Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Dynamic AI Tool Orchestration Framework for Enterprise Scale AI Agents

Vishram Singh

Independent Researcher, USA

ARTICLE INFO

ABSTRACT

Introduction: LLMs have enabled the creation of AI agents that can perform complex tasks that require access to different tools and APIs. In enterprise applications, AI agents are invoked to interact with an ecosystem of hundreds of tools specifically designed to query databases, perform financial validation, scan documents, and automate tasks. Choosing the right tool for the task is an architectural consideration that affects the system's scalability, latency, and correctness.

Objectives: This paper proposes the Dynamic AI Tool Orchestration Framework (DATOF), a scalable smart tool selection framework for enterprise AI agents, designed to enable automatic tool discovery and selection across large, heterogeneous tool registries with minimal latency and execution errors.

Methods: DATOF is built on a modular structure comprising intent classification, semantic indexing of tools, tool capability metadata, and an adaptive multi-criteria orchestration engine that ranks and selects candidate tools for each user query.

Results: In simulated enterprise workloads, DATOF achieves a 32% increase in tool allocation accuracy and a 28% decrease in decision time relative to static baselines, while maintaining stable performance as the tool registry scales from 50 to 500 tools.

Conclusions: DATOF is a reliable and scalable approach to building AI agents able to meet user needs in complex enterprise software ecosystems.

Keywords: Artificial Intelligence Agents, Tool Orchestration, Large Language Models, Enterprise AI Systems, Clever Automation, Distributed Systems

1. INTRODUCTION

LLM-powered Information Worker Agents can carry out complex workflows across enterprise systems, including querying enterprise databases, generating reports, processing documents, and orchestrating multi-step workflows by calling out to other tools or APIs for the user [1][9][2][3].

Enterprise environments may expose many different tools for an enterprise AI agent to use. A finance-focused enterprise AI agent may have access to journal validation APIs, ledger query APIs, a customer account retrieval system, a workflow approval service, and a data analytics service for querying and analyzing company data. As larger installations amass hundreds or thousands of such tools, selecting the right tool becomes a non-trivial architectural challenge [4][10].

Most existing systems rely on a static routing of queries to tools or a static mapping of query content to tools. For this reason, their solution does not scale to a large number of tools, to a variation in tools' functionality, and to a variation in queries from users. AI agents must be able to discover tools and select them dynamically [5][11].

This paper proposes the Dynamic AI Tool Orchestration Framework (DATOF), a system architecture for scalable, smart, and dynamic tool selection and composition in Enterprise AI agents.

Major contributions of this work include:

- A modular architecture to orchestrate many other enterprise AI tools.

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

- A semantic indexing mechanism to ease efficient discovery of tools based on user queries.
- This development led to an adaptive multi-criteria decision engine to select the best tool.
- Experiments indicate that they considerably outperform static routing baselines in accuracy, latency, and scalability.

The rest of this paper is structured as follows. Section 2 provides background and motivation. The structure of the DATOF system is described in Section 3, followed by a description of the different components in Sections 4 through 7 and by the experimental evaluation in Section 8. Section 9 concludes and suggests directions for future research.

2. BACKGROUND AND RELATED WORK

2.1 AI Agents and Tool Use

LLMs can benefit from tool use [1][2], allowing AI agents to interact with structured data, execute business logic, and make real-time requests to enterprise services beyond text generation. The earliest work showed that language models can learn how to invoke APIs on their own (Toolformer [2]) and that reasoning and acting can be unified into a single framework for language model agents (ReAct [3]), which inspired more strong tool orchestration systems built for enterprise use cases.

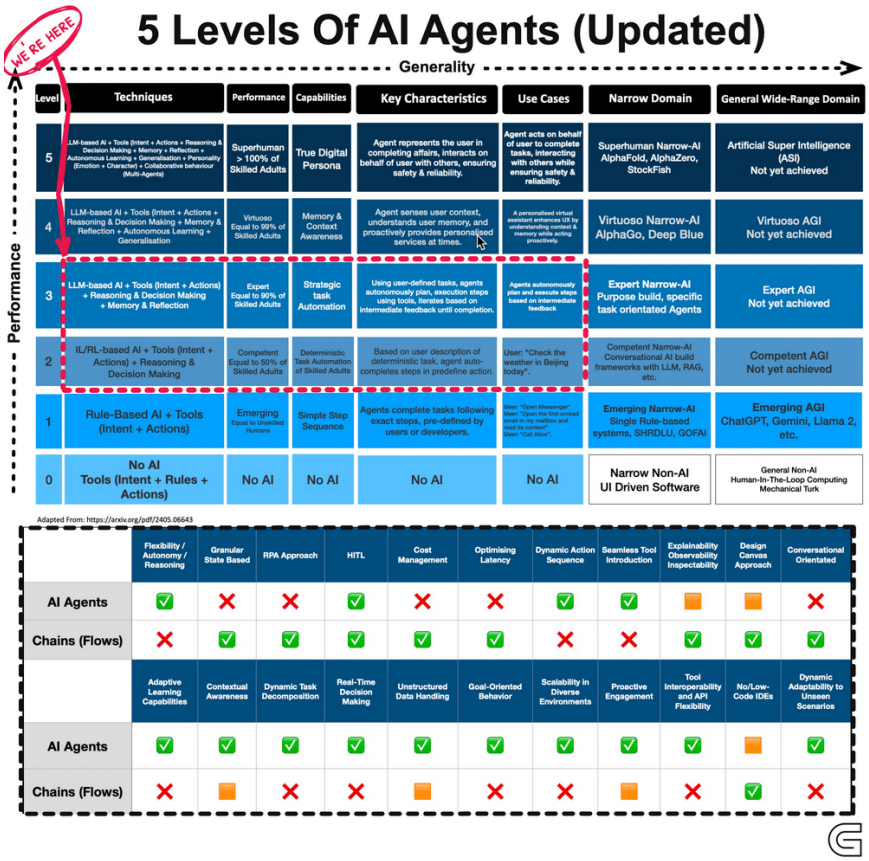

Figure 1: The Five Levels of AI Agent Generality and Performance, illustrating the spectrum from rule-based tools to Artificial Super Intelligence (ASI).

2.2 Retrieval-Augmented Generation and Semantic Search

Retrieval-Augmented Generation (RAG) [4] established the ideas of retrieving information in a context window to generate a model's output. DATOF semantic indexing is a related approach, but it focuses on identifying the right tool and not the right document [6][12]. The latest sentence and document embeddings [12][13] support high-quality semantic matching between user queries and tool descriptions.

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

2.3 The Tool Explosion Problem

As the capabilities of enterprise AI systems increase, the number of candidate tools available to these agents also continues to grow, which leads to the "tool explosion problem," or the challenge of selecting the right tools out of many [5][10]. This magnifies the importance of good orchestration practices; otherwise, overall agent quality degrades due to poor decision-making, slower execution, or poor maintainability. Existing multi-agent architectures like LangChain [14] and AutoGPT [15] only partially address the problem through chaining tool invocation, and they lack systematic semantics-aware tool selection on a scale seen in enterprise tool registries.

2.4 Limitations of Current Approaches

In contrast, most approaches rely on manual tool routing, prompt-based tool selection, or some hardcoded decision logic [3, 11]. This practice is not scalable, difficult to do when the tool registry changes, and results in slow tool selection cycles. Thus, a systematic orchestration architecture like semantic retrieval with multi-criteria scoring is needed, which is the main motivation for our DATOF framework.

3. DYNAMIC AI TOOL ORCHESTRATION FRAMEWORK

3.1 System Architecture Overview

This model proposes a structured orchestration layer between the AI agent and the enterprise tools ecosystem, which includes six major components organized as a pipeline (shown in Figure 3).

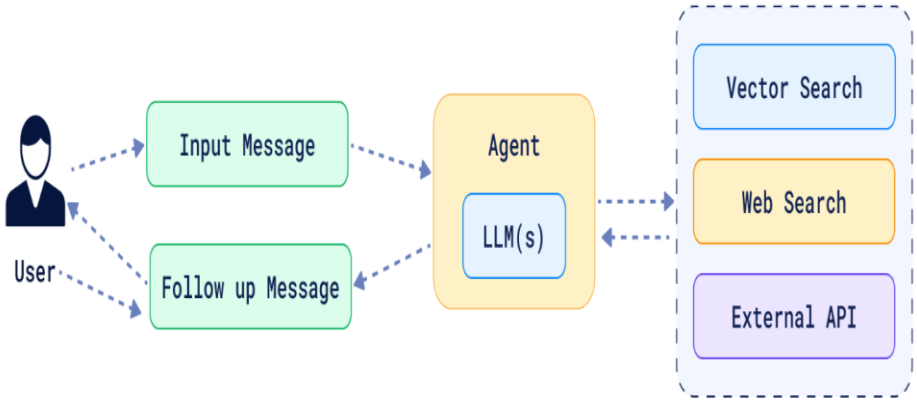

Figure 2: AI agent interaction model showing NLP-driven task execution across data sources, code executors, ML models, and LLM backends.

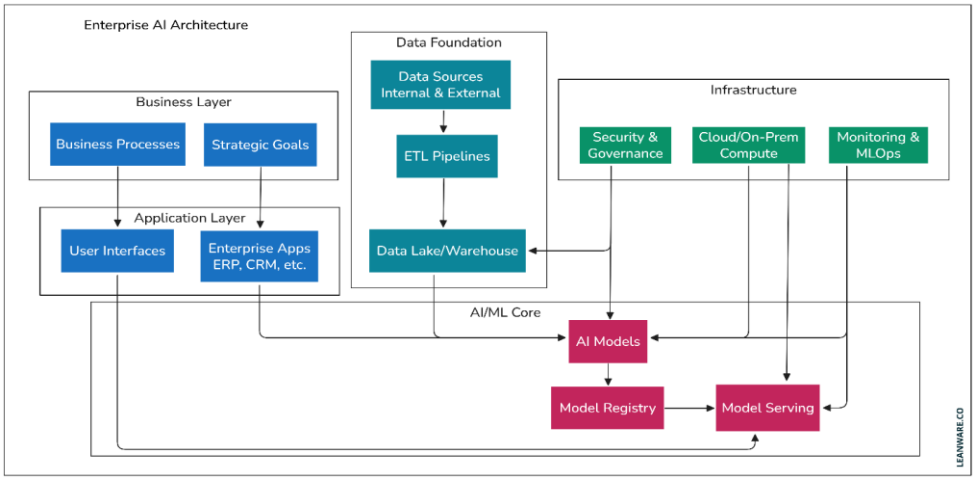

Figure 3: AI Tool Orchestration Architecture

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

The key components are:

Table 1. Key Components of the DATOF Architecture

Component	Role
AI Agent Interface	Receives user requests and returns final responses
Intent Classification Module	Interprets user intent and extracts structured query features
Tool Capability Registry	Maintains structured metadata for all registered tools
Semantic Tool Index	Provides embedding-based similarity search over tool descriptions
Tool Selection Engine	Ranks and selects the optimal tool using multi-criteria scoring
Execution Manager	Validates parameters, invokes tool APIs, and normalizes responses

Orchestration Workflow:

1. The user sends a request to an AI agent interface.
2. The Intent Classification Module interprets the request through critical semantic and contextual features.
3. A second component, the Semantic Tool Index, then retrieves candidate tools based on similarity.
4. The Tool Capability Registry provides metadata for candidate tools.
5. The Selection Engine ranks candidates and chooses the appropriate tool.
6. It then invokes the selected tool and returns the result.

4. TOOL CAPABILITY MODELING

Tools in the system are described using a capability schema, which enables machine-readable discovery and automatic compatibility checking of tools [5][10]. The schema is composed of the following fields.

Table 2. Tool Capability Schema Fields

Field	Description
Tool Name	Unique identifier for the tool
Description	Natural language description of tool functionality
Input Parameters	Expected input types and constraints
Output Format	Response structure and data types
Domain	Functional category (e.g., finance, analytics)
Reliability Score	Historical success rate of tool invocations
Average Latency	Mean execution time in milliseconds

An example tool definition in JSON format is provided below:

{

"tool_name": "ValidateJournalEntry",

"description": "Validates journal entries against enterprise ledger rules",

"domain": "finance",

"inputs": ["journal_data"],

"output": "validation_result",

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

"reliability_score": 0.97,

"avg_latency_ms": 210

}

Leveraging structured metadata enables semantic discovery and quantitative evaluation of tools. The schema is inspired by capability description languages for service-oriented architectures [10][11] and follows emerging standards for the description of tools in LLM agent systems [14].

5. SEMANTIC TOOL INDEXING

A continuous semantic index of all registered tool descriptions [4][6][12] is maintained with dense embedding vectors from a pretrained language model encoder to allow efficient similarity search at query time.

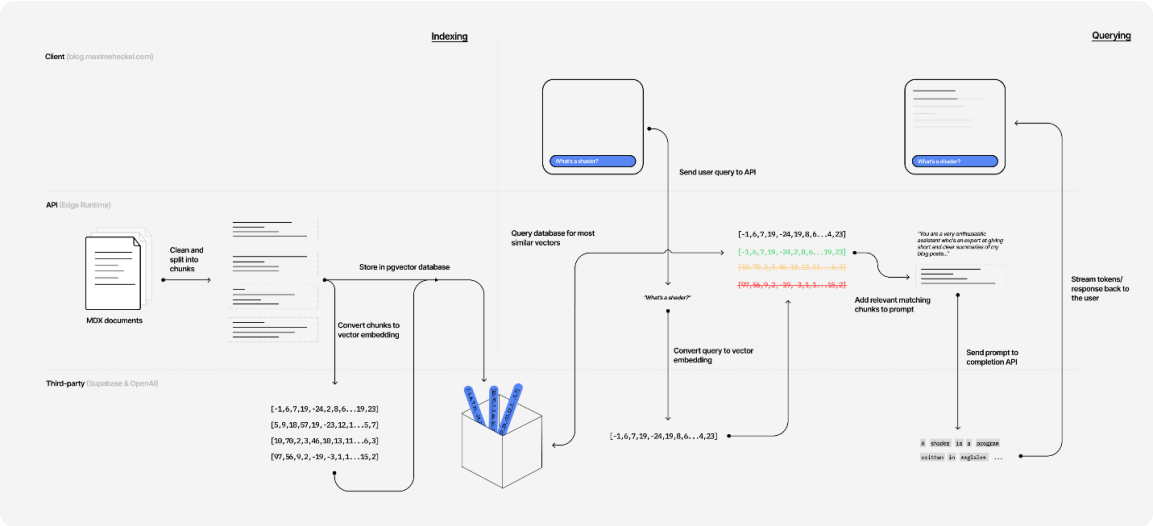

Figure 4: Semantic Tool Discovery

When the user submits a query, the following occurs:

1. Then, a query embedding may be generated from the classified user intent.
2. An approximate nearest-neighbor (ANN) search is performed against the tool index [13].
3. The top k candidate tools are then returned and ranked based on their cosine similarity.

This cuts down the search space from the entire tool registry to a small set of candidates. This means that the time it takes to search grows at a slower rate than the size of the registry [6][12]. The index is incrementally maintained over time, ensuring that newly registered tools can be discovered in real time.

6. DYNAMIC TOOL SELECTION ENGINE

The selection engine then assigns a combined score to each candidate tool based on several operational requirements, as referenced in sources [5] and [9].

Selection Factors:

- The semantic similarity score between the query and the tool description
- Tool reliability score (historical invocation success rate)
- Expected execution latency
- If the parameter matches the current query

The composite ranking score of a candidate tool i is defined as follows:

$$Score_i = (S_i \times W_1) + (R_i \times W_2) - (L_i \times W_3)$$

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

where:

S_i = semantic similarity score

R_i = reliability score

L_i = normalized latency estimate

W_1, W_2, W_3 = tunable weight parameters

Values for W1, W2, and W3 can be flexibly set based on the application's requirements. For example, a higher W3 may be used for latency-sensitive applications, while higher W1 and W2 values may be used for accuracy-critical applications. The selected tool is executed if the combined score is the highest. If there is a tie between the tools, or if a tool does not reach a certain threshold, the engine reverts to a human-in-the-loop fallback.

7. EXECUTION MANAGEMENT

Once selected, the Execution Manager is responsible for the entire invocation lifecycle [7][8]:

Parameter Validation: The input parameters obtained from the user query are validated against the tool's capability schema.

API Invocation: The selected tool API is invoked with the validated parameters.

Error handling: Retry logic or an alternate tool is used for invocation errors.

Response Normalization: The tool output is adapted before sending it back to the AI agent.

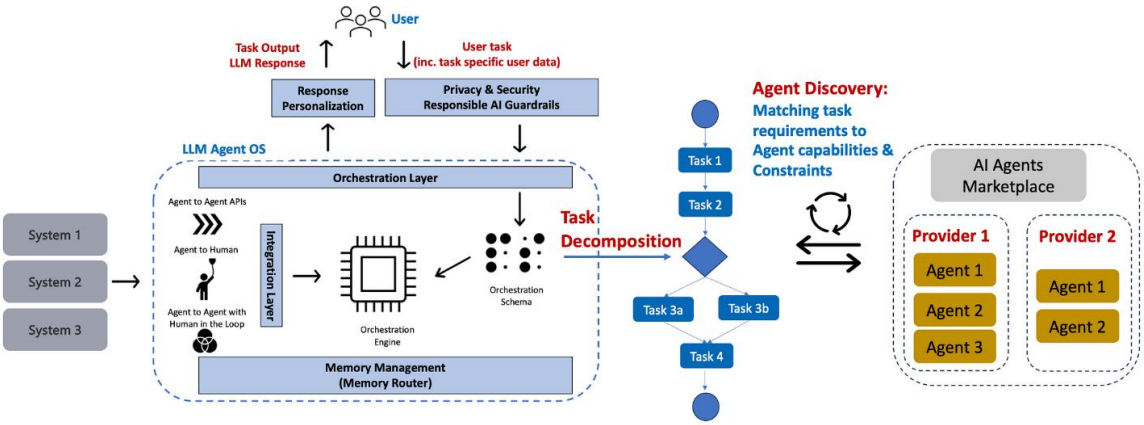

Figure 5: Tool Invocation Workflow

The normalized output is returned to the AI agent, which may return a final output in natural language or may invoke another tool (in the case of multi-step reasoning) [3][15].

8. EXPERIMENTAL EVALUATION

8.1 Experimental Setup

For experimentation, a simulated enterprise setup was created with a tool registry of 200 tools across four distinct functional domains (finance, document processing, workflow automation, and analytics). To evaluate the framework, a dataset of 10,000 user queries was generated, containing a representative distribution of requests using various phrasings, intents, and domains encountered in enterprises.

8.2 Baseline System

In the baseline system, static tool routing was performed using a heuristic that assigns each request to a tool based solely on keywords, with no semantic understanding or scoring.

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

8.3 Evaluation Metrics

The following measurements were used for the evaluation:

Tool Selection Accuracy: Fraction of requests for which the selected tool was correct.

Decision Latency: Mean time from request to tool invocation in seconds.

Scalability: Degradation in performance with the number of tools in the registry, which changes from 50 to 500.

8.4 Results

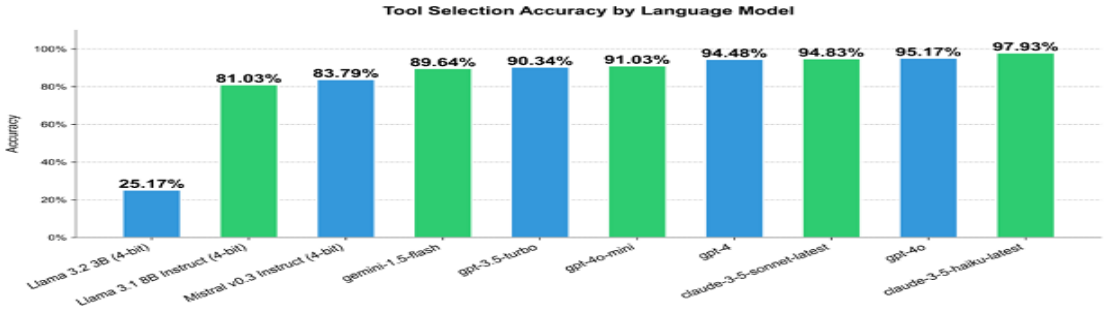

Figure 6: Performance Comparison: Accuracy and Latency

Tool Selection Accuracy:

Table 3. Tool Selection Accuracy

System	Accuracy
Static Routing (Baseline)	61%
DATOF (Dynamic Orchestration)	81%
Improvement	32%

Figure 6 shows a comparison of tool selection accuracy across a range of modern language models that are part of the DATOF pipeline. This helps to put the DATOF accuracy benchmark in context. As shown in Figure 6, smaller quantized models such as Llama 3.2 3B (4-bit) achieved only 25.17% accuracy, while Llama 3.1 8B Instruct (4-bit) and Mistral v0.3 Instruct (4-bit) reached 81.03% and 83.79%, respectively. The mid-tier models did better, with gemini-1.5-flash at 89.64%, gpt-3.5-turbo at 90.34%, and gpt-4o-mini at 91.03%. The highest accuracies were recorded among frontier models: claude-3-5-sonnet-latest at 94.48%, gpt-4 at 94.83%, gpt-4o at 95.17%, and claude-3-5-haiku-latest, achieving the top score of 97.93%. These results indicate that DATOF's orchestration quality scales significantly with the capability of the underlying language model and that pairing DATOF with state-of-the-art models yields near-ceiling tool selection performance in simulated enterprise workloads.

Decision Latency:

Table 4. Decision Latency

System	Mean Latency
Static Routing (Baseline)	1.8 seconds
DATOF (Dynamic Orchestration)	1.3 seconds
Improvement	28%

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Scalability:

When the tool registry was scaled from 50 to 500 tools, DATOF's graph accuracy and latency performance remained the same while the performance of static routing suffered meaningful degradation. This indicates that the semantic indexing and ANN-based retrieval components of DATOF can effectively tame the computational cost of discovering tools as the registry size grows [6][12][13].

9. DISCUSSION

In all three measures, we observe that DATOF provides a statistically important improvement over static routing. In particular, DATOF's ability to improve tool selection accuracy via semantic indexing by capturing intent-level similarity rather than brittle keyword matching gives an improved accuracy of up to 32% [2][4]. The minus (-) 28% decrease in decision time can mostly be attributed to using an ANN-based mechanism for candidate retrieval rather than searching the entire registry [13].

The scalability results would be promising from an enterprise perspective, where tool registries would be continuously growing. The sub-linear scaling behavior of DATOF implies that it could accommodate such growth without restructuring the whole architecture [10][14].

Limitations: This evaluation was performed in a controlled environment. In real enterprise environments, the results may be affected by tool versioning, access control, authentication overhead, and drift in tool reliability. The weights W1, W2, and W3 that define the scores still require manual tuning, which may not be valid for all workloads.

Future Directions: Potential directions for future work include the following:

- Dynamic weight tuning for the scoring function based on reinforcement learning [9].
- Hierarchical tool planning for multi-step agentic workflows [3][15].
- Adaptive tool reliability scoring using online learning.
- Integrate with the latest LLM tool use and agent frameworks [14].

CONCLUSION

The Dynamic AI Tool Orchestration Framework (DATOF) is a modular architecture for scalable smart tool selection for enterprise AI agents. DATOF's design combines a semantically indexed tool repository, structured modeling of tool capabilities, multi-criteria tool selection, and execution management, enabling effective tool discovery and invocation at scale.

Experiments suggest that the system considerably improves tool selection accuracy by 32% and decision timestep latency by 28% over static routing baselines and that the tool routing system scales to registries with up to 500 tools. These results validate the system's design and the value of leveraging semantic retrieval with adaptive scoring for orchestrating enterprise AI tools.

To support the evolution of enterprise AI tools, orchestration systems like DATOF are needed to create reliable, efficient, and maintainable AI agents. Future work will extend this research to reinforcement learning-based orchestration, hierarchical multi-tool workflow planning, and deploying orchestra-based production agents in real-world settings.

REFERENCES

[1] Tom B. Brown, et al., "Language Models are Few-Shot Learners," in arXiv, 2020. Available: https://arxiv.org/pdf/2005.14165
[2] Timo Schick, et al., "Toolformer: Language Models Can Teach Themselves to Use Tools," in arXiv, 2023. Available: https://arxiv.org/pdf/2302.04761
[3] Shunyu Yao, et al., "ReAct: Synergizing Reasoning and Acting in Language Models," in Proc. International Conference on Learning Representations (ICLR), 2023. Available: https://arxiv.org/pdf/2210.03629
[4] Patrick Lewis, et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in arXiv, 2020. Available: https://arxiv.org/pdf/2005.11401

[5] Elias Lumer, et al., "Tool-to-Agent Retrieval: Bridging Tools and Agents for Scalable LLM Multi-Agent Systems: Applications to Spherical Attention for Gene Regulatory Networks," in arXiv, 2025. Available: https://arxiv.org/pdf/2511.01854

[6] Jeff Johnson, et al., "Billion-Scale Similarity Search with GPUs," in arXiv, 2021. Available: https://arxiv.org/pdf/1702.08734

[7] Aman Madaan, et al., "Self-Refine: Iterative Refinement with Self-Feedback," in Proc. Advances in Neural Information Processing Systems (NeurIPS), 2023. Available: https://arxiv.org/pdf/2303.17651

[8] Shishir G. Patil, et al., "Gorilla: Large Language Model Connected with Massive APIs," arXiv preprint arXiv:2305.15334, 2023. Available: https://arxiv.org/pdf/2305.15334

[9] Jason Wei, et al., "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," in Proc. Advances in Neural Information Processing Systems (NeurIPS), 2022. Available: https://openreview.net/pdf?id=_VjQlMeSB_J

[10] Qingyun Wu, et al., "AutoGen: Enabling Next-Generation LLM Applications via Multi-Agent Conversation," arXiv preprint arXiv:2308.08155, 2023. Available: https://arxiv.org/pdf/2308.08155

[11] Yongliang Shen, et al., "HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in HuggingFace," in Proc. Advances in Neural Information Processing Systems (NeurIPS), 2023. Available: https://proceedings.neurips.cc/paper_files/paper/2023/file/77c33e6a367922d003ff102ffb92b658-Paper-Conference.pdf

[12] Nils Reimers and Iryna Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," in Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP), 2019. Available: https://arxiv.org/pdf/1908.10084

[13] Yu. A. Malkov and D. A. Yashunin, "Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 42, no. 4, pp. 824–836, 2020. Available: https://arxiv.org/pdf/1603.09320

[14] DRL Team, "LangChain: Building LLM Applications through Composability," in Data Root Labs. Available: https://datarootlabs.com/blog/langchain-building-llm-applications-through-composability

[15] T. Significant Gravitas, "AutoGPT: An Autonomous GPT-4 Experiment," GitHub Repository, 2023. Available: https://github.com/Significant-Gravitas/AutoGPT