

# ACE-Router: Generalizing History-Aware Routing from MCP Tools to the Agent Web

Zhiyuan Yao<sup>1,\*</sup>, Zishan Xu<sup>2,\*</sup>, Yifu Guo<sup>4,\*</sup>, Zhiguang Han<sup>5</sup>, Cheng Yang<sup>6</sup>,  
Shuo Zhang, Weinan Zhang<sup>2,†</sup>, Xingshan Zeng<sup>3,†</sup>, Weiwen Liu<sup>2,†</sup>

<sup>1</sup>Zhejiang University, <sup>2</sup>Shanghai Jiao Tong University, <sup>3</sup>Huawei Technologies Co. Ltd,  
<sup>4</sup>Sun Yat-sen University, <sup>5</sup>Nanyang Technological University, <sup>6</sup>Hangzhou Dianzi University

## Abstract

With the rise of the Agent Web and Model Context Protocol (MCP), the agent ecosystem is evolving into an open collaborative network, exponentially increasing accessible tools. However, current architectures face severe scalability and generality bottlenecks. To address this, we propose ACE-Router, a pipeline for training history-aware routers to empower precise navigation in large-scale ecosystems. By leveraging a dependency-rich candidate Graph to synthesize multi-turn trajectories, we effectively train routers with dynamic context understanding to create the plug-and-play Light Routing Agent. Experiments on the real-world benchmarks MCP-Universe and MCP-Mark demonstrate superior performance. Notably, ACE-Router exhibits critical properties for the future Agent Web: it not only generalizes to multi-agent collaboration with minimal adaptation but also maintains exceptional robustness against noise and scales effectively to massive candidate spaces. These findings provide a strong empirical foundation for universal orchestration in open-ended ecosystems. Our code is available at <https://github.com/euyis1019/ACE-Router>.

## 1 Introduction

Recent advances in reasoning and tool utilization have transformed LLMs (Guo et al., 2025a; Achiam et al., 2023) into capable Agents (Tran et al., 2025; Sapkota et al., 2025; Fang et al., 2025; Chang et al., 2025). By invoking external tools, they transcend static parametric limits to tackle diverse real-world challenges (Yang et al., 2024b; Guo et al., 2025b; Li et al., 2025b; Xu et al., 2026). However, most existing systems are monolithic with hardcoded, predefined toolsets, which limit flexibility and prevent seamless integration of different tools and domains.

\*Equal contribution.

†Corresponding authors.

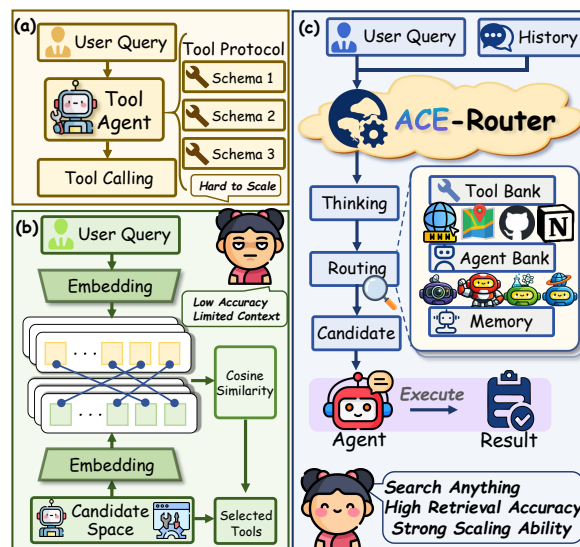


Figure 1: Comparison of ACE-Router with other existing paradigms. (a) **Static Injection**: Constrained by finite context windows and rigid schemas. (b) **Embedding-based Retrieval**: Limited by static semantic matching and lack of historical context awareness. (c) **ACE-Router (Ours)**: A robust router that leverages reasoning and interaction history to achieve high-accuracy retrieval within a massive candidate space.

To break these boundaries, the emerging Agent Web (Yang et al., 2025b) envisions an open ecosystem where agents act as autonomous nodes accessing a massive, expanding repository of resources. However, existing multi-agent systems, constrained by static orchestration, are ill-suited for this dynamic scale. To bridge this gap, the paradigm must shift toward "On-demand Teaming": host agents must dynamically discover and schedule optimal collaboration nodes based on real-time states (Lù et al., 2025; Petrova et al., 2025). Realizing this adaptive orchestration necessitates a robust *Router*, as illustrated in Figure 1(c), capable of navigating the vast search space to identify the most suitable tools, agents and so on.

Concretely, the Agent Web implies an unprece-

mented scale of heterogeneous tools and agents, calling for a unified interface to consistently discover and invoke such external resources. The Model Context Protocol (MCP) (Anthropic, 2024) is standardizing access to millions of tools. However, current "Static Injection" architectures (Shi et al., 2025; Hong et al., 2025; Lin et al., 2025), as illustrated in Figure 1 (a), face dual bottlenecks. *Scalability* is restricted by finite context windows, which cannot accommodate massive tool descriptions in a single pass. Meanwhile, *Generality* is undermined by rigid prompt structures, where hard-coded designs lack the flexibility to support dynamic collaboration across heterogeneous architectures.

To manage tool proliferation, retrieval-based tool selection is widely used (Gan and Sun, 2025), yet existing selectors typically rely on static embedding-based matching (Mo et al., 2025; Qin et al., 2023), as shown in Figure 1 (b). However, this approach faces three critical limitations: (1) It lacks fine-grained discriminability for functionally similar tools due to semantic overlap; (2) It typically ignores the multi-turn trajectory, omitting crucial state information like intermediate outcomes, historical performance, and tool correlations; (3) Even if history is incorporated, encoding long contexts into fixed-size vectors causes information compression, failing to resolve subtle distinctions in complex agent states. Consequently, this precludes the model from leveraging past interactions for informed, context-aware decisions.

To bridge these gaps, we propose **ACE-Router**, a pipeline for training high-performance, history-aware routers. Our approach begins with Graph-based Expansion, which employs self-evolutionary mutation to synthesize behaviorally diverse tools within a structured Candidate Graph, enabling the distinction of subtle functional nuances. Building on this, we implement Trajectory Synthesis by sampling tool subsets via random walks. These subsets drive a multi-agent framework to generate context-rich trajectories, yielding explicit supervision signals that align multi-turn histories with correct routing decisions. Finally, we introduce the Light Routing Agent, a plug-and-play module that operates through a minimal interface (i.e., Router Invocation and Execution tools). This abstraction decouples routing logic from specific tool definitions, improving generality and enabling seamless adaptation across diverse architectures.

Experimental results demonstrate that ACE-

Router achieves superior performance on real-world MCP benchmarks. Crucially, ACE-Router unveils *Cross-domain Transferability*, generalizing to multi-agent tasks with minimal adaptation. Furthermore, it demonstrates *Robustness against Noise*, effectively filtering out irrelevant distractions and hard negatives within massive candidate spaces.

Overall, our contributions are summarized as follows:

- We propose **ACE-Router**, a router training framework that integrates graph-based tool expansion and multi-agent trajectory synthesis. By rigorously aligning multi-turn history with routing decisions, this framework constructs high-quality supervision specifically tailored for router training.
- We train a history-aware router that effectively captures dynamic, multi-turn dependencies. This model transcends the limitations of static semantic matching by maintaining precise context awareness throughout complex interaction trajectories.
- We develop the Light Routing Agent, a plug-and-play module designed for both tool and agent selection. Experimental results demonstrate its superior performance and robustness on MCP benchmarks and validate its seamless generalization from tool routing to multi-agent orchestration.

## 2 Related Work

### 2.1 Large-Scale Tool Learning

With the emergence of open protocols like MCP, the tool ecosystem is transitioning from closed to open systems. This paradigm shift has spurred the development of diverse MCP-specific evaluation benchmarks, ranging from large-scale coverage and multi-domain diversity (Fan et al., 2025; Luo et al., 2025; Mo et al., 2025) to real-world service integration (Wu et al., 2025; Guo et al., 2025c; Mo et al., 2025) and multi-dimensional frameworks assessing accuracy, efficiency, and latency (Gao et al., 2025; Luo et al., 2025). However, existing tool learning methods face two fundamental bottlenecks under this new paradigm.

**Scalability Bottlenecks.** Mainstream approaches adopt two architectural patterns. Hard-coding predefined tool sets into system prompts

leads to context saturation as tool numbers grow (Yao et al., 2023; Schick et al., 2023; Shen et al., 2023; Yang et al., 2024b). Alternatively, "retrieve-inject" pipelines filter tool subsets through retrieval before context injection (Patil et al., 2023; Qin et al., 2023; Zhang et al., 2024; Song et al., 2023; Lumer et al., 2025), though processing numerous schemas still incurs substantial context overhead.

**Training Data Gaps.** Current datasets (Li et al., 2023; Qin et al., 2023) operate at scales below MCP levels. Traditional synthesis methods (Lu et al., 2024; Patil et al., 2025; Chen et al., 2024) generate isolated query-tool pairs from flat collections, lacking multi-step reasoning patterns and inter-tool dependencies.

## 2.2 Dynamic Tool Routing

**Static Semantic Matching.** Tool selection typically relies on embedding-based similarity between user queries and tool descriptions (Patil et al., 2023; Song et al., 2023), where single-turn matching determines relevance without considering multi-turn dynamics.

**Context-Aware Approaches.** Recent methods incorporate execution context through two paradigms: statistics-driven approaches match via probabilistic patterns in usage history (Yang et al., 2024a; Patel et al., 2025), while graph-based methods model dependencies through neural networks or search algorithms (Zhang et al., 2024; Zhuang et al., 2023; Li et al., 2025a). These approaches compress dialogue information into fixed representations without directly leveraging raw conversation history for routing decisions in evolving multi-agent scenarios.

# 3 Method

## 3.1 Overview

Figure 2 illustrates the overall framework of ACE-Router. The pipeline operates in a sequential manner: first, we perform a Graph-based Extension with Self-Evolutionary Mutation on the initial candidate set; subsequently, we leverage a multi-agent system to synthesize interaction trajectories, from which we derive supervision signals for the router; finally, we deploy the Light Routing Agent, which serves as the practical implementation of the router trained via ACE-Router, designed to seamlessly integrate into existing agent pipelines. We provide a detailed elaboration of these components in the

following sections.

## 3.2 Problem Formulation.

We formulate routing as the problem of selecting an appropriate candidate from a given candidate space  $\mathcal{C}$ , conditioned on the current user query  $Q$  and the dialogue history  $H$ .

At each routing step, the candidate space  $\mathcal{C}$  is specified beforehand and is drawn from a predefined set of candidate types, such as a tool set  $\mathcal{T}$  or an agent set  $\mathcal{A}$ :

$$\mathcal{C} \in \{\mathcal{T}, \mathcal{A}, \dots\}. \quad (1)$$

Formally, each candidate  $c \in \mathcal{C}$  is associated with a structured specification  $\phi(c)$ . For tools,  $\phi(c)$  includes the tool description and schema, while for agents,  $\phi(c)$  corresponds to the agent profile and its available tool, characterizing the agent’s specific capability scope.

Given  $(Q, H)$  and the specified candidate space  $\mathcal{C}$ , we train a parameterized router  $\pi_\theta$  to model a conditional distribution over candidates within  $\mathcal{C}$ :

$$\pi_\theta(c \mid Q, H, \mathcal{C}), \quad c \in \mathcal{C}. \quad (2)$$

At inference time, the router selects the candidate with the highest posterior probability:

$$c^* = \arg \max_{c \in \mathcal{C}} \pi_\theta(c \mid Q, H, \mathcal{C}). \quad (3)$$

## 3.3 Candidate Graph-based Extension With Self-Evolutionary Mutation

The goal of the router is to select candidates that best match the current state from a set of semantically and functionally related options. To support this objective during trajectory synthesis, it is crucial to expose the router to candidates that are not only relevant to the current query, but also closely related in terms of functionality or dependency structure.

To effectively scale the candidate space and bolster the discriminative capability against semantically close candidates, we construct a candidate graph over the initial candidate set, where nodes correspond to candidates (e.g., tools or agents), and edges capture semantic similarity or functional dependencies between them. Building on this graph, we further enrich the candidate space through a self-evolutionary mutation process that synthesizes new candidate variants from existing ones, inspired by methods such as Self-Evolve (Novikov et al., 2025).

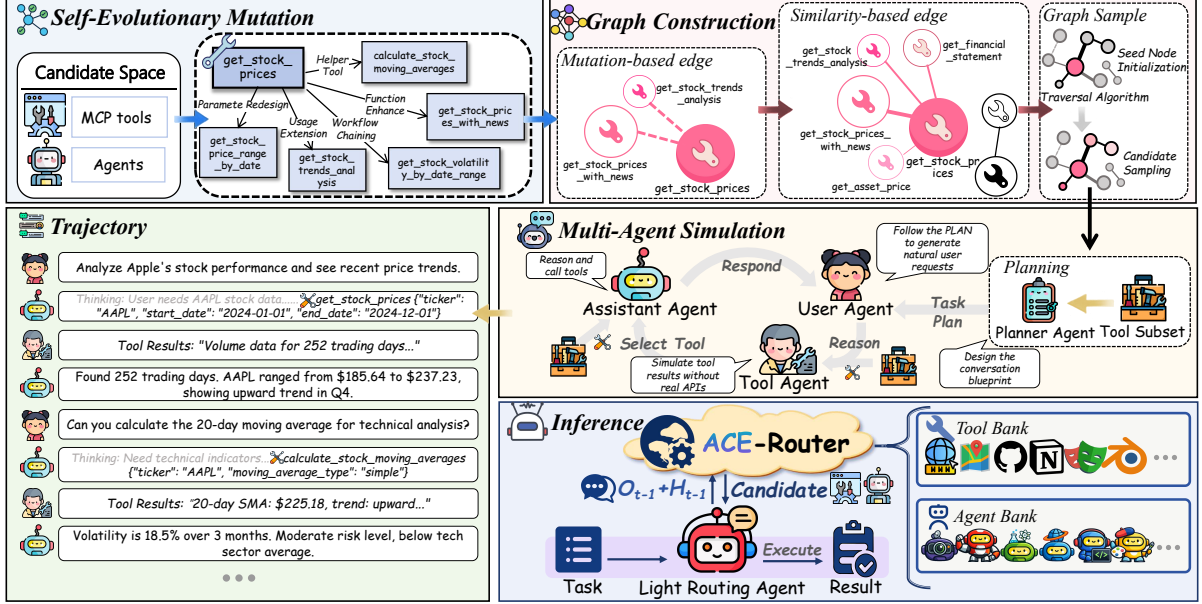


Figure 2: The overall framework of ACE-Router. It consists of three key stages: (1) **Self-evolutionary Graph Construction**, which expands and structures the candidate space via mutation and relation modeling; (2) **Multi-Agent Simulation**, which synthesizes interaction trajectories to extract history-aware supervision signals; and (3) **The Light Routing Agent**, designed to seamlessly integrate the trained router into the inference pipeline.

### 3.3.1 Graph Construction

Given a candidate set  $\mathcal{C} = \{c_1, c_2, \dots, c_N\}$ , we first derive a vector representation for each candidate by encoding its structured specification  $\phi(c)$ . Using a pretrained embedding model  $\mathcal{E}$ , the embedding vector  $\mathbf{h}_i \in \mathcal{R}^d$  for candidate  $c_i$  is computed as  $\mathbf{h}_i = \mathcal{E}(\phi(c_i))$ . For instance, within the tool subset  $\mathcal{T} \subseteq \mathcal{C}$ ,  $\phi(c_i)$  serializes the tool’s textual description and input schema.

To capture latent relationships, we define the semantic similarity between any pair of candidates  $c_i$  and  $c_j$  as the cosine similarity of their embeddings:

$$\text{sim}(c_i, c_j) = \cos(\mathbf{h}_i, \mathbf{h}_j) = \frac{\mathbf{h}_i \cdot \mathbf{h}_j}{\|\mathbf{h}_i\| \|\mathbf{h}_j\|}. \quad (4)$$

We construct an undirected edge between nodes  $c_i$  and  $c_j$  if their similarity exceeds a predefined threshold  $\tau$  (empirically set to 0.82), i.e.,  $\text{sim}(c_i, c_j) > \tau$ . This procedure yields an initial connectivity graph  $\mathcal{G} = (\mathcal{C}, \mathcal{E}_{\text{sim}})$ , which captures the local semantic neighborhoods among candidates.

### 3.3.2 Self-Evolutionary Mutation

To mitigate overfitting caused by an overly narrow candidate space  $\mathcal{C}$ , we introduce a novel Self-Evolutionary strategy to construct new candidate elements. The key idea is to iteratively expand

the candidate graph with controlled mutations that preserve semantic relevance to existing candidates.

Specifically, we define a set of mutation operators  $\mathcal{M}$  for tool, which include *Function Enhancement*, *Parameter Mutation*, *Workflow Chaining*, *Helper Operation*, and *Usage Extension*. Detailed specifications of these operators, along with the agent mutation strategies, are provided in Appendix C. At each iteration, we randomly sample an existing candidate  $c \in \mathcal{C}$  and a mutation operator  $m \in \mathcal{M}$ . We then prompt a large language model (LLM) to synthesize a new candidate  $c' = m(c)$  based on the selected mutation.

The newly generated candidate  $c'$  is added as a new node to the candidate graph, and an edge is created between  $c'$  and the original candidate  $c$  to explicitly encode their mutation relationship. This Self-Evolutionary process progressively enriches the candidate space while maintaining local structural consistency in the graph.

### 3.4 History-Aware Supervision for Router Training

We begin by sampling candidate subsets from the constructed candidate graph via a random walk-based traversal, aiming to select candidates that exhibit semantic similarity or functional dependencies. Specifically, we initiate the process from a set of seed nodes and perform a DFS-style traversal

to visit neighboring nodes. The collected nodes form a sampled subset, ensuring local coherence in terms of semantics and functionality.

Inspired by prior work on tool-oriented trajectory synthesis (Liu et al., 2024; Wang et al., 2025), we further synthesize a task description and a coarse-grained execution plan conditioned on the sampled subset. Based on this plan, we generate multi-turn dialogue trajectories through role-based simulation. Formally, each trajectory is represented as a sequence:

$$\mathcal{P} = (o_0, a_0, o_1, a_1, \dots, o_n, a_n), \quad (5)$$

where  $o_0$  denotes the initial user query, and  $o_t$  represents the user feedback or environment response following the assistant action  $a_t$  (including invocation results of candidate elements). Crucially, both the trajectory generation and the simulated responses are produced by Large Language Models. This environment-free simulation design enables scalable and flexible synthesis without requiring access to real execution APIs, thereby facilitating the efficient expansion of training data.

Upon acquiring the synthesized trajectories, we proceed to extract supervision signals for router training. Specifically, we identify time steps  $t$  where the assistant action  $a_t$  involves invoking a specific candidate  $c \in \mathcal{C}$ , which we extract as the ground-truth label. To construct the corresponding input, we designate the preceding interaction sequence  $(o_0, \dots, a_{t-1})$  as the history context  $H$ , while explicitly treating the immediate observation  $o_{t-1}$  as the current query. This strategy effectively transforms complex, multi-step trajectories into a large-scale dataset of history-aware routing instances. Depending on the definition of the candidate space  $\mathcal{C}$ , these supervision signals can be universally applied to train various router types, including both Tool Routers and Agent Routers. As validated in Section 4.5, this history-aware formulation yields significant accuracy gains over stateless baselines.

### 3.5 Light Routing Agent

To seamlessly integrate the trained router into existing agent workflows and evaluation benchmarks, we design a lightweight routing agent, termed the Light Routing Agent (LRA), which decouples routing decisions from concrete task execution. Unlike conventional agents that tightly couple planning, tool selection, and execution logic, LRA serves

solely as a minimal wrapper around the trained router.

Specifically, LRA is equipped with only two tools. The first is a router invocation tool, which queries the trained router based on the current dialogue history and contextual information to select the most appropriate candidate from a given candidate set. The second is an execution tool, responsible for invoking or executing the candidate returned by the router. With this design, the agent no longer needs to explicitly inject large candidate set information (e.g., tool descriptions or agent functionalities) into the context. Instead, it dynamically selects and dispatches the required operations at runtime via the router, thereby maintaining a lightweight agent structure while enabling efficient execution of diverse and complex tasks.

## 4 Experiment












### 4.1 Experiment Setup

#### 4.1.1 Dataset and model

Our initial tool bank consisted of 627 MCP tools collected from the MCP Universe (Luo et al., 2025) and the LiveMCP (Mo et al., 2025) benchmark. By applying the mutation operators described in the previous section, we expanded this initial set into 2,005 tools. For the toolgraph construction, we employed all-MiniLM-L6-v2 to generate semantic embeddings. Subsequently, leveraging GPT-4o for trajectory synthesis, we utilized this augmented tool bank to construct a comprehensive dataset of over 15,092 training samples for the tool router. Although trained on tool selection, the router captures transferable decision patterns, enabling generalization to agent routing tasks without additional training.

Our Tool Router is trained on top of Qwen3-8B (Yang et al., 2025a). We evaluate the proposed router against a diverse set of baseline methods, including the native Qwen3-8B model as well as several representative closed-source large language models, such as GPT-4o (Hurst et al., 2024), Claude-Sonnet-4 (Anthropic, 2025), Gemini-2.5-Pro (Comanici et al., 2025) and so on. In addition to model-based routing approaches, we also include embedding-based routing strategies as baseline methods, utilizing all-MiniLM-L6-v2 and text-embedding-3-large as the underlying encoders. These approaches select candidates by computing vector similarity between the query and candidates, and we consider multiple input settings, including

Table 1: **Accuracy (%) comparison on MCP-Universe and MCP-Mark benchmarks.** **Q** denotes methods using only the current query, while **Q+H** incorporates both the query and interaction history. For **MCP-Universe**, we evaluate six specific domains: Location Navigation (Loc.), Repository Management (Repo.), Financial Analysis (Fin.), 3D Designing (3D), Browser Automation (Browser), and Web Searching (Web). **MCP-Mark** assesses performance on specific real-world tool environments including Notion, GitHub, PostgreSQL, Playwright, and Filesystem. The best result is marked in **bold** and the second best result is underlined.

Methods	MCP-Universe						MCP-Mark						
	 Loc.	 Repo.	 Fin.	 3D	 Browser	 Web	Overall	 Notion	 GitHub	 PostgreSQL	 Playwright	 Filesystem	Overall
<i>Embedding-based Retrieval</i>													
Text-Emb-3-Large (Q)	46.11	38.62	62.71	31.58	28.74	16.00	40.95	20.00	10.00	70.00	10.00	20.00	26.00
Text-Emb-3-Large (Q+H)	45.81	14.48	50.85	26.32	34.48	14.00	35.20	20.00	0.00	60.00	10.00	10.00	20.00
all-MiniLM-L6-v2 (Q)	44.91	34.48	67.80	26.32	34.48	<u>20.00</u>	40.67	10.00	10.00	70.00	10.00	30.00	26.00
all-MiniLM-L6-v2 (Q+H)	52.10	33.10	66.10	28.95	37.93	12.00	43.62	20.00	10.00	80.00	20.00	30.00	32.00
<i>ReAct Agents</i>													
ReAct (Gemini-2.5-Pro)	42.81	43.45	66.10	55.26	27.59	16.00	41.80	20.00	20.00	<u>90.00</u>	20.00	50.00	40.00
<i>LLM-based Routers</i>													
Qwen3-8B	48.50	49.66	<u>69.49</u>	50.00	29.89	18.00	46.14	<u>30.00</u>	20.00	<u>90.00</u>	20.00	50.00	42.00
GLM-4.5	53.29	46.90	52.54	47.37	35.63	10.00	46.42	20.00	10.00	80.00	<u>30.00</u>	50.00	38.00
DeepSeek-V3.2	49.10	44.83	64.41	50.00	33.33	8.00	44.74	20.00	10.00	<b>100.00</b>	<u>30.00</u>	50.00	42.00
Claude-sonnet-4	53.59	43.45	61.02	55.26	42.53	16.00	48.25	<u>30.00</u>	<u>40.00</u>	<u>90.00</u>	20.00	40.00	44.00
Gemini-2.5-Pro	<u>54.19</u>	46.21	61.02	55.26	<u>47.13</u>	18.00	<u>49.79</u>	20.00	30.00	<b>100.00</b>	20.00	<u>60.00</u>	46.00
Gemini-2.5-flash	50.90	<u>50.34</u>	66.10	52.63	28.74	16.00	46.98	<u>30.00</u>	<b>50.00</b>	90.00	20.00	50.00	<u>48.00</u>
GPT-4.1	45.51	48.97	61.02	52.63	34.48	18.00	44.60	10.00	20.00	90.00	10.00	50.00	36.00
GPT-4o	47.90	48.97	62.71	<u>65.79</u>	44.83	12.00	47.41	20.00	30.00	90.00	10.00	40.00	38.00
<b>ACE-Router (Ours)</b>	<b>54.49</b>	<b>51.03</b>	<b>72.88</b>	<b>71.05</b>	<b>49.43</b>	<b>24.00</b>	<b>53.44</b>	<b>40.00</b>	<b>50.00</b>	<b>100.00</b>	<b>40.00</b>	<b>70.00</b>	<b>60.00</b>

using only the current query (query-only), incorporating historical context. We also include an LLM-driven ReAct (Yao et al., 2023) agent as a baseline for tool selection.

To ensure a fair comparison of routing capability across different models, we fix the downstream execution (reasoning) model to Gemini-2.5-Pro for all router-based methods.

#### 4.1.2 Benchmark and Evaluation

We conduct a systematic evaluation of the proposed router on several widely used MCP benchmarks, including MCP-Universe (Luo et al., 2025) and MCP-Mark (easy mode) (Wu et al., 2025).

To further evaluate the router’s generalization ability in cross-agent scenarios, we construct an evaluation setup tailored to the agent routing task. We systematically collect and normalize over 40 mainstream agents, unifying them into a consistent JSON format to form an initial Agent Bank as the candidate space. Based on the proposed Self-Evolutionary mutation mechanism and multi-agent trajectory synthesis strategy, we generate a total of 156 agent router test samples. All samples are utilized to assess the router’s generalization performance under unseen agent combinations and complex candidate spaces. For a detailed taxonomy of agent Self-Evolutionary mutation types and examples of the benchmark tasks, please refer to Appendix C and Appendix D.

#### 4.1.3 Implementation Details

Given resource constraints, We fine-tune the model using LoRA (Hu et al., 2022) applied to all linear layers with a rank of  $r = 8$ . Training runs for 3 epochs with a global batch size of 64, utilizing a learning rate of  $1 \times 10^{-4}$  with a cosine annealing schedule and a 0.1 warmup ratio. The maximum sequence length is set to 32,768 tokens in BF16 precision. For evaluation, we set the sampling temperature to 1 and report the average results over 5 independent runs (avg@5) to ensure stability.

#### 4.2 Main Result

As shown in Table 1, ACE-Router consistently outperforms all baseline methods, significantly enhancing the agent’s capability to solve tasks using MCP tools. Specifically, on the MCP-Universe benchmark, we achieve an overall performance of 53.44%, with the Financial Analysis domain reaching 72.88%. On MCP-Mark, our method attains 60.00%.

Crucially, our results demonstrate that the router-based paradigm significantly surpasses both Embedding-based retrieval and ReAct-based agents. Furthermore, a key finding is that our 8B-parameter specialized router outperforms massive generalist models, including GPT-4o (47.41% on MCP-Universe) and Gemini-2.5-Pro (49.79% on MCP-Universe). This highlights a critical limitation in generalist LLMs: despite their reasoning prowess, they struggle with the precise discrim-

ination required for tool selection. Collectively, these findings validate the effectiveness of the *light routing agent* design. Our results demonstrate that employing a specialized router represents a superior strategy for enabling efficient and reliable tool usage, thereby providing a robust foundation for the emerging open Agent Web.

### 4.3 Scalability and Robustness Analysis

To validate the router’s adaptability to realistic and challenging scenarios, we evaluate its performance under two distinct conditions: expanded tool spaces and noisy input environments.

**Scalability to Large-Scale Tool Spaces.** We first evaluate the router’s scalability by aggregating tools from all MCP servers into a unified candidate pool, extending beyond single-server experiments. This setup allows us to assess performance within a heterogeneous, large-scale tool space. As shown in Table 9, competing methods experience notable performance degradation when facing this expanded search space. For instance, on the MCP-Universe benchmark, ReAct agents drop from 41.80% to 36.47%. In contrast, ACE-Router demonstrates exceptional stability, maintaining an accuracy of 53.02% (marginally shifted from 53.44%). This demonstrates the effectiveness of ACE-Router in handling large candidate pools, suggesting that with increased training data and model capacity, it can reliably scale to retrieve tools from web-scale open tool ecosystems.

**Robustness against Tool Noise.** We evaluate the robustness of the router by introducing additional noisy tools from two distinct sources: (1) *External Benchmarks* (+LiveMCP), which consists of callable tools drawn from real-world environments within the same or related domains; and (2) *Self-Evolutionary Mutations* (+Mutation), comprising automatically synthesized non-callable variants that are semantically similar to the target tools and introduce complex functional dependencies.

Table 9 illustrates the impact of these noisy settings on model performance. On MCP-Mark under the +LiveMCP setting, even advanced generalist models struggle to filter out high-interference noise; for instance, GPT-4o and Gemini-2.5-Pro achieve accuracies of only 28.00% and 32.00%, respectively. In contrast, ACE-Router demonstrates superior resilience, maintaining a high accuracy of 56.00%. Similarly, under the +Mutation setting—where injected tools are highly confusable

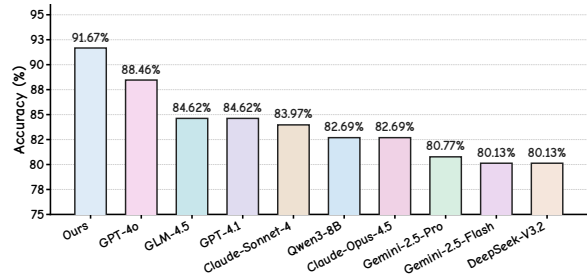


Figure 3: **Performance evaluation on the Agent Route Benchmark.** Comparative analysis of agent route accuracy between ACE-Router and representative baselines.

with targets—ACE-Router exhibits minimal performance degradation. Specifically, accuracy dips only slightly from 53.44% to 53.02% on MCP-Universe and from 60.00% to 54.00% on MCP-Mark.

These results indicate that while generalist models are susceptible to distraction, our specialized router remains robust against both real-world noise and fine-grained hard negatives. This resilience stems from our rigorous training methodology, where the self-evolutionary mutation mechanism forces the router to distinguish between targets and semantically close distractors, thereby establishing stable and fine-grained discriminative boundaries.

### 4.4 Generalization to Agent Routing

We extended our evaluation to the constructed **Agent Route Benchmark**, assessing ACE-Router alongside a series of representative state-of-the-art models on their ability to accurately select the optimal agent for subsequent operations based on the given task query and interaction history.

As illustrated in Figure 3, ACE-Router significantly outperforms all baselines in agent selection tasks, achieving an accuracy of 91.6%. This exceptional performance highlights a critical advantage of our approach: it learns the fundamental logic of "capability matching" rather than overfitting to specific tool schemas. Despite being trained primarily on tool data, the router successfully transfers this abstract decision-making pattern to the agent domain without additional fine-tuning. This generalization capability is pivotal for the envisioning of the Agent Web—an interconnected ecosystem comprising millions of specialized agents. In such a decentralized landscape, our router serves as a universal dispatcher, enabling dynamic, on-demand teaming by accurately identifying and orchestrating diverse agents to collaborate on complex tasks,

Table 2: **Robustness and Scalability Analysis.** We evaluate model performance on MCP-Universe and MCP-Mark under four settings: **Clean** (single-server baseline), **Multi** (merged multi-server tools setting), **+Mutation** (adding Self-Evolutionary mutation tools), and **+LiveMCP** (adding real external tools). The best result is marked in **bold** and the second best result is underlined.

Method	MCP-Universe				MCP-Mark			
	Clean	Multi	+Mutation	+LiveMCP	Clean	Multi	+Mutation	+LiveMCP
<i>Embedding-based Retrieval</i>								
Text-Emb-3-Large (Q)	40.95	40.11	39.69	39.00	26.00	18.00	20.00	14.00
Text-Emb-3-Large (Q+H)	35.20	34.23	34.37	33.24	20.00	14.00	12.00	10.00
all-MiniLM-L6-v2 (Q)	40.67	39.69	39.83	38.57	26.00	20.00	22.00	14.00
all-MiniLM-L6-v2 (Q+H)	43.62	42.92	42.78	41.79	32.00	26.00	20.00	18.00
<i>ReAct Agents</i>								
ReAct (Gemini-2.5-Pro)	41.80	36.47	40.95	40.11	40.00	32.00	26.00	20.00
<i>LLM-based Routers</i>								
Qwen3-8B	46.14	45.30	44.88	44.46	42.00	38.00	32.00	30.00
GLM-4.5	46.42	45.44	45.30	44.32	38.00	32.00	30.00	26.00
DeepSeek-V3.2	44.74	43.76	44.04	43.07	42.00	38.00	36.00	30.00
Claude-Sonnet-4	48.25	47.41	47.13	46.84	44.00	<u>40.00</u>	32.00	30.00
Gemini-2.5-Pro	<u>49.79</u>	<u>49.09</u>	<u>48.81</u>	<u>48.11</u>	46.00	<u>40.00</u>	36.00	32.00
Gemini-2.5-flash	46.98	46.28	45.99	43.76	<u>48.00</u>	<u>40.00</u>	<u>44.00</u>	<u>36.00</u>
GPT-4.1	44.60	43.90	43.48	42.64	36.00	32.00	30.00	28.00
GPT-4o	47.41	46.56	46.42	45.86	38.00	34.00	32.00	28.00
<b>ACE-Router (Ours)</b>	<b>53.44</b>	<b>53.02</b>	<b>53.02</b>	<b>52.60</b>	<b>60.00</b>	<b>56.00</b>	<b>54.00</b>	<b>56.00</b>

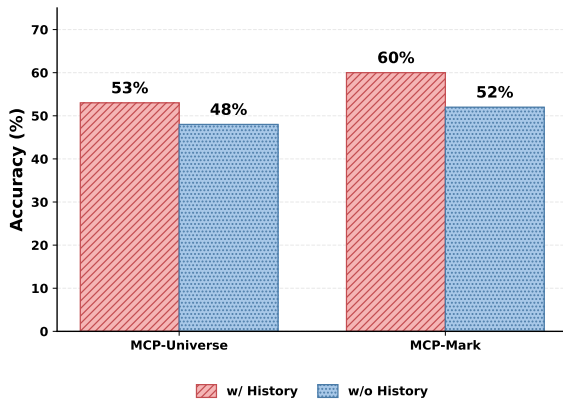


Figure 4: **Impact of Historical Context.** A performance comparison between the history-aware model and an ablation variant trained without historical context.

thereby serving as a foundational infrastructure for future multi-agent systems.

#### 4.5 Significance of History-Aware Routing

A distinct advantage of ACE-Router lies in its capability to effectively leverage the interaction history of the primary reasoning agent to inform routing decisions. This historical context encapsulates critical information—including intermediate outcomes, prior successes and failures, and latent tool usage correlations—that cannot be adequately captured by the current query alone.

To validate the efficacy of this history-aware

mechanism, we conducted an ablation study by intentionally stripping historical context information from the training data. As illustrated in Figure 4, the removal of historical context leads to a significant decline in overall routing accuracy. Specifically, the performance drops from 53% to 48% on MCP-Universe and from 60% to 52% on MCP-Mark. This significant performance gap underscores the critical role of interaction history in two key dimensions: (1) *Sequential Dependency Reasoning*, where the model must track task progress to respect logical prerequisites—for example, ensuring a user profile is retrieved before attempting to access specific GitHub repository details; and (2) *Error Recovery*, where the model utilizes prior execution feedback to recognize failures and pivot to alternative strategies rather than repeating erroneous calls. These findings confirm that effective routing is inherently a dynamic, history-dependent reasoning process, far exceeding the capabilities of static semantic matching.

#### 4.6 Ablation on Mutation and Vanilla Fine-Tuning

To isolate the contributions of our core components and verify that our performance gains stem from architectural design rather than merely large-scale supervision, we conduct two key ablation studies. First, we evaluate a variant without Self-Evolutionary Mutation (*w/o Mutation*) to demon-

strate the critical impact of our dynamic data synthesis. Second, to further isolate the effectiveness of our architectural components, we provide an additional variant (*Vanilla Fine-Tuning*) that removes both the historical context and the Self-Evolutionary Mutation techniques proposed in our paper, relying solely on naive fine-tuning.

As shown in Table 3, removing the mutation module causes a substantial performance drop. Expanding the tool candidate pool from 627 to 2,005 effectively mitigates the risk of overfitting to a narrow candidate space. Furthermore, the full ACE-Router framework significantly outperforms this vanilla baseline on both benchmarks. This confirms that our core designs—history-aware reasoning and self-evolutionary data synthesis—fundamentally drive the model’s strong generalization, providing benefits far beyond standard fine-tuning.

Table 3: **Ablation Results.** Performance impact of the Self-Evolutionary Mutation module and comparison with standard fine-tuning baselines.

Method	MCP-Universe (Acc. %)	MCP-Mark (Acc. %)
ACE-Router (Ours)	<b>53.44</b>	<b>60.00</b>
w/o Mutation	47.12	44.00
Vanilla Fine-Tuning	46.70	42.00

## 5 Conclusion

In this paper, we introduced ACE-Router, a general framework designed for training robust history-aware router models. Our approach begins by expanding an initial candidate pool via self-evolving mutation operators to construct a comprehensive Candidate Graph. Subsequently, we generate effective supervisory signals for the router by employing random walk sampling on the graph coupled with multi-agent trajectory synthesis. Experimental results demonstrate that the router trained on our synthesized data not only achieves superior performance and robustness on MCP tool benchmarks but also exhibits strong generalization capabilities in agent retrieval tasks. These findings pave the way for a router-centric paradigm in future multi-agent collaboration within the Agent Web ecosystem.

## Limitations

Due to computational resource constraints, we exclusively implemented LoRA fine-tuning on the Qwen3-8B architecture. Nevertheless, we posit that our constructed dataset possesses inherent scalability, suggesting that performance gains could be

substantially amplified when applied to larger-scale foundation models.

Furthermore, our current routing mechanism is predominantly trained on tool-use data. While preliminary results indicate that this tool-oriented router generalizes effectively to agent retrieval tasks, we plan to develop specialized routing models explicitly tailored for multi-agent scenarios in future work. Ultimately, we aim to extend this routing training paradigm to encompass universal, massive-scale retrieval requirements, such as long-term memory management.

## Acknowledgments

The work is supported by National Natural Science Foundation of China (62502310,62322603).

## References

- Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.
- Anthropic. 2024. Introducing the Model Context Protocol. <https://www.anthropic.com/news/model-context-protocol>.
- Anthropic. 2025. Introducing Claude 4. <https://www.anthropic.com/news/claude-4>.
- Ge Chang, Jinbo Su, Jiacheng Liu, Pengfei Yang, Yuhao Shang, Huiwen Zheng, Hongli Ma, Yan Liang, Yuanchun Li, and Yunxin Liu. 2025. Grail: Learning to interact with large knowledge graphs for retrieval augmented reasoning. *arXiv preprint arXiv:2508.05498*.
- Zehui Chen, Weihua Du, Wenwei Zhang, Kuikun Liu, Jiangning Liu, Miao Zheng, Jingming Zhuo, Songyang Zhang, Dahua Lin, Kai Chen, and Feng Zhao. 2024. T-eval: Evaluating the tool utilization capability of large language models step by step. *arXiv preprint arXiv:2312.14033*.
- Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.
- Shiqing Fan, Xichen Ding, Liang Zhang, and Linjian Mo. 2025. Mcptoolbench++: A large scale ai agent model context protocol mcp tool use benchmark. *arXiv preprint arXiv:2508.07575*.

- Jinyuan Fang, Yanwen Peng, Xi Zhang, Yingxu Wang, Xinhao Yi, Guibin Zhang, Yi Xu, Bin Wu, Siwei Liu, Zihao Li, and 1 others. 2025. A comprehensive survey of self-evolving ai agents: A new paradigm bridging foundation models and lifelong agentic systems. *arXiv preprint arXiv:2508.07407*.
- Tiantian Gan and Qiyao Sun. 2025. Rag-mcp: Mitigating prompt bloat in llm tool selection via retrieval-augmented generation. *arXiv preprint arXiv:2505.03275*.
- Xuanqi Gao, Siyi Xie, Juan Zhai, Shiqing Ma, and Chao Shen. 2025. Mcp-radar: A multi-dimensional benchmark for evaluating tool use capabilities in large language models. *Preprint*, arXiv:2505.16700.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025a. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.
- Yifu Guo, Zishan Xu, Zhiyuan Yao, Yuquan Lu, Jiaye Lin, Sen Hu, Zhenheng Tang, Huacan Wang, and Ronghao Chen. 2025b. Octopus: Agentic multimodal reasoning with six-capability orchestration. *Preprint*, arXiv:2511.15351.
- Zikang Guo, Benfeng Xu, Chiwei Zhu, Wentao Hong, Xiaorui Wang, and Zhendong Mao. 2025c. Mcp-agentbench: Evaluating real-world language agent performance with mcp-mediated tools. *Preprint*, arXiv:2509.09734.
- Jack Hong, Chenxiao Zhao, ChengLin Zhu, Weiheng Lu, Guohai Xu, and Xing Yu. 2025. Deep-eyesv2: Toward agentic multimodal model. *Preprint*, arXiv:2511.05271.
- Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.
- Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.
- Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*.
- Minghao Li, Feifan Song, Bowen Yu, 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*, pages 3102–3116. Association for Computational Linguistics.
- Sijia Li, Yuchen Huang, Zifan Liu, Zijian Li, Jingjing Fu, Lei Song, Jiang Bian, Jun Zhang, and Rui Wang. 2025a. Sit-graph: State integrated tool graph for multi-turn agents. *Preprint*, arXiv:2512.07287.
- Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. 2025b. Search-o1: Agentic search-enhanced large reasoning models. *arXiv preprint arXiv:2501.05366*.
- Jiaye Lin, Yifu Guo, Yuzhen Han, Sen Hu, Ziyi Ni, Licheng Wang, Mingguang Chen, Hongzhang Liu, Ronghao Chen, Yangfan He, Daxin Jiang, Binxing Jiao, Chen Hu, and Huacan Wang. 2025. Se-agent: Self-evolution trajectory optimization in multi-step reasoning with llm-based agents. *Preprint*, arXiv:2508.02085.
- Weiwen Liu, Xu Huang, Xingshan Zeng, Xinlong Hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, and 1 others. 2024. Toolace: Winning the points of llm function calling. *arXiv preprint arXiv:2409.00920*.
- Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Felix Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, Zirui Wang, and Ruoming Pang. 2024. Toolsandbox: A stateful, conversational, interactive evaluation benchmark for llm tool use capabilities. *arXiv preprint arXiv:2408.04682*.
- 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*.
- Ziyang Luo, Zhiqi Shen, Wenzhuo Yang, Zirui Zhao, Prathyusha Jwalapuram, Amrita Saha, Doyen Sahoo, Silvio Savarese, Caiming Xiong, and Junnan Li. 2025. Mcp-universe: Benchmarking large language models with real-world model context protocol servers. *arXiv preprint arXiv:2508.14704*.
- Xing Han Lù, Gaurav Kamath, Marius Mosbach, and Siva Reddy. 2025. Build the web for agents, not agents for the web. *Preprint*, arXiv:2506.10953.
- Guozhao Mo, Wenliang Zhong, Jiawei Chen, Xuanang Chen, Yaojie Lu, Hongyu Lin, Ben He, Xianpei Han, and Le Sun. 2025. Livemcpbench: Can agents navigate an ocean of mcp tools? *arXiv preprint arXiv:2508.01780*.
- Alexander Novikov, Ngãn Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, and 1 others. 2025. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*.
- Bhrij Patel, Davide Belli, Amir Jalalirad, Maximilian Arnold, Aleksandr Ermovol, and Bence Major. 2025. Dynamic tool dependency retrieval for efficient function calling. *arXiv preprint arXiv:2512.17052*.

- Shishir G. Patil, Huanzhi Mao, Charlie Cheng-Jie Ji, Fanjia Yan, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. 2025. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning*.
- Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. 2023. [Gorilla: Large language model connected with massive apis](#). *arXiv preprint arXiv:2305.15334*.
- Tatiana Petrova, Boris Bliznioukov, Aleksandr Puzikov, and Radu State. 2025. [From semantic web and mas to agentic ai: A unified narrative of the web of agents](#). *Preprint*, arXiv:2507.10644.
- 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*.
- Stephen Robertson and Hugo Zaragoza. 2009. *The probabilistic relevance framework: BM25 and beyond*, volume 4. Now Publishers Inc.
- Ranjan Sapkota, Konstantinos I Roumeliotis, and Manoj Karkee. 2025. Ai agents vs. agentic ai: A conceptual taxonomy, applications and challenges. *arXiv preprint arXiv:2505.10468*.
- Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. [Toolformer: Language models can teach themselves to use tools](#). *arXiv preprint arXiv:2302.04761*.
- Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. [Hugging-gpt: Solving ai tasks with chatgpt and its friends in hugging face](#). *arXiv preprint arXiv:2303.17580*.
- Yexuan Shi, Mingyu Wang, Yunxiang Cao, Hongjie Lai, Junjian Lan, Xin Han, Yu Wang, Jie Geng, Zhenan Li, Zihao Xia, Xiang Chen, Chen Li, Jian Xu, Wenbo Duan, and Yuanshuo Zhu. 2025. [Aime: Towards fully-autonomous multi-agent framework](#). *Preprint*, arXiv:2507.11988.
- Yifan Song, Weimin Xiong, Dawei Zhu, Cheng Li, Ke Wang, Ye Tian, and Sujian Li. 2023. [Restgpt: Connecting large language models with real-world restful apis](#). *arXiv preprint arXiv:2306.06624*.
- Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, Quoc-Viet Pham, Barry O’Sullivan, and Hoang D Nguyen. 2025. Multi-agent collaboration mechanisms: A survey of llms. *arXiv preprint arXiv:2501.06322*.
- Zezhong Wang, Xingshan Zeng, Weiwen Liu, Liangyou Li, Yasheng Wang, Lifeng Shang, Xin Jiang, Qun Liu, and Kam-Fai Wong. 2025. Toolflow: Boosting llm tool-calling through natural and coherent dialogue synthesis. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4246–4263.
- Zijian Wu, Xiangyan Liu, Xinyuan Zhang, Lingjun Chen, Fanqing Meng, Lingxiao Du, Yiran Zhao, Fanshi Zhang, Yaoqi Ye, Jiawei Wang, and 1 others. 2025. Mcpmark: A benchmark for stress-testing realistic and comprehensive mcp use. *arXiv preprint arXiv:2509.24002*.
- Zhihao Xu, Rumei Li, Jiahuan Li, Rongxiang Weng, Jingang Wang, Xunliang Cai, and Xiting Wang. 2026. [Unlocking implicit experience: Synthesizing tool-use trajectories from text](#). *Preprint*, arXiv:2601.10355.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.
- Jian Yang, Zhao Wang, Yuxiang Li, Hao Chen, Ke Wang, Yingwei Li, and Jingrui He. 2024a. [Auto-tool: Efficient tool selection for large language model agents](#). *arXiv preprint arXiv:2511.14650*.
- John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. 2024b. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652.
- Yingxuan Yang, Mulei Ma, Yuxuan Huang, Huacan Chai, Chenyu Gong, Haoran Geng, Yuanjian Zhou, Ying Wen, Meng Fang, Muhao Chen, and 1 others. 2025b. Agentic web: Weaving the next web with ai agents. *arXiv preprint arXiv:2507.21206*.
- Shunyu Yao, Dian Zhao, Jeffrey Yu, Nan Shafran, Karthik Narasimhan, and Yuan Cao. 2023. [React: Synergizing reasoning and acting in language models](#). In *International Conference on Learning Representations*.
- Xukun Zhang, Zhiyuan Zhu, Mingyu Wang, Lingfei Wang, Haoran Li, Jingjing Zhang, and Dongsheng Li. 2024. [Toolnet: Connecting large language models with massive tools via tool graph](#). *arXiv preprint arXiv:2403.00839*.
- Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, and 1 others. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.
- Yuchen Zhuang, Xiang Chen, Tong Yu, Saayan Mitra, Victor Bursztyn, Ryan A. Rossi, Somdeb Sarkhel, and Chao Zhang. 2023. [Toolchain\\*: Efficient action space navigation in large language models with a\\* search](#). *arXiv preprint arXiv:2310.13227*.

## A Comparison with Stronger Retrieval Baselines

We expand our evaluation by replacing the initial retrieval baselines with stronger alternatives: a sparse retriever (BM25 (Robertson and Zaragoza, 2009)) and advanced dense embedding models (Contriever (Izacard et al., 2021) and Qwen3-Embedding-8B (Zhang et al., 2025)).

As shown in Table 4, ACE-Router consistently outperforms these baselines on both benchmarks. This demonstrates the significant advantage of our history-aware routing mechanism over standard retrieval approaches in complex tool selection.

Table 4: **Comparison with stronger baselines.** Performance evaluation of ACE-Router against advanced sparse and dense retrieval approaches.

Method	MCP-Universe (Acc. %)	MCP-Mark (Acc. %)
BM25	30.00	40.00
Contriever	33.66	40.00
Qwen3-Embedding-8B	44.88	42.00
<b>ACE-Router (Ours)</b>	<b>53.44</b>	<b>60.00</b>

## B Validity of Simulated Ground Truth

A critical prerequisite for our framework is ensuring that the simulated training data provides high-quality and reliable supervision. To rigorously quantify the accuracy of the LLM-generated tool calls used as ground truth, we conducted a manual human evaluation. We sampled a random subset of 100 synthesized trajectories and had human experts independently annotate the correct tool selections. The agreement between our synthesized labels and expert annotations was measured using Cohen’s Kappa ( $\kappa$ ). As shown in Table 5, the evaluation yields a  $\kappa$  score of **0.93**, indicating near-perfect agreement. This confirms that our self-evolutionary data synthesis mechanism generates highly reliable supervision signals.

Table 5: **Reliability of Synthesized Trajectories.** Agreement between LLM-generated tool selections and human expert annotations.

Metric	Score
Cohen’s Kappa ( $\kappa$ )	<b>0.93</b>

## C Self-Evolutionary Mutation

Table 6 presents the taxonomy of mutation types for tools, while Table 7 outlines the corresponding

mutation strategies designed for agents. Both tables include detailed descriptions and examples

Table 6: Taxonomy of Tool Mutation Types

Mutation Type	Description & Examples
<b>Usage Extension</b>	Apply the tool’s core logic to related new scenarios or domains. <i>Example:</i> <code>analyze_code_quality</code> → <code>analyze_document_quality</code> (applies code analysis concepts to documents).
<b>Function Enhancement</b>	Substantially expand the tool’s capabilities to enable entirely new use cases while maintaining the core purpose (add 2+ major user-visible features). <i>Example:</i> <code>compress_image</code> → <code>image_optimization_suite</code> (adds format conversion + batch processing + quality presets + metadata editing).
<b>Workflow Chain</b>	Create a tool that works immediately before or after the original tool in a workflow, providing better inputs or processing outputs. <i>Example:</i> <code>search_web</code> → <code>prepare_search_keywords</code> (pre-processes queries) or <code>summarize_search_results</code> (post-processes results).
<b>Helper Tool</b>	Create an independent supporting tool that enhances the ecosystem around the original tool. <i>Example:</i> <code>create_chart</code> → <code>validate_chart_data</code> (checks data format before charting) or <code>suggest_chart_colors</code> (recommends color schemes).
<b>Parameter design</b>	<b>Re-</b> Modify the tool’s parameter structure to enable different input patterns or interaction approaches. Focus on meaningful parameter changes that shift how users provide data or configure behavior. <i>Example:</i> <code>get_user(user_id: string)</code> → <code>query_users(filters: object, sort: string, limit: number)</code> (from single lookup to flexible querying).

## D Agent Route Benchmark

### D.1 Standardized Agent Definition

In this section, we present the standardized schema used to define agent capabilities within our benchmark. Below are two representative examples: `SWE_agent` and `WebVoyager_agent`.

#### Agent Definition Examples

```
[
  {
    "name": "SWE_agent",
    "description": "An autonomous software engineering agent that can understand codebases, modify source code, execute tests, and resolve real GitHub issues through iterative interaction with
```

```

    a development environment.",
    "tools": [
      "find_file", "search_code", "view_file",
      "edit_file", "apply_patch", "run_tests",
      "exec_shell", "search_dir", "scroll_up"
    ],
    "inputSchema": {
      "type": "object",
      "properties": {
        "repo_path": {
          "type": "string",
          "description": "Local filesystem path to the
target code repository"
        },
        "issue_description": {
          "type": "string",
          "description": "Issue title and detailed
problem description, typically from a GitHub
issue"
        },
        "language": {
          "type": "string",
          "description": "Primary programming language
of the codebase"
        },
        "test_command": {
          "type": "string",
          "description": "Shell command used to
execute the project's test suite"
        },
        "model": {
          "type": "string",
          "description": "Identifier of the language
model used by the agent"
        }
      }
    },
    "tags": ["General"]
  },
  {
    "name": "WebVoyager_agent",
    "description": "A multimodal web agent that uses
both text and visual input to interact with real
websites end-to-end, completing user
instructions by browsing, clicking, typing, and
interpreting pages.",
    "tools": [
      "observe_dom", "click", "type_text",
      "scroll", "navigate"
    ],
    "inputSchema": {
      "type": "object",
      "properties": {
        "instruction": {
          "type": "string",
          "description": "User's natural language
instruction describing the web task to complete"
        },
        "start_url": {
          "type": "string",
          "description": "Initial webpage URL where
the agent begins browsing"
        },
        "multimodal_model": {
          "type": "string",
          "description": "Identifier of the vision-
language model used for perception and reasoning"
        },
        "timeout": {
          "type": "integer",
          "description": "Maximum allowed wait time
for browser interactions"
        }
      }
    },
    "tags": ["General"]
  }
]

```

Table 7: Taxonomy of Agent Mutation Types

Mutation Type	Description & Examples
<b>Domain Transfer</b>	Apply the agent's architecture and workflow to a different but related domain. <i>Example:</i> SWE_agent → doc_review_agent (adapts the edit/search/validate pattern from code to documents).
<b>Capability Enhancement</b>	Substantially expand the agent's capabilities by adding new tools and extending its scope. <i>Example:</i> code_search_agent → code_intelligence_agent (adds semantic analysis, dependency tracking, and refactoring suggestions).
<b>Workflow Specialization</b>	Create a more focused agent that specializes in a subset of the original agent's workflow. <i>Example:</i> full_stack_dev_agent → api_testing_agent (focuses exclusively on API testing with specialized validation tools).
<b>Tool Composition</b>	Recombine and restructure the agent's tools to create new workflow patterns. <i>Example:</i> data_pipeline_agent → realtime_streaming_agent (reorganizes batch processing tools into streaming-compatible tools).
<b>Scenario Adaptation</b>	Adapt the agent to handle different use case scenarios or user contexts. <i>Example:</i> general_qa_agent → customer_support_agent (adapts general QA capabilities specifically for customer service scenarios).

## D.2 Agent Route Benchmark Examples

### Task Example

```
{
  "system": "You are an Agent Router.
  Your task is to analyze the meaning of a user query
  and select the most relevant agents based on the
  agents' descriptions and schemas.

  Guidelines:
  1. Consider both the agent descriptions and input
  schemas when judging relevance.
  2. Use the inputSchema to understand what parameters
  each agent accepts.
  3. Do not infer hidden capabilities or invent agents.
  4. Return only one agent that is most relevant.
  5. Output strictly in the required format: ["
  agent_name"], no extra commentary.",
  "user": "Below are examples of the user's past
  interactions, including queries and system
  responses:
<history>
User: Initiate a comprehensive code review of the
  backend systems located at the server path "/var
  /www/backend" using PHP as the primary language.
  ...
Assistant: <think>...</think>
<agent_call>code_review_agent{"repo_path": "/var/www/
  backend", "language": "PHP", "review_criteria":
  "Performance optimization and best practices
  adherence", "report_format": "markdown"}</
  agent_call>
Tool results: ...
Assistant: ## Code Review Report ...

User: I would like to proceed with an analysis of
  sentiment in the trading data for our company's
  stocks, symbolized as "COMP", ...
Assistant: <think>...</think>
<agent_call>sentiment_analysis_trading_agent{"
  market_symbols": ["COMP"], "timeframe": "
  intraday", "sentiment_threshold": 0.7, "
  data_source": "auto"}</agent_call>
<agent_call>risk_management_agent{"user_instruction":
  "Evaluate financial risks ...", "llm_model": "
  default", "risk_tolerance": "medium"}</
  agent_call>
Tool results: ...
Assistant: The sentiment analysis for "COMP" indicates
  ...
</history>

Current user query:
<current_query>"Please execute a financial forecast to
  evaluate the impact on overall business
  profitability given the recent trading
  adjustments and risk evaluations. Use a
  predictive model to analyze the potential
  outcomes and suggest appropriate policy changes.
  Ensure that the analysis accounts for the
  sentiments and risks previously identified, and
  provide a detailed report on strategic
  recommendations."</current_query>

Available agents:
<agents>[
{
  "name": "risk_management_agent",
  "description": "The Risk Management Agent
  specializes in evaluating financial risks based
  on historical data and current market conditions
  , providing targeted risk mitigation strategies
  and recommendations.",
  "tools": ["parse_risk_instruction", "
  plan_risk_analysis", "generate_risk_model", "
  execute_risk_assessment", "
  validate_risk_strategies"],
  "inputSchema": {
    "type": "object",
    "properties": {
      "user_instruction": {"type": "string"},
      "spreadsheet_id": {"type": "string"},
      "llm_model": {"type": "string"},
```

```
      "risk_tolerance": {"type": "string"}
    },
    "required": []
  }
},
{
  "name": "economy_forecasting_agent",
  "description": "An agent designed to forecast
  economic trends and guide policy recommendations
  by simulating various economic scenarios and
  assessing potential outcomes.",
  "tools": ["simulate_economic_scenario", "
  forecast_trends", "recommend_policy", "
  analyze_impact"],
  "inputSchema": {
    "type": "object",
    "properties": {
      "economic_indicators": {"type": "object"},
      "forecast_horizon": {"type": "integer"},
      "policy_options": {"type": "object"},
      "evaluation_criteria": {"type": "object"}
    },
    "required": []
  }
},
... (other agents)
]</agents>

Task:
<task>
Analyze the current query in the context of the user's
  past queries and agent descriptions.
Return the most relevant agent based on their
  descriptions and schemas.
</task>

Output requirements:
###
- First, think through your reasoning inside <think></
  think> tags
- Then output only one agent name as a JSON array
- Format:
<think>
Your reasoning about which agent to select...
</think>

["agent_name"]
###,
"expected_agent": ["economy_forecasting_agent"]
}
```

## E Prompt

### Self-Evolutionary Mutation Prompt for Tool

#### # Role: Expert Tool Designer

You are an expert tool designer specializing in creating innovative software tools through genetic algorithm-inspired mutations. Your expertise includes API design, parameter optimization, and functional enhancement.

#### ## Your Task

Perform a **MUTATION OPERATION** on the given tool to create a new, related but distinct tool that serves a similar domain but with meaningful innovations.

#### ## Original Tool Analysis

```
{json.dumps(base_tool,ensure_ascii=False,indent=2)}
```

#### ## Mutation Strategy: {mutation\_type}

#### ## Design Requirements

#### ### Functional Requirements:

- **Innovation:** Create meaningful functional differences while maintaining domain relevance

- **Utility:** Ensure the new tool solves a real problem or improves upon existing functionality
- **Compatibility:** Maintain similar complexity level and use case applicability

### ### Technical Requirements:

- **Parameters:** Design intuitive, well-typed parameters following JSON Schema standards
- **Naming:** Use clear, descriptive names that immediately convey purpose
- **Documentation:** Write concise but comprehensive descriptions
- **Validation:** Include appropriate parameter validation and constraints

### ### Constraints:

- Keep the same domain tags: `{base_tool.get('tags', [])}`
- Avoid direct copying – ensure meaningful differentiation
- Maintain professional tool naming conventions
- Focus on practical, implementable functionality

### ## Expected Output

Return **ONLY** valid JSON in this exact format (no markdown, no extra text):

```
{
  "name": "descriptive_tool_name",
  "description": "Clear, actionable
  description of what this
  tool does and why it's
  useful",
  "inputSchema": {
    "type": "object",
    "properties": {
      "parameter_name": {
        "type": "appropriate_type",
        "description": "What this
        parameter does and how
        to use it",
        "default": "
        optional_default_value"
      }
    },
    "required": ["
    list_required_parameters"
  ],
  "tags": {base_tool.get('tags',
  [])}
}
```

**CRITICAL:** Use only double quotes, no single quotes. No markdown formatting.

**Note:** Only include a "results" field if the tool produces structured output that requires explicit definition.

### ## Quality Checklist

- Tool name is descriptive and unique
- Description clearly explains purpose and value
- Parameters are well-designed with proper types
- Required parameters are logically necessary
- JSON syntax is valid and complete

## Self-Evolutionary Mutation Prompt for Agent

### # Role: Expert Agent Architect

You are an expert AI agent architect specializing in designing autonomous agents through genetic algorithm-inspired mutations. Your expertise includes agent workflow design, tool orchestration, and capability planning.

### ## Your Task

Perform a **MUTATION OPERATION** on the given agent to create a new, related but distinct agent that serves a similar purpose but with meaningful innovations in its capabilities and tool composition.

### ## Original Agent Analysis

**Agent Name:** {agent\_name}

**Description:** {agent\_description}

### Tools Used by This Agent:

`{json.dumps(agent_tools, ensure_ascii=False, indent=2)}`

### Agent InputSchema (Parameters):

`{json.dumps(agent_args, ensure_ascii=False, indent=2)}`

### ## Mutation Strategy: {mutation\_type}

### ## Design Requirements

### ### Agent Design Principles:

- **Coherent Toolset:** The tools should work together to accomplish the agent's goals
- **Clear Workflow:** The agent should have a logical flow of operations
- **Practical Utility:** The agent should solve real-world problems
- **Tool Synergy:** Tools should complement each other, not duplicate functionality

### ### Tool Evolution Guidelines:

- You may **ADD** new tools that enhance the agent's capabilities
- You may **MODIFY** existing tools to better fit the new agent's purpose
- You may **REMOVE** tools that don't align with the new agent's focus

- You may RENAME tools to reflect their new context
- Aim for 4–8 tools per agent (not too few, not too many)

### ### Naming Convention:

- Agent name MUST end with "\_agent" suffix
- Use snake\_case format
- Name should clearly indicate the agent's primary function
- Example: "code\_review\_agent", "data\_analysis\_agent", "document\_qa\_agent"

### ### Tags Guidelines:

- Tags should categorize the agent's primary domain or capability
- Use descriptive tags like: "code agent", "search agent", "web agent", "data agent", "research agent", "automation agent", "analysis agent", "multimodal agent", etc.
- Can include multiple tags if the agent spans multiple domains

### ## Expected Output

Return **ONLY** valid JSON in this exact format (no markdown, no extra text):

```
{
  "name": "descriptive_name_agent",
  "description": "Clear description of what
    this agent does, its primary use
    cases, and how it accomplishes its
    goals",
  "tools": [
    "tool_name_1",
    "tool_name_2",
    "tool_name_3"
  ],
  "inputSchema": {
    "type": "object",
    "properties": {
      "parameter_name": {
        "type": "appropriate_type",
        "description": "Detailed description
          of what this parameter
          configures for the agent"
      }
    }
  },
  "tags": ["category agent"]
}
```

### CRITICAL REQUIREMENTS:

- Agent name MUST end with "\_agent"
- Use only double quotes, no single quotes
- No markdown formatting
- Tools array should contain 4–8 tool names
- Each tool name should be descriptive and use snake\_case

- Tags should be descriptive category labels (e.g., "code agent", "search agent", "web agent")
- Each parameter in inputSchema.properties MUST have a detailed "description" field

### ## Quality Checklist

- Agent name ends with "\_agent" and clearly describes purpose
- Description explains the agent's workflow and capabilities
- Tools form a coherent set that enables the agent's goals
- Tools are appropriately evolved from the original (not just copied)
- Parameters make sense for configuring this agent
- Each parameter has a clear, detailed description in inputSchema
- Tags accurately categorize the agent's domain
- JSON syntax is valid and complete

## F Confidence Intervals and Variance Analysis

To rigorously demonstrate the reliability of our results and address the robustness of our evaluations, we calculate and report the Standard Deviation (STD) across 5 independent runs for our main experiments.

Table 8 updates the overall benchmark performance, presenting the variance for the final accuracy on MCP-Universe and MCP-Mark. Furthermore, Table 9 provides the comprehensive robustness and scalability analysis with full confidence intervals under all experimental settings. These results confirm that ACE-Router maintains statistically significant improvements over the baselines.

Table 8: **Overall Performance.** Accuracy (%) with standard deviation across 5 independent runs on the MCP-Universe and MCP-Mark benchmarks. The best result is marked in **bold** and the second best is underlined.

Method	MCP-Universe (Overall)	MCP-Mark (Overall)
<i>Embedding-based Retrieval</i>		
Text-Emb-3-Large (Q)	40.95 ± 1.43	26.00 ± 2.45
Text-Emb-3-Large (Q+H)	35.20 ± 1.51	20.00 ± 2.00
all-MiniLM-L6-v2 (Q)	40.67 ± 1.95	26.00 ± 3.46
all-MiniLM-L6-v2 (Q+H)	43.62 ± 1.25	32.00 ± 3.46
<i>ReAct Agents</i>		
ReAct (Gemini-2.5-Pro)	41.80 ± 1.65	40.00 ± 2.00
<i>LLM-based Routers</i>		
Qwen3-8B	46.14 ± 1.56	42.00 ± 2.00
GLM-4.5	46.42 ± 0.99	38.00 ± 3.46
DeepSeek-V3.2	44.74 ± 1.48	42.00 ± 2.00
Claude-Sonnet-4	48.25 ± 1.93	44.00 ± 2.45
Gemini-2.5-Pro	<u>49.79 ± 1.36</u>	46.00 ± 2.45
Gemini-2.5-flash	46.98 ± 1.66	<u>48.00 ± 2.83</u>
GPT-4.1	44.60 ± 1.45	36.00 ± 2.00
GPT-4o	47.41 ± 1.50	38.00 ± 2.83
<b>ACE-Router (Ours)</b>	<b>53.44 ± 0.57</b>	<b>60.00 ± 2.00</b>

Table 9: **Robustness and Scalability Analysis.** We evaluate model performance on MCP-Universe and MCP-Mark under four settings: **Clean** (single-server baseline), **Multi** (merged multi-server tools setting), **+Mutation** (adding Self-Evolutionary mutation tools), and **+LiveMCP** (adding real external tools). The best result is marked in **bold** and the second best is underlined.

Method	MCP-Universe				MCP-Mark			
	Clean	Multi	+Mutation	+LiveMCP	Clean	Multi	+Mutation	+LiveMCP
<i>Embedding-based Retrieval</i>								
Text-Emb-3-Large (Q)	40.95 ± 1.43	40.11 ± 1.89	39.69 ± 1.38	38.99 ± 1.27	26.00 ± 2.45	18.00 ± 3.16	20.00 ± 2.83	14.00 ± 2.83
Text-Emb-3-Large (Q+H)	35.20 ± 1.51	34.22 ± 1.96	34.36 ± 1.38	33.24 ± 1.95	20.00 ± 2.00	14.00 ± 3.46	12.00 ± 2.00	10.00 ± 3.46
all-MiniLM-L6-v2 (Q)	40.67 ± 1.95	39.69 ± 0.59	39.83 ± 1.99	38.57 ± 1.97	26.00 ± 3.46	20.00 ± 2.45	22.00 ± 2.83	14.00 ± 2.45
all-MiniLM-L6-v2 (Q+H)	43.62 ± 1.25	42.92 ± 1.17	42.78 ± 1.69	41.80 ± 0.98	32.00 ± 3.46	26.00 ± 2.45	20.00 ± 2.83	18.00 ± 3.46
<i>ReAct Agents</i>								
ReAct (Gemini-2.5-Pro)	41.80 ± 1.65	36.47 ± 1.02	40.95 ± 1.39	40.11 ± 1.36	40.00 ± 2.00	32.00 ± 2.83	26.00 ± 3.16	20.00 ± 1.41
<i>LLM-based Routers</i>								
Qwen3-8B	46.14 ± 1.56	45.30 ± 1.46	44.88 ± 1.83	44.46 ± 1.75	42.00 ± 2.00	38.00 ± 2.45	32.00 ± 1.41	30.00 ± 2.83
GLM-4.5	46.42 ± 0.99	45.44 ± 1.70	45.30 ± 0.75	44.32 ± 1.61	38.00 ± 3.46	32.00 ± 3.16	30.00 ± 3.16	26.00 ± 3.46
DeepSeek-V3.2	44.74 ± 1.48	43.76 ± 1.94	44.04 ± 1.30	43.06 ± 0.95	42.00 ± 2.00	38.00 ± 3.16	36.00 ± 2.83	30.00 ± 2.83
Claude-Sonnet-4	48.25 ± 1.93	47.41 ± 1.59	47.12 ± 1.49	46.84 ± 1.97	44.00 ± 2.45	<u>40.00 ± 3.16</u>	32.00 ± 3.46	30.00 ± 2.83
Gemini-2.5-Pro	49.79 ± 1.36	49.09 ± 1.81	48.81 ± 1.90	48.11 ± 1.34	46.00 ± 2.45	40.00 ± 2.45	36.00 ± 3.16	32.00 ± 2.83
Gemini-2.5-flash	46.98 ± 1.66	46.28 ± 1.89	46.00 ± 0.82	43.76 ± 1.76	<u>48.00 ± 2.83</u>	<u>40.00 ± 2.00</u>	<u>44.00 ± 2.45</u>	36.00 ± 3.46
GPT-4.1	44.60 ± 1.45	43.90 ± 1.02	43.48 ± 1.44	42.64 ± 1.48	36.00 ± 2.00	32.00 ± 2.00	30.00 ± 2.00	28.00 ± 2.83
GPT-4o	47.41 ± 1.50	46.56 ± 1.62	46.42 ± 0.74	45.86 ± 1.75	38.00 ± 2.83	34.00 ± 3.16	32.00 ± 2.83	28.00 ± 2.45
<b>ACE-Router (Ours)</b>	<b>53.44 ± 0.57</b>	<b>53.02 ± 1.21</b>	<b>53.02 ± 0.87</b>	<b>52.59 ± 0.50</b>	<b>60.00 ± 2.00</b>	<b>56.00 ± 1.41</b>	<b>54.00 ± 2.45</b>	<b>56.00 ± 2.83</b>