolset to correctly evaluate the retrieval system. For our study, we selected Seal-Tools, BFCL, and UltraTool as primary datasets. Seal-Tools (Wu et al., 2024a) includes out-of-domain test examples that call single and multiple tools over a very large toolset. BFCL (Patil et al., 2024) has a simple single-turn tool calling dataset. It contains a one-to-one query to tool mapping that allows us to test our tool merging strategy efficiently. UltraTool (Huang et al., 2024) is specifically designed to improve an LLM's tool utilization within real-world scenarios. Details on the benchmark can be found in Table 1.

Evaluation Metrics. We follow prior work (Lumer et al., 2024; Wu et al., 2024a) and use standard tool selection and retrieval-based metrics. Correct Selection Rate (CSR), adapted from MetaTool (Huang et al., 2023), measures the percentage of queries for which the predicted toolset exactly matches the ground-truth set. CSR@k specifies the Correct Selection Rate using the top-k tools retrieved from ToolScopeRetriever, as shown in Equation 6:

$$\text{CSR@}k = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \mathbb{I}[\hat{T}_q^{(k)} = T_q^*] \tag{6}$$

Here, $\mathcal{Q}$ is the set of all evaluation queries, T_q^* is the ground-truth toolset for query q, $\hat{T}_q^{(k)}$ is the predicted top-k toolset, and $\mathbb{I}[\cdot]$ is the indicator function.

Recall@k quantifies the proportion of ground-truth tools correctly retrieved among the top-k predictions as shown in Equation 7.

$$\text{Recall@}k = \frac{1}{|\mathcal{Q}|} \sum_{q=1}^{|\mathcal{Q}|} \frac{|T_q^k \cap T_q^*|}{|T_q^*|} \tag{7}$$

Baselines. We evaluate two retrieval baselines in all main CSR experiments: BM25 (Robertson et al., 2009), which ranks query–tool pairs by TF-IDF term overlap. Dense Embeddings (Qu et al., 2025), which encodes queries and tool descriptions into a shared vector space and ranks them by cosine similarity. Both engines are evaluated on the original, unmerged tool sets, allowing any performance gap to be attributed to ToolScope. In multi-tool setting (e.g: UltraTool), we perform query decomposition and score normalization to select the top k tools.

For completeness, we also include the additional baselines DPR (Wu et al., 2024a) and ToolShed (Lumer et al., 2024) alongside BM25 and Dense Embeddings; their Seal-Tools retrieval scores appear in Table 5 in Appendix D.

Implementation.
ToolScopeMerger: Each tool is embedded using `thenlper/gte-large` and indexed in FAISS. Tool pairs with cosine similarity above 0.82 are selected as merge candidates, a threshold determined via sensitivity analysis on Seal-Tools (Fig. 4; Sec. 4.4). GPT-4o performs relationship classification, Auto-Correction, and toolset updating. **ToolScopeRetriever:** We evaluate on the full toolset to reflect

production settings, where the agent does not rely on pre-filtered tools (unlike UltraTool (Huang et al., 2024)). Following results in Sec. 4.4, we use a dense-only retriever (α=1) and rerank the top-50 candidates with a cross-encoder using min–max normalization. **Main Agent:** Tool selection are generated using 3 state-of-the-art models: GPT-4o (OpenAI, 2024), Cohere-Command-R-08-2024 (Cohere, 2024), and LLaMA-3.3-70B (Meta Platforms, Inc., 2024). **Result Analysis:** GPT-4o was used as LLM-as-a-Judge (Zheng et al., 2023) for grading documentation quality in Section 4.4.

Full implementation details and hyperparameters: Full pseudo-code, prompts, hyperparameters, and hardware configurations are provided in Appendices G, H and I.

4.2 Experimental Results

We present the main results for LLM agent tool selection in Table 2, with full results provided in Appendix C. Based on the results, we have the following observations:

ToolScope yields state-of-the-art CSR@k across all 3 benchmarks and 3 LLM models. We observe in Table 2 that ToolScope, with or without Auto-Correction, achieves the highest CSR@k on Seal-Tools, BFCL, and UltraTool across all k values and LLMs including GPT-4o, Llama-3.3-70B, and Command-R. Gains are large on challenging datasets such as Seal-Tools (+34.6%) and UltraTool (+38.6%), consistent on BFCL (+8.8%). These results highlight ToolScope's strong generalization capability across different LLMs. Among the models evaluated, GPT-4o exhibits the highest CSR, indicating that ToolScope can benefit from continuous advancements in foundation models.

Robustness across k. Gains are robust at all evaluated k. On Seal-Tools, the performance gap widens as k increases, consistent with the hypothesis that the original toolset contain higher rates of overlap; the merged toolset avoids confusing the retriever with overlapped tools, preserving precision at larger k. On BFCL, improvements remain stable across k, and on UltraTool, ToolScope continues to outperform the baselines, showing scalability to larger number of retrieved tools sent to LLM agent.

Auto-Correction enhances CSR across datasets. Auto-Correction is a lightweight LLM-based module from ToolScopeMerger that audits and refines merge decisions after graph-based consolidation. Across models, it consistently improves CSR. For example, it yields gains of 1.5–2.9% on Seal-Tools and 2.9–7.9% on UltraTool, where redundant or ambiguous tools are more common. Improvements on BFCL are modest for GPT-4o (+2.6%) and LlaMa-3.3-70b (+5.0%), but we observe a slight regression with command-r (–1.2%), likely due to over-correction when the baseline coverage is already strong. These results suggest that Auto-Correction is most beneficial in complex or fast-evolving toolsets and can be deployed with guardrails (e.g., confidence thresholds or fallback merges) to prevent rare regressions. Appendix J illustrates this process with two case studies: one highlighting a successful merge and the other a rejected merge. These show the module's ability to correct errors.

4.3 Ablation Study

Table 3 evaluates the contributions of the three core components of ToolScope: the Reranker from ToolScopeRetriever, the Merger from ToolScope-Merger, and the Auto-Correction LLM module from ToolScopeMerger. Each dataset is assessed under different combinations of these modules. We also retain the base retrieval setup, including query decomposition, dense embedding, and score normalization. This serves as a stable retrieval foundation for reranking and merging decisions.

ToolScopeMerger provides the strongest gains. With Reranker off, enabling Merger boosts CSR by 22.0% on Seal-Tools, 5.0% on BFCL, and 7.0% on UltraTool. The larger improvements on Seal-Tools and UltraTool stem from their higher semantic redundancy, where merging prevents confusing overlaps in the retrieved pool. Beyond accuracy, ToolScopeMerger also reduces raw toolset size: -14.0% on BFCL, -2.1% on Seal-Tools, and -25.3% on UltraTool (Table 6, Appendix D), providing a cleaner foundation for retrieval.

Auto-Correction further improves performance by refining merge errors, as seen in Table 2. When added on top of Reranker + Merger, it raises CSR by 2.9% on Seal-Tools, 5.0% on BFCL, and 7.9% on UltraTool (Table 2). These gains indicate that even after graph-based consolidation, some false merges remain and can be efficiently corrected by a lightweight LLM audit.

Reranker helps most on large, noisy toolsets. Its gains are $+1.3\%$ on Seal-Tools and $+0.9\%$ on UltraTool, but negligible (0.0%) on BFCL. This reflects dataset differences: BFCL is a single-tool benchmark where retrieval already isolates the correct tool, while Seal-Tools and UltraTool contain

Dataset	Reranker	Merger	AutoCorrect	CSR
Seal-Tools	✓	✓	✓	0.931
	✓	✗	✗	0.912
	✗	✓	✓	0.918
	✗	✗	✗	0.694
BFCL	✓	✓	✓	0.938
	✓	✗	✗	0.888
	✗	✓	✓	0.938
	✗	✗	✗	0.888
UltraTool	✓	✓	✓	0.752
	✓	✗	✗	0.622
	✗	✓	✓	0.743
	✗	✗	✗	0.673

Table 3: Configuration usage (✓/✗) for each dataset and corresponding CSR score. BFCL is evaluated at $k = 5$, Seal-Tools is evaluated at $k = 30$, and UltraTool is evaluated at $k = 30$. All results are generated using GPT-4o.

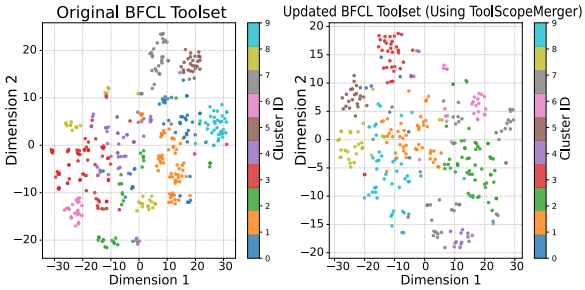

Figure 3: T-SNE visualization of original BFCL tool embedding and merged BFCL tool embedding

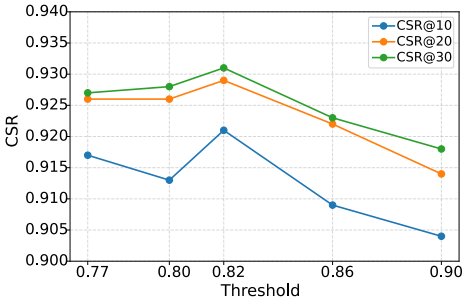

Figure 4: Sensitivity analysis of ToolScopeMerger on Seal-Tools using different cosine similarity filtering thresholds.

larger, noisier multi-tool corpora where reranking helps disambiguate overlapping candidates. We therefore keep Reranker in the pipeline, as it consistently adds value on more challenging datasets.

In summary, ToolScopeMerger delivers the dominant accuracy gain, Auto-Correction provides consistent refinements, and Reranker offers selective improvements. These components help ToolScope achieve strong and consistent performance across all three benchmarks.

4.4 Discussion

The findings of this study provide important insights into the following points.

ToolScopeMerger effectively reduces functional overlap. ToolScopeMerger removes semantically similar tools, improving the distribution in embedding space. We also evaluated the impact using the silhouette coefficient (Rousseeuw, 1987), where lower values indicate less semantic overlap. In addition, we used t-SNE plots (Van der Maaten and Hinton, 2008) to visualize the tool semantic distribution. As shown in Figure 3 (details in Appendix E.2), Seal-Tools, UltraTool and BFCL all exhibit sparser distributions. As seen in Appendix E.1, all datasets also have lower silhouette scores after merging with varying cluster sizes, confirming that ToolScopeMerger reduces semantic redundancy.

ToolScopeMerger is robust to threshold tuning. As shown in Figure 4, we conducted a sensitivity study on the cosine similarity threshold in ToolScopeMerger, using the most heterogeneous corpus, Seal-Tools, and sweeping from 0.77 up to 0.90. CSR remains stable in the 0.77 to 0.82 band, with the three lowest thresholds differing by less than 0.5%. CSR reaches its maximum at 0.82. For thresholds above 0.86, CSR declines, indicating that overly conservative merging can discard useful semantically similar tool pairs. We therefore fix the cosine similarity threshold at 0.82 for all experiments. The consistently high CSR values across thresholds from 0.77 to 0.90 indicate that ToolScopeMerger is robust to a broad range of reasonable settings.

ToolScopeMerger ensures merge validation without the need for manual intervention. We eliminate manual validation by introducing Auto-Correction, an LLM-based module that audits and refines merge decisions (Section 3.2). It corrects merge errors and improves CSR by 1.5–2.9% on Seal-Tools and 2.9–7.9% on UltraTool, where tool redundancy is high. Gains on BFCL are moderate: 2.6% (GPT-4o) and 5.0% (LlaMa-3.3-70b). Beyond accuracy, Auto-Correction reduces the need of manual inspection and enables developers to review and improve tool definitions as needed.

Robustness Analysis of Documentation Quality. We evaluate whether ToolScope depends on high-quality documentation by scoring each tool's name, signature, and description on a 1–5 scale

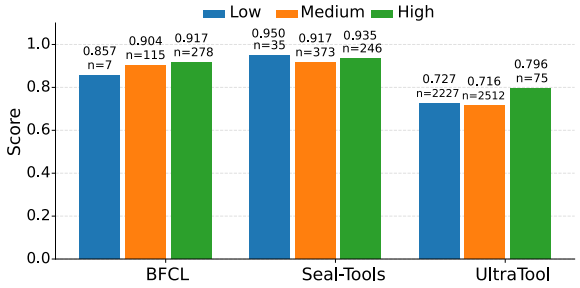

Figure 5: CSR scores for BFCL, Seal-Tools, and UltraTool by tool quality (Low/Medium/High). n denotes the number of evaluation queries for each quality level used to compute the CSR.

using GPT-4o as an LLM judge (Appendix H.5). Tools are grouped into Low (1–2), Medium (3), and High (4–5) quality bands, and CSR is analyzed within each band. As shown in Figure 5, ToolScope maintains strong CSR even with low-quality documentation: 85.7% (BFCL), 95.0% (Seal-Tools), and 72.7% (UltraTool). Although Seal-Tools and UltraTool contain mostly medium or low-quality tools (79% and 96%, respectively), ToolScope remains robust to imperfect descriptions. Human validation of GPT-4o's scoring confirms strong alignment with human judgment (Appendix F.2), supporting the reliability of this evaluation. These results show that ToolScope remains effective even with brief or imperfect documentation, as long as each tool includes a name, signature, and short description, conditions typical of real-world toolsets.

ToolScope improves retrieval performance. ToolScopeMerger enhances retrieval accuracy by reducing semantic overlap in the toolset (Table 5 in Appendix D). Recall@10 increases significantly, from 0.550 to 0.935 on Seal-Tools and from 0.945 to 0.985 on BFCL. ToolScopeRetriever uses a hybrid strategy combining sparse (BM25) and dense embeddings, with a weighting parameter α controlling their balance. Our α tuning experiment (Figure 7, Appendix D) shows that retrieval@k peaks at $\alpha = 1$, where dense-only retrieval performs best. This configuration outperforms baselines across all k on BFCL and at $k = 5$ on Seal-Tools.

Reranking improves CSR at small retrieval k. As shown in Figure 6 from Appendix D, enabling the reranker significantly improves CSR at smaller retrieval sizes ($k = 5$ or 10). As k increases, the performance gap between reranked and non-reranked configurations narrows, suggesting diminishing returns. Overall, $k = 5$ or $k = 10$ with reranking provides a good balance between performance and retrieval efficiency.

ToolScopeRetriever reduces context length. ToolScopeRetriever drastically shrinks the number of tokens passed to the agent by ranking and selecting only the top-k relevant tools per query. As shown in Table 8 in Appendix K, average total context length is reduced from 32,563 to 469 tokens on BFCL (98.6% reduction), from 292,107 to 317 on Seal-Tools (99.9%), and from 136,352 to 2,076 on UltraTool (98.5%). These reductions make the approach efficient and scalable even for very large toolsets and strict prompt budgets.

Tool overlap remains a limitation of open-source benchmarks. While Toolshed (Lumer et al., 2024) notes some overlap in Seal-Tools (Wu et al., 2024a), we identified many semantically redundant tools that compromise retrieval and selection accuracy without a solution like ToolScope-Merger. Similar issues appear in BFCL (Patil et al., 2024) and UltraTool (Huang et al., 2024). Such overlap introduces ambiguity that degrades retrieval metrics and causes selection errors. It also limits the reliability of benchmarking research in tool learning. Benchmark creators should enforce clearer distinctions or merge overlapping tools to enable more consistent evaluation. ToolScope offers a practical remedy for both production and research scenarios involving noisy toolsets.

5 Conclusion

In this paper, we present **ToolScope**, a two-part framework aimed at addressing key challenges in LLM agent tool selection: tool overlap and limitations of long context length. **ToolScopeMerger** consolidates overlapping tools using an automated framework; its Auto-Correction module automatically audits and fixes merges, reducing semantic redundancy in large toolsets. **ToolScopeRetriever** improves retrieval accuracy through query decomposition, hybrid retrieval, reranker and robust score normalization for tool retrieval. Together, our results demonstrate that ToolScope significantly improves tool selection accuracy across standard benchmarks, achieving gains of 34.6% on Seal-Tools, 38.6% on UltraTool, and 8.8% on BFCL over strong baselines. We also identify a persistent tool overlap issue in currently available open-source public benchmarks, which can be improved using ToolScope. Overall, ToolScope provides a scalable solution for improving LLM-agent tool selection in real-world settings.

Limitations

Our retrieval currently relies on tool names, signatures, and descriptions. Enriching tools with metadata such as typical use cases, domains, and input/output schemas could further strengthen ranking, particularly in very large toolsets. Complementary strategies like multi-index retrieval may also improve scalability. Finally, evaluating with reasoning-oriented models could influence the effectiveness of Auto-Correction in ToolScope-Merger and the quality of tool selection.

Ethical Considerations

While ToolScope improves LLM agent performance in tool selection tasks, it is not yet suitable for deployment in environments where errors in tool selection could result in significant harm or consequences such as those in medical, legal, or financial fields. ToolScope relies on LLM generated tool merging and retrieval mechanisms which are inherently probabilistic. As with many LLM solutions, this is subject to hallucination, bias, and misclassifications of overlapped tools.

References

Emre Can Acikgoz, Jeremiah Greer, Akul Datta, Ze Yang, William Zeng, Oussama Elachqar, Emmanouil Koukoumidis, Dilek Hakkani-Tür, and Gokhan Tur. 2025. Can a single model master both multi-turn conversations and tool use? calm: A unified conversational agentic language model. *arXiv preprint arXiv:2502.08820*.

Lisa Alazraki and Marek Rei. 2025. Meta-reasoning improves tool use in large language models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, page 7885–7897. Association for Computational Linguistics.

Cohere. 2024. Command models get an august refresh. Accessed: 2025-05-18.

Shijue Huang, Wanjun Zhong, Jianqiao Lu, Qi Zhu, Jiahui Gao, Weiwen Liu, Yutai Hou, Xingshan Zeng, Yasheng Wang, Lifeng Shang, Xin Jiang, Ruifeng Xu, and Qun Liu. 2024. Planning, creation, usage: Benchmarking llms for comprehensive tool utilization in real-world complex scenarios. *Preprint*, arXiv:2401.17167.

Yue Huang, Jiawen Shi, Yuan Li, Chenrui Fan, Siyuan Wu, Qihui Zhang, Yixin Liu, Pan Zhou, Yao Wan, Neil Zhenqiang Gong, and 1 others. 2023. Metatool benchmark for large language models: Deciding whether to use tools and which to use. *arXiv preprint arXiv:2310.03128*.

Tatsuro Inaba, Hirokazu Kiyomaru, Fei Cheng, and Sadao Kurohashi. 2023. Multitool-cot: Gpt-3 can use multiple external tools with chain of thought prompting. *arXiv preprint arXiv:2305.16896*.

Weiwen Liu, Xu Huang, Xingshan Zeng, Xinlong Hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, and 1 others. 2024a. Toolace: Winning the points of llm function calling. *arXiv preprint arXiv:2409.00920*.

Xukun Liu, Zhiyuan Peng, Xiaoyuan Yi, Xing Xie, Lirong Xiang, Yuchen Liu, and Dongkuan Xu. 2024b. Toolnet: Connecting large language models with massive tools via tool graph. *arXiv preprint arXiv:2403.00839*.

Elias Lumer, Anmol Gulati, Vamse Kumar Subbiah, Pradeep Honaganahalli Basavaraju, and James A Burke. 2025. Scalemcp: Dynamic and auto-synchronizing model context protocol tools for llm agents. *arXiv preprint arXiv:2505.06416*.

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

Meta Platforms, Inc. 2024. Model cards and prompt formats – llama 3.3. Accessed: 2025-05-18.

Nexusflow. 2023. Nexusraven-v2: Surpassing gpt-4 for zero-shot function calling. https://nexusflow.ai/blogs/ravenv2. Accessed: 2025-05-04.

OpenAI. 2024. Hello gpt-4o. Accessed: 2025-05-18.

OpenAI. 2024. A practical guide to building agents. https://cdn.openai.com/business-guides-and-resources/a-practical-guide-to-building-agents.pdf. Accessed: 2025-05-16.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. 2024. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems*, 37:126544–126565.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. 2025. Toolrl: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, and 1 others. 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. 2024. From exploration to mastery: enabling llms to master tools via self-driven interactions. *arXiv preprint arXiv:2410.08197*.

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.

Stephen Robertson, Hugo Zaragoza, and 1 others. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Peter J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65.

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.

Zhuocheng Shen. 2024. Llm with tools: A survey. *arXiv preprint arXiv:2409.18807*.

Yifan Song, Weimin Xiong, Dawei Zhu, Wenhao Wu, Han Qian, Mingbo Song, Hailiang Huang, Cheng Li, Ke Wang, Rong Yao, and 1 others. 2023. Restgpt: Connecting large language models with real-world restful apis. *arXiv preprint arXiv:2306.06624*.

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*.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Hongru Wang, Cheng Qian, Wanjun Zhong, Xiusi Chen, Jiahao Qiu, Shijue Huang, Bowen Jin, Mengdi Wang, Kam-Fai Wong, and Heng Ji. 2025. Otc: Optimal tool calls via reinforcement learning. *arXiv preprint arXiv:2504.14870*.

Mengsong Wu, Tong Zhu, Han Han, Chuanyuan Tan, Xiang Zhang, and Wenliang Chen. 2024a. Seal-tools: Self-instruct tool learning dataset for agent tuning and detailed benchmark. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 372–384. Springer.

Shirley Wu, Shiyu Zhao, Qian Huang, Kexin Huang, Michihiro Yasunaga, Kaidi Cao, Vassilis Ioannidis, Karthik Subbian, Jure Leskovec, and James Y Zou. 2024b. Avatar: Optimizing llm agents for tool usage via contrastive reasoning. *Advances in Neural Information Processing Systems*, 37:25981–26010.

Rui Yang, Lin Song, Yanwei Li, Sijie Zhao, Yixiao Ge, Xiu Li, and Ying Shan. 2023. Gpt4tools: Teaching large language model to use tools via self-instruction. *Advances in Neural Information Processing Systems*, 36:71995–72007.

Lifan Yuan, Yangyi Chen, Xingyao Wang, Yi R Fung, Hao Peng, and Heng Ji. 2023. Craft: Customizing llms by creating and retrieving from specialized toolsets. *arXiv preprint arXiv:2309.17428*.

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*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.

A Code Release

We are in the legal process of open-sourcing our code. The code will be release upon legal approval.

B Future Work

We expect additional improvements in our future work. First, we aim to adopt a more extensible retrieval method such as a multi-index framework that which enhance scalability and relevance in large toolsets. Second, we plan to enrich the tool metadata with descriptive metadata (e.g: common use case descriptions, domain and user intents) and structural metadata (e.g: input and output schemas) that could provide stronger retrieval context and improve the retrieval accuracy. Finally, we want to expand the scope of the evaluation to analyze ToolScope's impact on tool calling and response generation. This would help to establish a clear picture of holistic improvements to LLM agent tool learning.

C Full CSR Scores for Baseline and ToolScope

We report the full Correct Selection Rate (CSR@k) results for tool selection across BFCL, Seal-Tools, and UltraTool in Table 4 on different k (where k refers to the top k tools retrieved for LLM Agent tool selection). With or without Auto-Correction, ToolScope consistently outperforms BM25 and Dense baselines, resulting in large improvements on Seal-Tools (+34.6%) and UltraTool (+38.6%), and stable improvements on BFCL (+8.8%). These results demonstrate its strong cross model generalization capability, with GPT-4o as the generation model achieving the highest CSR.

Dataset	Model	Config	CSR@5	CSR@10	CSR@15	CSR@20	CSR@25	CSR@30
Seal-Tools	GPT-4o	BM25	0.544	0.593	0.646	0.666	0.702	0.727
		Dense	0.548	0.603	0.631	0.657	0.676	0.694
		ToolScope	0.875	0.896	0.905	0.908	0.902	0.908
		ToolScope+AC	**0.890**	**0.921**	**0.924**	**0.929**	**0.931**	**0.931**
	LLaMA-3.3-70B	BM25	0.527	0.574	0.618	0.653	0.692	0.709
		Dense	0.531	0.584	0.611	0.637	0.657	0.663
		ToolScope	0.883	0.900	0.910	0.910	0.910	0.910
		ToolScope+AC	**0.889**	**0.916**	**0.922**	**0.924**	**0.926**	**0.928**
	Command-R-08-2024	BM25	0.463	0.512	0.551	0.573	0.593	0.608
		Dense	0.523	0.569	0.603	0.621	0.647	0.666
		ToolScope	0.873	0.888	0.889	0.891	0.890	0.887
		ToolScope+AC	**0.878**	**0.900**	**0.902**	**0.904**	**0.908**	**0.905**

Dataset	Model	Config	CSR@1	CSR@5	CSR@10	CSR@15
BFCL	GPT-4o	BM25	0.695	0.850	0.875	0.870
		Dense	0.780	0.888	0.900	0.915
		ToolScope	**0.878**	0.912	0.915	0.915
		ToolScope+AC	0.860	**0.938**	**0.935**	**0.940**
	LLaMA-3.3-70B	BM25	0.692	0.832	0.848	0.865
		Dense	0.782	0.840	0.845	0.862
		ToolScope	**0.872**	0.888	0.892	0.882
		ToolScope+AC	0.858	**0.935**	**0.942**	**0.942**
	Command-R-08-2024	BM25	0.695	0.785	0.805	0.807
		Dense	0.785	0.802	0.825	0.822
		ToolScope	**0.872**	0.902	**0.912**	**0.920**
		ToolScope+AC	0.858	**0.908**	0.900	0.900

Dataset	Model	Config	CSR@20	CSR@30
UltraTool	GPT-4o	BM25	0.333	0.366
		Dense	0.647	0.673
		ToolScope	0.691	0.723
		ToolScope+AC	**0.702**	**0.752**
	LLaMA-3.3-70B	BM25	0.330	0.364
		Dense	0.649	0.664
		ToolScope	0.692	0.729
		ToolScope+AC	**0.710**	**0.759**
	Command-R-08-2024	BM25	0.338	0.368
		Dense	0.648	0.669
		ToolScope	0.676	0.702
		ToolScope+AC	**0.688**	**0.718**

Table 4: Correct Selection Rate (CSR) of top k tools across datasets (BFCL, Seal-Tools, UltraTool) and models. k refers to the top k tools retrieved for LLM Agent tool selection. Configurations compared: BM25, Dense, ToolScope, and ToolScope + AC (ToolScope with Auto-Correction). Seal-Tools is evaluated at $k \in \{5, 10, 15, 20, 25, 30\}$, BFCL at $k \in \{1, 5, 10, 15\}$, and UltraTool at $k \in \{20, 30\}$.

D Retrieval Performance for ToolScope

We evaluate tool retrieval performance across both the Seal-Tools and BFCL benchmarks using Recall@k, which measures the proportion of ground-truth tools appearing in the top-k retrieved candidates. Table 5 summarizes the results for several representative configurations. We observe that our solution achieves the highest recall on both benchmarks.

Benchmark / Configuration	**Recall@k**		
	$k = 1$	$k = 5$	$k = 10$
Seal-Tools / BM25	-	0.490	0.540
Seal-Tools / Dense	-	0.589	0.649
Seal-Tools / DPR	-	0.480	0.680
Seal-Tools / ToolShed	-	0.876	**0.965**
Seal-Tools / ToolScope	-	**0.884**	0.935
BFCL / BM25	0.693	0.913	0.945
BFCL / Dense	0.818	0.973	0.985
BFCL / ToolScope	**0.880**	**0.973**	**0.985**

Table 5: Recall@k scores for selected retrieval configurations on Seal-Tools and BFCL benchmarks. ToolScope achieves the highest recall across both benchmarks for $k = 5$ for Seal-Tools and all $k \in \{1, 5, 10\}$ for BFCL, and competitive recall score for $k = 1$ for Seal-Tools.

Figure 6 illustrates the Correct Selection Rate (CSR@k) with and without the Reranker. Across all datasets, ToolScope with Reranker consistently outperforms ToolScope without Reranker. As k increases, the scores gradually converge, indicating that the benefit of reranking diminishes when enough tools are retrieved. This highlights the Reranker's effectiveness in improving tool selection, particularly when fewer candidate tools are considered.

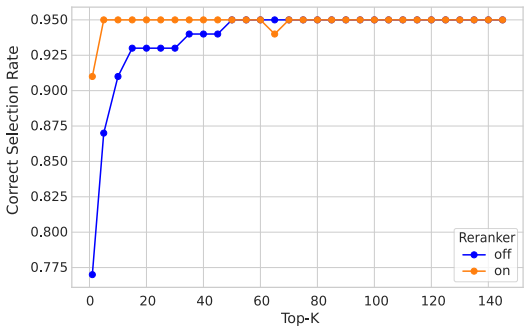

Figure 6: The CSR results (%) of top k tools with/without Reranker

We analyze the effect of the hybrid retrieval weight α on retrieval performance in Seal-Tools, as shown in Figure 7. α controls the weightage given to BM25 and Dense scores. Performance steadily increases with higher values of α, peaking at $\alpha = 1$, which shows that prioritizing the dense retriever results in the best recall@k scores overall.

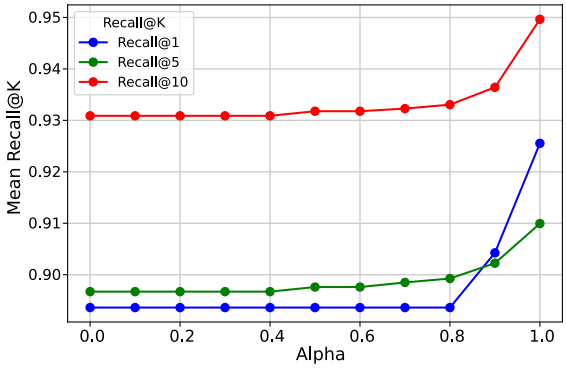

Figure 7: Recall@k for different values of α on the Seal-Tools dataset.

ToolScopeMerger reduces toolset redundancy across all benchmarks (Table 6). Merging results in moderate reductions for BFCL (–14.0%) and Seal-Tools (–2.1%), and a substantial reduction for UltraTool (–25.3%).

Toolset Size Comparison After Merging

Dataset	**Original Size**	**Merged Size**	**% Change**
BFCL	400	344	-14.0%
Seal-Tools	4076	3992	-2.1%
UltraTool	1885	1408	-25.3%

Table 6: Comparison of benchmark toolset sizes before and after merging with ToolScopeMerger.

E ToolScopeMerger Results

E.1 Silhouette Scores on Seal-Tools, BFCL, UltraTool

Figures 8, 9, and 10 show the silhouette scores for each of our benchmark datasets before and after applying ToolScopeMerger with Auto-Correction. Silhouette score is used to assess clustering quality, ranging from -1 to 1. A high silhouette score can reflect a high tool overlap across clusters, since tools that are overly similar will cluster tightly.

After applying our solution, the silhouette scores consistently decrease across different values of k, indicating a reduction in tool overlap. This suggests that ToolScopeMerger with Auto-Correction

successfully merges redundant tools and increases the functional diversity across the toolset.

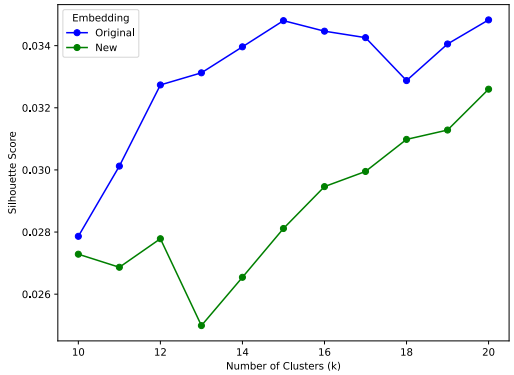

Figure 8: Seal-Tools Silhouette Scores

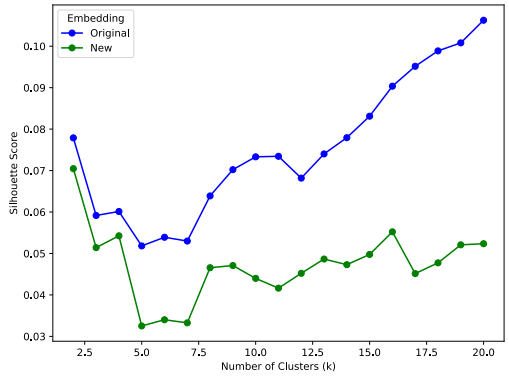

Figure 9: BFCL Silhouette Scores

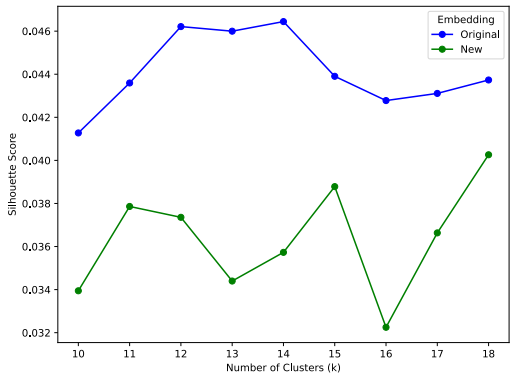

Figure 10: UltraTool Silhouette Scores

E.2 ToolScopeMerger T-SNE Plots on Seal-Tools, UltraTool

Figures 11 and 12 present T-SNE plots of the benchmark datasets before and after we apply ToolScopeMerger with Auto-Correction. T-SNE is a technique that allows us to visualize high-dimensional data in 2D space. After applying our solution, the plots show visibly reduced cluster overlap. This indicates that ToolScopeMerger effectively merges redundant tools and improves separation between tool groups.

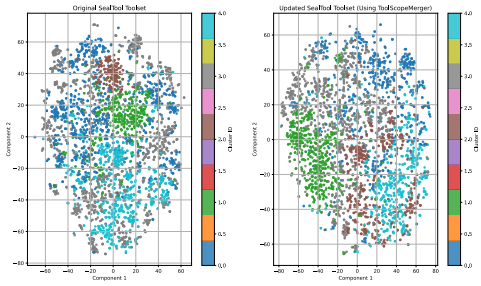

Figure 11: T-SNE visualization of original Seal-Tools tool embedding and merged and autocorrected Seal-Tools tool embedding

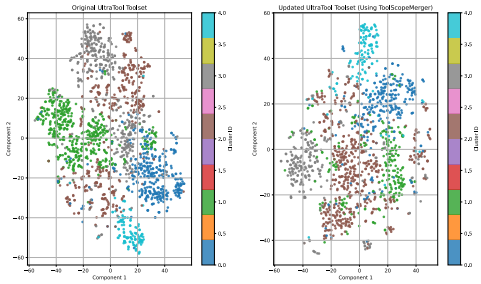

Figure 12: T-SNE visualization of original UltraTool tool embedding and merged and autocorrected UltraTool tool embedding

F Tool Documentation Quality Analysis

F.1 Tool Documentation Quality Overview by Dataset

Figure 13 shows the tool quality distribution for each dataset, adding extra context to our robustness analysis on ToolScope. As seen in the result, Seal-Tools and UltraTool contain a substantial number of medium and low-quality tools, while BFCL is dominated by high-quality ones. Upon human verification, we have also confirmed the low quality

tools from Seal-Tools and UltraTool still contain a basic structure of the tools: a name, signature, and a brief description.

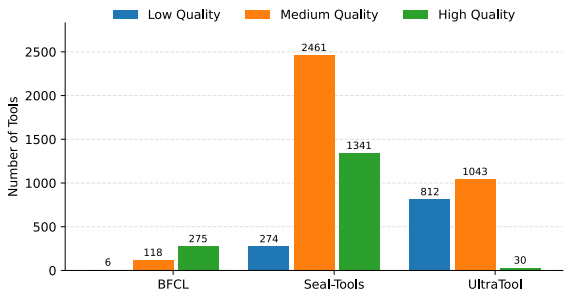

Figure 13: Tool Counts by Quality (Low, Medium, High) Across Benchmarks

F.2 Human Agreement and GPT-4o Alignment

To assess the reliability of our documentation quality scores and validate the use of GPT-4o as an LLM judge, we conducted a small-scale human annotation study.

Human Annotation Protocol. 3 independent annotators evaluated a sample of 30 tools drawn from Seal-Tools and UltraTool. Tools were stratified into three bins (Low, Medium, High) based on initial GPT-4o ratings, with 10 tools sampled per bin. Annotators rated each tool's documentation using the same rubric provided to GPT-4o (see Appendix H.5), which considers three dimensions: parameter clarity, input constraints, and functional behavior. Each rating was given on a 1–5 ordinal scale.

Score Binning. To reduce noise and better reflect practical groupings, we converted the 1–5 ratings into three discrete quality buckets: Low Quality (1-2), Medium Quality (3) and High Quality (4-5).

Inter-Annotator Agreement. We computed pairwise Cohen's κ between all annotators on the bucketed scores. Agreement scores were:

- Rater 1 vs Rater 2: $\kappa = 0.536$
- Rater 1 vs Rater 3: $\kappa = 0.644$
- Rater 2 vs Rater 3: $\kappa = 0.635$

These indicate moderate to substantial agreement under standard interpretation of κ values.

Correlation with GPT-4o. To evaluate alignment with GPT-4o, we computed Kendall's τ between the GPT-4o scores (bucketed) and the average of human bucketed scores. The result was:

$$\tau = 0.664, \quad p = 0.0001$$

This demonstrates strong ordinal correlation, suggesting that GPT-4o scores are a reliable proxy for human judgments of the tool documentation quality.

Conclusion. These results support the use of GPT-4o as an automated scorer for large-scale documentation quality assessment, with acceptable agreement with human judgment and strong correlation trends.

G Algorithm Pseudo-code

Algorithm 1 ToolScope: Automatic Toolset Merging and Retrieval Workflow

Input: Dataset β with queries, Initial toolset $T = \{t_1, \ldots, t_n\}$
Parameter: k for Top-k tools retrieved for tool selection, hybrid retrieval weight α
Output: Merged toolset T', updated ground truth dataset β', top-k most relevant tools retrieved per query q: $T_q^{(k)}$

ToolScopeMerger:
Tool Indexing & Filtering: Encode tool descriptions d_i into embeddings $v_i = f(d_i)$
Retrieve top-k similar tools for each t_i to form candidate pairs $T_i^{(k)}$
Perform cosine similarity threshold θ filtering to filter candidate pairs in $T_i^{(k)}$
Relationship Classification: Use LLM-based classifier M_C to classify semantic equivalence for each candidate pair (t_i, t_j)
Graph Construction: Build undirected graph $G = (T, E)$ where edges represent semantically equivalent pairs
Extract connected components $C = \{C_1, \ldots, C_a\}$
Tool Pruning: Per cluster C_b, create mapping of one representative tool t_b^* to remaining tools in cluster C_b
Output tool mapping between tools to keep and to merge $\phi : T \to T'$
Auto-Correction: Use LLM validator V to verify and refine each cluster $C_b \Rightarrow \{C'_1, \ldots, C'_m\}$
Toolset and Dataset Update: For each final cluster C'_j, synthesize merged tool signature and description d^* and update toolset: $\phi(t) \Rightarrow t_j^*,\ \forall t \in C'_j$
return merged toolset T'
Apply relabeling to benchmark dataset β using mapping ϕ to produce updated dataset β'
return updated ground truth dataset β'
ToolScopeRetriever:
for each query $q \in \beta'$ **do**
  Read decomposed queries from original query q into ordered steps $S_q = \{s_1, \ldots, s_m\}$
  for each step $s \in S_q$ **do**
    Retrieve top-k candidate tools $T_s^{(k)}$ via hybrid retrieval with weight α
    Candidates are reranked using a cross-encoder
    Select top-1 tool $t^{(1)}$ based on the reranker scores
  end for
  Apply min-max normalization on reranker scores for all steps across their remaining tools $t^{(j)}$ for $j = 2, \ldots, k$
  Select top-$(k - s)$ tools
  return top-k tools for q; $T_q^{(k)}$
end for

H Prompts

H.1 Prompt for LLM Merging Classifier M_C

You are an expert in software tool design and resolving function overlap issues.
Goal: Determine whether a single candidate function is semantically similar to a given target function — not necessarily equivalent — and should be considered for potential merging or consolidation if a high degree of similarity is detected.
Background: Overlapping function definitions can confuse both developers and LLM-based systems when multiple similar tools compete for the same user intent. Identifying function pairs that are highly similar in behavior, purpose, and parameter usage can improve clarity, reduce redundancy, and simplify tool invocation.
Definition of Similarity: Two functions are considered similar if:

- They perform the same high-level task or fulfill the same user intent, even if their internal implementations differ.
- Their inputs and outputs are comparable or can be aligned with minimal transformation (e.g., parameters are conceptually equivalent or combinable).
- They could be merged into a unified function with optional parameters or internal branching, without loss of functionality.

Examples of similarity:

- `translateToFrench(text)` and `translateToGerman(text)` → same operation (translation), different fixed parameter ⇒ merge into `translate(text, language)`
- `getUserDetails()` and `fetchUserInfo()` → same intent, different naming
- `addCrop(crop)` and `addCropToFarm(crop, farmId)` → same core action, one more specific ⇒ merge with optional parameter

Not similar if:

- They serve different user intents (`validateInput` vs `sanitizeInput`)
- They act on incompatible types or domains (`generatePDF()` vs `sendEmail()`)
- They overlap in name but not function (`searchUser()` vs `deleteUser()`)

Input

- **Target function:** `{target_tool_docstring}`
- **Candidate function:** `{candidate_tool_docstring}`

Instructions:

- Compare the target and candidate functions carefully.
- If they are similar based on the criteria above, return:
  - The **name** of the candidate function (recommended for potential merge)
  - A short explanation using chain-of-thought reasoning
- If they are **not similar**, return:
  - `None`
  - A brief explanation of why they are not similar enough to merge

Output Format:

- **Line 1:** Candidate function name or `None`. Return only the name — no parameters or signature.
- **Line 2:** Explanation of your reasoning

Figure 14: Prompt for LLM merging classifier M_C.

H.2 Prompt for Merge Auto-Correction LLM V

You are verifying whether the following Python functions can be merged into ONE API.
Similarity rubric (must satisfy 1 AND 2):

- 1. Same *high-level* user intent or task.
- 2. Inputs/outputs can be aligned by adding OPTIONAL parameters or renaming arguments (no functionality lost).

Non-merge only if INTENT clearly differs.
Output (JSON only)

```
{
  "merge": "MERGE_OK",
  "reason": "<optional <25 words>"
}
OR
{
  "merge": "MERGE_BAD",
  "clusters": [
    ["idA", "idB"],
    ["idC"]
  ],
  "reason": "<  <25 words >"
}
```

List of candidate functions (ID : signature — doc):
{GROUP_BLOCK}

Figure 15: Prompt for Merge AutoCorrection LLM V.

H.3 Prompt for LLM Tool Documentation Merger M_D

You are an expert Python API architect:
Task:
Given multiple semantically similar function definitions, merge them into **ONE** canonical function named `{keep_name}`.
Similarity rubric (all must hold to merge):

1. Same user intent/task.
2. Inputs & outputs can be aligned via optional parameters.
3. No functionality is lost after merge.

Instructions:

- Keep `{keep_name}`'s parameter order & defaults first.
- Add **only** unique parameters from the other functions; make them optional.
- Combine docstrings: start with 1–2 concise sentences of purpose, then list arguments (name, type, default, description).
- Do **NOT** output implementation code or markdown fences.
- Output **only** the final signature and full docstring.

`{keep_block} {prune_block}`

Figure 16: Prompt for LLM Tool Documentation Merger M_D.

H.4 Prompt for LLM Agent Tool Selection

You are a tool selection agent.
Your task is to determine the most appropriate tool function (if any) to use for a specific step in a multi-step plan derived from a user request.
You are given:

- The original user query to provide context
- A single step from the plan (typically a short sentence)
- A list of available tools, including their names, descriptions, argument schemas, and expected results

Your job is to return the name of the **single most appropriate function**(from the list below) that can execute this step.
Instructions:

- Only output the function name exactly as written in the list below, or ‘None‘ if no tool is applicable.
- Do not add parentheses, arguments, or explanations.
- Do not make assumptions beyond what is provided in the tool descriptions and step text.

Original User Query:
`{question}`
Plan Step:
`{input}`
Candidate Tools:
`{tools}`
Output:

Figure 17: Prompt for LLM Agent Tool Selection.

H.5 Prompt for LLM Tool Documentation Quality

Figure 18 shows the prompt for Tool Documentation Quality using an LLM.

You are an expert evaluator assessing the quality of tool documentation.
Given a tool's name, function signature, and docstring, decide how clear and helpful the documentation is for someone who needs to call the tool correctly.
Judge the documentation on three evidence pillars:

P Parameters

- Do parameter names and counts match the signature?
- Are required parameters described accurately?
- Are optional parameters or default values acknowledged?

I Inputs & Constraints

- Are input types, roles, and typical values clear?
- Are important constraints, ranges, or usage conditions provided?

B Behavior & Purpose

- Is it clear—at least at a high level—what the tool does when invoked?
- Does the description indicate what the caller can expect in return (or is it self-evident)?

Scoring rubric (1–5):
Score 5 – Excellent
Parameters (P): All required and optional parameters are fully documented; naming matches the signature; defaults are stated when non-trivial.
Inputs & Constraints (I): Types plus examples or constraints are provided for almost every field.
Behavior & Purpose (B): Clear, precise description of functionality and return/output.
Overall Quality: Ready for production use.

Score 4 – Good
Parameters (P): Required parameters are fully documented; most optional parameters are mentioned, with only minor detail gaps.
Inputs & Constraints (I): Types are present, but a few examples or constraints are missing.
Behavior & Purpose (B): Behavior is described clearly, but return/output is only briefly mentioned or assumed obvious.
Overall Quality: Very usable; small refinements desirable.

Score 3 – Fair
Parameters (P): Required parameters are documented and correct, but several optional parameters are missing or vague.
Inputs & Constraints (I): Basic types or roles are stated, but little constraint information is given.
Behavior & Purpose (B): Behavior is conveyed in one or two somewhat vague sentences but is still understandable.
Overall Quality: Usable with modest guesswork.

Score 2 – Poor
Parameters (P): Some required parameters are undocumented or mismatched with the signature.
Inputs & Constraints (I): Types are unclear or absent; no constraint details are given.
Behavior & Purpose (B): Behavior is barely described, and purpose is hard to infer.
Overall Quality: High risk of misuse.

Score 1 – Very Poor
Parameters (P): Few or no parameters are documented, or documentation is misleading.
Inputs & Constraints (I): Input expectations are absent or incorrect.
Behavior & Purpose (B): The tool's purpose is not conveyed.
Overall Quality: Documentation offers virtually no guidance.

Respond with exactly two lines:
1. **Score** (integer 1–5)
2. **One-sentence justification**

Now, evaluate the following tool:
```
Tool name: {tool_name}
Tool signature: {tool_signature}
Tool description: {tool_description}
```

Figure 18: Prompt for Tool Documentation Quality.

I Hyperparameters and Infrastructure

To support reproducibility, we report here all relevant hyperparameters, thresholds, and hardware configurations used in our experiments.

Module	**Hyperparameter**	**Value / Notes**
Models	Tool selection	GPT-4o, Cohere Command-R, LLaMA-3.3-70B
	Embedding Model	thenlper/gte-large
	Reranker model	cross-encoder/ms-marco-MiniLM-L6-v2
	Auto-Correction model	GPT-4o
Retrieval / Reranking Parameters	Hybrid weight α	1.0 (dense-only)
	Top m for reranking	50
	Score normalization	min–max
Clustering	Pair generation size	30 (top cosine neighbors)
	Auto-Correction passes	1 (single GPT-4o prompt)
Prompts	LLM Merging Classifier	Refer to Appendix H for prompt details
	Merge Auto-Correction	
	Tool Documentation Merger	
	LLM Agent Tool Selection	
	LLM Tool Doc. Quality	
Inference	Temperature	0.0
	Max. Tokens	4000
	Top k	50
	Top p	1
	Freq. Penalty	0.0
Sensitivity	Merging threshold range	0.77 to 0.86
	Selected merging threshold	0.82
Hardware	GPUs	3 × NVIDIA A100

Table 7: Hyperparameter settings by module

J Auto-Correction Case Studies (BFCL)

To illustrate how Auto-Correction operates in practice, we provide two case studies: one example where semantically equivalent functions are correctly merged (`MERGE_OK`), and another where an invalid merge is rejected and the cluster is split (`MERGE_BAD`).

Case A: MERGE_OK — Triangle area functions

Candidate set (ID : signature — doc).

10 : `calculate_area(base: int, height: int, unit: str = None)` — Calculate the area of a right-angled triangle given the lengths of its base and height.
0 : `calculate_triangle_area(base: int, height: int, unit: str = None)` — Calculate the area of a triangle given its base and height.
104 : `geometry_area_triangle(base: int, height: int, unit: str = None)` — Calculate the area of a triangle.
11 : `calculate_triangle_area(base: int, height: int)` — Calculate the area of a triangle using its base and height.
95 : `calc_area_triangle(base: int, height: int)` — Calculate the area of a triangle with the formula area = 0.

Auto-Correction output.

```
{
"merge": "MERGE_OK",
"reason": "All functions calculate the area of a triangle."
}
```

Rationale. All functions implement the same intent (*triangle area from base and height*). Minor signature differences (e.g., optional `unit`) are reconciled. Docstring noise (e.g., "area = 0") does not affect intent.

Canonical signature (post-merge).

```
calculate_triangle_area(base: int, height: int, unit: Optional[str] = None)
```

Case B: MERGE_BAD — Final velocity vs. free-fall speed

Candidate set (ID : signature — doc).

27 : `final_velocity(initial_velocity: int, acceleration: int, time: int)` — Final velocity given initial velocity, acceleration, and time.
29 : `calculate_final_speed(time: int, initial_speed: int = None, gravity: float = None)` — Final speed in free fall after a certain time (neglecting air resistance).
31 : `calculate_final_velocity(initial_velocity: int, acceleration: float, time: int)` — Final velocity under constant acceleration.

Auto-Correction output.

```
{
"merge": "MERGE_BAD",
"clusters": [ ["27","31"], ["29"] ],
"reason": "Function 29 assumes free fall, differing from general acceleration in 27 and 31."
}
```

Rationale. IDs `27` and `31` describe the same general constant-acceleration formula and can be merged. ID `29` assumes a free-fall case (gravity parameter), so Auto-Correction correctly split it into a separate cluster.

K Context Length Reduction with ToolScopeRetriever

Table 8: Average prompt tokens required to present the retrieved tools before and after applying ToolScopeRetriever. Prompt tokens are measured at k=5 for BFCL and Seal-Tools, and k=30 for UltraTool.

Dataset	Original	Merged	% Reduction
BFCL	32,563	469	98.56%
Seal-Tools	292,107	317	99.89%
UltraTool	136,352	2,076	98.48%

L Dataset Licensing

L.1 Seal-Tools Dataset License

The Seal-Tools dataset and associated code are licensed under the **Apache License 2.0**. Details can be found at: https://github.com/fairyshine/Seal-Tools?tab=Apache-2.0-1-ov-file

L.2 BFCL Dataset License

The BFCL dataset and associated code are licensed under the **Apache License 2.0**. Details can be found at: https://github.com/ShishirPatil/gorilla/blob/main/LICENSE

L.3 UltraTool Dataset License

The UltraTool dataset and associated code are licensed under the **Apache License 2.0**. Details can be found at: https://github.com/JoeYing1019/UltraTool