Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

AWMF: Agentic Workflow and Multi-Model Fusion for Enhancing Domain-Specific AI

Zhiyong He
Nanjing University of Industry Technology
Zhengyang Li
DigiPen Institute of Technology Redmond
Kun Qi
kunqishenzhen@126.com

Shenzhen Polytechnic University
Duo Jia
Inner Mongolia Institute of Metrology Testing and Research

Research Article

Keywords: Large Language Models, Agentic Workflow, Multi-Model Fusion, Retrieval-Augmented Generation,Resource Efficiency

Posted Date: June 8th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-8486988/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

AWMF: Agentic Workflow and Multi-Model Fusion for Enhancing Domain-Specific AI

Zhiyong He[1],**Zhengyang Li**[2] ,**Kun Qi**[3]*,**Duo Jia**[4]

[1]School of Electronic and Information Engineering/School of Integrated Circuits,Nanjing University of Industry Technology,Nanjing,China
[2]Department of Computer Science,DigiPen Institute of Technology Redmond,Washington,USA
[3]School of Electronic and Communication Engineering, Shenzhen Polytechnic University, Shenzhen, China
[4] Inner Mongolia Institute of Metrology Testing and Research,Hohhot ,China

*** Correspondence:**
Kun Qi
kunqishenzhen@126.com

Abstract: The increasing complexity of real-world tasks in AI highlights the need for more efficient and adaptable systems that can effectively leverage various specialized models to manage complex, multi-dimensional challenges. In this paper, we propose the Automated Workflow and Multi-Model Fusion (AWMF) framework, a computational framework designed to coordinate the execution of specialized AI models in multi-task scenarios. By automating key steps such as task decomposition, model selection, and workflow optimization, AWMF enables intelligent task execution while maintaining a balance among performance, accuracy, and resource constraints. We present a hybrid approach that combines Large Language Models (LLMs) with predefined semantic operators, supporting accurate task parsing, dynamic workflow creation, and the seamless integration of external data sources. We evaluate AWMF through several case studies, including the MMLU benchmark, Chinese sentiment classification, external data augmentation via a RAG approach, and historical execution optimization. Experimental results demonstrates that AWMF outperforms individual models in accuracy and adaptability, particularly in tasks requiring external knowledge and cross-lingual capabilities. The integration of historical data further enhances performance, demonstrating AWMF's effectiveness in diverse, real-world environments.

Keywords: Large Language Models; Agentic Workflow; Multi-Model Fusion; Retrieval-Augmented Generation;Resource Efficiency

1 Introduction

The rapid progress in large language models (LLMs) has significantly advanced artificial intelligence [1,2], offering efficient tools for understanding and generating text. These models have demonstrated their effectiveness across a variety of natural language processing tasks, from answering simple queries to addressing complex questions [3]. Despite their capabilities, LLMs still encounter important limitations when handling tasks that require deep reasoning or specialized domain knowledge. Moreover, fine-tuning these models for specific applications is computationally demanding, often requiring large amounts of data and extensive computing resources. Such challenges pose difficulties for researchers and organizations aiming to use LLMs effectively in practical scenarios, especially when computational resources are limited.

To address these limitations, recent efforts have explored using multiple smaller models, each specializing in different aspects of a task. This modular approach efficiently combines the strengths of various models, improving overall performance without the extensive computational requirements of large-scale models. However, effectively coordinating these smaller models remains challenging, as it involves intelligently selecting, managing, and executing task-specific models to ensure accuracy and efficient resource use. While systems such as AutoGPT, BabyAGI, and MetaGPT have made progress in multi-agent systems, they still face issues like poor controllability, instability, and reproducibility, especially in specialized domains [1,5,6,7,8]. Furthermore, frameworks like Retrieval-Augmented Generation (RAG) emphasize retrieval-based approaches but struggle with unstructured data or complex query planning [9,10,11]. While recent frameworks such as SPP [12], EvoAgent [13], AutoAgents [14] and SwarmAgentic [15] have made strides in agentic system automation, but they still fall short of meeting all autonomy criteria, particularly in From-Scratch Agent Generation and self-optimizing collaboration. While these systems rely on predefined templates or seed agents, they lack the flexibility to autonomously generate and adapt agents based on task requirements. In contrast, our proposed semantic operator-based framework enables From-Scratch Agent Generation, self-optimizing agent functionality, and dynamic collaboration, offering a more scalable and fully autonomous solution. This innovation allows agents to adapt autonomously and collaborate effectively in complex environments, addressing limitations present in frameworks like EvoAgent and AFlow [16].

This paper introduces the Agentic Workflow and Multi-Model Fusion (AWMF) framework, an innovative approach designed to overcome these challenges. While AWMF leverages established techniques, such as LLMs, ensemble methods, RAG, and vector databases, it uniquely integrates key components—automated task parsing, intelligent model selection, multi-model fusion, and external data augmentation—into a unified, adaptive pipeline. This integration enables AI systems to efficiently manage tasks involving diverse data sources, multiple specialized models, and varied resource constraints. AWMF dynamically constructs and optimizes workflows based on specific task requirements, maximizing both accuracy and resource efficiency. Unlike traditional fixed-structure models, AWMF adapts workflows dynamically to optimize computational resources while maintaining high performance across diverse tasks. This flexibility is especially valuable in real-world settings, where resource constraints—such as limited computing power, time limitations, or varying task complexities—often pose significant obstacles. Moreover, AWMF's ability to intelligently select and orchestrate specialized models significantly enhances performance in specialized domains, overcoming the limitations of single-model approaches. By leveraging the strengths of multiple models, AWMF ensures the best model is selected for each task component, leading to improved overall outcomes. Additionally, the integration of external data via APIs enriches the framework's context in real-time, enabling more accurate and timely responses. This dynamic, feedback-driven adaptation further contributes to AWMF's robustness and continuous improvement in task execution.

In summary, this paper offers three primary contributions:

- **Automated Workflow and Multi-Model Fusion Framework (AWMF):** We propose the AWMF framework, which automates the orchestration, selection, and execution of multiple specialized models to address complex multi-task scenarios. The framework dynamically constructs optimized workflows and integrates external data, improving efficiency, accuracy, and adaptability for diverse real-world tasks.
- **Hybrid Task Parsing and Semantic Operator Identification:** AWMF presents a hybrid method for parsing tasks and identifying semantic operators. By combining Large Language Models (LLMs) with predefined semantic operators and rule-based validation, this approach effectively decomposes complex tasks into manageable, executable components. It ensures

reliable execution even with ambiguous or poorly specified inputs, providing a strong foundation for efficiently constructing subsequent workflows.

- **Comprehensive Experimental Validation and Performance Gains:** We validate the AWMF framework through extensive experiments, demonstrating significant performance improvements across various tasks, including multi-task scenarios, sentiment classification, and external data integration. Our results show notable gains in accuracy, efficiency, and robustness, highlighting how multi-model fusion, dynamic workflow optimization, and external data integration enhance system performance, making AWMF highly effective for real-world applications.

2 Overview of AWMF

2.1 Motivation

Recent advancements in LLMs have greatly improved their ability to understand and generate human-like text with increased accuracy and context awareness. Despite these improvements, sophisticated models still face challenges in consistently providing accurate responses, especially in complex or highly specialized tasks requiring detailed reasoning. Furthermore, fine-tuning these extensive models demands significant computational resources and large datasets, creating substantial obstacles for many researchers and organizations. To overcome these issues, orchestrating multiple smaller models provides a more efficient and modular solution, enabling different specialized models to collaboratively achieve higher precision in demanding question-answering tasks. This approach also mitigates challenges associated with limited hardware capabilities, where running large-scale models might be impractical; instead, strategically deploying smaller, task-specific models ensures feasible inference and improved accuracy. Additionally, by combining smaller models in an ensemble, we are able to reduce the instability inherent in single large models, increasing the overall robustness of the system. This is a core design principle of our framework: balancing the accuracy and efficiency across multiple models to ensure reliable performance across a variety of real-world tasks. Moreover, this modular approach helps reduce the computational costs, making the system more accessible for a wider range of users and use cases. The ability to dynamically select and orchestrate specialized models, rather than relying on a single monolithic model, ensures that performance is optimized while still adhering to resource constraints. Therefore, there is an urgent need for techniques capable of effectively integrating multiple specialized models to enhance performance while addressing accuracy demands and resource constraints.

2.2 Key Idea

We propose the Automated Workflow and Multi-Model Fusion (AWMF) framework, which automates the orchestration, selection, and execution of AI models for complex multi-task scenarios. Traditional AI model applications often face limitations such as rigid execution patterns, limited adaptability, and insufficient integration of external knowledge. The AWMF framework addresses these challenges by dynamically constructing automated workflows using predefined execution templates and intelligent scheduling mechanisms.

We predefine a set of task-specific execution modules, including intelligent model selection, multi-model fusion, external API integration, and automated script generation. These modules combine rule-based methods with optimizations driven by LLMs. When presented with a task, the framework leverages LLMs and workflow optimization methods to automatically interpret requirements, select suitable models, orchestrate multi-step workflows, optimize them for efficiency and accuracy, and iteratively execute with real-time feedback and adaptive adjustments.

By integrating automated workflow generation, intelligent model selection, and real-time external data integration, AWMF substantially improves model adaptability, execution efficiency, and overall task performance in complex and dynamic environments.

2.3 Challenges and Solutions

To develop an efficient automated workflow system with intelligent decision-making capabilities, dynamic optimization mechanisms, and adaptive execution characteristics, AWMF must address challenges posed by three key issues: task parsing, model scheduling, and execution optimization. Additionally, it must explore corresponding solutions to ensure optimal performance and adaptability.

C-I: Difficulty in Accurate Multi-Step Task Decomposition. A significant methodological challenge lies in automatically breaking down complex tasks into clear, semantically coherent steps (such as data preprocessing, model invocation, and result integration). Traditional rule-based parsing methods frequently struggle with nuanced user requests, resulting in inaccurate or inefficient task decomposition.

S-I: LLM Reasoning and Rule-Based Validation. To address this challenge, we propose a hybrid approach that integrates large language models with a predefined set of semantic operators. In this approach, the large language models are responsible for high-level parsing and logical reasoning to decompose a given query into constituent sub-tasks. Concurrently, a rule-based mechanism refines and validates every step to ensure accuracy and consistency. This dual-layer strategy facilitates reliable task decomposition without requiring extensive manual rule development.

C-II: Trade-off Between Accuracy and Efficiency in Multi-Model Fusion. When multiple models (e.g., transformer-based, graph-based, or traditional machine learning models) are available for a given task, determining which models to select and in what sequence to execute them becomes a critical methodological challenge. Prioritizing accuracy alone can significantly increase computational costs, whereas emphasizing efficiency may lead to reduced performance.

S-II: Multi-Objective Optimization for Balancing Accuracy and Efficiency. We propose a multi-objective optimization framework that evaluates models based on accuracy, inference speed, and resource consumption. By utilizing techniques such as weighted Pareto optimization and adaptive scheduling, the framework dynamically adjusts integration weights and execution sequences for each model. This approach optimizes overall performance utility while concurrently reducing latency and minimizing resource utilization.

C-III: Enhancing Adaptability Through Iterative Optimization and Self-Reflective Reasoning. A purely static pipeline risks missing opportunities for improvement when facing ambiguous or evolving tasks. Designing a framework capable of iteratively refining pipelines, updating execution strategies, and performing self-correction to maintain alignment with user objectives presents a significant methodological challenge.

S-III: Feedback-Driven Loop for Adaptive and Self-Reflective Pipelines. We implement a feedback-driven loop that allows the system to evaluate intermediate outcomes and self-reflect on possible shortcomings. By utilizing LLMs for meta-reasoning, the pipeline dynamically adjusts operators, updates model hyperparameters, or re-invokes specific modules. This iterative approach progressively enhances system performance, reduces reliance on human intervention, and increases adaptability in practical scenarios.

3 System Architecture

In this section, we first present an overview of the Adaptive Workflow Multi-model Fusion (AWMF) architecture, followed by an in-depth description of each constituent module.

3.1 Design of AWMF

The proposed AWMF framework orchestrates complex tasks through a structured, multi-phase pipeline, as illustrated in Figure 1. ① First, user requests undergo Task Parsing and *Semantic Operator Identification*, leveraging an LLM alongside predefined validation rules to translate high-level user inputs into detailed semantic operators, such as Retrieve, Summarize, and Generate. Additional operators and their respective functions are elaborated in Table 1. ② Subsequently, the Dynamic *Workflow Generation and Optimization* module constructs and optimizes task-specific workflows via intelligent scheduling strategies and predefined templates, balancing objectives such as accuracy and resource efficiency. ③To enrich the contextual depth and ensure real-time data incorporation, the *External API Integration and Data* Augmentation module systematically fetches supplemental data from external sources. ④ Following this, the *Multi-Model Fusion and Adaptive* Model Selection module strategically selects optimal models from a diverse repository, employing ensemble techniques (e.g., voting or stacking) when advantageous for task performance. ⑤The *Automated Script Generation and Execution* module subsequently compiles auxiliary processing tasks into executable scripts, ensuring efficient management and runtime monitoring during runtime. ⑥Lastly, the *Feedback-Driven Self-Optimization* module iteratively generates user responses and refines workflow configurations based on performance metrics and intermediate outcomes. Through this comprehensive, end-to-end design, AWMF effectively achieves a robust balance between accuracy, operational efficiency, and adaptability in complex multitask scenarios.

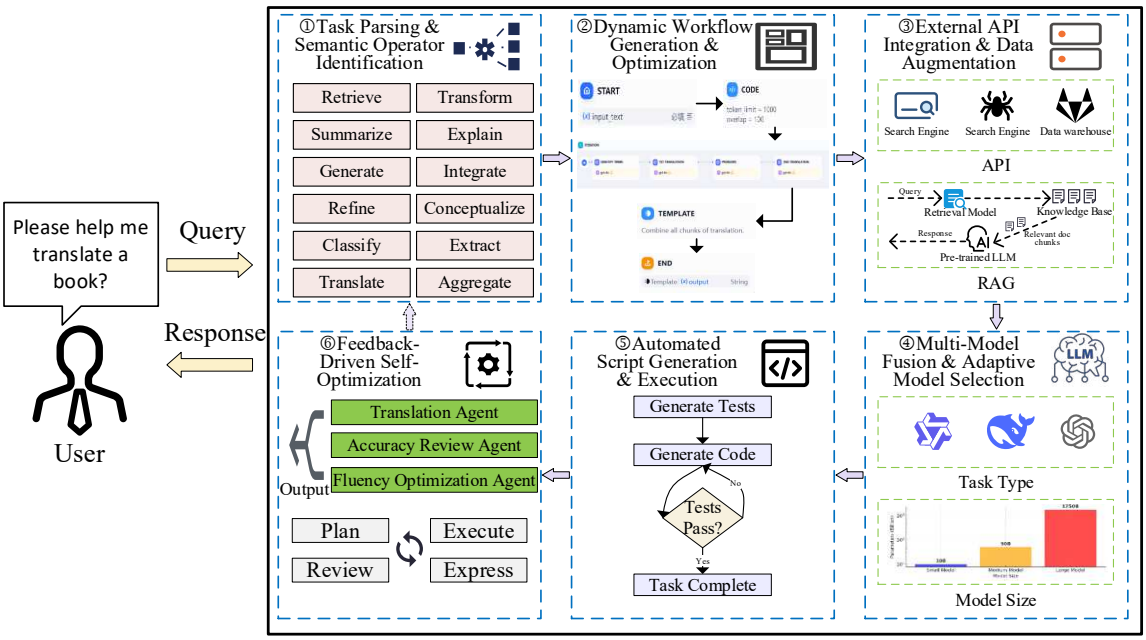


Figure 1. The Workflow of AWMF

3.2 Task Parsing and Semantic Operator Identification

The *Task Parsing and Semantic Operator Identification* module initiates its workflow by processing user-specified tasks, typically expressed in natural language, and transforming them into structured representations suitable for downstream workflow construction. This transformation process can be mathematically modeled as mapping the input task T_{user} to a set of semantic operators $\{O_1, O_2, \ldots, O_n\}$. An LLM acts as the core parsing engine, systematically interpreting complex linguistic constructs, including task objectives, data requirements, and contextual constraints. The parsing process aims to optimize the decomposition of the task into executable components. Concurrently, the outputs generated by the LLM

are rigorously validated and refined by a rule-based mechanism to guarantee robustness and minimize semantic ambiguity, ensuring the parsed tasks remain contextually precise and executable.

Specifically, the LLM ingests user requests, such as "Identify sentiment and summarize recent news articles," and maps them to corresponding semantic operators, including "Extract," "Classify," and "Summarize." This mapping can be viewed as a function $f(T_{\text{user}}) \rightarrow \{O_1, O_2, \ldots, O_n\}$, where f represents the LLM's parsing function. Subsequently, these operators are organized into a coherent logical sequence. We categorize and elaborate upon 12 distinct operator types, as illustrated in Table 1. Furthermore, each operator is tagged with detailed metadata (such as required inputs, expected outputs, and processing time constraints) to enable subsequent modules to optimize execution scheduling effectively. The validation mechanism, $V(O_i)$, ensures that each operator satisfies the constraints and is ready for execution. The validation mechanism $V(O_i)$ is defined as follows:

$$V(O_i) = \begin{cases} 1 & \text{if operator } O_i \text{ is valid,} \\ 0 & otherwise. \end{cases} \tag{1}$$

Through a fusion of data-driven parsing and stringent rule-based checks, this module reliably produces semantically precise task decompositions even when faced with ambiguous user inputs. The total accuracy of the parsed operators can be quantitatively assessed as the proportion of valid operators: $\frac{\sum_{i=1}^{n} V(O_i)}{n}$. This ensures that the parsing process is both accurate and efficient.

Table 1. Summary of Semantic Operators

Operator	Description
Retrieve	Access external databases or resources to gather relevant and current information, enhancing responses with precise and updated details.
Summarize	Condense extensive textual content into succinct summaries, facilitating efficient information assimilation and comprehension.
Generate	Produce structured, contextually appropriate textual outputs from given input data, primarily employed in crafting finalized responses.
Refine	Modify and enhance input texts to ensure greater clarity, coherence, and accuracy, thus improving subsequent processing stages.
Classify	Categorize or tag entities through either large language models or external machine learning classifiers, ensuring precise categorization tasks.
Translate	Convert textual content from a source language into a target language by utilizing LLMs or external translation services, ensuring semantic fidelity.
Transform	Convert data formats, such as transforming structured tables into natural-language descriptions, thereby facilitating interpretability.
Explain	Provide clear rationales or insights behind model decisions or outcomes, enabling introspection into the model's inferential processes.
Integrate	Synthesize information from diverse sources into a unified and coherent textual response.
Conceptualize	Extract and emphasize essential concepts within complex textual data, thus simplifying query comprehension and execution.
Extract	Selectively retrieve specific and pertinent information from datasets, analogous to the projection operation in SQL.

Aggregate	Conduct aggregation operations, including calculating sums, averages, or maxima based on input data, for comprehensive statistical insights.

3.3 Dynamic Workflow Generation and Optimization

Once the semantic operators $O_1, O_2, \ldots, O_n$ are identified, the *Dynamic Workflow Generation and Optimization* module begins constructing them into an executable pipeline. This process can be modeled as mapping the set of identified operators into an initial workflow W_{initial}. Using a predefined library of workflow templates, the module then dynamically adjusts this workflow by inserting, rearranging, or omitting steps to align with specific operator constraints. The initial workflow is refined to create an optimized version, W_{initial}, that fits the task requirements more efficiently.

A key feature of this module is its multi-objective optimization approach, which balances several conflicting objectives such as accuracy, execution time, resource consumption, and cost constraints. The optimization objective is to maximize the accuracy A while minimizing execution time T, resource consumption R, and cost C. This can be represented as:

$$\text{Objective}(A, T, R, C) = \alpha A - \beta T - \gamma R - \delta C. \tag{2}$$

Here, $\alpha, \beta, \gamma, \delta$ are weights that determine the relative importance of each factor. The optimization process seeks to find the best balance, adjusting the workflow to maximize performance while staying within resource and time constraints.

Depending on these constraints, operators may be parallelized to decrease latency or serialized to enhance the reliability of intermediate outputs. Internally, the optimization process leverages sophisticated strategies, such as weighted Pareto fronts [17] and priority-based heuristics [18], ensuring the resultant workflows achieve optimal equilibrium among performance, cost-effectiveness, and reliability, irrespective of task complexity.

3.4 External API Integration and Data Augmentation

Workflows often benefit from the integration of external data sources, such as database queries, real-time knowledge graphs, or specialized APIs that provide domain-specific insights like financial data, weather forecasts, or medical records. The *External API Integration and Data Augmentation module* facilitates this process by systematically interfacing with these external sources to acquire, preprocess, and incorporate supplemental data into the workflow. The module relies on an internal API management system, which manages key details such as connection configurations, authentication keys, and rate-limiting strategies for various external data providers. The system is designed to automatically detect when a workflow operator requires external data, and it subsequently invokes the appropriate API.

When an API request is made, raw data responses are returned in various formats (e.g., JSON, CSV). These responses are then converted into a standardized format suitable for the workflow. The raw data D_{raw} represents the unprocessed data from the external source, and $Adapter(D_{raw}, F)$ is a function that processes and converts this raw data into an adapted form that fits the workflow's needs. The adapter function can be expressed as:

$$D = Adapter(D_{raw}, F). \tag{3}$$

where F refers to the format (e.g., JSON or CSV) that the data needs to be converted to.

To enhance efficiency and minimize latency, the module incorporates a caching layer, which temporarily stores frequently accessed data. This caching mechanism reduces redundant API calls and ensures faster data retrieval. The caching function can be described as:

$$V(O_i) = \begin{cases} Retrieve\ from\ cache & if\ D\ is\ cached, \\ Fetch\ from\ API & if\ D\ is\ not\ cached \end{cases} \tag{4}$$

where D represents the adapted data required by the workflow operator O_i, which could be any type of external data needed for a given task (e.g., financial data, weather forecast, or medical records). The system checks whether the data is already cached. If so, it retrieves it directly; otherwise, it fetches the data from the external source.

This integration allows operators to gain immediate access to essential external data, enriching the overall task by improving its comprehensiveness and accuracy. The incorporation of up-to-date, domain-specific data significantly enhances the quality of the task's execution, especially in scenarios requiring real-time or specialized information.

3.5 Multi-Model Fusion and Adaptive Model Selection

Given that different tasks and subtasks may be best addressed by distinct computational models, the *Multi-Model Fusion and Adaptive Model Selection* module takes an adaptive approach to model utilization. It manages a comprehensive repository $\mathcal{M}$, which includes classical machine learning models (e.g., random forests, SVMs), state-of-the-art deep learning architectures (e.g., transformers, graph neural networks), and LLMs of various sizes (e.g., 3B, 7B, 70B parameters). The repository $\mathcal{M}$ contains models with different capabilities, allowing the system to choose the most appropriate model depending on the task requirements.

The model selection engine operates by referencing task-specific metadata, which includes the input domain (the type of data being processed, such as text, images, or structured data), historical performance metrics (e.g., accuracy, execution time), and resource requirements (e.g., computational power and memory usage). The engine compares these requirements with the characteristics of the models in the repository to identify the model that best matches the task at hand. This selection process can be mathematically described by the function $S(\mathcal{D})$, where $\mathcal{D}$ represents the task's metadata. The engine selects the model M_{selected} that minimizes the difference between the task's needs and the model's capabilities.

If the selected model M_{selected} does not meet the performance threshold (such as accuracy, resource efficiency, or execution time), the system will switch to a larger model M_{larger}. This adjustment is based on a performance evaluation where the initial model's performance is compared to the predefined threshold. If P_{selected} (the performance of the selected model) does not meet the required criteria, the system will evaluate a larger model with more parameters or computational resources. This ensures that the system maintains high performance across different tasks, particularly when computational resources allow for scaling.

To ensure continuous improvement, the system regularly updates its repository by integrating newly available models and performance data. As new models are developed and evaluated, they are added to $\mathcal{M}$, and the selection process is updated to reflect the latest advancements in model architectures and performance. This dynamic update process ensures that the model selection engine always has access to the best available models, ensuring sustained optimality and adaptability to evolving task requirements.

3.6 Automated Script Generation and Execution

Once a workflow has been assembled, the system must operationalize it by generating and executing executable code. The *Automated Script Generation and Execution* module automates this transformation by mapping operator-level instructions (e.g., "Classify whether a sentence is sarcastic") into scripts written in Python, SQL, or shell commands. Formally, this process can be described as a translation function $G: O \rightarrow C$, where an operator O is converted into a corresponding code snippet C. An LLM-based generator ensures that each operator is translated into a precise and deterministic function, yielding clearly defined outputs such as Boolean labels or structured data tables.

After code generation, an execution engine takes responsibility for deploying the scripts within the computational environment. It manages resource allocation, monitors runtime monitoring, and records execution logs $L = \{l_1, l_2, \dots, l_m\}$, where each l_i corresponds to a record of inputs, execution states, and outputs. This logging mechanism not only supports reproducibility but also provides a basis for performance evaluation and subsequent feedback-driven optimization.

In the event of operational failures (such as runtime errors, invalid outputs, or system crashes), the execution engine triggers a fault-tolerant recovery mechanism. This can be expressed as an error-handling function $R(E)$, where an error E invokes either parameter adjustment and retry, or restoration from a checkpointed state. Such recovery ensures that transient issues do not disrupt the workflow, while stable fallbacks guarantee task completion.

3.7 Feedback-Driven Self-Optimization

The *Feedback-Driven Self-Optimization* module closes the loop of the AWMF framework by delivering workflow-generated responses back to the user while simultaneously collecting performance metrics (such as latency, error rates, and user satisfaction) to drive continuous improvement. Each operator execution is accompanied by structured logging, producing a performance dataset

$$\mathcal{P} = \{(O_i, t_i, e_i, u_i) \mid i = 1, \dots, n\}, \tag{5}$$

where O_i is the operator, t_i is execution time, e_i is the error rate, and u_i is a measure of user satisfaction. This dataset enables precise identification of bottlenecks and suboptimal execution points.

A feedback optimization engine then analyzes these metrics to propose refinements to the workflow. Adjustments may include reordering operator sequences, modifying ensemble weights, or introducing new operators into the workflow. For consistently underperforming models, adaptive measures such as re-selection of alternative models or hyperparameter tuning are automatically triggered. Similarly, persistent failures in external API calls result in fallback strategies, such as retrieving cached data or switching to more reliable endpoints.

This optimization process can be described as an iterative refinement cycle:

$$W^{(k+1)} = \mathcal{F}\left(W^{(k)}, \mathcal{P}^{(k)}\right), \tag{6}$$

where $W(k)$is the workflow configuration at iteration k, $\mathcal{P}^{(k)}$ is the performance dataset from that iteration, and $\mathcal{F}$ is the feedback function that applies modifications based on observed outcomes. Through repeated application of this cycle, the workflow gradually evolves to improve effectiveness and efficiency.

3.8 Domain Knowledge Base

We construct a knowledge repository that is primarily focused on domain-specific knowledge to optimize the performance of intelligent agents. This repository consists of three primary data types: web search data, domain-specific knowledge, and historical execution records, each contributing uniquely to the agent's learning and task execution.

The first category, web search data, plays a crucial role in acquiring up-to-date information. The internet serves as a vast repository, and web searches through open APIs, such as Google Search, facilitate rapid retrieval of diverse content. These search engines leverage advanced technologies to gather information from global websites, news outlets, academic literature, and more. The real-time nature and diversity of this data are particularly valuable for intelligent agents, as they provide timely updates on global events and trends. Additionally, the variety of formats and domains enables the integration of cross-domain knowledge, expanding the language model's understanding and supporting complex, multi-domain problem-solving tasks.

The second category, domain-specific knowledge, refers to specialized information crucial for refining a model's problem-solving capabilities. This data is sourced from various avenues, such as academic

literature, web crawling, and expert interviews. Upon acquisition, the data undergoes rigorous cleaning and organization to eliminate errors, duplicates, and incomplete entries, ensuring the knowledge base's reliability and accuracy. Advanced data cleaning algorithms are employed to remove irrelevant information, impute missing values, and resolve inconsistencies. The data is then categorized and annotated with specific classification schemes, enabling systematic organization and improved usability for intelligent agents.

The third category, historical execution records, includes detailed data generated by intelligent agents during task execution. These records, which encompass operational processes, parameters, outcomes, and feedback, provide valuable insights for learning and optimization. From a learning perspective, historical records allow agents to refine their knowledge and performance through analysis. Successful cases offer strategies for future tasks, while failures provide opportunities for root cause analysis and error prevention. In terms of optimization, historical records help agents identify weaknesses in their performance. For example, if recurring tasks expose inefficient parameter settings that lead to suboptimal results, the agents can adjust these parameters for improved future outcomes. Through systematic analysis, agents can evaluate the effectiveness of different methods and strategies, ultimately enhancing operational quality and efficiency.

Knowledge vectorization, the process of converting textual knowledge into vector representations, is another key component of the repository. This transformation allows for the numerical storage and efficient retrieval of knowledge. Once vectorized, the knowledge is stored in vector databases, which use specialized data structures and indexing methods to facilitate rapid similarity searches, improving the efficiency of information retrieval. After constructing the knowledge repository, a unified API query interface is implemented to provide real-time access to the knowledge base for external applications. This interface ensures that intelligent agents can efficiently retrieve and apply domain-specific knowledge, enhancing their problem-solving capabilities.

4 Experiment

This section introducesthe experimental objectives, setup, and evaluation methodologies employed to rigorously assess the performance of the AWMF framework.

4.1 Experimental Settings

Experimental Objectives. The primary goal of this experiment is to evaluate the effectiveness of the AWMF framework across multiple dimensions. First, we aim to assess the impact of intelligent model selection and multi-model fusion on improving accuracy and robustness in multi-task scenarios. To achieve this, we implement ensemble learning techniques and analyze their ability to improve performance in diverse subject areas. Second, we seek to validate the framework's capability to automatically generate and dynamically configure predefined workflows, evaluating their efficiency, reliability, and flexibility in real-world task execution. Lastly, we examine the integration of external APIs and automated script generation, exploring their potential to reduce task execution time and improve accuracy by dynamically adapting task strategies based on the context.

Fairness Considerations. In this study, the experimental evaluation is conducted under an open-book task setting, where the AWMF framework leverages RAG and historical data to supplement the model's knowledge. In contrast, closed-book models (e.g., OpenAI's o1) rely solely on their internal knowledge base and do not have access to external information. To ensure a fair comparison, all models were evaluated using the same prompt settings and decoding strategies.

We recognize the potential concern regarding the overlap between historical execution records and test items. To address this issue and ensure the fairness of our evaluation, we take the following precautions:

- **Strict Separation of Training and Test Data**: We ensure that the historical execution records used for model optimization do not overlap with the evaluation datasets. These records are only used during model training and fine-tuning phases to improve model decision-making, and are excluded from the test tasks.
- **Use of External Data**: While external APIs and historical data may aid in supplementing the model's knowledge, they are not directly used to influence the specific test items in the MMLU benchmark. We maintain a strict boundary between training, testing, and external data to avoid any inadvertent bias during the evaluation.
- **Cross-validation for Consistency**: To further guarantee fairness and minimize bias, we utilize cross-validation techniques. This ensures that the model's performance is consistently evaluated across multiple, independent test splits, thereby reducing the risk of overfitting to any specific data, including historical data. Unless otherwise specified, all experiments are repeated five times with different random seeds. We report the mean and standard deviation of accuracy across runs. For small sample settings, bootstrapped confidence intervals were also estimated to assess robustness.

By implementing these safeguards, we ensure that the evaluation results reflect the true capabilities of the AWMF framework without being influenced by prior exposure to test data or historical task-specific information.

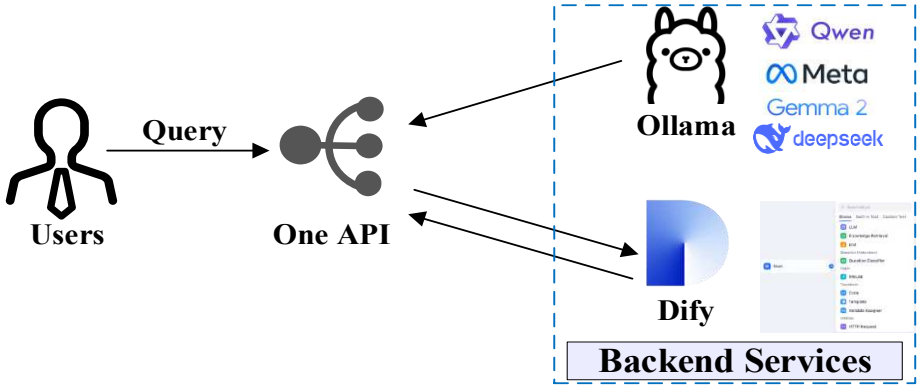


Figure 2. Experimental setup

Dataset and Evaluation Metrics. The performance of the AWMF framework is evaluated using two distinct datasets: the *MMLU* [19] benchmark for multi-domain knowledge tasks and the *Chinese NLP Corpus* [20] for sentiment classification, with accuracy as the primary evaluation metric. The MMLU benchmark is a widely recognized standard for evaluating AI models across a broad range of real-world tasks, covering domains such as STEM, business, healthcare, social sciences, and humanities. It provides a strong validation of the framework's core capabilities. In addition, the Chinese NLP Corpus is used to evaluate the framework's cross-lingual performance in sentiment classification tasks. Although these datasets are representative, cross-dataset validation remains a critical next step. Testing models trained on one dataset and evaluated on another will be included in future work to further evaluate the framework's ability to generalize to unseen data across diverse domains. In our experiments, we primarily use accuracy as the evaluation metric, which is calculated by comparing the model-generated responses to the ground-truth labels provided in the dataset. This allows for a robust assessment of the model's ability to answer questions correctly within each domain. Furthermore, for tasks involving cross-lingual evaluation, we incorporate the Chinese NLP Corpus, a dataset designed for sentiment classification in Mandarin. This dataset consists of over 7,000 hotel reviews, with more than 5,000 positive reviews and 2,000 negative reviews, providing a substantial resource for evaluating sentiment analysis models in a Chinese-language

context. Accuracy is again used as the primary evaluation metric, measured by comparing the predicted sentiment (i.e., positive/negative) to the true sentiment labels in the dataset.

Experimental Environment. All experiments are conducted on a high-performance Linux workstation running Ubuntu 22.04, equipped with dual AMD EPYC 7513 CPUs, four NVIDIA A6000 GPUs, and 512GB of RAM. To ensure reproducibility and facilitate systematic experimentation, we containerize the experimental environment using Docker. Neural network operations are executed with CUDA 11.8 and PyTorch 2.2.2. Each experimental run is meticulously logged, and reported performance metrics represent averaged results from multiple runs to guarantee reliability and statistical validity.

As illustrated in Figure 2, all LLMs deployed in this research are orchestrated via Ollama [21], an efficient platform for large language model management and scaling. Workflow execution is streamlined through the integration of Dify [22], enabling dynamic task orchestration and facilitating efficient model selection, task management, and externaldata retrieval. Custom agents and model integrations are managed through One API [23], providing a unified and efficient interface for orchestrating interactions within the AWMF framework.

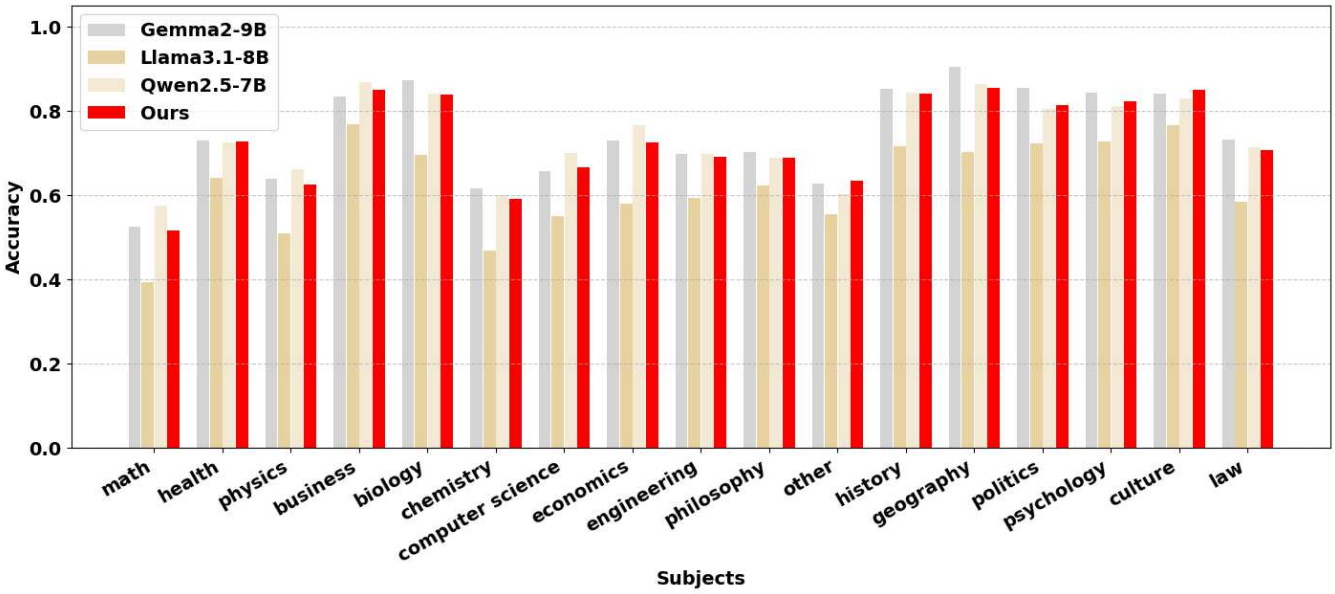


Figure 3. Comparison of Accuracy Across Different Subjects

4.2 Case Study: Question Answering Task

Setting. The principal objective of our experiments is to evaluate the effectiveness of the AWMF framework on the MMLU [19] benchmark, covering a broad spectrum of academic disciplines. We initially assess individual model capabilities by independently evaluating Gemma2-9B, Llama3.1-8B, and Qwen2.5-7B, followed by evaluating the effectiveness of a voting-based ensemble mechanism designed to integrate these models.

Table 2. Accuracy (mean ± std) Comparison on MMLU

Model	Accuracy (mean ± std)
Gemma2-9B	0.717 ± 0.024
Llama3.1-8B	0.603 ± 0.039
Qwen2.5-7B	0.707 ± 0.022

Ours (AWMF)	**0.7000 ± 0.017**

Result. Figure 3 presents several critical insights. Although Gemma2-9B and Qwen2.5-7B generally demonstrate superior accuracy compared to Llama3.1-8B across various subjects, the ensemble model exhibits more consistent and robust performance across all domains. Despite minor accuracy trade-offs in disciplines such as mathematics and physics, the ensemble approach significantly enhances consistency and robustness by effectively mitigating individual model variability. Notably, the ensemble model compensates for the weaknesses observed in lower-performing models (e.g., Llama3.1-8B), thereby optimizing performance through strategic integration. These findings underscore the AWMF framework's ability to enhance adaptability, reliability, and overall robustness in complex and heterogeneous task scenarios. In Table 2, it can be observed that Gemma2-9B and Qwen2.5-7B outperform Llama3.1-8B in terms of accuracy, while the AWMF model, through the integration of multiple models, demonstrates more consistent and stable performance, despite minor trade-offs in certain domains.

4.3 Case Study: Sentiment Classification Task

Setting. To further investigate the cross-lingual and interdisciplinary capabilities of the AWMF framework, a series of experiments are conducted using a Chinese sentiment classification dataset [20]. This dataset consists of over 7,000 hotel reviews, including more than 5,000 positive reviews and approximately 2,000 negative reviews, providing a comprehensive foundation for evaluating the framework's performance in sentiment analysis tasks within Mandarin contexts. For comparative purposes, the AWMF framework is benchmarked against several prominent models, including Qwen2.5:7B, Phi4:14B, and Gemma2:9B, which represent some of the leading pre-trained models in the field of sentiment analysis. All models are evaluated on the same dataset to ensure consistency and fairness in comparison. The primary performance metric for the evaluation is accuracy, which measures the proportion of correctly classified sentiments, offering a clear indicator of each model's ability to handle the complexities inherent in Chinese sentiment analysis tasks.

Results. Each experiment on the Chinese Sentiment Corpus is conducted five times under identical conditions to estimate performance stability. Table 3 reports the mean accuracy and standard deviation. The results show that the AWMF framework significantly outperforms all competing models in the sentiment classification task, achieving an accuracy rate of 0.8409. This notable performance underscores the framework's superior ability to capture nuanced sentiment cues in Chinese text. In contrast, the best-performing benchmark model, Phi4:14B, achieves an accuracy of 0.7800, which, while commendable, still falls short of the AWMF framework's capabilities. Other models, such as Qwen2.5:7B and Gemma2:9B, report accuracy rates of 0.7604 and 0.7246, respectively, highlighting their limitations in handling the intricacies of Mandarin sentiment classification.

These findings provide compelling evidence of the AWMF framework's exceptional performance in cross-lingual tasks, especially in the context of sentiment analysis in Chinese. The results indicate that the framework not only excels at processing sentiment data within the target language but also adapts effectively to the unique challenges posed by non-English languages.

Table 3. Accuracy (mean ± std) Comparison on Chinese Sentiment Classification

Model	Accuracy (mean ± std)
Qwen2.5:7b	0.760 ± 0.008
Qwen2.5:7b instruct-fp16	0.754 ± 0.010
Phi4:14b-fp16	0.780 ± 0.009
Gemma2:9b-instruct-fp16	0.725 ± 0.012

Ours (AWMF)	**0.841 ± 0.007**

4.4 Case Study: External Data Augmentation

Setting. This experiment assesses the efficacy of external knowledge integration via a RAG approach within the AWMF framework, leveraging the Deepseek R1 7B model to dynamically generate relevant knowledge bases. Real-time API retrieval provides contextual augmentation to improve model response accuracy, particularly across STEM, Other (such as business, health, miscellaneous), Social Sciences, and Humanities domains. Individual performances of Gemma2-9B, Llama3.1-8B, and Qwen2.5-7B with external knowledge augmentation are benchmarked against the integrated AWMF framework.

Result. The results, summarized in Table 4, clearly illustrate the advantages of incorporating external knowledge via RAG. The AWMF framework significantly outperforms individual models, achieving accuracy scores of 0.8972 (STEM), 0.8281 (Other), 0.8995 (Social Sciences), and 0.8471 (Humanities). These improvements underscore the substantial effectiveness of external data augmentation in enhancing model performance across both technical and subjective domains.

Additionally, we conducted comparative experiments against leading state-of-the-art models, including OpenAI's o1, DeepSeek-R1, and Llama3.1-405B, using the comprehensive MMLU benchmark for evaluation. Figure 4 demonstrates that the AWMF framework achieves a competitive accuracy of 86.8%, closely trailing top performers OpenAI o1 (91.8%) and DeepSeek-R1 (90.8%). Importantly, the AWMF framework surpasses several prominent models such as Gemini 2.0 (86.5%) and Qwen2.5-72B (86.1%), highlighting its exceptional balance of accuracy, adaptability, and computational efficiency, thereby positioning it as a viable solution for diverse, real-world multi-domain tasks.

Table 4. Accuracy Comparison Across Subjects After Data Augmentation

Model	STEM	Other	Social Sciences	Humanities
Gemma2-9B	0.6372	0.7293	0.8229	0.7550
Llama3.1-8B	0.5066	0.6484	0.6925	0.6416
Qwen2.5-7B	0.6610	0.7290	0.8049	0.7413
Ours	**0.8972**	**0.8281**	**0.8995**	**0.8471**

In addition to effectively integrating external knowledge through the RAG methodology, we conduct a comparative experiment designed to rigorously evaluate the performance of the AWMF framework against several state-of-the-art large language models. The benchmarked models encompass industry-leading variants, such as OpenAI's o1, DeepSeek-R1, and Llama3.1-405B, selected to comprehensively represent current advancements in the domain. The accuracy of these models was quantitatively assessed using the MMLU benchmark, an established and extensive dataset that spans diverse academic disciplines, thereby providing a robust and comprehensive evaluation of model efficacy.

As depicted in Figure 4, the AWMF framework attains an accuracy rate of 86.8%, positioning it prominently among the top-performing models. Although OpenAI's o1 (91.8%) and DeepSeek-R1 (90.8%) achieved superior raw accuracy metrics, the AWMF framework outperforms Gemini 2.0 (86.5%), Qwen2.5-72B (86.1%), and several other leading competitors. This comparative analysis clearly highlights the efficacy of the AWMF framework, particularly when augmented by multi-model fusion strategies and external knowledge integration through the RAG system. Despite not consistently exceeding the largest models purely in terms of raw accuracy scores, the AWMF framework exhibits

highly competitive performance, reflecting its significant potential for practical deployment. Its balanced capability, encompassing accuracy, adaptability, and computational efficiency, positions AWMF as a compelling candidate for complex, multi-domain applications.

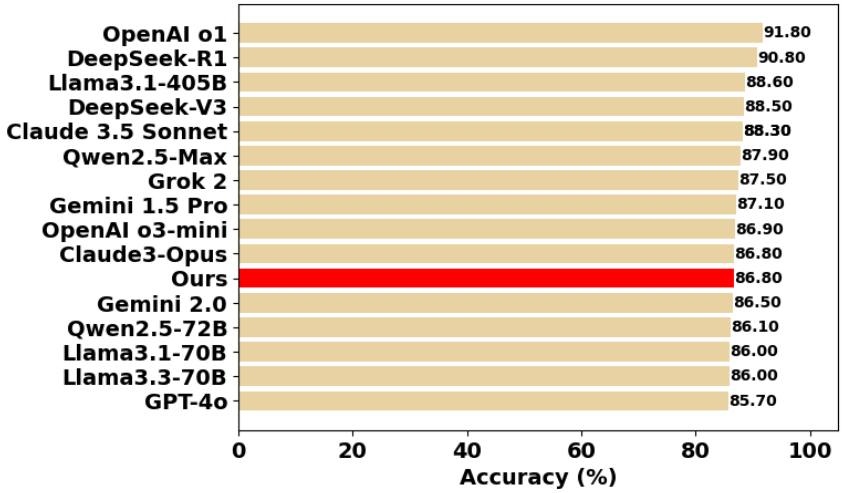


Figure 4. Comparison of AWMF with Other Large Models

4.5 Case Study: History Optimization Task

Setting. To investigate the impact of historical execution records on model performance, we conducted an experiment that incorporated previous execution records and feedback into the MMLU framework. This setup was compared to a baseline scenario where historical data was absent. The primary focus was to evaluate how the inclusion of historical records influences the accuracy and efficiency of the MMLU framework across 17 diverse categories, including mathematics, health, physics, business, chemistry, and politics.

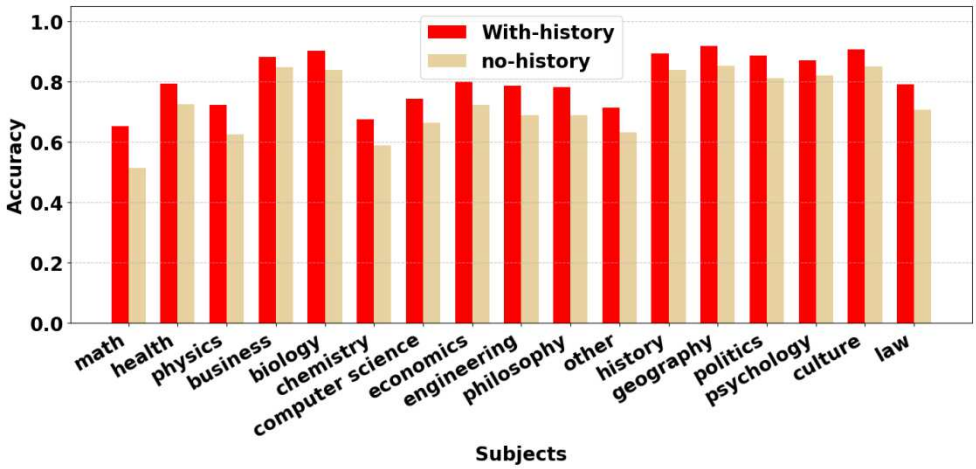


Figure 5. Accuracy Comparison with and without Historical Records

Results. As illustrated in Figure 5, the results from the experiment reveal that the inclusion of historical execution records leads to a clear and consistent improvement in accuracy across all 17 categories in the MMLU benchmark. In every domain tested, the model that incorporates historical data outperforms the baseline model without historical records. This improvement is not limited to specific areas, but is consistently observed across a broad range of subjects, including mathematics, health, physics, chemistry,

and politics. Particularly notable are the substantial gains in chemistry and politics, where the model's accuracy benefits significantly from the contextual reinforcement provided by historical data. This added context allows the model to better refine its understanding of complex, task-specific patterns, ultimately leading to improved performance in these areas. Even in domains where the improvement is more marginal, such as "other" and "law," the model that utilizes historical execution data still maintains a consistent edge in terms of accuracy. This consistent superiority underscores the general efficacy of incorporating historical records into the model's decision-making process.

5 Limitations and Future Works

Baseline Comparison: One limitation of the current study is the lack of comparison with recent multi-agent frameworks, such as AutoGen and MetaGPT. The primary focus of this work was on evaluating the novel contributions of the proposed framework, particularly the integration of semantic operators and multi-model fusion. Due to time and resource constraints, direct comparisons with these advanced systems were not included in this version. However, we recognize that such comparisons are crucial for assessing the competitive advantages of our framework in relation to existing multi-agent systems. Future work will address this limitation by incorporating comparisons with state-of-the-art systems to provide a more comprehensive evaluation of our approach's performance and capabilities.

Ablation Studies: Another limitation of this study is the absence of ablation studies, which are essential for isolating and understanding the contributions of key components such as semantic operator validation, multi-model fusion, and external data integration. While the current work demonstrates the effectiveness of the proposed framework as a whole, it does not provide detailed insights into the individual impact of these components on overall performance. Future research will address this by conducting ablation studies to assess how each component contributes to the framework's overall performance. This will help clarify the role of each element and strengthen the understanding of the framework's operational dynamics.

Scalability Analysis: Scalability remains a significant concern when considering the framework's performance as the number of models or task complexity increases. While this aspect is important, scalability testing was not part of the current evaluation due to the scope and limitations of the study. Future work will include scalability tests to assess how the framework performs as the number of models or task complexity increases, helping to evaluate its potential for large-scale applications.

Threats to Validity: The present implementation employs Ollama for model orchestration. Although Ollama offers lightweight deployment and convenient containerization, it lacks support for distributed multi-GPU parallelism, fine-grained caching, and high-throughput serving capabilities that are essential for large-scale, reproducible inference. In contrast, inference frameworks such as vLLM [24,25,26], Text Generation Inference (TGI) [27], and Ray Serve [28] provide optimized batching, token-level streaming, and memory-efficient scheduling for concurrent multi-model execution. Consequently, the scalability and latency metrics reported in this study should be interpreted as functional proofs of concept rather than upper-bound performance estimates. Future work will integrate AWMF with these high-efficiency backends to ensure reproducible and statistically robust throughput benchmarking.

6 Related Work

Multi-agent Systems. Multi-agent systems, exemplified by AutoGPT, BabyAGI, CAMEL, MetaGPT, and AutoGen [1,5,6,7,8,25,29,30], have demonstrated substantial promise in enabling autonomous artificial intelligence agents. Nonetheless, achieving full autonomy in operational contexts remains profoundly challenging. These dynamic process agents, frequently described as autonomous intelligent systems capable of independently perceiving environmental changes, executing real-time decisions,

performing context-appropriate actions, evaluating resultant outcomes, and adaptively planning subsequent operations, consistently encounter significant obstacles. Despite theoretical adaptability across diverse scenarios, persistent critical challenges include limited controllability, instability, low reproducibility, and suboptimal task completion rates in specialized contexts [8,25]. Concurrent research efforts continue to explore leveraging historical operational data to distill actionable insights, subsequently enhancing the agents' reasoning capabilities [31,32,33,34]. Nevertheless, such strategies invariably impose substantial demands on the instruction-following capabilities of models, presenting acute challenges particularly for those with limited parameterization.

Retrieval-Augmented Generation (RAG). The Retrieval-Augmented Generation (RAG) framework represents an advanced hybrid methodology integrating retrieval-based techniques with generative models, fundamentally aimed at substantially enhancing the precision and contextual relevance of responses produced by large language models. Conventionally, RAG entails a meticulously orchestrated workflow: initially segmenting documents into optimally sized, manageable units, indexing these segments for efficient retrieval, subsequently retrieving segments most closely aligned with a user's query, and ultimately synthesizing this retrieved content alongside the original query through generative model processes [9,10,11]. Several strategic optimizations have been proposed to augment RAG efficacy, including refined document segmentation techniques, decomposition of complex queries into constituent simpler queries, and iterative or recursive retrieval-generation cycles [9]. Recent innovations [35] have broadened the RAG paradigm's applicability to heterogeneous data structures, such as knowledge graphs [36,37] and structured data tables [38], significantly expanding the utility of LLMs across diverse data formats. However, the intrinsic reliance on point-wise retrieval within the RAG paradigm presupposes queries can be fully resolved through retrieved segments, a condition often problematic in the analysis of unstructured datasets [39].

Agentic LLM Systems. Recent frameworks, such as SPP [12], EvoAgent [13], AutoAgents [14], and SwarmAgentic [15], have made significant strides in the automation of agentic systems. These frameworks have developed sophisticated agentic workflows, often involving the integration of multiple large language models (LLMs) to tackle intricate and multi-faceted computational tasks [8,40,41]. These workflows typically deploy specialized agents analogous to fundamental data analytics operators (such as Filter and Scan) that are carefully engineered for precise task execution. While their efficacy in handling complex scenarios is well-documented, manually constructed agentic systems frequently incur significant development and operational maintenance costs, posing substantial barriers to broader implementation [42, 43]. Despite these advances, these systems often rely on predefined templates or seed agents, lacking the flexibility to autonomously generate and adapt agents based on dynamic task requirements. As a result, they fall short in fully meeting autonomy criteria, particularly in areas such as From-Scratch Agent Generation and self-optimizing collaboration. In contrast, our proposed semantic operator-based framework enables From-Scratch Agent Generation, self-optimizing agent functionality, and dynamic collaboration. These innovations allow agents to autonomously adapt and collaborate effectively in complex environments, overcoming the limitations present in frameworks like EvoAgent and AFlow [16]. While existing agentic systems often incur significant development and operational maintenance costs, our approach offers a more scalable and fully autonomous solution, addressing critical challenges in broader agentic system implementation.

Prompt Engineering. Prompt Engineering is a pivotal field at the intersection of natural language processing and human-computer interaction, dedicated to designing input instructions that maximize the performance of LLMs. Its primary goal is to bridge the gap between human intent and machine interpretation through carefully crafted prompts. The foundational framework includes four main components: semantic disambiguation for clarifying ambiguous user queries, cognitive scaffolding to

guide model reasoning, output constraint specification to enforce specific formats or styles, and contextual augmentation through the addition of supplementary information [44,45,46]. Recent advancements have introduced novel techniques such as in-context learning for few-shot tasks, self-consistency sampling to reduce response variability, and adversarial prompting to test robustness [47,48]. To overcome the limitations of static prompting, dynamic optimization approaches like prompt chains for multi-step reasoning and adaptive prompt tuning, which leverages reinforcement learning from human feedback (RLHF), have emerged [49,50,51]. RLHF has significantly advanced the adaptability and efficiency of prompting systems by allowing models to learn from user feedback iteratively. These innovations have broadened the scope of prompt engineering to specialized domains, including multimodal prompting that combines text with visual/audio inputs, cross-lingual prompting for low-resource languages, and ethical prompting to mitigate harmful outputs [52,53,54].

Vector Database. A vector database is a specialized system designed to efficiently store and retrieve high-dimensional vector embeddings produced by machine learning models. Unlike traditional relational databases, vector databases are optimized for similarity search and semantic retrieval, making them essential in fields such as natural language processing, computer vision, and recommendation systems. The architecture of a vector database generally consists of three key components: vector indexing for fast approximate nearest neighbor (ANN) search, storage engines tailored to high-dimensional data, and query optimization mechanisms that balance recall and latency [55,56,57].

7 Conclusion

In this paper, we present the AWMF framework, which overcomes the limitations of traditional LLMs by orchestrating and collaboratively executing multiple smaller, specialized models tailored to complex multi-task scenarios. AWMF integrates dynamic task parsing, intelligent model selection, advanced multi-model fusion, and real-time external data integration, significantly boosting the adaptability, accuracy, and overall performance of AI systems. By combining hybrid reasoning techniques, iterative optimization, and real-time feedback, AWMF ensures consistent optimal performance while maintaining computational efficiency. Experimental results show that AWMF outperforms individual specialized models, achieving competitive results in complex, multi-domain tasks. With enhancements such as external knowledge integration through RAG and the use of historical data, AWMF demonstrates notable improvements in accuracy, especially in STEM, social sciences, and sentiment classification tasks. These findings emphasize the framework's scalability, efficiency, and robustness, positioning it as a promising solution for diverse real-world AI applications across various industries.

8 Conflict of Interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

9 Author Contributions

Zhiyong He: Writing–original draft, Methodology, Investigation, Conceptualization, Design of experiment. Zhengyang Li: Writing–review&editing, Data curation. Kun Qi and Duo Jia: Writing–review&editing, Formal analysis, Supervision.

10 Acknowledgements

This work was supported by the Research Center of Intelligent Sensor and Network Engineering Technology of Jiangsu Province.

11 Data Availability Statement

All experimental workflows, configuration files, and code generated for the AWMF framework will be made available upon acceptance. Restrictions may apply to proprietary model weights or external API keys, which are not publicly distributable.

References

1. Mohammadi M, Tajik E, Martinez-Maldonado R, et al. Artificial intelligence in multimodal learning analytics: A systematic literature review. Comput Educ Artif Intell (2025):100426. Elsevier.
2. Mohammadabadi SMS, Entezami M, Moghaddam AK, et al. Generative artificial intelligence for distributed learning to enhance smart grid communication. Int J Intell Netw 5 (2024):267–274. Elsevier.
3. Nejadshamsi S, Eicker U, Wang C, et al. Data sources and approaches for building occupancy profiles at the urban scale—A review. Build Environ 238 (2023):110375. Elsevier.
4. Talebirad Y, Nadiri A. Multi-agent collaboration: Harnessing the power of intelligent LLM agents. arXiv preprint arXiv:2306.03314 (2023). Available at: https://arxiv.org/pdf/2306.03314.
5. Hong S, Zheng X, Chen J, et al. Metagpt: Meta programming for multi-agent collaborative framework. arXiv preprint arXiv:2308.00352 (2023). Available at: https://arxiv.org/pdf/2308.00352.
6. Li G, Hammoud H, Itani H, et al. Camel: Communicative agents for "mind" exploration of large language model society. Advances in Neural Information Processing Systems 36 (2023):51991–52008.
7. Wu Q, Bansal G, Zhang J, et al. Autogen: Enabling next-gen LLM applications via multi-agent conversation framework. arXiv preprint arXiv:2308.08155 (2023). Available at: https://arxiv.org/pdf/2308.08155.pdf.
8. Wang L, Ma C, Feng X, et al. A survey on large language model based autonomous agents. Frontiers of Computer Science 18(6) (2024):186345.
9. Asai A, Wu Z, Wang Y, et al. Self-rag: Learning to retrieve, generate, and critique through self-reflection. arXiv preprint arXiv:2310.11511 (2023). Available at: https://arxiv.org/pdf/2310.11511.
10. Gao Y, Xiong Y, Gao X, et al. Retrieval-augmented generation for large language models: A survey. arXiv preprint arXiv:2312.10997 (2023). Available at: https://arxiv.org/pdf/2312.10997.
11. Shao Z, Gong Y, Shen Y, et al. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. arXiv preprint arXiv:2305.15294 (2023). Available at: https://arxiv.org/pdf/2305.15294.
12. Wang Z, Mao S, Wu W, et al. Unleashing the emergent cognitive synergy in large language models: A task-solving agent through multi-persona self-collaboration. arXiv preprint arXiv:2307.05300 (2023).
13. Yuan S, Song K, Chen J, et al. Evoagent: Towards automatic multi-agent generation via evolutionary algorithms. arXiv preprint arXiv:2406.14228 (2024).
14. Chen G, Dong S, Shu Y, et al. Autoagents: A framework for automatic agent generation. arXiv preprint arXiv:2309.17288 (2023).
15. Zhang Y, Lin C, Tang S, et al. SwarmAgentic: Towards fully automated agentic system generation via swarm intelligence. arXiv preprint arXiv:2506.15672 (2025).
16. Zhang J, Xiang J, Yu Z, et al. Aflow: Automating agentic workflow generation. arXiv preprint arXiv:2410.10762 (2024).
17. Van Veldhuizen DA, Lamont GB. Evolutionary computation and convergence to a Pareto front. In: Proceedings of the Genetic Programming 1998 Conference (GP1998); 1998. p. 221–228.
18. Almeida BF, Correia I, Saldanha-da-Gama F. Priority-based heuristics for the multi-skill resource constrained project scheduling problem. Expert Syst Appl 57 (2016):91–103.

19. Hendrycks D, Burns C, Basart S, et al. Measuring massive multitask language understanding. In: Proceedings of the International Conference on Learning Representations (ICLR); 2021.
20. jinhuakst,iamzn. ChineseNlpCorpus. Available at: https://github.com/songquanpeng/one-api; 2025.
21. Ollama. Ollama. Available at: https://github.com/ollama/ollama; 2025.
22. Dify. Dify. Available at: https://github.com/langgenius/dify; 2025.
23. songquanpeng. one-api. Available at: https://github.com/songquanpeng/one-api; 2025.
24. vllm-project, vllm, Available at: https://github.com/vllm-project/vllm; 2025.
25. Kwon, Woosuk, et al. Efficient Memory Management for Large Language Model Serving with PagedAttention. Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles, 2023.
26. de Curtò, J., and de Zarzà, I. Comparative Analysis of Reasoning Capabilities in Foundation Models. In: Proceedings of the 2nd International Conference on Foundation and Large Language Models (FLLM), Dubai, UAE, pp. 141–149.
27. Hugging Face. Text Generation Inference (TGI). Available at: https://huggingface.co/docs/text-generation-inference/en/index; 2025.
28. Moritz P, Nishihara R, Wang S, et al. Ray: A distributed framework for emerging AI applications. In: Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI); 2018. p. 561–577.
29. Xue S, Qi D, Jiang C, et al. Demonstration of DB-GPT: Next generation data interaction system empowered by large language models. arXiv preprint arXiv:2404.10209 (2024). Available at: https://arxiv.org/pdf/2404.10209.
30. Chu Z, Hao H, Ouyang X, et al. Leveraging large language models for pre-trained recommender systems. arXiv preprint arXiv:2308.10837 (2023). Available at: https://arxiv.org/pdf/2308.10837.
31. Wang B, Fang H, Eisner J, et al. LLMs in the Imaginarium: Tool learning through simulated trial and error. arXiv preprint arXiv:2403.04746 (2024). Available at: https://arxiv.org/pdf/2403.04746.
32. Song Y, Yin D, Yue X, et al. Trial and error: Exploration-based trajectory optimization for LLM agents. arXiv preprint arXiv:2403.02502 (2024). Available at: https://arxiv.org/pdf/2403.02502.
33. Tao Z, Lin T, Chen X, et al. A survey on self-evolution of large language models. arXiv preprint arXiv:2404.14387 (2024). Available at: https://arxiv.org/pdf/2404.14387.
34. Zhao A, Huang D, Xu Q, et al. Expel: LLM agents are experiential learners. In: Proceedings of the 38th AAAI Conference on Artificial Intelligence (AAAI); 2024. p. 19632–19642.
35. Jeong S, Baek J, Cho S, et al. Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity. arXiv preprint arXiv:2403.14403 (2024).
36. Edge D, Trinh H, Cheng N, et al. From local to global: A graph RAG approach to query-focused summarization. arXiv preprint arXiv:2404.16130 (2024). Available at: https://arxiv.org/pdf/2404.16130.
37. Peng B, Zhu Y, Liu Y, et al. Graph retrieval-augmented generation: A survey. arXiv preprint arXiv:2408.08921 (2024). Available at: https://arxiv.org/pdf/2408.08921.
38. Luo Z, Xu C, Zhao P, et al. Augmented large language models with parametric knowledge guiding. arXiv preprint arXiv:2305.04757 (2023). Available at: https://arxiv.org/pdf/2408.08921.
39. Patel L, Jha S, Guestrin C, et al. Lotus: Enabling semantic queries with LLMs over tables of unstructured and structured data. arXiv preprint arXiv:2407.11418 (2024). Available at: https://arxiv.org/pdf/2407.11418.
40. Liu Z, Zhang Y, Li P, et al. Dynamic LLM-agent network: An LLM-agent collaboration framework with agent team optimization. arXiv preprint arXiv:2310.02170 (2023). Available at: https://arxiv.org/pdf/2310.02170.

41. Zhou X, Li G, Sun Z, et al. D-bot: Database diagnosis system using large language models. arXiv preprint arXiv:2312.01454 (2023).
42. Hu S, Lu C, Clune J. Automated design of agentic systems. arXiv preprint arXiv:2408.08435 (2024). Available at: https://arxiv.org/pdf/2408.08435.
43. Zhang J, Xiang J, Yu Z, et al. Aflow: Automating agentic workflow generation. arXiv preprint arXiv:2410.10762 (2024). Available at: https://arxiv.org/pdf/2410.10762.
44. Brown T, Mann B, Ryder N, et al. Language models are few-shot learners. Adv Neural Inf Process Syst 33 (2020).
45. Wei J, Wang X, Schuurmans D, et al. Chain of thought prompting elicits reasoning in large language models. In: International Conference on Learning Representations; 2022.
46. Liu W, Du Z, He D, et al. Parameter-efficient prompt tuning makes language models better few-shot learners. Neural Comput Appl 35(5) (2023):4325–4338. Springer.
47. Min S, Lee X, Lewis M, et al. Rethinking the role of demonstrations: What makes in-context learning work? Adv Neural Inf Process Syst 35 (2022).
48. Zou C, Chen J, Du Z, et al. On the robustness of in-context learning. In: International Conference on Machine Learning; 2023.
49. Ouyang L, Wu J, Jiang X, et al. Training language models to follow instructions with human feedback. Adv Neural Inf Process Syst 35 (2022).
50. Stiennon N, Ouyang L, Wu J, et al. Learning to summarize from human feedback. Adv Neural Inf Process Syst 33 (2020):3008–3021.
51. Yang Z, Sun A, Zhao Y, et al. RLHF fine-tuning of LLMs for alignment with implicit user feedback in conversational recommenders. arXiv preprint arXiv:2508.05289 (2025).
52. Li W, Tang Z, Li X, et al. Multimodal prompting for large language models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2023. p. 2005–2015.
53. Hu P, Liu Y, Li M, et al. Cross-lingual prompting for low-resource language tasks. In: Proceedings of the Annual Meeting of the Association for Computational Linguistics; 2024.
54. Ribeiro MT, Singh S, Guestrin C. Explaining predictions when you can't explain the model. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2016. p. 1135–1144.
55. Johnson J, Douze M, Jégou H. Balancing accuracy and speed in approximate nearest neighbor search. ACM Trans Inf Syst (TOIS) 35(4) (2017):1–33.
56. Malkov YA, Yashunin DA. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. IEEE Trans Pattern Anal Mach Intell 42(4) (2018):824–836.
57. Dou Z, Song L, King I. Approximate nearest neighbor search: A survey. Found Trends Inf Retrieval 14(3-4) (2020):231–373.