ToolExpNet: Optimizing Multi-Tool Selection in LLMs with Similarity and Dependency-Aware Experience Networks

Anonymous ACL submission

Abstract

Tool learning enhances Large Language Models' (LLMs) dynamic interaction with external tools, improving their ability to solve complex problems. However, current empirical methods, which primarily focus on isolated tools learning, still struggle with accurate multi-tool selection due to issues like confusing similar tools and neglecting dependencies. To address these challenges, we propose the Tool Experience Network (ToolExpNet), which integrates tools and trial-and-error experiences into a network characterized by semantic similarity and dependency relationships. ToolExpNet iteratively conducts simulated experiments using adaptive sampling to explore subtle differences and connections between tools, and summarizes these experiences to provide insightful guidance for LLM tool selection. Our experiments demonstrate that learning the relationships between tools helps achieve more comprehensive tool learning. Evaluations on multiple real-world API datasets show that ToolExpNet effectively addresses common challenges in multi-tool selection, significantly outperforming existing baselines across different foundation LLMs.

1 Introduction

Tool learning (Qin et al., 2024a; Qu et al., 2025b) empowers Large Language Models to dynamically interact with external tools, enhancing their problem-solving capabilities for complex tasks (Nakano et al., 2021; Xu et al., 2023; Schick et al., 2023; Zhao et al., 2024b). This paradigm significantly boosts performance in knowledge acquisition (Gu et al., 2024; Schick et al., 2023), expertise enhancement (Kadlcík et al., 2023; He-Yueya et al., 2023; Bran et al., 2024), automation efficiency (Schick et al., 2023; Yao et al., 2022a), and interaction capabilities (Yang et al., 2023b; Wang et al., 2024b). To invoke external tools, LLMs typically conduct task planning and tool selection, generating final answers based on tool execution results (Song et al., 2023; Shen et al., 2023).

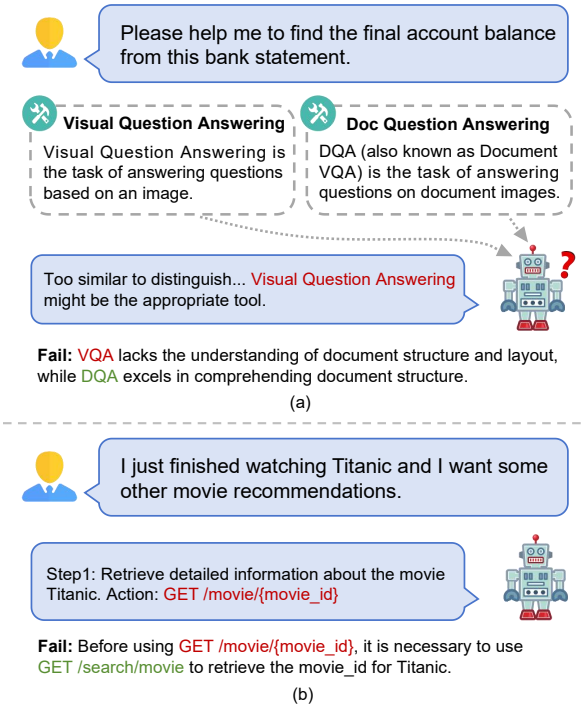

Figure 1: Two common failure modes in real-world tool invocation scenarios with existing methods: The top illustration shows an incorrect tool selection due to semantic similarity, while the bottom illustration demonstrates a planning error due to overlooking functional dependency.

While tuning-based methods effectively enable LLMs to use external tools (Lu et al., 2023; Liang et al., 2023; Qiao et al., 2024), tuning-free methods are irreplaceable due to their ability to learn new tools without parameter changes and their applicability to closed-source models (Liu et al., 2024c; Zhang et al., 2024; Liu et al., 2024b). These methods primarily rely on feeding tool documentation or memory into the LLM's context to select the correct tool sequence, highlighting the importance of comprehensive and accurate tool descriptions. Re-

cent studies have improved tool understanding by rewriting documentation (Yuan et al., 2024; Chen et al., 2024) or incorporating trial-and-error experiences into the model's context (Zhao et al., 2024a; Wang et al., 2024a; Qu et al., 2025a). However, multi-tool selection accuracy remains a challenge in real-world complex tasks.

We observe that, with enhanced foundation model capabilities, the proportions of previously identified error types (Song et al., 2023; Shi et al., 2024a), such as API hallucination and format non-compliance, have decreased. However, LLMs still face two key challenges in selecting external tools: confusing similar tools due to ambiguous documentation and overlooking tool dependencies. Figure 1 illustrates examples of these challenges. Previous work on LLM tool understanding has typically focused on isolated tools (Wang et al., 2024a; Qu et al., 2025a), neglecting potential inter-tool associations in real scenarios. This limits the LLMs' accurate and comprehensive understanding of tools, resulting in suboptimal performance in addressing these challenges.

Inspired by human cognitive learning theories (Smelser et al., 2001; Barsalou, 2014), humans integrate new knowledge by associating it with existing knowledge systems, forming structured cognitive schemas. This associative mechanism is particularly helpful in learning similar concepts, where comparative analysis enables learners to grasp subtle differences more accurately. Following this principle, we suggest that LLMs' tool learning strategy should not be limited to the functional attributes of individual tools but should focus on establishing a network of relationships between tools.

Based on this concept, we propose the Tool Experience Network (ToolExpNet), which organizes the available toolset and trial-and-error experiences into a network to enhance comprehensive tool learning. Specifically, ToolExpNet's graph structure includes two types of edges: semantic similarity edges (E_s), which connect tools with similar functional descriptions, and functional dependency edges (E_d), representing the sequential invocation dependencies between tools. This structure systematically addresses the challenges of distinguishing between similar tools (via E_s) and captures the opportunities for combining tools (via E_d).

ToolExpNet employs an iterative contrastive-relation trial and error process and tool insight refinement to explore tool interactions. Tool pairs are sampled based on two types of links with adaptive weights. It generates simulated queries to highlight functional differences and dependencies between tool pairs, answers these queries, and updates the weights based on error rates. In the subsequent tool insight refinement stage, the LLM summarizes the usage experiences of tools guided by these links, forming comprehensive tool guidance to enhance the tool selection process.

Our contributions are as follows: (1) We propose ToolExpNet, a novel tool network based on similarity and dependency relationships. It rewrites tool guidance to emphasize inter-tool associations, unlike existing methods focused on isolated tools. This approach highlights the importance of modeling tool relationships during tool learning phase and provides insights for future methods. (2) We introduce a holistic tool learning strategy that simulates confusing and dependency queries to guide LLMs through trial-and-error learning, forming tool insights that significantly enhance the accuracy of multi-tool invocation. (3) Through extensive experiments on multiple foundation models and real-world datasets, we demonstrate that ToolExpNet outperforms existing methods and provide an in-depth analysis of its mechanisms.

2 Tool Experience Network

We propose the Tool Experience Network (ToolExpNet), as shown in Figure 2, which organizes the available toolset into a network based on similarity and dependency relationships, facilitating more systematic and comprehensive tool learning.

2.1 Graph Structure

Formally, we model the tool ecosystem as a graph $G = (V, E_s \cup E_d)$, where nodes represent individual tools and edges capture complex inter-tool relationships. Each tool node $v_i \in V$ is defined as a tuple (e_i, φ_i), where e_i includes API metadata and φ_i represents functional insights distilled through LLM-based experience summarization. These insights can serve as empirical knowledge for LLMs during the tool selection phase. The edges explicitly characterize two fundamental relationships:

Semantic Similarity Edges (E_s): These edges connect tool pairs (v_i, v_j) with partial functional overlap or semantically analogous descriptions, which may mislead LLMs into conflating their distinct capabilities during tool selection.

Functional Dependency Edges (E_d): These edges denote relationships where one tool's func-

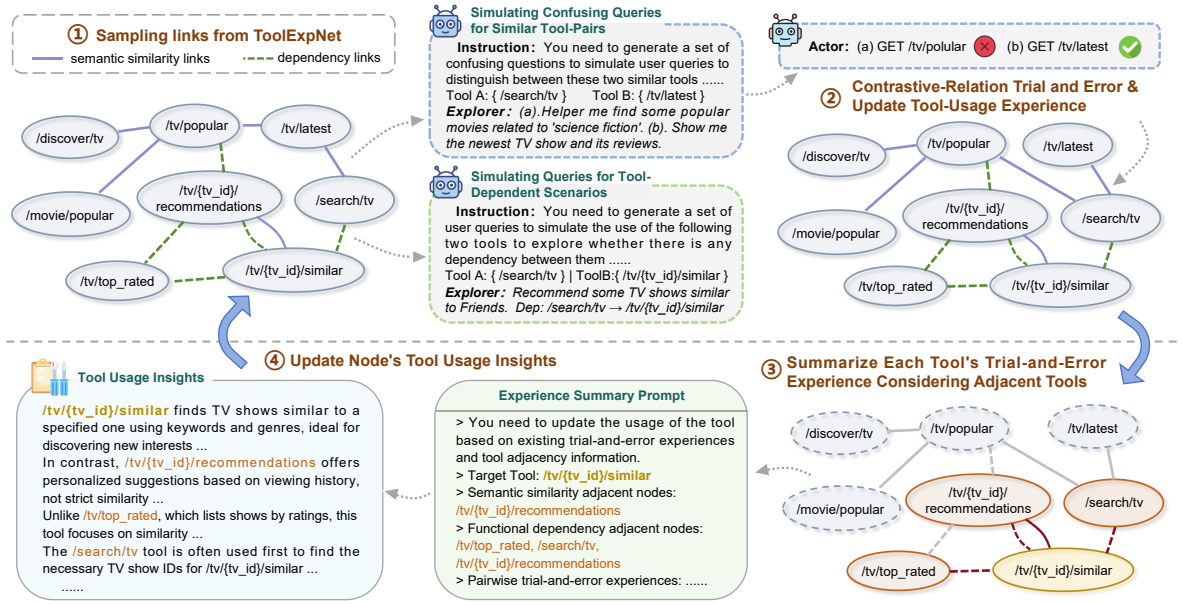

Figure 2: The ToolExpNet framework enhances tool usage insights by leveraging semantic similarity and dependency links to guide trial-and-error exploration. Contrastive-relation trial and error experiments simulate user queries, revealing functional differences and dependencies. These experiences update the tool's experiential network. Insights from these trials update node usage profiles, highlighting functional differences and interdependencies. This structured approach optimizes tool usage through comprehensive relational understanding.

tionality extends or depends on another. This often occurs when the input parameters of certain APIs are reliant on the outputs from the execution of other functions.

This dual-relational structure enables systematic modeling of both the selection challenges (via E_s) and compositional opportunities (via E_d) inherent in tool-augmented LLM systems. The explicit graph formulation facilitates structured reasoning about tool relationships while maintaining computational tractability.

2.2 ToolExpNet Initialization

Given a toolset Γ, we instantiate each tool as a node $v_i \in V$, initializing its functional insight φ_i directly from raw API documentation, even when such documentation is verbose or incomplete (Yuan et al., 2024; Qu et al., 2025a). Semantic edges (E_s) are formed between tool pairs whose documentation embeddings exceed a similarity threshold Φ. Dependency edges (E_d) are established between tools where the output data types of one tool overlap with the input data types of another tool.

3 Tool Learning Strategy

We suggest that the role of a tool within a toolkit is determined not only by its intrinsic properties but also by its toolset-context. Therefore, during the tool learning phase, we iterate the processes of contrastive-relation trial-and-error and tool insight refinement. This iterative process, similar to how humans learn through trial and error and then summarize their experiences, helps organize knowledge into structured cognition and memory. The prompts for this section are provided in Appendix A.

3.1 Contrastive-Relation Trial and Error

Prior studies (Shinn et al., 2024; Anokhin et al., 2024; Zhao et al., 2024a) have demonstrated the effectiveness of LLMs in learning through trial-and-error experiences. Experience learning serves as a plug-and-play approach, requiring no explicit gradient updates, making it compatible with closed-source LLMs. However, existing work typically focuses on self-exploration with single tools, lacking structured preservation of cross-tool usage patterns and inter-tool relationships.

To address this, we propose a Pairwise Exploration framework to capture LLMs' cross-tool operational knowledge. Each self-exploration iteration generates simulated user queries and golden solutions for targeted tool pairs, enabling systematic trial-and-error learning.

Explorer ($\mathcal{H}$): During each iteration, the explorer samples a subset of edges (maximum

max_try) from E_s and E_d with initial sampling weights $w^0(e_{ij}) = 1$. For edge e_{ij} in iteration t, the sampling probability is proportional to $w^t(e_{ij})$. For semantic similarity edges (E_s), $\mathcal{H}$ generates contrastive user queries emphasizing functional distinctions between tools v_i and v_j. For dependency edges (E_d), it simulates queries requiring sequential tool invocation while pruning spurious dependencies. This process outputs query-label pairs $(Q, L) = \mathcal{H}(p, v_i, v_j)$, where p denotes task-specific prompts. Here, Q represents the simulated queries, and L denotes the corresponding labels.

Actor ($\mathcal{A}$): The actor attempts to answer queries using tools Γ, producing responses $A = \mathcal{A}(p, Q, \Gamma)$. Execution traces and tool selection outcomes are logged into edge-specific experience pools. The error rate $1 - \text{ACC}(A, L)$ updates the sampling weight $w^{t+1}(e_{ij})$, prioritizing challenging tool pairs in subsequent iterations. Particularly, if $\mathcal{A}$ confuses two tools without existing links, a new semantic similarity link is added. Conversely, if tools linked by E_d show no actual dependencies during execution, the edge is removed.

3.2 Tool Insight Refinement

To systematically distill cross-tool operational knowledge, we propose an experience aggregation mechanism inspired by graph-structured message passing. For each tool node $v_i = (e_i, \varphi_i^t)$, we gather its local context from semantic neighbors $V_s = \{v_j | e_{ij} \in E_s\}$ and dependency neighbors $V_d = \{v_k | e_{ik} \in E_d\}$, along with their interaction histories. This contextualized experience is processed through LLM-based reflection to update functional insights φ_i^{t+1}.

Formally, the insight refinement process operates as:

$$\varphi_i^{t+1} = \text{Reflect}\Big(p, \underbrace{\{\varphi_j^t | v_j \in V_s \cup V_d\}}_{\text{Neighbor informations}}, \underbrace{\{(Q, A, L) | e_{ij} \in E_s \cup E_d\}}_{\text{Relevant trial experience}}\Big)$$

where p is a Chain-of-Thought prompt guiding the LLM to: (1) Identify *capability boundaries* by contrasting v_i with V_s, analyzing failure/success cases in E_s edges to clarify functional distinctions. (2) Discover *compositional patterns* by examining V_d relationships, synthesizing multi-tool workflows from E_d execution traces.

This graph-aware reflection enables dynamic evolution of tool understanding without model retraining. The updated φ_i^{t+1} is subsequently used as additional contextual information to inform the LLM's tool selection.

4 Experimental Setup

4.1 Datasets and Evaluation Metrics

Datasets. We conducted experiments on two widely-used benchmarks: RestBench (Song et al., 2023) and ToolBench (Qin et al., 2024b), across three scenarios. RestBench comprises two real-world scenarios with manually curated high-quality data. It includes TMDB, featuring 54 movie-related APIs, and Spotify, with 40 music-related APIs. ToolBench is a dataset collected from the RapidAPI and BMTools, containing over 16,000 real APIs spanning multiple categories. Due to budget constraints, we focused on the most challenging subset of ToolBench, I3-Instruction, which involves complex user requests requiring multiple tools from different categories.

Evaluation metrics. Following Song et al. (2023); Yuan et al. (2024); Qu et al. (2025a); Shi et al. (2024b), we utilized two common metrics: (1) Correct Path Rate (CP%), which measures the proportion of instances where the model-generated sequence of tool calls includes the golden tool path as a subsequence, to assess the accuracy of the model's tool invocation. (2) Win Rate (WR%): This metric evaluates the win rate of tool invocation sequences and planning processes generated by different methods compared to ReAct. The assessment is conducted through pairwise comparisons using a ChatGPT-based judger.

4.2 Baselines

We primarily compare our method with well-established baselines, includint:(1) ReAct (Yao et al., 2022b), which integrates CoT reasoning with action selection. It uses feedback to generate subsequent actions.(2) Easytool (Yuan et al., 2024), which addresses issues of inconsistency, redundancy, and incompleteness in real-world tool documentation. It rewrites documents with ChatGPT and incorporates guidelines. This enhances the Large Language Models' understanding of tool functionalities and parameter requirements.(3) DRAFT (Qu et al., 2025a), a trial-and-error-based approach that analyzes feedback from LLMs' interactions with external tools via three stages: experience collection, learning from experience, and document rewriting. This method dynamically re-

fines tool documentation to promote a deeper understanding and more effective utilization of tools by LLMs.

4.3 Implementation Details

We selected several leading large language models to validate the applicability of our method. These include the larger-scale GPT-4o and LLaMA3-70B, as well as the smaller-scale Qwen2.5-7B. For the initialization of ToolExpNet, we set the similarity threshold $\Phi = 0.8$. The temperature of all models are set to 0.

5 Results and Analysis

5.1 Overall Performance

We present our experimental results in Table 1. Our framework generally outperforms existing baselines across various real-world API scenarios. Specifically, it achieves superior performance on the CP and WR metrics compared to trial-and-error-based methods and document-driven tool learning approaches. This indicates better accuracy in tool selection and more effective multi-tool planning. Furthermore, our framework demonstrates robust adaptability to different foundation LLMs. Even when tested with a smaller model, Qwen2.5-7B, which has relatively limited tool comprehension capabilities, it consistently delivers performance improvements. These improvements validate the effectiveness of our model and suggest that our tool-learning methods, which summarize the distinctions and connections between tools, could be more effective in enhancing an LLM's understanding of tool capabilities.

5.2 Error Analysis for Tool Selection

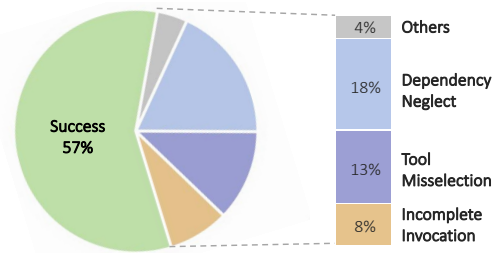

Figure 3: Statistics of Different Types of Errors in the ReAct Framework Based on GPT-4o on the TMDB Dataset.

We meticulously annotated and analyzed the failure cases of the ReAct framework based on GPT-4o in the TMDB task. These failures are categorized into four main types: (1)**Incomplete Invocation**: Missing critical tool calls due to overlooked user intents or flawed task planning. (2)**Dependency Neglect**: Ignoring dependencies between function calls, leading to errors or parameter hallucinations. (3)**Tool Misselection**: Selecting incorrect similar tools due to ambiguous documentation or overlapping functionalities. (4)**Others**: Miscellaneous errors, such as failures in instruction adherence or incorrect invocation formats.The statistics of these failures are illustrated in Figure 3.

We also observed that with the improvement in foundation model capabilities, the proportion of failures due to instruction adherence or tool hallucination has improved compared to previous observations (Shi et al., 2024a; Song et al., 2023; Wang et al., 2024a). However, a substantial portion of tool selection failures still stemmed from neglecting dependencies or being confused by ambiguous intents and similar tool documentation, with these two error types accounting for 72.09% of failures on TMDB. Appendix B provides examples of these error types.

This finding suggests that during the tool learning phase, LLMs should place greater emphasis on understanding the relationships and distinctions between tools to enhance comprehension of cross-tool dependencies and similarities.

5.3 Why ToolExpNet Works

In Section 5.2, we summarize two common error types in LLM tool usage. In this section, we explain how ToolExpNet effectively addresses these issues to achieve optimal outcomes.

While methods such as experience-based memory (Zhao et al., 2024a; Wang et al., 2024a) or tool documentation rewriting (Hsieh et al., 2023; Yuan et al., 2024) have been shown to improve LLMs' ability in task planning and tool selection, previous studies often focus on summarizing trial-and-error processes for isolated tools. However, in most real-world complex scenarios, multiple tools must be invoked in a specific sequence, forming a unidirectional flow of information. Although revising documentation for individual tools can help LLMs better understand when and how to use a particular tool, it does not enhance their ability to distinguish between similar tools or plan dependencies among tools directly. This limitation often leads to two key errors: Dependency Neglect and Tool Misselection.

We adopt trial-and-error guided by two types of edges, E_s and E_d, to establish relationships be-

Model	Method	RestBench-TMDB		RestBench-Spotify		ToolBench	
		CP%	WR%	CP%	WR%	CP%	WR%
Llama3-70B	ReAct	72.00	50.00	49.12	50.00	41.00	50.00
	EasyTool	76.00	58.00	57.89	59.65	46.00	55.00
	DRAFT	86.00	59.00	66.67	63.16	**53.00**	**61.50**
	ToolExpNet (Ours)	**86.00**	**61.00**	**70.17**	**64.91**	51.00	60.00
GPT-4o	ReAct	57.00	50.00	50.87	50.00	37.00	50.00
	EasyTool	74.00	60.50	61.40	57.89	45.00	62.50
	DRAFT	86.00	63.00	70.17	64.91	51.00	65.00
	ToolExpNet (Ours)	**90.00**	**69.00**	**75.44**	**68.42**	**53.00**	**68.50**
Qwen-2.5-7B	ReAct	38.00	50.00	21.05	50.00	16.00	50.00
	EasyTool	49.00	**69.00**	29.82	66.67	24.00	65.00
	DRAFT	46.00	64.00	31.58	70.17	23.00	65.00
	ToolExpNet (Ours)	**49.00**	67.50	**38.60**	**77.19**	**29.00**	**69.50**

Table 1: Performance comparison of different methods across three datasets. CP% and WR% denote the Correct Path Rate and Win Rate, respectively. The best result for each LLM is highlighted in **bold**.

	Error Type	ReAct	ToolExpNet
TMDB	D.N.	0.17	0.03
	T.M.	0.13	0.04
Spotify	D.N.	0.30	0.09
	T.M.	0.14	0.08

Table 2: Proportion of two common failure types in total sample count across different datasets and methods. D.N. and T.M. denote Dependency Neglect and Tool Misselection, respectively.

Method	TMDB(ΔSL)	Spotify(ΔSL)
ReAct	+0.76	+0.53
EasyTool	+0.24	+0.25
DRAFT	+0.22	+0.37
ToolExpNet	**+0.17**	**+0.23**

Table 3: Comparison of ΔSolution Length (ΔSL) across different scenarios for various methods, representing the additional number of API calls relative to the golden solution.

tween similar and dependent tools. Table 2 demonstrates that our model significantly reduces the error rates in two common categories: Dependency Neglect and Tool Misselection. This indicates that ToolExpNet effectively optimizes these errors to enhance the tool-using capabilities of LLMs.

In the self-explore stage of tool learning, tools with similar or dependent functions are grouped together for targeted trial-and-error experiences. During the reflection and summary stages, these tools are jointly analyzed to identify subtle differences and explore functional extensions through combinations with other tools. This learning process helps extract insights from trial-and-error experiences, highlighting the distinctions and connections between tools. These insights are then injected into the LLM through in-context learning to guide its planning and tool selection processes.

Following RestGPT (Song et al., 2023), we adopt ΔSolution Length(ΔSL) to measure the mean number of additional API calls required to successfully execute an instruction:

$$\Delta\text{SL} = \frac{1}{N_s} \sum_{i=0}^{N} \left(L^i_{\text{real}} - L^i_{\text{gold}} \right) \cdot \mathbb{I}(i, \text{success})$$

where N_s is the number of successfully completed instructions, L^i_{real} and L^i_{gold} are the actual and gold-standard API call counts for the i-th instruction, and $\mathbb{I}(i, \text{success})$ is an indicator function that equals 1 if the i-th instruction is successfully completed, otherwise 0.

We evaluate our approach on two real-world datasets, TMDB and Spotify. As shown in Table 3, ToolExpNet outperforms existing baselines on the ΔSL metric. This demonstrates that the insights generated by our method enhance its planning and tool selection processes, reducing unnecessary API calls caused by dependency neglect and subsequent backtracking. Detailed examples of this behavior

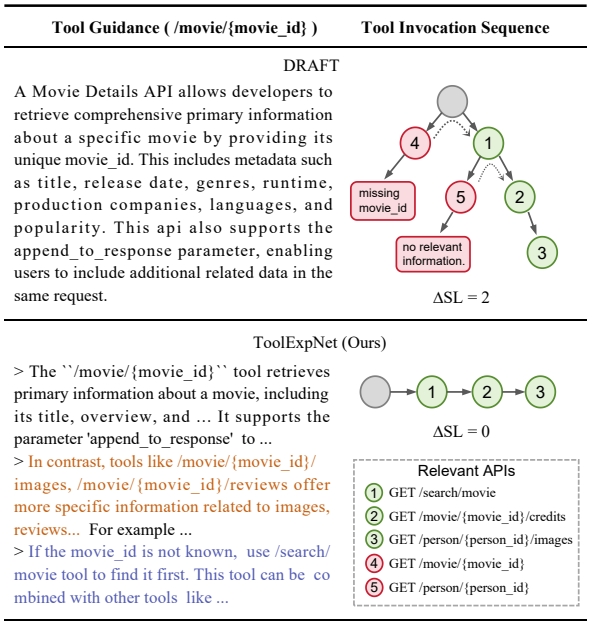

Figure 4: Case Study: This figure compares the tool guidance and performance of DRAFT and ToolExpNet in solving the query *"What does the lead actor of Titanic look like?"*. DRAFT, lacking dependency modeling, results in backtracking and an increased sequence length ($\Delta\text{SL} = 2$). In contrast, ToolExpNet's tool guidance provides a detailed description of semantic similarity tools and dependency tools to efficiently plan the tool sequence, avoiding unnecessary steps and achieving an optimal sequence length ($\Delta\text{SL} = 0$).

are provided in Appendix B.

Figure 4 shows a concrete example. It compares the tool-calling process guided by our method's tool insights with the process using DRAFT's tool documentation as context. Specifically, we compare the tool guidance for the same API endpoint *movie/{movie_id}* revised by DRAFT and ToolExpNet, as well as their performance on a given user query. The case demonstrates that our model, by summarizing the distinctions and connections between tools during the tool learning phase, achieves better task planning and tool selection.

5.4 Ablation Study

We conducted ablation studies on TMDB and Spotify, to evaluate the impact of different components in ToolExpNet . Specifically, we assessed the contributions of semantic similarity links E_s, functional dependency links E_d, and the contrastive-relation trial-and-error phase to the overall performance. The results in Table 4 show that removing any of these components leads to a performance drop. To further understand the role of these components, we analyzed how the tool usage insights generated under different settings influence the results.

Model	TMDB(CP)	Spotify(CP)
ToolExpNet	90.00	75.44
w/o E_s	85.00	70.17
w/o E_d	83.00	63.16
w/o Trial	82.00	70.17

Table 4: Ablation Study Results on TMDB and Spotify.

Semantic similarity links (E_s) enhance the LLM's capacity to differentiate between similar tools. A comparison of experiments with and without E_s reveals that the absence of E_s leads to a higher rate of Tool Misselection errors (4% $\rightarrow$ 10%). When E_s is removed from ToolExpNet , the LLM, during the reflection phase, can only consider tools dependency relationships. It also cannot leverage the error experiences where confusion occurred between two similar tools. This hinders the reflection on subtle differences between similar tools, making the LLM more prone to interference from similar tool documentation and more likely to select the wrong tool during the tool selection phase.

Functional dependency links E_d improve the efficiency of tool planning. The use of certain tools often depends on the results obtained from other tools. This dependency may stem from the inherent nature of the tools (e.g., parameter filling related to IDs) or the task logic implied by the user's intent. When E_d is removed, it leads to a higher rate of Dependency Neglect errors (3% $\rightarrow$ 12%). E_d enables the LLM to perform more effective planning before executing a task. As shown in Table 3 and the examples (Figure 4), E_d encourages the LLM to plan dependencies before invoking the target tool. A smaller ΔSL indicates a more efficient tool invocation process.

5.5 Further Analysis

Furthermore, we discuss the impact of different learning iterrations in the tool learning strategy on final performance. We conducted experiments using the models GPT-4o and Qwen2.5-7B-Instruct on the I3 subset of ToolBench. Each iteration allows a maximum sample size of 100, with all processes using greedy decoding. As illustrated in Figure 5, performance improves with more iterations and saturates around the third iteration.

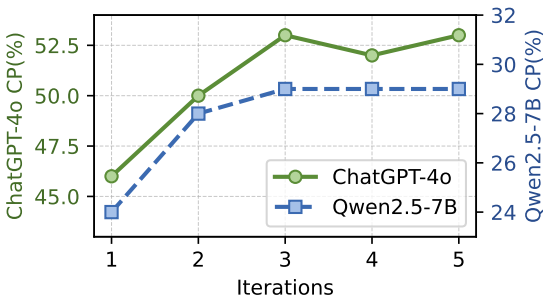

Figure 5: Performance of GPT-4o and Qwen2.5-7B on the I3 subset of ToolBench across different iterations.

6 Related Work

6.1 LLM Tool learning

Recent studies have demonstrated that large language models can significantly enhance their capabilities and tackle complex problems by leveraging external tools (Qu et al., 2025b; Shen et al., 2023; Qin et al., 2024b). Specifically, with the assistance of external tools, LLMs can acquire up-to-date information (Schick et al., 2023; Komeili et al., 2022; Gou et al., 2024), enhance their expertise (Inaba et al., 2023; Bran et al., 2024), and automate various tasks (Schick et al., 2023; Yao et al., 2022a; Zhuang et al., 2023). Existing methods be broadly categorized into two types: tuning-based and tuning-free (Qu et al., 2025b). Tuning-based methods involve further training LLMs on tool-related datasets to improve their tool usage capabilities (Liu et al., 2024a; Yang et al., 2023a; Hao et al., 2023; Patil et al., 2024). However, these methods are typically applicable only to open-source models and require substantial computational resources. In contrast, non-fine-tuning methods rely on the context learning ability of LLMs by providing tool documentation or a small number of usage examples, enabling the LLMs to understand how to use the tools (Wei et al., 2022; Hsieh et al., 2023; Qu et al., 2025a; Zhao et al., 2024a). These methods offer greater flexibility but are prone to errors in tool selection and parameter filling due to insufficient tool understanding (Shi et al., 2024a; Qu et al., 2025a). In this paper, we propose a novel approach that organizes tools and trial-and-error experiences into a network structure to facilitate more comprehensive tool understanding by LLMs.

6.2 Experience Enhenced LLM

Large Language Models face significant challenges in multi-tool calling tasks (Qu et al., 2025b; Anokhin et al., 2024). To enhance the performance of LLMs in complex real-world tasks, researchers are exploring how to enable LLMs to learn from their own experiences and thereby strengthen their tool-calling capabilities (Shinn et al., 2024; Zhao et al., 2024a; Wang et al., 2024a). For instance, Reflexion (Shinn et al., 2024) allows LLMs to reflect on their actions after task completion, identify the causes of failures, and improve subsequent attempts. ExpeL (Zhao et al., 2024a) enables LLMs to gather experiences through trial and error across multiple tasks, extract lessons from both successes and failures, and use these insights to optimize decision-making in subsequent tasks. Wang et al. (2024a) enhance LLMs' understanding of tools by incorporating trial and error, imagination, and memory mechanisms. Other methods often involve summarizing and updating tool documentation from trial experiences, transforming ambiguous, redundant, or incomplete tool documentation into more structured tool memories with model insights (Yuan et al., 2024; Qu et al., 2025a; Wu et al., 2024). Such approaches exhibit enhanced adaptability to new tools. However, prior work has predominantly focused on single-tool trial-and-error processes. In real-world multi-tool task scenarios, LLMs are required to accurately select and execute tools in sequence from a pool of interdependent and potentially confusing tools (Lu et al., 2023; Li et al., 2023; Gao et al., 2024). This raises higher demands on the LLM's ability to comprehend cross-tool interactions. To address this, we explicitly model two common types of tool relationships, namely dependency and similarity, to enhance LLMs' comprehensive understanding of cross-tool utilization in real-world scenarios.

7 Conclusion

In this paper, we propose ToolExpNet, a novel framework that organizes tool usage insights and trial-and-error experiences into a network based on semantic similarity and dependency relations, addressing the limitations of existing methods that focus on isolated tools learning. Experimental results on various foundation models and real-world datasets demonstrate that ToolExpNet outperforms existing methods, providing a comprehensive understanding of tool usage and improving multi-tool invocation accuracy.

Limitations

Although our method significantly reduces the number of tokens in tool usage guidelines compared to redundant original documents (Yuan et al., 2024; Qu et al., 2025a), it results in a larger token length than single-tool-focused trial-and-error and document rewriting methods (e.g., +42.86% compared to DRAFT (Qu et al., 2025a) using the Qwen2.5-7B-Instruct tokenizer). This increase is due to the detailed tool distinctions and dependency information that our method incorporates to enhance tool selection and invocation accuracy. While these enhancements improve tool invocation outcomes, the larger context length may pose challenges for Large Language Models with limited context windows. Future work will focus on optimizing the generation of tool guidelines to achieve higher information density and exploring the use of rewritten tool guidelines in the tool retrieval phase (Qu et al., 2025b) to improve efficiency. These advancements aim to balance the trade-off between detailed tool descriptions and the practical constraints of large language models, ultimately enhancing the applicability and performance of our method in real-world scenarios.

References

Petr Anokhin, Nikita Semenov, Artyom Sorokin, Dmitry Evseev, Mikhail Burtsev, and Evgeny Burnaev. 2024. Arigraph: Learning knowledge graph world models with episodic memory for llm agents. *Preprint*, arXiv:2407.04363.

Lawrence W Barsalou. 2014. *Cognitive psychology: An overview for cognitive scientists*. Psychology Press.

Andres M. Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D. White, and Philippe Schwaller. 2024. Augmenting large language models with chemistry tools. *Nat. Mac. Intell.*, 6(5):525–535.

Yanfei Chen, Jinsung Yoon, Devendra Singh Sachan, Qingze Wang, Vincent Cohen-Addad, Mohammad-Hossein Bateni, Chen-Yu Lee, and Tomas Pfister. 2024. Re-invoke: Tool invocation rewriting for zero-shot tool retrieval. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pages 4705–4726. Association for Computational Linguistics.

Shen Gao, Zhengliang Shi, Minghang Zhu, Bowen Fang, Xin Xin, Pengjie Ren, Zhumin Chen, Jun Ma, and Zhaochun Ren. 2024. Confucius: Iterative tool learning from introspection feedback by easy-to-difficult curriculum. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18030–18038.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2024. CRITIC: large language models can self-correct with tool-interactive critiquing. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Yu Gu, Yiheng Shu, Hao Yu, Xiao Liu, Yuxiao Dong, Jie Tang, Jayanth Srinivasa, Hugo Latapie, and Yu Su. 2024. Middleware for llms: Tools are instrumental for language agents in complex environments. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 7646–7663. Association for Computational Linguistics.

Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. 2023. Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Joy He-Yueya, Gabriel Poesia, Rose E Wang, and Noah D Goodman. 2023. Solving math word problems by combining language models with symbolic solvers. *arXiv preprint arXiv:2304.09102*.

Cheng-Yu Hsieh, Si-An Chen, Chun-Liang Li, Yasuhisa Fujii, Alexander Ratner, Chen-Yu Lee, Ranjay Krishna, and Tomas Pfister. 2023. Tool documentation enables zero-shot tool-usage with large language models. *Preprint*, arXiv:2308.00675.

Tatsuro Inaba, Hirokazu Kiyomaru, Fei Cheng, and Sadao Kurohashi. 2023. Multitool-cot: GPT-3 can use multiple external tools with chain of thought prompting. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 1522–1532. Association for Computational Linguistics.

Marek Kadlcík, Michal Stefánik, Ondrej Sotolár, and Vlastimil Martinek. 2023. Calc-x and calcformers: Empowering arithmetical chain-of-thought through interaction with symbolic systems. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 12101–12108. Association for Computational Linguistics.

Mojtaba Komeili, Kurt Shuster, and Jason Weston. 2022. Internet-augmented dialogue generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 8460–8478. Association for Computational Linguistics.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang,

and Yongbin Li. 2023. Api-bank: A comprehensive benchmark for tool-augmented llms. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 3102–3116. Association for Computational Linguistics.

Yaobo Liang, Chenfei Wu, Ting Song, Wenshan Wu, Yan Xia, Yu Liu, Yang Ou, Shuai Lu, Lei Ji, Shaoguang Mao, Yun Wang, Linjun Shou, Ming Gong, and Nan Duan. 2023. Taskmatrix.ai: Completing tasks by connecting foundation models with millions of apis. *Preprint*, arXiv:2303.16434.

Weiwen Liu, Xu Huang, Xingshan Zeng, Xinlong Hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, Zezhong Wang, Yuxian Wang, Wu Ning, Yutai Hou, Bin Wang, Chuhan Wu, Xinzhi Wang, Yong Liu, Yasheng Wang, Duyu Tang, Dandan Tu, Lifeng Shang, Xin Jiang, Ruiming Tang, Defu Lian, Qun Liu, and Enhong Chen. 2024a. Toolace: Winning the points of llm function calling. *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*.

Yanming Liu, Xinyue Peng, Yuwei Zhang, Jiannan Cao, Xuhong Zhang, Sheng Cheng, Xun Wang, Jianwei Yin, and Tianyu Du. 2024c. Tool-planner: Dynamic solution tree planning for large language model with tool clustering. *arXiv preprint arXiv:2406.03807*.

Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. 2023. Chameleon: Plug-and-play compositional reasoning with large language models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

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

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2024. Gorilla: Large language model connected with massive apis. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Shuofei Qiao, Honghao Gui, Chengfei Lv, Qianghuai Jia, Huajun Chen, and Ningyu Zhang. 2024. Making language models better tool learners with execution feedback. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 3550–3568. Association for Computational Linguistics.

Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Xuanhe Zhou, Yufei Huang, Chaojun Xiao, et al. 2024a. 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, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024b. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

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

Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-rong Wen. 2025b. Tool learning with large language models: a survey. *Frontiers of Computer Science*, 19(8).

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugginggpt: Solving AI tasks with chatgpt and its friends in hugging face. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Zhengliang Shi, Shen Gao, Xiuyi Chen, Yue Feng, Lingyong Yan, Haibo Shi, Dawei Yin, Zhumin Chen, Suzan Verberne, and Zhaochun Ren. 2024a. Chain of tools: Large language model is an automatic multi-tool learner. *arXiv preprint arXiv:2405.16533*.

Zhengliang Shi, Shen Gao, Xiuyi Chen, Yue Feng, Lingyong Yan, Haibo Shi, Dawei Yin, Pengjie Ren, Suzan Verberne, and Zhaochun Ren. 2024b. Learning to use tools via cooperative and interactive agents. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pages 10642–10657. Association for Computational Linguistics.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2024. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36.

Neil J Smelser, Paul B Baltes, et al. 2001. *International encyclopedia of the social & behavioral sciences*, volume 11. Elsevier Amsterdam.

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

Boshi Wang, Hao Fang, Jason Eisner, Benjamin Van Durme, and Yu Su. 2024a. Llms in the imaginarium: Tool learning through simulated trial and error. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 10583–10604. Association for Computational Linguistics.

Chenyu Wang, Weixin Luo, Qianyu Chen, Haonan Mai, Jindi Guo, Sixun Dong, Zhengxin Li, Lin Ma, Shenghua Gao, et al. 2024b. Tool-lmm: A large multi-modal model for tool agent learning. *arXiv preprint arXiv:2401.10727*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Shirley Wu, Shiyu Zhao, Qian Huang, Kexin Huang, Michihiro Yasunaga, Kaidi Cao, Vassilis N Ioannidis, Karthik Subbian, Jure Leskovec, and James Zou. 2024. Avatar: Optimizing llm agents for tool usage via contrastive reasoning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

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. 2023a. Gpt4tools: Teaching large language model to use tools via self-instruction. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. 2023b. Mm-react: Prompting chatgpt for multimodal reasoning and action. *Preprint*, arXiv:2303.11381.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022a. Webshop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022b. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Siyu Yuan, Kaitao Song, Jiangjie Chen, Xu Tan, Yongliang Shen, Kan Ren, Dongsheng Li, and Deqing Yang. 2024. EASYTOOL: Enhancing LLM-based agents with concise tool instruction. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*.

Yinger Zhang, Hui Cai, Xierui Song, Yicheng Chen, Rui Sun, and Jing Zheng. 2024. Reverse chain: A generic-rule for llms to master multi-api planning. In *Findings of the Association for Computational Linguistics: NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 302–325. Association for Computational Linguistics.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024a. Expel: LLM agents are experiential learners. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 19632–19642. AAAI Press.

Yuyue Zhao, Jiancan Wu, Xiang Wang, Wei Tang, Dingxian Wang, and Maarten De Rijke. 2024b. Let me do it for you: Towards llm empowered recommendation via tool learning. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1796–1806.

Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. 2023. Toolqa: A dataset for LLM question answering with external tools. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

A Prompts in Tool Learning Strategy

Here, we present the primary prompts used in the Tool Learning Strategy. These prompts are designed to guide the Explorer in generating simulated queries involving pairs of similar tools, creating queries that require multiple tools to solve a problem to explore potential dependencies, and facilitating the reflection process of summarizing tool usage experience to form concise tool guidance through a Chain of Thought process.

Simulating Confusing Queries for Semantic Similar Tool-Pairs

Instruction
You have an opportunity to further explore the subtle differences between the following two similar tools. You need to imagine scenarios where users might use these tools and produce simulated queries.
Tool Pair
Below are two similar tools and their descriptions:
1. {tool1_info}
2. {tool2_info}
Requirements
- Create a diverse set of user request examples (at least 5) for each tool to simulate its usage scenarios. Note that user requests typically do not mention the specific name of the tool, as the tool is abstracted from the user's perspective.
- The generated user requests should be as diverse as possible, covering different usage scenarios.
- There should be a certain level of complexity and potential for confusion between the two sets of simulated queries, highlighting the subtle differences between the two tools.
Output Format
The output user request examples should be in a JSON format, as in the example below:
{output_example}

Simulating Queries for Tool-Dependent Scenarios

Instruction
You have an opportunity to further explore the dependencies between the following two tools. You need to imagine scenarios where users might use these tools and generate simulated subtasks. Subtasks are part of a task planning decomposition.
Tool Pair
Below are two potentially dependent tools and their descriptions:
1. {tool1_info}
2. {tool2_info}
Requirements
- Generate a set of subtasks for these two tools to simulate their usage scenarios. These problems must be solved using both tools. Note that user requests typically do not mention the specific name of the tool, as the tool is abstracted from the user's perspective.
- The generated user requests should be as diverse as possible, covering different usage scenarios and input-output conditions.
- Additionally, you need to indicate whether there is a parameter dependency or functional expansion relationship between the two tools, and whether they must be called in sequence.
- Provide the specific parameters necessary for calling the API, especially if the parameter cannot be obtained from the result of any API.
Output Format
The user request examples should be a JSON, indicating the sequence of tool calls. For example: {output_format}

Process of Tool Guidance Refinement

Instruction
You need to leverage existing tools' trial-and-error experience and related information to optimize the guidance for the tool {tool_name}.
Background
- Tool documentation: {tool_doc}
- Potentially dependent adjacent tools: {composition_tools}
- Functionally similar adjacent tools: {similar_tools}
- Trial-and-error experience, each record includes the user question (question), your decision result (pred), and the golden path: {trial_exp}
CoT Guideline

1. Analyze the subtle differences between this tool and similar tools, determine under what circumstances to use which tool, and how to distinguish them.
2. Analyze how this tool forms more complex functions with dependent tools and whether it needs to depend on other tools before being called.
3. Summarize the typical scenarios in which this tool is used in user requests.
4. Rewrite and optimize the tool's description to make it more concise and clear. Remove redundancy, distinguish easily confused tools, explain dependencies on other tools, and possible use cases.
Output Format
Generate a JSON dictionary in the following format: {output_example}

B Error Type Examples

Table 5 shows specific cases corresponding to several types of errors in the tool selection phase of LLMs. Note that IDs and other parameters have been anonymized to protect privacy. To enhance readability, we have only presented the effective tool invocation paths, omitting the intermediate thought processes and retries caused by issues such as API call exceptions.

C Tool Insight Refinement Examples

In Table 6, we present several specific examples. During the tool insight refinement phase, the LLM identifies tools that share semantic similarities and dependencies with the tool that requires refinement. It also reviews related trial-and-error records to facilitate reflection and synthesis, leading to the creation of tool guidance.

D Ablation Studies

Table 7 shows the differences in the insights formed for tool usage after removing semantic dependency links and dependency links in the ablation experiments.

Removing a type of edge implies that in both the trial-and-error phase and the Tool Insight Refinement phase, the LLM cannot perceive the presence of other tools that have relevant associations with the given tool. This results in a fragmented tool usage insight, where only the retained type of connection can be perceived. Removing any type of edge leads to a decline in the overall performance of multi-tool invocation and an increase in the failure rate related to those specific error types. This indicates the effective role of modeling inter-tool relationships in facilitating tool learning.

Error Type: Dependency Neglect

Question: I'm watching the tv series The Last of Us and I need some more recommendations.
Golden Path: "GET /search/tv" → "GET /tv/{tv_id=1024}/recommendations"
LLM Path: "GET /tv/ {tv_id=3566} /recommendations" → "GET /tv/{tv_id=3566}"
Reason: The function of /search/tv is to search for a TV show, while /tv/{tv_id} retrieves a show's details by ID. To find the ID for "The Last of Us", the process should start with /search/tv. However, the LLM ignored this dependency, hallucinated a fictional tv_id, and produced an incorrect result.

Error Type: Tool Confusion

Question: Please recommend me some TV shows similar to Breaking Bad.
Golden Path: "GET /search/tv" → "GET /tv/{tv_id}/similar"
LLM Path: "GET /search/tv" → "GET /tv/{tv_id}/recommendations"
Reason: The failure occurred because the LLM confused two similar tools. The endpoint /tv/{tv_id}/similar is intended to find TV shows similar to a given show by analyzing keywords and genres, while /tv/{tv_id}/recommendations is used for getting recommendations based on the show's existing data. The LLM incorrectly used the recommendations endpoint instead of the similar endpoint.

Error Type: Incomplete Invocation

Question: When is the lead actor of The Mandalorian born?
Golden Path: "GET /search/tv" → "GET /tv/{tv_id}/credits" → "GET /person/{person_id}"
LLM Path: "GET /search/tv" → "GET /person/{person_id}"
Reason: The LLM overlooked the user's implicit need to find the lead actor's information. This required retrieving the TV show's credits to identify the lead actor's person_id. By skipping this step, the LLM failed to gather sufficient information to correctly answer the question.

Error Type: Others

Question: Add the first song of The Dark Side of the Moon in my playback queue.
Golden Path: "/search" → "/albums/{id}/tracks" → "/me/player/queue"
LLM Path: None
Reason: The LLM misunderstood the task, producing an incorrect and unparsable call format that cannot be executed.

Table 5: Examples of failure types for ReAct using GPT-4o as the backbone model on the TMDB and Spotify datasets. Note that for privacy reasons, all sensitive function parameters, such as specific IDs, have been anonymized. The parameters in the examples are fictional. Placeholders such as {id} represent specific parameters.

Table 6: Examples of tool trial-and-error experience summaries. The process of Chain-of-Thought (CoT) reflection and forming insightful tool guidance during the tool insight refinement stage.

Tool Name: /search/collection
> Semantic similarity: The /search/collection tool is used to search for collections based on a text query, which is useful when you dońt have specific collection IDs. In contrast, /collection/{collection_id} and /collection/{collection_id}/images are used when you already know the collection ID and need detailed information or images for that specific collection. Use /search/collection to find collections by keywords, and use the other two tools to get detailed information or images once you have the collection ID. > Dependency and composition: To perform a comprehensive search and retrieve detailed information about collections, you can first use /search/collection to find collections based on a query. Once you have the collection IDs from the search results, you can use /collection/{collection_id} to get detailed information about each collection and /collection/{collection_id}/images to get images associated with the collections. Typically, users can provide the query for /search/collection, but they will need to use the search results to obtain the collection IDs required for the other two tools. > Examples: 1. I want to find collections related to modern art. 2. I need detailed information about the collection with A. 3. Can you show me the images for the collection with A? > Tool guidance: The /search/collection tool allows you to search for collections using a text query. This tool is particularly useful when you do not have specific collection IDs and need to find collections based on keywords. The query parameter is required and should be URI encoded. You can also specify the page number for paginated results. Once you have the collection IDs from the search results, you can use /collection/{collection_id} to get detailed information about a specific collection and /collection/{collection_id}/images to retrieve images associated with that collection.
Tool Name: /movie/{movie_id}/similar
> Semantic similarity: The /movie/{movie_id}/similar tool is specifically designed to fetch a list of movies that are similar to a given movie based on keywords and genres. This is different from the /movie/{movie_id}/keywords tool, which only retrieves the keywords associated with a movie, and the /search/movie tool, which allows for a broader search of movies based on a text query and various filters. The /movie/{movie_id} tool provides comprehensive information about a specific movie, but does not focus on finding similar movies. > Dependency and composition: To use the /movie/{movie_id}/similar tool effectively, you may need to first retrieve the movie ID using the /search/movie tool if you do not already have it. Once you have the movie ID, you can use it to get similar movies. This tool does not directly depend on other tools, but combining it with /movie/{movie_id}/keywords can provide a deeper understanding of why certain movies are considered similar. > Examples: 1. Can you show me movies similar to Inception? (First, use /search/movie to find the movie ID for Inception, then use /movie/{movie_id}/similar with the obtained ID) 2. I want to find movies like The Matrix. What are some recommendations? (First, use /search/movie to get the ID for The Matrix, then use /movie/{movie_id}/similar with the obtained ID) > Tool guidance: Fetch a list of movies similar to a specified movie based on keywords and genres. This tool is distinct from the recommendation system and focuses on similarity in terms of content. To use this tool, you need the movie ID, which can be obtained using the /search/movie tool if not already known. The response includes a paginated list of similar movies. Example usage: "Find movies similar to The Godfather."

ToolExpNet w/o E_s

Tool Name:/person/{person_id}
Experience Insights: Retrieve primary details of a person using their ID. This tool supports the /append_to_response parameter for additional data. It can be used in conjunction with other tools like /search/person to find the person ID, and /person/{person_id}/tv_credits, /person/{person_id}/movie_credits, and /person/{person_id}/images for more detailed information. Recent updates include the addition of the /known_for_department field and support for translations.
Cases: The tool /person/{person_id} can be used in combination with other tools to provide comprehensive details about a person. For instance, to get a person's TV show credits, you can use /person/{person_id}/tv_credits. If you need to find a person's ID, you can use /search/person with a query parameter to search for the person by name. Once you have the person ID, you can use /person/{person_id} to get primary details, /person/{person_id}/images to get images, and /person/{person_id}/movie_credits to get movie credits.

ToolExpNet w/o E_d

Tool Name: /movie/{movie_id}/reviews
Experience Insights: Fetch user reviews for a specific movie using its movie_id. This tool allows you to retrieve detailed feedback and opinions from users about a particular movie. It is useful for understanding audience reception and sentiments. To specify which page of reviews to query, use the *page* parameter. This tool is distinct from /search/movie, which is used for finding movies based on search criteria.
Cases: The tool /movie/{movie_id}/reviews is specifically designed to fetch user reviews for a particular movie identified by its movie_id. In contrast, the /search/movie tool is used to search for movies based on various criteria such as title, release year, and other filters. While /search/movie helps in finding movies, /movie/{movie_id}/reviews provides detailed user feedback for a specific movie. Use /movie/{movie_id}/reviews when you need to gather opinions and reviews about a particular movie, and use /search/movie when you need to find movies that match certain search criteria.

Table 7: Examples comparing the tool usage experiences and insights formed by the LLM after removing the two different types of edges, E_s and E_d from ToolExpNet .