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 include tools with overlapping names and descriptions, leading to ambiguity in tool selection and degradation in reasoning performance. Furthermore, the limited input context available to LLM agents constrains their ability to effectively utilize information from a large number of tools. To address these challenges, we propose ToolScope, a two-part workflow which contains: (1) ToolScopeMerger, an automated tool-merging framework that reduces semantic redundancy on tools and offers observability to users to refine their toolset. (2) ToolScope-Filter, which mitigates the limitations imposed by LLMs' context length by retrieving the top-k relevant tools for a given query. This selective filtering reduces input length, enabling more efficient tool usage without compromising selection accuracy. Experimental results on open-source benchmarks demonstrate that ToolScope significantly improves tool selection accuracy, achieving a 34.5% improvement on Seal-Tools and a 18.3% improvement on BFCL compared to baseline methods.

1 Introduction

Tool learning (Qu et al., 2025), which refers to LLMs being proficient in using 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). Within these stages, tool selection, where the model identifies the most appropriate tool from a set to address a given query, is a critical yet underdeveloped subtask (Yang et al., 2023; Qin et al., 2023). Previous work in tuning-free tool selection methods (Qu et al., 2025) has focused on systematically improving tool documentation with frameworks such as DRAFT (Qu et al., 2024) and EASYTOOL (Yuan et al., 2024). Furthermore, work on retrieval-based tool selection explored "term-based" (e.g. BM25) (Robertson et al., 2009) and "semantic-based" methods such as CRAFT (Yuan et al., 2023) or off-the-shelf embeddings (Qu et al., 2025) as the retrieval technique to improve tool selection. The key challenge in tool selection is that LLMs often fail to choose the most appropriate tools. Two major factors which contribute to this are: (1) overlapping tool descriptions which introduce ambiguity that reduces both retrieval and tool selection accuracy (OpenAI, 2024; Lumer et al., 2024; Huang et al., 2023), (2) the limited input context of LLMs constrains their ability to reason effectively over large toolsets (Huang et al., 2023). As toolsets grow in size and complexity, these issues become more pronounced. Therefore, addressing both tool overlap and context limitations is essential for improving tool selection performance.

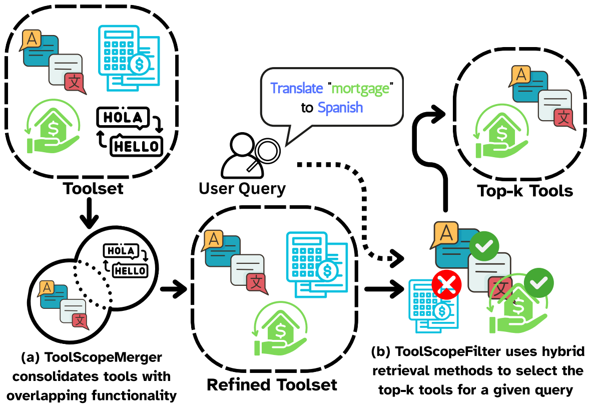

Figure 1: A simplified overview of using ToolScope for a tool selection task. A toolset is fed into ToolScope-Merger (a) creating a refined toolset that is passed to ToolScopeFilter (b). The retrieved best tools for a query are then passed to an LLM agent for selection.

We propose **ToolScope**, a two-part solution that automatically merges tools through ToolScope-Merger and retrieve most relevant tools using ToolScopeFilter. This addresses two main challenges: overlapping tool descriptions and context length limitations.

ToolScopeMerger helps improve toolset quality by identifying and merging overlapping tools. It begins by forming tool pairs based on their semantic similarity. An LLM determines whether tool pairs are functionally similar enough to be combined. These relationships are used to build a graph that defines the mapping between the tools to keep, and the tools to prune. Finally, we use another LLM to guide the merging, and we further update the toolset and the ground truth in the dataset. ToolScopeMerger also offers the option for users to manually review and adjust the results, ensuring flexibility and transparency.

ToolScopeFilter is a retrieval system that handles single and multi-tool queries by retrieving candidate tools using a hybrid strategy that combines dense and sparse search. The extracted tools are then reranked based on contextual relevance, with score normalization applied to ensure fair comparison between all candidates. ToolScopeFilter enhances the relevance of retrieved tools by capturing both semantic and exact matches, and improves results for retrieval and tool selection accuracy.

In summary, our contribution is as follows: (1) **ToolScopeMerger**, a novel graph-based framework that merges semantically similar tools to address tool overlap issues, improving toolset clarity. (2) **ToolScopeFilter**, a hybrid retrieval system that integrates both sparse and dense scores to reduce context length. (3) We demonstrate that combining **ToolScopeMerger** with **ToolScopeFilter** leads to significant improvements in both retrieval and selection accuracy across open-source benchmarks. Together, these methods improve tool selection performance, increasing tool selection accuracy on Seal-Tools by 34.5% and on BFCL by 18.3%, as reported in Table 1.

2 Related Work

2.1 Tool Learning

Tool-augmented large language model (LLM) agents enhance reasoning via external tools such as calculators, APIs, and search engines (Yehudai et al., 2025; Qin et al., 2024; Chen et al., 2023; Qin et al., 2023; Xu et al., 2023; Jin et al., 2025). These agents consistently outperform baseline models on complex benchmarks and real-world tasks (Nakano et al., 2021; Qin et al., 2023; Winston and Just).

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), and reinforcement learning (Wang et al., 2025; Qian et al., 2025). However, these approaches are computationally expensive. 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). Furthermore, MetaTool (Huang et al., 2023) attempts to resolve the overlapped tool issue through clustering and manual curation, but did not address the problem at scale through automated merging solution.

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 combining lexical and semantic methods or the benefits of integrating enhanced toolsets (Lumer et al., 2025). Finally, no study has combined tool overlap solutions and hybrid retrieval methods to tackle improvement of LLM agent tool selection.

3 Methods

3.1 Overview of ToolScope

ToolScope is a comprehensive framework designed to optimize the performance of LLM agents by addressing the two key challenges identified in tool selection: tool overlap and tool selection accuracy. ToolScopeMerger simplifies a toolset by systematically consolidating overlapping tools. Then, this updated toolset is used in ToolScopeFilter to retrieve relevant tools. ToolScope combines the approaches to further improve tool selection accuracy.

3.2 ToolScopeMerger

ToolScopeMerger is a graph-based, auto-merging framework designed to address the problem of tool overlap in large-scale toolsets used by LLM agents. A simplified overview can be seen in Figure 2. Overlapping tools have become an increasingly common issue in both production and benchmark toolsets, especially those designed to cover multiple domains and handle thousands of queries. Tool overlap introduces ambiguity during tool selection, negatively impacting retrieval and tool selection accuracy, as LLM agents struggle to select between two tools with very similar descriptions and functionality. ToolScopeMerger resolves these issues by merging semantically similar tools into a single tool that covers the functionality of all tools to merge. ToolScopeMerger can alternatively be used in a manual form for observability, where semantically overlapping tools will be identified and presented to the user.

ToolScopeMerger workflow is structured into five key phases: (1) Tool Indexing - similar tool pairing, (2) Tool Relationship Classification - identify overlap tool pairs, (3) Graph Construction - structure global pruning pairs, (4) Tool Pruning - unify functions to merge, and (5) Toolset and Dataset Update. Each phase is described in the following.

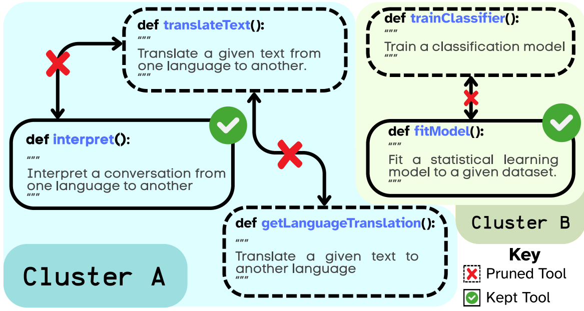

Figure 2: ToolScopeMerger uses an undirected, graph-based approach to identify and then prune overlapping tools. To ensure observability, the proposed merges are provided to tool developers for merge auditing.

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 $M_C : T \times T \rightarrow \{0, 1\}$ 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 C. 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.

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

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 $\phi(t) = t_j^*, \forall t \in C_b$.

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 D 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 2:

$$\forall (q, l) \in \beta,\ t \Longrightarrow \phi(t) \tag{2}$$

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.

ToolScopeMerger also allows users to perform reviews after merging, helping refine and identify edge cases in the toolset. To complement automated merging, in our experiments, we also manually inspect selected clusters, pruning decisions, and failure cases in our experiments. These include similar tools that were not merged due to vague descriptions, and incorrect merges based on surface-level similarity.

3.3 ToolScopeFilter

Effective tool selection is essential for enhancing the capabilities of large language models (LLMs) when solving complex tasks involving external tools. Inspired by recent work (Chen et al., 2024; Qu et al., 2024), 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.

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:

$$s(q, t) = \alpha \cdot s_{\text{dense}}(q, t) + (1 - \alpha) \cdot s_{\text{sparse}}(q, t) \tag{3}$$

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, selecting the top-1 tool $t^{(1)}$ with the highest reranker score. The remaining tools $t^{(j)}$ for $j = 2, \ldots, M$ are normalized using min-max normalization:

$$s_{\text{norm}}(t^{(j)}) = \frac{s(t^{(j)}) - s_{\min}}{s_{\max} - s_{\min} + \varepsilon} \tag{4}$$

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), RestBench (Song et al., 2023), and SealTools (Wu et al., 2024a). Despite covering a range of tool-use tasks, existing benchmarks have key limitations: (1) some tool descriptions were lacking 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 SealTools and BFCL as primary datasets. SealTools (Wu et al., 2024a) includes out-of-domain test examples that calls single and multiple tools over a large toolset with 4076 tools. BFCL (Patil et al., 2024) has a simple single-turn tool calling dataset which contains 400 queries and 400 tools with one to one query to tool mappings that allow us to test our tool merging strategy efficiently. Together, these datasets support evaluation across different retrieval challenges.

Benchmarks	Configuration	CSR						
		$k=1$	$k=5$	$k=10$	$k=15$	$k=20$	$k=25$	$k=30$
Seal-Tools	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	-	**0.889**	**0.908**	**0.919**	**0.921**	**0.919**	**0.926**
BFCL	BM25	0.695	0.850	0.875	0.870	0.868	0.878	0.883
	Dense	0.780	0.888	0.900	**0.915**	**0.912**	**0.912**	**0.915**
	ToolScope	**0.878**	**0.912**	**0.915**	**0.915**	**0.912**	**0.912**	**0.915**

Table 1: Correct Selection Rate (CSR) of top k tools across Seal-Tools and BFCL datasets. DPR and ToolShed do not have CSR results since the implementation code is not available.

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, as shown in Equation 5. Recall@k quantifies the proportion of ground-truth tools correctly retrieved among the top-k predictions (see Equation 6 in Appendix A).

$$\text{CSR} = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \mathbb{I}[\hat{T}_q = T_q^*] \tag{5}$$

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

Baselines. We benchmarked our methods against established baselines on tool selection and retrieval including: (1) BM25 (Robertson et al., 2009), a term based retrieval method that scores query-tool relevance using term frequency and inverse document frequency (TF-IDF). (2) Dense embeddings (Qu et al., 2025), which encodes both queries and tool descriptions into dense vector representations using a pretrained embedding model. For retrieval performance on Seal-Tools, we also report performance from DPR (Wu et al., 2024a) and ToolShed (Lumer et al., 2024). All baselines operate over the original, unmodified toolsets. Furthermore, for multi-tool queries, we retrieve the top-k tools from the original query, then later use query decomposition at tool selection.

Implementation Details. For main experiments of ToolScopeMerger, we use thenlper/gte-large embedding (Li et al., 2023), FAISS index (Douze et al., 2024), GPT-4o (OpenAI, 2024) as the backbone model. For main experiments of ToolScopeFilter, we use BM25 as sparse retrieval method, thenlper/gte-large (Li et al., 2023) as the dense embedding, ms-marco-MiniLM-L6-v2 (Cross-Encoder, 2024) as the cross encoder and GPT-4o (OpenAI, 2024) as the backbone model.

4.2 Experimental Results

We present our experimental results on LLM agent tool selection in Table 1. Based on the results, we have the followng observations: ToolScope consistently outperforms baselines in Correct Selection Rate (CSR) across both Seal-Tools (Wu et al., 2024a) and BFCL (Patil et al., 2024) benchmarks. As seen in Table 3, even when applied independently, ToolScopeFilter substantially improves CSR, and combining it with ToolScopeMerger further boosts performance by reducing the toolset semantic redundancy and aligning better with the LLM's reasoning behavior. By jointly optimizing the tool index and toolset granularity, ToolScope enables more accurate and scalable function selection for LLM agents.

The improvements in CSR are robust across all evaluated values of k, suggesting that ToolScope streamlines the retrieval process while reducing functional overlap. As the value of of k increases, we observe a higher performance difference in Seal-Tools. This could be attributed to the fact that as we increase the value of k, there is a higher percentage of overlap tools in the original dataset compared with the merged dataset.

4.3 Result Analysis

Based on the experiment results, we have the following observations:

ToolScope improves both retrieval and selection performance. ToolScopeMerger increases retrieval accuracy by reducing semantic overlap in the toolset and making the retrieval space clearer, as reported in Table 2. Recall@10 improves significantly from 0.550 to 0.935 on the SealTools (Wu et al., 2024a) benchmark and from 0.945 to 0.985 on BFCL (Patil et al., 2024), highlighting the bene-

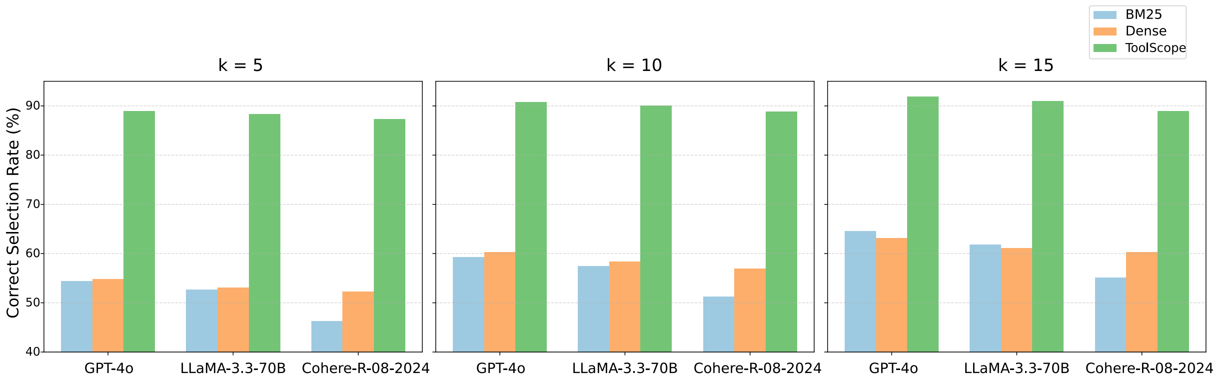

Figure 3: Analysis of cross model generalization on ToolScope with top k tools across Seal-Tool.

fit of refining the toolset. ToolScopeFilter applies a hybrid retrieval strategy that combines sparse (BM25) and dense embeddings. The weighting parameter α balances the contribution of lexical and semantic similarity, while the cross-encoder reranker evaluates retrieved candidates using contextual information. However, retrieval accuracy does not consistently improve with hybrid search. The highest Recall@k scores are obtained when $\alpha = 1$, indicating that dense retrieval alone performs best for retrieval, as seen in Figure 4. Accordingly, the ToolScope results reported in Table 2 use dense-only retrieval, where our method outperforms all baselines across all k values on BFCL and at $k = 5$ on SealTools.

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[1]	-	0.480	0.680
Seal-Tools / ToolShed[2]	-	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**

[1] DPR results are from (Wu et al., 2024a)
[2] ToolShed results are from (Lumer et al., 2024)

Table 2: Recall@k scores for selected configurations on Seal-Tools and BFCL benchmarks

ToolScope has demonstrated cross model generalization in improving CSR. We compared ToolScope to the baselines which use BM25 and Dense retrieval techniques. We evaluated three foundation models: GPT-4o (OpenAI, 2024), Cohere-Command-R-08-2024 (Cohere, 2024), and LLaMA-3.3-70B (Meta Platforms, Inc., 2024). As shown in Figure 3, ToolScope consistently outperforms both baselines across all k values (5, 10, and 15) in CSR. 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.

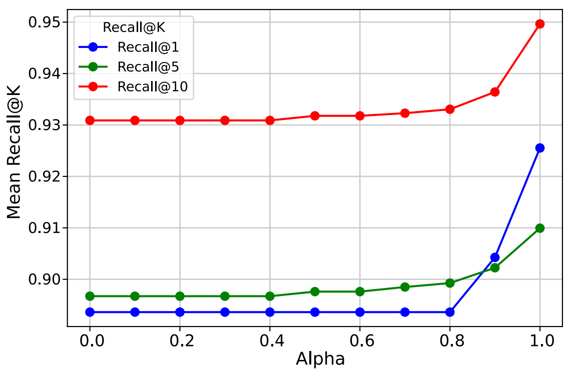

Figure 4: Recall@k for different values of α on the Seal-Tools dataset.

Ablation Study. We conduct ablation studies on both BFCL and Seal-Tools to evaluate the contributions of the two core components in ToolScope: ToolScopeMerger and ToolScopeFilter. As shown in Table 3, ToolScope consistently achieves the highest CSR across all top-k values, demonstrating the effectiveness of combining tool pruning and targeted retrieval.

In BFCL, where each query corresponds to exactly one gold tool, CSR is particularly sensitive to the quality of the tool input. We observe that ToolScopeMerger alone achieves comparable performance to ToolScopeFilter, and the full ToolScope system achieves the best CSR, particularly at lower k. This highlights the importance of

Methods	CSR for BFCL		
	$k = 1$	$k = 5$	$k = 10$
ToolScope	**0.878**	**0.912**	**0.915**
ToolScopeFilter *only*	0.820	0.882	0.888
ToolScopeMerger *only*	0.825	0.902	0.915

Methods	CSR for Seal-Tools		
	$k = 10$	$k = 20$	$k = 30$
ToolScope	**0.908**	**0.921**	**0.926**
ToolScopeFilter *only*	0.895	0.906	0.912
ToolScopeMerger *only*	0.613	0.701	0.732

Table 3: Ablation Study of ToolScope on BFCL and Seal-Tools. ToolScopeFilter *only* uses original data with retrieval improvement, while ToolScopeMerger *only* uses merged data without retrieval improvement.

ensuring high-quality, non-redundant tools, as even slight ambiguity in tool definitions can negatively affect retrieval and selection in tool settings like BFCL.

For Seal-Tools, removing ToolScopeMerger leads to a moderate CSR drop. This can be explained by the fact that only 20% of queries need to invoke tools were either modified or merged. This indicates that pruning semantically overlapping tools improves the diversity and coverage of top-k candidates. Excluding ToolScopeFilter results in a more significant CSR decline, highlighting the necessity of an effective retrieval mechanism as Seal-Tool has a large original toolset (4076 tools).

Reranking enables high CSR with smaller retrieval sizes of ToolScopeFilter. As shown in Figure 6, enabling the reranker significantly improves CSR (Correct Selection Rate), at smaller retrieval sizes, such as $k = 5$ or 10. As k increases, the performance gap between reranked and non-reranked configurations narrows, suggesting diminishing returns from retrieving larger toolsets. Overall, $k = 5$ or $k = 10$ with reranking provide a good balance between performance and retrieval efficiency.

ToolScopeMerger effectively reduces functional overlap. ToolScopeMerger removes semantically similar tools, improving the distribution in embedding space. We reduced the toolset in BFCL (Patil et al., 2024) from 400 to 344 tools by removing tools with similar names and descriptions. SealTools (Wu et al., 2024a), which started with 4,076 tools, also showed overlap: we merged 84 tools despite earlier claims of minimal redundancy. We evaluated the impact using the silhouette co-

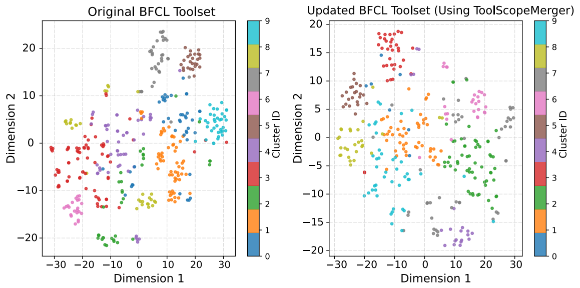

Figure 5: T-SNE visualization of original BFCL tool embedding and merged BFCL tool embedding

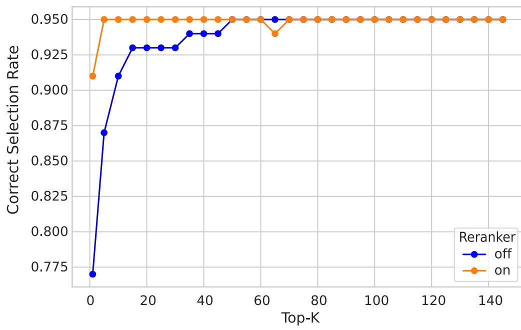

Figure 6: The CSR results (%) of top k tools with/without Reranker

efficient (Rousseeuw, 1987), where lower values indicate less overlap. We also used t-SNE plots (Van der Maaten and Hinton, 2008) to visualize tool distribution. As shown in Figure 5 and Appendix E.2, both datasets exhibit sparser distributions. As seen in Appendix E.1, both datasets also have lower silhouette scores after merging, confirming that ToolScopeMerger reduces semantic redundancy.

ToolScopeMerger increases observability for tool developers. We provide outputs that enable tool developers to audit merge decisions, as illustrated in Figure 2. This allows them to verify correct merges, such as *translateText* and *getLanguageTranslation* as shown in Listing 1, and spot incorrect ones caused by unclear documentation. For example, in SealTools, *getGeologyData* and *getGeologyInfo* seen in Listing 2 were merged despite lacking enough detail to confirm they serve the same function. By making similarity-driven merges transparent, tool developers can confirm or override merge decisions with minimal overhead and improve tool descriptions to better reflect functionality.

Correct merge:

Listing 1: Function definition for `translateText` and `getLanguageTranslation`

```
translateText:
translateText(text: str, source_language: str,
    target_language: str)
Translate a given text from one language to
    another

Args:
   text (str): The text to be translated (e.g.,
       Hello, how are you?)
   source_language (str): The source language
       of the text (e.g., English, Spanish)
   target_language (str): The target language
       for translation (e.g., Spanish, French)
```

```
getLanguageTranslation:
getLanguageTranslation(text: str,
    source_language: str, target_language: str)
Translate a given text to another language

Args:
   text (str): The text to be translated
   source_language (str): The source language
       of the text (e.g., English, Spanish,
       French)
   target_language (str): The target language
       for translation (e.g., English,
       Spanish, French)
```

Incorrect merge due to Poor Documentation

Listing 2: Function definition for texttgetGeologyData and texttgetGeologyInfo

```
getGeologyData:
getGeologyData(location: str)
Retrieve geological data for a specific location

Args:
   location (str): The location for which you
       want to retrieve geological data (e.g.,
       mountain range, river, city)
```

```
getGeologyInfo:
getGeologyInfo(location: str)
Retrieve geological information

Args:
   location (str): The location for which you
       want to retrieve geological information
       (e.g., mountains, lakes, caves)
```

The absence of query decomposition significantly impacts performance in multi-tool scenarios. In our baseline methods, which do not incorporate query decomposition, we observe a sharp decline in CSR and Recall@k on multi-tool queries. Without breaking down complex queries into atomic sub-queries for the LLM agent, the retrieval and selection process struggles to identify *all* necessary tools in the top-k. This results in partial matches, where only a subset of the necessary tools are retrieved for a given query, reducing CSR since the necessary tools are not available in the LLM agent's perspective. These findings suggest that query decomposition is a critical component for effective tool selection in multi-tool scenarios.

Tool overlap remains a prevalent and often unaddressed issue across open-source benchmarks. For example, although Toolshed (Lumer et al., 2024) reports that the SealTools benchmark contains few overlapping tools, we found multiple instances of semantically redundant tools that can hinder retrieval and selection accuracy without a solution like ToolScopeMerger.

Similarly, BFCL-simple (Patil et al., 2024), by design, includes overlapping tools with highly similar functionality and descriptions. While this may serve other evaluation purposes, it poses challenges for experiments focused on tool selection accuracy, such as ours. These inconsistencies highlight a critical limitation in current benchmarks: the prevelance of overlapping tools. As our results demonstrate, tool overlap introduces ambiguity that can distort retrieval metrics and, more importantly, cause selection errors. For future research in tool learning and tool selection, it is essential to address this issue. Benchmark designers should strive for clearer manual distinction between tools or provide annotations of overlapping to ensure more accurate performance reporting.

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 and increases observability for tool developers to audit merge decisions, while **ToolScopeFilter** improves retrieval accuracy through query decomposition, hybrid retrieval and reranker and robust score ranking logic for tool retrieval. Together, our results show that these components significantly improve tool selection accuracy across standard benchmarks Seal-Tools (Wu et al., 2024a) and BFCL (Patil et al., 2024), increasing CSR on Seal-Tools by 34.5% and CSR on BFCL by 18.3% when compared to baselines. We also identify a persistent tool overlap issue in currently available open-source public benchmarks, which calls for a higher quality toolset curation. Overall, ToolScope provides a scalable solution for improving LLM-agent tool selection in real-world settings.

Limitations

ToolScope currently focuses on reducing tool overlap and increasing retrieval performance. The quality of the tool documentations can be further improved using automatic documentation refinement framework such as DRAFT (Qu et al., 2024). This may bring additional improvements to LLM agent tool selection accuracy.

Future Work

We expect additional improvements for the future work. First, adopting additional advanced retrieval methods, such as multi-index frameworks that may improve scalability and relevance in large tool repositories. Second, given the limitation to the overlap issues identified in existing benchmarks we evaluated, it is crucial to expand on additional benchmarks of diverse toolset. It is crucial to evaluate whether it is better to modify the existing toolset and update single-label ground truth or introducing multi-label ground truth so that we can have a more comprehensive and accurate benchmark to evaluate LLM agent system. Finally, expanding the scope of the evaluation to analyze ToolScope's impact on tool calling and response generation would help establish a clear picture on holistic improvements to LLM agent tool learning.

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*.

Nuo Chen, Hongguang Li, Baoyuan Wang, and Jia Li. 2023. From good to great: Improving math reasoning with tool-augmented interleaf prompting. *arXiv preprint arXiv:2401.05384*.

Yanfei Chen, Jinsung Yoon, Devendra Singh Sachan, Qingze Wang, Vincent Cohen-Addad, Mohammadhossein Bateni, Chen-Yu Lee, and Tomas Pfister. 2024. Re-invoke: Tool invocation rewriting for zero-shot tool retrieval. *arXiv preprint arXiv:2408.01875*.

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

Cross-Encoder. 2024. cross-encoder/ms-marco-minilm-l6-v2. Accessed: 2025-05-18.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library. *arXiv preprint arXiv:2401.08281*.

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*.

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*.

Yixin Li and 1 others. 2023. Gte-large model card. `https://huggingface.co/thenlper/gte-large`. Accessed: 2025-05-18.

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.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, and 1 others. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

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, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Xuanhe Zhou, Yufei Huang, Chaojun Xiao, and 1 others. 2024. Tool learning with foundation models. *ACM Computing Surveys*, 57(4):1–40.

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*.

Cailin Winston and René Just. A taxonomy of failures in tool-augmented llms.

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.

Qiantong Xu, Fenglu Hong, Bo Li, Changran Hu, Zhengyu Chen, and Jian Zhang. 2023. On the tool manipulation capability of open-source large language models. *arXiv preprint arXiv:2305.16504*.

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.

Asaf Yehudai, Lilach Eden, Alan Li, Guy Uziel, Yilun Zhao, Roy Bar-Haim, Arman Cohan, and Michal Shmueli-Scheuer. 2025. Survey on evaluation of llm-based agents. *arXiv preprint arXiv:2503.16416*.

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*.

A Metrics Definition

Recall@k measures the proportion of ground-truth tools that are correctly retrieved within the top-k predictions. It is defined as:

$$\text{Recall}@K = \frac{1}{|Q|} \sum_{q=1}^{|Q|} \frac{|T_q^K \cap T_q^*|}{|T_q^*|} \tag{6}$$

where Q is the set of queries, T_q^* is the ground-truth toolset, and T_q^k is the top-k predicted tools.

B Dataset Licensing

B.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

B.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

C Prompt for LLM Merging Classifier M_C

You are an expert in software tool design and resolving function overlap issues.

Goal: Determine whether any candidate functions are semantically equivalent to a given target function - in the strictest sense - and recommend one for merging only if true equivalence is detected.

An overlapped issue arises when a user query can be handled by multiple similar functions. This ambiguity can reduce LLM accuracy in selecting the correct function to call.

Merging should only be suggested if the functions are equivalent:

Equivalent: Two functions are considered equivalent only if: They perform exactly the same core operation. They are fully interchangeable in real-world usage - i.e., replacing one with the other does not alter the behavior, side effects, or required inputs. Their parameter lists match exactly - in both name and count (or have trivially renamable arguments with the same semantics).

Examples of equivalence: trainClassifier(data) and predictModel(data) (if both imply model training despite different names) fetchUserInfo() and getUserDetails()

Do not consider two functions equivalent if: They differ by a fixed value (e.g., translateToHebrew vs translateToItalian) One modifies the input while the other checks it (e.g., sanitizeInput vs validateInput) One function is a logical generalization or specialization of the other (e.g, addCrop vs. addCropToFarm, translateText vs. translateSpanish)

If none of the candidate functions provide semantically similar capabilities with the target function, return None. Think about this from the perspective of how an LLM might confuse or suggest the function to prune based on natural language queries.

Target function:
{target_tool_docstring}

Candidate functions:
{candidate_output_str}

Instructions: Compare each candidate function to the target function. If any of the candidate functions are sufficiently overlapping or logically mergeable with the target function, return the candidate function based on the rules below. The candidate function selected and target function should have exact same list of parameters. If the function names are semantically equivalent or fixed-value variants (e.g., hardcoded languages), return the name of the candidate function to prune (as equivalent). If no functions meet the merge criteria, return None.

Output only two lines:

- **The chosen candidate function** from the list of candidate functions. Only return the function name - do not return parameters or signature.
- **A short reasoning** using chain-of-thought that explains the relationship, and why this choice was made.

D Prompt for LLM Description Merger M_D

You are given multiple Python function definitions, each with a signature and docstring.

Your task is to merge all of them into a **single function** using the name **{keep_tool}**'.

Instructions:

- The function to merge into is listed first.
- The other functions may include additional or overlapping parameters and docstring details.

- Your goal is to:

- Combine all **unique arguments**
- Prefer parameter types, defaults, and naming from the canonical function {keep_tool}: Additional parameters added could be added as optional parameter. Do not hallucinate extra parameters not present in the function to merge into and functions to merge from.
- Carefully integrate all relevant docstring content

- The final result must include:

- One complete **function signature**
- One consolidated **docstring**, insert all descriptions about the function first, make sure descriptions are well summarized and concise. Then insert the argument explaination.
- Do **not** include implementation code.
- Do **not** include markdown, explanation, or commentary.
- Only output the final **signature and docstring**, nothing else.

{keep_block}{prune_block}

E ToolScopeMerger Results

E.1 Silhouette Scores on Seal-Tools and BFCL

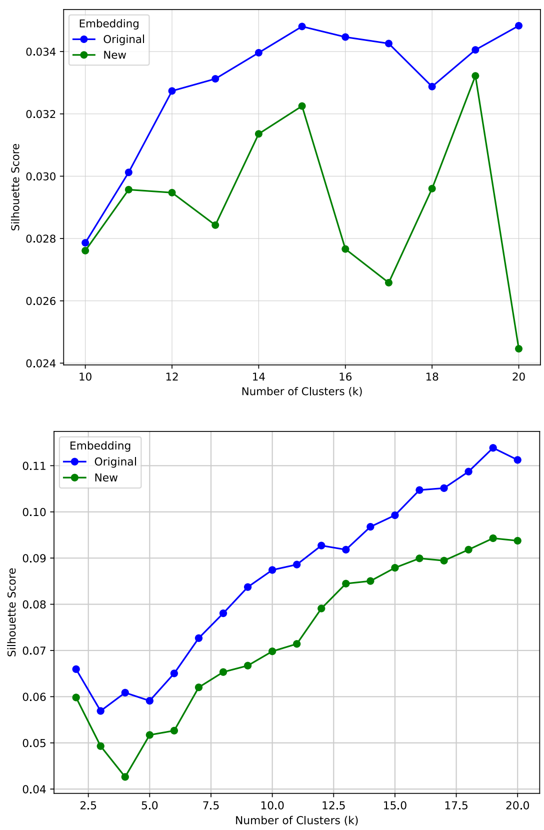

Figure 7: Seal-Tools (top) and BFCL (bottom) silhouette scores comparison

E.2 ToolScopeMerger Result on Seal-Tools

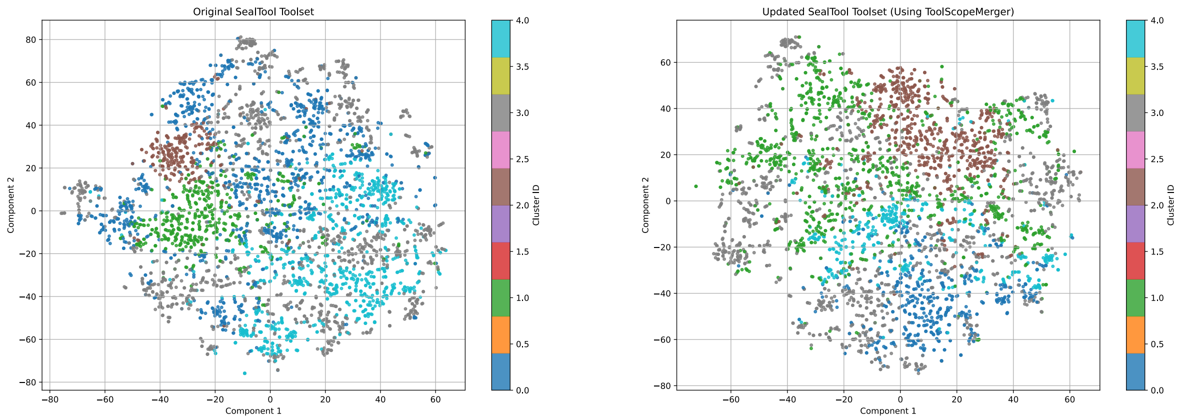

Figure 8: T-SNE visualization of original Seal-Tools tool embedding and merged Seal-Tools tool embedding