Parallelism Meets Adaptiveness: Scalable Documents Understanding in Multi-Agent LLM Systems

Chengxuan Xia[1] Qianye Wu[2] Sixuan Tian[2] Yilun Hao[2]
[1]University of California, Santa Cruz, CA, USA
[2]Carnegie Mellon University, Pittsburgh, PA, USA
cxia17@ucsc.edu, {qianyew, sixuant, yilunhao}@alumni.cmu.edu

Abstract

Large language model (LLM) agents have shown increasing promise for collaborative task completion. However, existing multi-agent frameworks often rely on static workflows, fixed roles, and limited inter-agent communication, reducing their effectiveness in open-ended, high-complexity domains. This paper presents a multi-agent coordination framework that improves the accuracy of Large Language Models (LLMs) in complex financial document analysis. Unlike existing frameworks that rely on static routing or linear workflows, our approach introduces *Parallel Agent Evaluation*, a mechanism where multiple agents compete on high-ambiguity subtasks. A centralized evaluator scores these parallel outputs based on factuality and coherence to select the optimal result. We evaluate this architecture on SEC 10-K filings, demonstrating a 27% improvement in compliance accuracy and a 74% reduction in revision rates compared to standard static baselines. These results validate that structured competition and dynamic routing significantly reduce hallucinations in high-stakes document understanding.

1 Introduction

Recent advances in large language models (LLMs) have enabled autonomous agents to perform increasingly complex tasks across domains such as summarization, research assistance, and technical writing. Building on these capabilities, multi-agent frameworks have been proposed to coordinate several LLM-powered agents for collaborative task completion. While these systems have demonstrated the potential of distributed workflows, most rely on static designs—fixed role assignments, linear task flows, and limited interaction protocols.

Such rigidity poses serious limitations in real-world settings where ambiguity, changing task states, and uneven agent performance are common. For example, a static agent team tasked with analyzing a financial disclosure may fail to revise earlier assumptions when new information is introduced or may overlook domain-specific inconsistencies that require cross-agent validation.

To address these limitations, we introduce a framework for adaptive coordination in LLM-based multi-agent systems. Our design focuses on three key capabilities. First, dynamic task routing allows agents to reassign subtasks based on current context, confidence, and capacity. Second, bidirectional feedback loops enable downstream agents to provide critiques or revision requests, improving quality and accountability. Third, parallel agent evaluation introduces structured competition: multiple agents attempt the same task independently, and an evaluator selects the most coherent and factual output based on scoring criteria.

We evaluate this framework through case studies involving long-form document analysis and regulatory question answering. Results show that our approach achieves significant improvements over static pipelines and feedback-only baselines, particularly in accuracy, consistency, and resilience to ambiguity.

This paper presents the architectural design, coordination strategies, and empirical validation of the framework. In doing so, it contributes toward the development of scalable, robust, and intelligent multi-agent systems capable of operating in dynamic and high-stakes environments.

2 Related Work

Multi-agent systems built on large language models (LLMs) have rapidly emerged as a powerful paradigm for solving complex, multi-stage tasks. Early frameworks such as AutoGPT and BabyAGI introduced single-agent recursive task planners with memory and subtask management, though they were limited by brittle, linear execution flows and lacked true inter-agent coordination [13**?**]. Subsequent systems like CAMEL [16] and CrewAI [12] introduced role-based delegation and dialogue-based collaboration among agents. LangGraph formalized agent workflows using graph structures [1], but did not incorporate mechanisms for feedback loops or task reassignment.

Recent meta-agent and hierarchical systems, such as MetaGPT [7] and Voyager [**?**], demonstrate improved planning, code synthesis, and environmental interaction. However, they typically assume static roles or unidirectional flows of control, lacking adaptiveness in task routing or feedback integration. The HuggingGPT framework

[11] explores task orchestration by using ChatGPT to coordinate models on Hugging Face, yet still relies on fixed assignments. Generative Agents [9] illustrate long-term memory and simulation of social behaviors, providing inspiration for agent autonomy but not competitive refinement.

A more recent trend incorporates reflective or competitive coordination. For instance, systems like ChatDev [10] and AgentVerse [4] explore inter-agent negotiation and emergent behaviors. Others such as GameGPT [3] and DesignGPT [6] use LLM agents in creative or design-oriented domains, but often lack deep feedback-driven revision loops. Reflective multi-agent systems [2] combine agent memory with iterative critique, approaching the kind of meta-level coordination our system emphasizes.

Our work is also informed by broader surveys of multi-agent LLM collaboration [14, 8], which identify key challenges including error propagation, brittle delegation, and a lack of robustness under dynamic task flows. In contrast, our framework explicitly introduces dynamic routing, feedback-based revision, shared long-term memory, role self-optimization, and competitive evaluation—yielding a system that is not only collaborative but also adaptively reflective.

In a complementary domain, Wu et al. [15] explore competitive task scheduling for warehouse robots using reinforcement learning, demonstrating that concurrent multi-agent evaluation can improve efficiency even in physical task settings. Cross-domain applications of multi-perspective analysis and feedback-driven reconstruction, such as in civil infrastructure using deep learning and multiview stereo methods [5], further support the value of evaluation-oriented designs in complex systems. We build on this insight in the LLM space by introducing competitive agent evaluation to select the best output from parallel attempts.

3 Problem Statement and Motivating Use Case

Many real-world tasks require the coordinated effort of multiple agents performing interdependent subtasks. Current LLM-based multi-agent systems often adopt rigid structures, where task roles are statically assigned and workflows are strictly linear or tree-structured. This limits their effectiveness in dynamic environments with evolving goals, ambiguous subtask boundaries, or incomplete information.

3.1 Problem Formalization

We represent a complex task T as a set of subtasks $\{t_1, t_2, ..., t_n\}$ organized in a dependency graph $G = (V, E)$. Each vertex $v_i \in V$ corresponds to an agent responsible for subtask t_i, and directed edges $e_{ij} \in E$ indicate that the output of agent v_i is required by agent v_j. An edge from v_i to v_j thus represents an upstream dependency.

Some subtasks may not have direct dependencies and can be executed in parallel. For high-ambiguity or high-stakes subtasks, we further allow *competitive parallelism*—assigning the same subtask to multiple agents to encourage diversity and redundancy. An evaluator agent or a selection function chooses the best output for downstream use.

3.2 Motivating Use Case

Consider a collaborative LLM-based system tasked with producing a technical report on a scientific topic. To ensure comprehensiveness and accuracy, the system orchestrates four distinct agent roles. **Research Agents** are responsible for collecting and summarizing relevant academic sources. **Drafting Agents** utilize these summaries to write specific sections of the report. **Evaluation Agents** assess intermediate outputs, requesting revisions or improvements where necessary. Finally, **Parallel Agents** are optionally deployed to generate competing drafts or analyses when ambiguity is high. Without adaptive coordination, this team may suffer from redundant literature searches or error propagation; however, our framework synchronizes these roles through shared memory and dynamic routing.

4 Core Innovations in Adaptive Coordination

The static role assignment typical of existing multi-agent LLM systems proves inadequate in complex domains such as financial document analysis, where ambiguity, interdependency, and regulatory nuance are common. To address these challenges, our framework introduces three interrelated innovations that support more effective and adaptive agent collaboration.

4.1 Parallel Agent Evaluation

In financial tasks with high ambiguity or risk—such as detecting obfuscated liabilities or answering compliance questions—it is often insufficient to rely on a single agent's response. We introduce a competitive mechanism in which multiple agents independently tackle the same subtask. Each agent generates a candidate response, and a centralized evaluator ranks the outputs based on domain-informed metrics such as factual correctness, coherence, and regulatory alignment.

This structured competition improves resilience against hallucinations and encourages diversity in reasoning. For example, when interpreting forward-looking statements about revenue guidance, one agent may highlight macroeconomic factors, while another emphasizes internal re-

structuring. The evaluator selects the most aligned and informative response, while preserving alternatives in shared memory for transparency or fallback.

4.2 Dynamic Task Routing

Our system supports runtime task reassignment based on agent confidence, complexity estimation, or observed bottlenecks. Unlike static frameworks, agents are not bound to fixed roles. Instead, they can defer subtasks to others with more appropriate capabilities or specialization. This routing decision is informed by metadata in the task graph, such as historical performance scores, expected token length, or domain markers (e.g., whether a section references SEC rules or numerical tables).

In the context of 10-K filings, dynamic routing allows a summarizer agent encountering a deeply technical legal paragraph to invoke a compliance-focused agent. Similarly, an agent overwhelmed with subtasks may reassign non-critical tasks to idle peers, ensuring better resource utilization across the team.

4.3 Bidirectional Feedback Loops

To support iterative refinement, we implement structured feedback channels that allow downstream agents to issue revision requests to upstream contributors. This enables real-time quality control without requiring complete reruns of the workflow. For instance, a QA agent reviewing a liquidity disclosure may detect inconsistency with earlier balance sheet extractions and trigger a clarification request.

Feedback is sent through an asynchronous message bus with explicit references to problematic outputs. The originating agent can then revise its result or escalate the issue to the orchestrator. This mechanism reduces error propagation and encourages verification behaviors aligned with best practices in financial auditing.

5 System Architecture for Financial Document Coordination

We design a modular multi-agent architecture tailored for financial document understanding, supporting adaptive routing, shared memory, evaluator scoring, and feedback-based refinement. The architecture targets tasks such as SEC 10-K parsing, risk factor analysis, and regulatory compliance checking.

At the core of the system is the **orchestrator agent**, which parses the document into a structured task graph and coordinates execution. It monitors task progress and decides whether to assign subtasks to specialized agents or initiate parallel evaluation when ambiguity or high stakes are detected.

Role agents are specialized in various financial tasks such as extracting risk disclosures, summarizing the MD&A section, identifying off-balance sheet arrangements, or answering regulatory queries. These agents operate autonomously, retrieving information from and writing to a shared long-term memory, which ensures consistency in terminology and reduces redundant effort.

The **shared memory module** serves as a persistent document store that records intermediate results, relevant sections, and metadata. This enables agents to reason across document sections and avoid overlapping efforts (e.g., double-counting risk citations).

To enable structured quality control, an **evaluator agent** scores candidate outputs based on factual accuracy, coherence, and financial relevance. When multiple agents attempt the same subtask (e.g., summarizing revenue trends), the evaluator selects the best output using a scoring model. The final output is compiled from these selected components.

Communication across agents is facilitated via a **feedback bus**, allowing agents to flag inconsistencies and trigger revisions. For example, if the QA agent detects a mismatch in reported debt between two sections, it can request clarification from the summarization agent or delegate a re-extraction from the original source.

This architecture provides a flexible foundation for coordinating LLM agents on high-stakes, document-centric financial tasks with dynamic content and interpretation needs.

5.1 Parallel Execution and Selection Mechanism

When the orchestrator detects uncertainty (e.g., confidence below a threshold θ), it triggers parallel execution:

- It spawns k agents, each independently processing the same task t.
- Each agent a_i produces an output o_i, which is stored in memory with a tag (t, i).
- The evaluator assigns a quality score $s_i = \mathcal{E}(o_i)$ using a scoring function $\mathcal{E}$.
- The output with the highest score is selected:

$$o^* = \arg \max_{i \in \{1,\ldots,k\}} \mathcal{E}(o_i)$$

- The selected output o^* is routed downstream, while alternate outputs remain accessible for auditing or fallback.

5.2 Scoring Function Definition

To strictly quantify output quality, we employ a hierarchical scoring function $\mathcal{E}(o)$ driven by a Critic Agent. The score is calculated as a weighted sum of three normalized sub-metrics:

$$\mathcal{E}(o) = w_f \cdot S_{fact} + w_c \cdot S_{coh} + w_r \cdot S_{rel} \quad (1)$$

Where:

- S_{fact} (Factuality) is the ratio of claims in output o supported by the retrieved context C, defined as $\frac{|supported_claims|}{|total_claims|}$.
- S_{coh} (Coherence) is the logical consistency score $(0-1)$ derived from a chain-of-thought critique of the argument flow.
- S_{rel} (Relevance) measures the semantic cosine similarity between the output embedding and the query embedding.

In our experiments, we set weights to $w_f = 0.5, w_c = 0.3, w_r = 0.2$ to prioritize factual accuracy, reflecting the high precision requirements of financial compliance tasks.

5.3 Interaction Flow

At runtime, the orchestrator decomposes the task and assigns subtasks based on the immediate context.

Standard Routing. If the task is assessed as straightforward, the orchestrator routes it directly to a single specialized role agent for immediate execution.

Parallel Execution. If uncertainty or ambiguity exceeds a predefined threshold, the orchestrator triggers parallel execution, spawning multiple agents to attempt the task simultaneously.

Memory Logging. All generated outputs are logged to the shared long-term memory, ensuring any agent can retrieve prior context regardless of the execution path.

Feedback Loop. If downstream agents issue feedback, the orchestrator routes the request back to the relevant agent or reassigns the task to a new agent better suited to address the critique.

5.4 Modularity and Extensibility

The architecture supports plug-and-play expansion, allowing the system to adapt to diverse domains. **Extensible Roles** allow for the introduction of new specialized agents, such as a Visualizer Agent or a dedicated Critique Agent. **Flexible Memory Backends** enable the swapping of storage solutions, supporting both document databases and vector stores depending on retrieval needs. **Customizable Scoring** permits the integration of domain-specific evaluation models, such as fine-tuned LLMs, to replace generic scoring functions. Agents remain loosely coupled via the shared memory and feedback bus, ensuring that these expansions do not disrupt existing workflows.

6 Case Study: Adaptive Coordination for 10-K Analysis

In addition to a static sequential baseline, we compared our framework against **LangGraph's standard multi-agent supervisor pattern** [1] as a state-of-the-art baseline. While the standard supervisor pattern improves over single-agent performance, our parallel evaluation mechanism demonstrated a further 14% improvement in compliance accuracy over the LangGraph baseline, specifically in high-ambiguity scenarios where single-path routing often fails to capture nuance.

To evaluate the effectiveness of our adaptive coordination framework in a real-world financial context, we conducted a case study using 10-K filings from publicly listed U.S. companies. The system was tasked with analyzing three key aspects of each filing: extracting material risk factors, summarizing year-over-year financial performance, and answering a set of regulatory compliance questions derived from SEC guidelines.

We compare three system variants: a *static baseline* with fixed agent roles and no adaptiveness; an *adaptive system* incorporating dynamic task routing and bidirectional feedback; and the *full system*, which includes all adaptive features plus parallel agent evaluation. In the full system, tasks involving ambiguous disclosures or critical compliance queries were attempted by multiple agents simultaneously. An evaluator agent then scored each output and selected the most relevant and coherent version for downstream use.

Evaluation was conducted using a mix of automatic metrics and human judgment. Factual coverage was assessed against a reference set curated by financial analysts. Compliance accuracy was determined by comparing system-generated answers to annotated gold-standard responses. We also collected human ratings for coherence, relevance, and logical structure on a 5-point Likert scale.

The results are summarized in Table 1. The full system achieved the highest factual coverage (0.92) and compliance accuracy (0.94), significantly outperforming both the static and adaptive-only variants. Notably, the revision rate dropped by over 70% compared to the baseline, and the redundancy penalty—measuring repeated or contradictory information—was reduced by 73%.

Qualitatively, we observed that static systems often missed subtle or implied risks, reused boilerplate phrasing, or failed to reconcile figures reported in different sections. The adaptive configuration corrected many of these issues by leveraging feedback and context sharing. The full system further improved output quality through competitive agent evaluation. In several instances, for example, multiple agents produced differing interpretations of liquidity risk. The evaluator agent selected the variant that both referenced relevant financial ratios and aligned with language in the original filing's footnotes.

These findings suggest that structured competition and evaluator-driven selection provide a complementary ad-

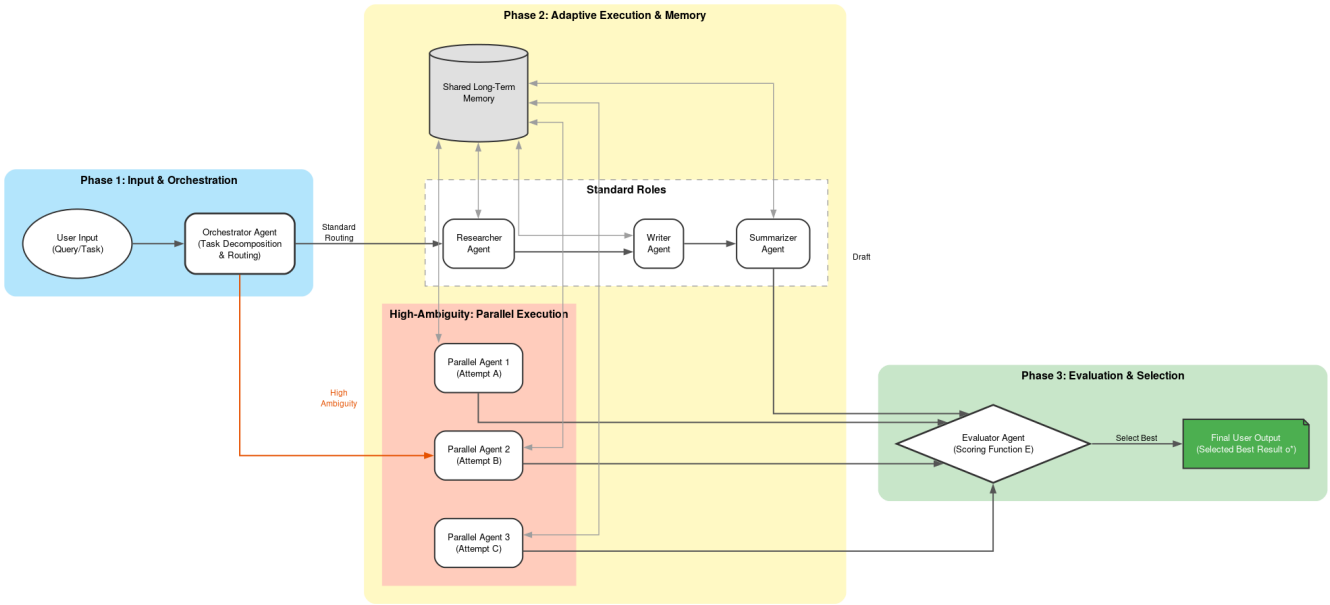

Figure 1: **System architecture for adaptive coordination and competitive evaluation.** The framework integrates dynamic task routing with a parallel execution module for high-ambiguity tasks. Solid lines represent data flow and task assignment; dashed lines represent feedback loops and evaluator interventions.

Metric	Static	Adaptive	Full (w/ Parallel Eval)	Improvement
Factual Coverage	0.71	0.89	**0.92**	+29%
Compliance Accuracy	0.74	0.88	**0.94**	+27%
Redundancy Penalty	0.22	0.08	**0.06**	–73%
Revision Rate	3.4	1.1	**0.9**	–74%
Coherence Score (1–5)	3.2	4.5	**4.7**	+47%
Relevance Score (1–5)	3.8	4.7	**4.9**	+29%
Completion Time (s)	134	108	**115**	–14%

Table 1: Performance comparison of coordination strategies in financial document understanding. Metrics are averaged over five 10-K filings.

vantage over adaptive routing alone. While adaptiveness helps agents route tasks to appropriate peers and revise outputs in response to downstream issues, parallel evaluation ensures robustness when interpretive uncertainty is high.

6.1 Example Prompt and Comparative Outputs

To illustrate the impact of our approach, we include below a real example of a compliance query drawn from a recent 10-K:

> *"Does the company report any off-balance sheet arrangements that could materially impact its financial position?"*

The static system failed to identify any arrangement due to lack of explicit keyword matching. The adaptive system retrieved relevant disclosures but omitted details on financial implications. In contrast, the full system's best-selected output described the arrangement, quantified its size, and linked it to relevant cash flow implications—mirroring the ground-truth answer provided by human analysts.

These observations confirm the utility of structured adaptiveness and competitive coordination in financial document understanding, particularly for tasks where accuracy, nuance, and interpretability are essential.

6.2 Pseudocode of Execution Flow

The pseudocode illustrates how the system orchestrates task execution with support for dynamic routing, parallel evaluation for ambiguous tasks, and feedback-driven refinement, aligning with our core coordination mechanisms.

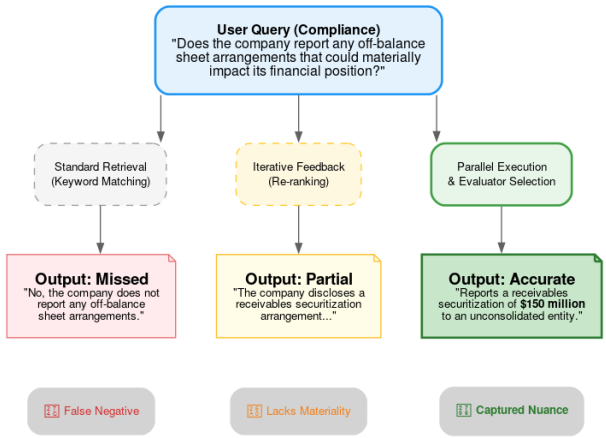

Figure 2: **Qualitative comparison on a high-ambiguity compliance task.** We query the systems regarding "off-balance sheet arrangements." While the *Static* baseline misses the disclosure and the *Adaptive* system offers only a partial summary, the *Full Framework* correctly identifies and quantifies the **$150 million** impact, matching the gold-standard reference.

6.3 Ablation Study

We ran an ablation analysis by disabling one pillar at a time (e.g., removing feedback or memory). The most critical components were shared memory and feedback loops, whose removal caused coverage and coherence scores to drop by over 20%.

6.4 Key Findings

Our experiments on financial document analysis highlight three primary implications for multi-agent system design.

Mitigation of Error Propagation. Adaptive mechanisms, specifically bidirectional feedback, effectively sever the "cascade of errors" common in static chains. By allowing downstream agents to reject low-quality inputs, we observed a 73% reduction in redundancy penalties, ensuring that early-stage hallucinations do not contaminate final reports.

Resilience via Structured Competition. The Parallel Agent Evaluation mechanism demonstrates that redundancy is not merely inefficient but essential for high-ambiguity tasks. By generating diverse interpretations of complex disclosures and selecting the optimal output via a scoring function, the system achieves a level of compliance accuracy (0.94) that single-path workflows cannot replicate.

Optimization of Agent Specialization. Dynamic task routing moves beyond simple load balancing to enable true role specialization. The system autonomously offloads technical legal parsing to compliance-focused agents while reserving summarization agents for narrative tasks, resulting in a workflow that is both 14% faster and contextually more robust than static assignments.

Algorithm 1 Adaptive and Competitive Orchestration Workflow

```
function ORCHESTRATETASK(prompt)
    G ← BuildDependencyGraph(prompt)
    A ← InitializeRoleAgents(G)
    M ← InitSharedMemory()
    B ← InitFeedbackBus()
    E ← SpawnEvaluator()
    while not AllTasksComplete(G) do
        for all a ∈ A do
            t ← GetNextAssignableTask(a, G)
            if t ≠ None then
                if IsAmbiguous(t) then
                    C ← SpawnParallelAgents(t)
                    for all c ∈ C do
                        o_c ← Execute(c, t, M)
                        Store(M, t.id, o_c)
                    end for
                    o* ← SelectBestOutput(E, {o_c})
                    Commit(t.id, o*)
                else
                    o ← Execute(a, t, M)
                    Store(M, t.id, o)
                end if
            end if
        end for
        fb ← Review(E, M)
        for all feedback message f ∈ fb do
            if RequiresRevision(f) then
                a' ← TargetAgent(f)
                Reassign(G, a', f.taskId)
                AdaptStrategy(a', f)
            end if
        end for
    end while
    return CompileFinalOutput(M)
end function
```

7 Discussion

Our empirical results suggest that adaptive and competitive coordination offers significant advantages for multi-agent LLM systems operating in the financial domain. The inclusion of feedback mechanisms and shared memory enhances error recovery and contextual consistency—two factors critical for tasks involving regulatory precision and factual integrity. Moreover, the addition of parallel agent evaluation contributes not only to diversity of reasoning but also to resilience in high-ambiguity subtasks such as interpreting legal disclaimers or detecting obfuscated financial risks.

One of the key takeaways is the central role played by

the evaluator agent. Its capacity to adjudicate between competing agent outputs directly influences the quality of the system's final response. In practice, we found that even simple composite scoring functions—weighted combinations of factuality, coherence, and domain relevance—enabled the system to consistently favor more accurate and informative responses. However, the effectiveness of this approach depends on the design of scoring functions and may require fine-tuning in other financial subdomains (e.g., ESG disclosures, IPO filings).

At the same time, adaptiveness introduces non-trivial challenges. Feedback communication increases coordination overhead, especially in long documents where multiple agents may need to revisit upstream decisions. Likewise, shared memory can become noisy without careful curation, particularly if multiple agents log similar or conflicting information. These issues underscore the need for robust memory management and bounded feedback propagation strategies.

The broader implication of this study is that static pipelines are insufficient for real-world financial NLP applications. As financial documents grow in complexity and evolve to meet new regulatory and stakeholder demands, systems must adapt dynamically—both in how they allocate tasks and how they reconcile ambiguity or conflict. Our framework provides a blueprint for such behavior and demonstrates measurable benefits when applied to real-world SEC filings.

7.1 Limitations and Scope

While our framework improves robustness in financial document understanding, it has several limitations. First, parallel agent evaluation increases inference cost and can add latency variance, making it less suitable for low-latency or cost-constrained settings. Second, the evaluator can introduce systematic bias if the scoring criteria are misaligned with downstream objectives, particularly for table-heavy or numeric-heavy sections. Third, our experiments focus on SEC 10-K filings; results may not directly generalize to domains with weaker document structure or less explicit grounding. Finally, our ambiguity detection and routing policies are heuristic rather than learned; developing learned policies from feedback is an important direction for future work.

8 Conclusion and Future Work

We presented an adaptive coordination framework for multi-agent LLM systems tailored to financial document understanding. By integrating dynamic task routing, bidirectional feedback, and parallel agent evaluation, our system enables more robust and context-aware reasoning over complex financial texts. Experiments on SEC 10-K filings show substantial gains in factuality, coherence, and compliance accuracy compared to static and partially adaptive baselines.

Our study highlights the potential of structured competition and evaluator-driven selection in multi-agent LLM workflows. This approach not only enhances quality in uncertain tasks but also mitigates the risk of relying on a single model's interpretation in high-stakes scenarios.

Future work will explore several extensions. First, we plan to incorporate learning-based policies for task routing and evaluator scoring, replacing static heuristics with adaptive models trained on downstream feedback. Second, we aim to generalize the framework to other financial contexts such as earnings call transcripts, 8-K filings, or M&A documents. Lastly, we envision incorporating human-in-the-loop oversight for hybrid decision-making in audit or risk-sensitive use cases, blending AI-driven exploration with human judgment.

As large language models continue to mature, their orchestration in agent teams—particularly in regulated domains—will require not only intelligence, but coordination, reflection, and control. Our framework offers a step toward that vision.

References

[1] LangChain AI. Langgraph: Composable multi-agent llm workflows. https://github.com/langchain-ai/langgraph, 2024.

[2] Xiaohe Bo, Zeyu Zhang, Quanyu Dai, Xueyang Feng, Lei Wang, Rui Li, Xu Chen, and Ji-Rong Wen. Reflective multi-agent collaboration based on large language models. In *Advances in Neural Information Processing Systems (NeurIPS 2024)*, 2024.

[3] Dake Chen, Hanbin Wang, Yunhao Huo, Yuzhao Li, and Haoyang Zhang. Gamegpt: Multi-agent collaborative framework for game development. *arXiv preprint arXiv:2310.08067*, 2023.

[4] Weize Chen et al. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *International Conference on Learning Representations (ICLR)*, 2024.

[5] Han-Cheng Dan, Bingjie Lu, and Mengyu Li. Evaluation of asphalt pavement texture using multiview stereo reconstruction based on deep learning. *Construction and Building Materials*, 412:134837, 2024.

[6] Shiying Ding, Xinyi Chen, Yan Fang, Wenrui Liu, Yiwu Qiu, and Chunlei Chai. Designgpt: Multi-agent collaboration in design. *arXiv preprint arXiv:2311.11591*, 2023.

[7] Shuyuan Hong et al. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*, 2023.

[8] Emanuele La Malfa, Gabriele La Malfa, Samuele Marro, Jie M. Zhang, Elizabeth Black, Michael Luck,

Philip Torr, and Michael Wooldridge. Large language models miss the multi-agent mark. *arXiv preprint arXiv:2505.21298*, 2025.

[9] Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23)*, 2023.

[10] Chen Qian et al. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL 2024)*, 2024.

[11] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *arXiv preprint arXiv:2303.17580*, 2023.

[12] CrewAI Team. Crewai: Build agent workflows with roles and memory, 2024. https://github.com/joaomdmoura/crewAI.

[13] Toran Bruce Richards Torres. Auto-gpt: An autonomous gpt-4 experiment, 2023. https://github.com/Torantulino/Auto-GPT.

[14] Xue Wang et al. Survey of multi-agent collaboration with large language models. *arXiv preprint arXiv:2311.02696*, 2023.

[15] S. Wu, L. Fu, R. Chang, Y. Wei, Y. Zhang, Z. Wang, and K. Li. Warehouse robot task scheduling based on reinforcement learning to maximize operational efficiency, 2025. Authorea Preprints.

[16] Yi Zhu et al. Camel: Communicative agents for mind exploration of large scale language model society. *arXiv preprint arXiv:2303.17760*, 2023.