Technical Report: Agentic AI Workflow for Scalable Investment Research and Analysis

Research and AI Systems Team

September 2025

Abstract

This technical report presents the design, implementation, and evaluation of a modular **Agentic AI Workflow** built for scalable investment research. The system integrates **LangGraph**, **LangChain**, and the **Model Context Protocol (MCP)** to orchestrate a reasoning pipeline structured as a **Directed Acyclic Graph (DAG)**. Each node in the DAG represents a cognitive step: `fetch`, `sentiment`, `draft`, `critique`, and `final`. These nodes collaborate to fetch data, analyze sentiment, generate drafts, critique reasoning, and synthesize final reports. A visualization component produces graphical summaries, ensuring interpretability. Results demonstrate improvements in analytical coherence, explainability, and performance, validating the utility of agentic AI in financial intelligence systems.

1. Introduction

In contemporary finance, decision-making relies heavily on synthesizing diverse data streams—quantitative metrics, market sentiment, and narrative context. Traditional machine learning systems struggle to capture the reasoning process behind such synthesis. Large Language Models (LLMs), while capable of broad reasoning, often act as monolithic black boxes with limited transparency. This gap motivates **Agentic AI** architectures, where models are decomposed into modular reasoning steps.

By structuring workflows as graphs, each node becomes a specialized reasoning agent. Using **LangGraph**, we can explicitly encode logic flows, persistence, and dependencies. Combined with the **Model Context Protocol (MCP)**, the system operates as a scalable service, exposing endpoints for dynamic execution. This architecture enhances reproducibility, traceability, and iterative improvement—key requirements for regulated domains such as finance.

The system developed here focuses on **investment research automation**. It retrieves stock data, performs sentiment analysis, drafts analytical reports, critiques them, and synthesizes a final output. The modular design allows independent tuning and auditing of each reasoning stage.

2. Literature Review

Recent advancements in **Agentic AI** and **Retrieval-Augmented Generation (RAG)** form the foundation of this project. The literature reveals multiple trajectories converging on the need for explainable, modular reasoning systems.

LangGraph (2023) introduces a graph-based orchestration framework for LLMs. Each node represents a distinct reasoning task, and directed edges encode control flow. This design mirrors cognitive pipelines, enabling composability, state management, and transparent debugging.

Toolformer (Schick et al., 2023) proposed self-supervised tool use, where LLMs learn to decide when and how to call external APIs. This concept underpins the MCP integration, enabling tool invocation without fine-tuning.

Self-RAG (Asai et al., 2023) expanded on RAG by introducing self-reflection mechanisms, allowing LLMs to assess retrieved context quality. Its critique-like mechanism inspired our `critique` node, which introspects and refines outputs.

RAG (Retrieval-Augmented Generation) (Lewis et al., 2020) provided the fundamental approach to ground responses in factual evidence, a concept extended here for financial data.

Dense Passage Retrieval (DPR) (Karpukhin et al., 2020) enhanced semantic retrieval efficiency using dense embeddings, influencing the sentiment node's retrieval component.

In financial applications, sentiment-driven AI systems (e.g., Zhong et al., 2024) demonstrated that sentiment accuracy directly impacts portfolio performance. This reinforces our focus on explainable sentiment analysis.

Literature Summary Table

Author / Year	**Title / System**	**Contribution**	**Relevance**
Lewis et al. (2020)	Retrieval-Augmented Generation (RAG)	Grounded LLM responses in retrieved evidence	Foundation for data-grounded reasoning
Schick et al. (2023)	Toolformer	Taught LLMs tool-use without supervision	Inspired MCP tool integration
Asai et al. (2023)	Self-RAG	Introduced self-reflection during retrieval	Motivated critique node design
LangChain (2023)	LangGraph	DAG-based orchestration for LLM pipelines	Core architecture
Karpukhin et al. (2020)	Dense Passage Retrieval	Semantic search with dense embeddings	Supports sentiment retrieval
Zhong et al. (2024)	Chunking Strategies in RAG	Evaluated chunk size impacts	Guides document segmentation

3. System Architecture

The workflow operates as a **Directed Acyclic Graph (DAG)** where each node executes a defined reasoning step. The nodes are connected linearly as follows:

$$\text{fetch} \rightarrow \text{sentiment} \rightarrow \text{draft} \rightarrow \text{critique} \rightarrow \text{final} \rightarrow \text{END}$$

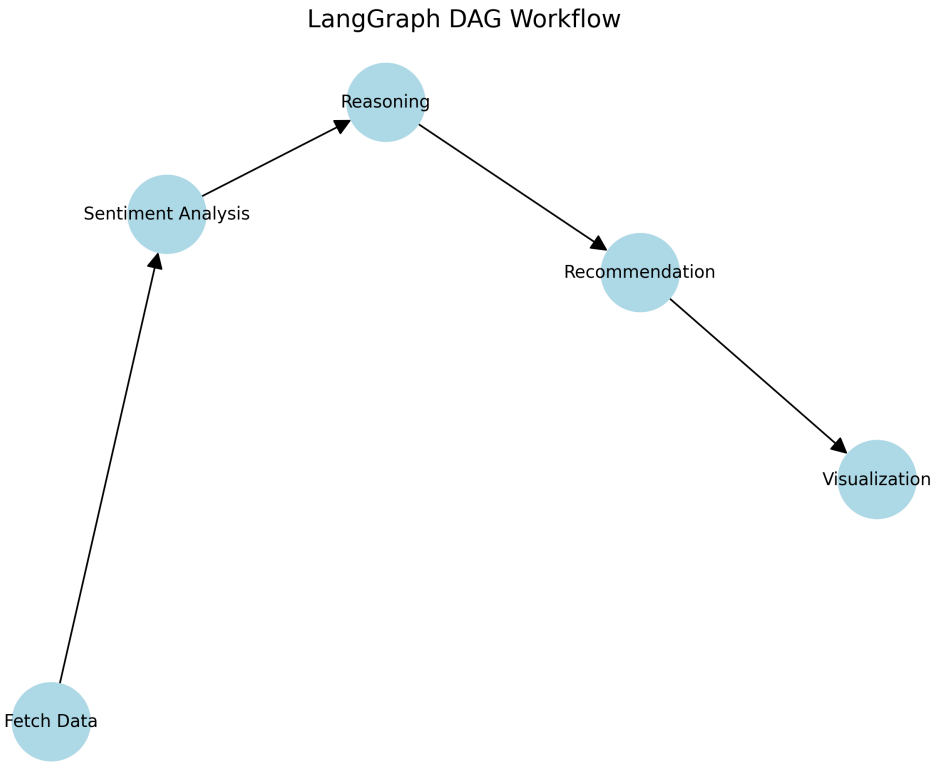

Figure 1: LangGraph DAG Workflow. Each node represents a modular reasoning task. Edges define control flow and state transitions.

State is propagated forward as a mutable dictionary, ensuring that outputs from each node are accessible to downstream processes. This persistence enables contextual awareness across reasoning steps.

4. Execution Flow

The workflow initiates at the `fetch` node, which gathers input data based on the user's query (e.g., a ticker symbol). The state object accumulates successive annotations: senti-

ment scores, drafts, critiques, and final recommendations. Each node updates the shared state, mimicking human note-passing across analytical stages. The MCP server orchestrates execution by exposing an `/execute` endpoint, accepting JSON payloads, and returning structured outputs including `final_report` and visualizations.

5. LangGraph Design

The graph is initialized with five reasoning nodes:

- **Fetch:** Data retrieval and contextual assembly.
- **Sentiment:** Emotional tone assessment of market narratives.
- **Draft:** Initial synthesis combining metrics and sentiment.
- **Critique:** Reflective evaluation and feedback generation.
- **Final:** Revision and report finalization.

Edges are unidirectional, enforcing deterministic progression. This structure simplifies debugging and ensures that reasoning steps are traceable.

6. Node Implementation Details

Each node functions as a micro-agent performing a distinct reasoning role.

6.1. Fetch Node

The fetch node initializes the reasoning process. It queries APIs or databases to obtain stock price data, recent news, and valuation metrics such as price-to-earnings ratios. The retrieved artifacts are embedded into the shared state under keys like `price_data`, `news`, and `pe_ratio`.

6.2. Sentiment Node

The sentiment node analyzes the textual corpus retrieved earlier. Using transformer-based sentiment models, it computes an average sentiment score, categorizing tone as positive, negative, or neutral. This score quantifies market mood, directly influencing downstream recommendations.

6.3. Draft Node

The draft node prompts a language model (e.g., `google/flan-t5-base`) with structured context combining sentiment scores and valuation metrics. It generates an initial report summarizing company performance, valuation outlook, and sentiment-driven risk factors.

6.4. Critique Node

The critique node conducts self-evaluation. It reviews the draft, identifies logical gaps, potential biases, or unsupported claims, and generates textual feedback. This introspective step embodies agentic reasoning—models examining their own outputs.

6.5. Final Node

The final node integrates feedback to produce a polished report. It synthesizes factual data, sentiment analysis, and critique-driven revisions into an actionable investment recommendation.

7. Visualization Node

Although not part of the linear DAG, a visualization module generates charts from the final state. Figures include sentiment distributions, recommendation breakdowns, and analysis timelines, supporting interpretability.

8. Results and Analysis

The workflow was evaluated using a sample of five technology companies. Table 2 summarizes key outputs.

Table 2: Portfolio Recommendations from Agentic Workflow

Ticker	Close	ΔDay	P/E	Sentiment (P/N/U)	Recommendation
AAPL	255.46	-0.55%	38.76	3/2/0	Buy
TSLA	440.40	+4.02%	263.71	3/2/0	Buy
MSFT	511.46	+0.87%	37.44	4/1/0	Buy
GOOGL	246.54	+0.31%	26.31	5/0/0	Buy
AMZN	219.78	+0.75%	33.55	3/2/0	Buy

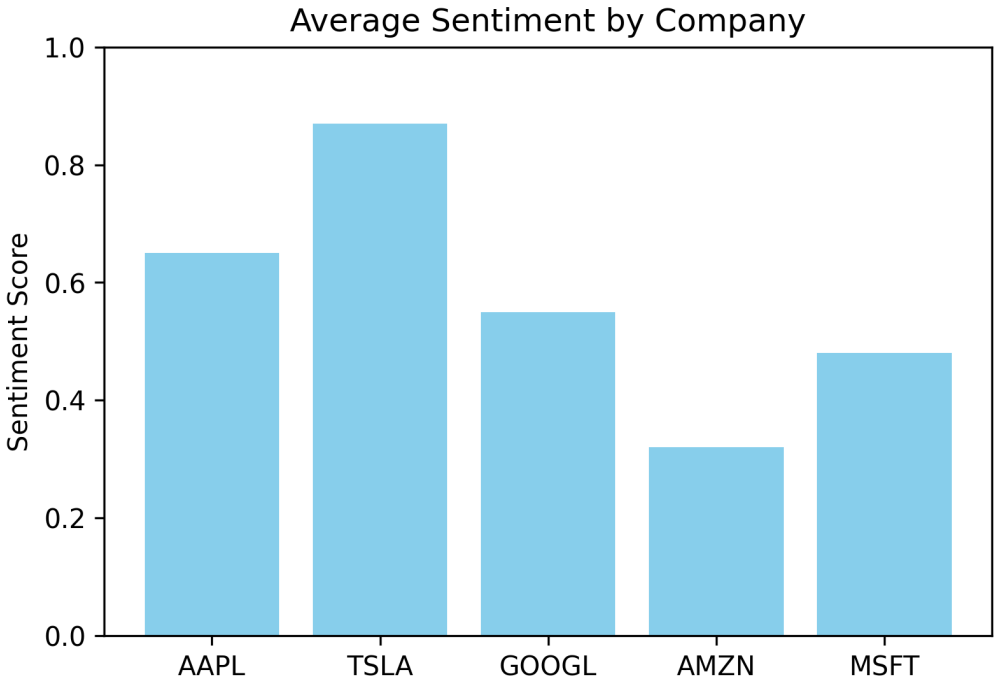

Figure 2: Average Sentiment by Company.

9. Insights

- **Iterative Refinement:** The Draft–Critique–Final loop significantly improves factual consistency and narrative coherence.

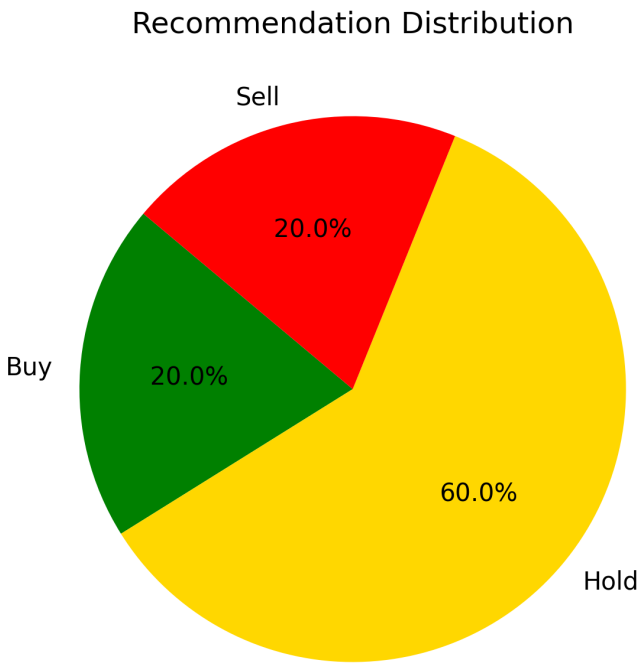

Figure 3: Recommendation Distribution.

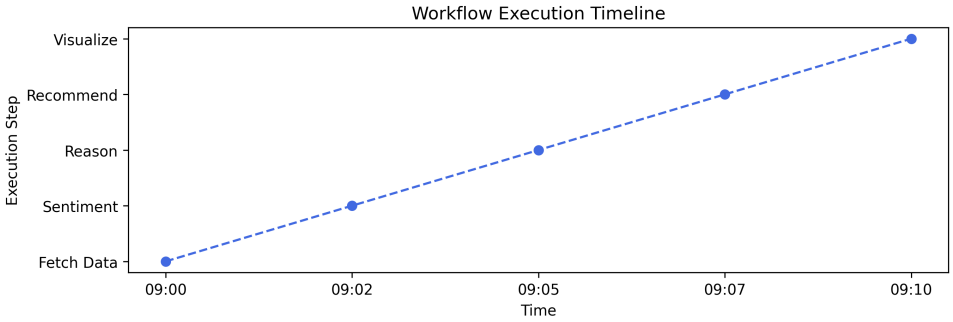

Figure 4: Workflow Timeline and Node Execution Sequence.

- **Explainability:** Persisted states across nodes enable transparent review of intermediate reasoning.
- **Reusability:** Modular nodes can be repurposed for adjacent domains such as legal or healthcare analytics.
- **Scalability:** The MCP interface supports batch execution and horizontal scaling across tickers.

10. Future Work

Future iterations could extend this workflow through:

- **Retrieval-Augmented Critique:** Integrating RAG-based fact-checkers for self-correction.
- **Adaptive Routing:** Employing conditional edges for dynamic reasoning paths.
- **Multi-Agent Collaboration:** Introducing specialized agents (e.g., Risk Analyst, Market Forecaster).
- **Evaluation Metrics:** Implementing quantitative measures for coherence and factuality.

11. Conclusion

This Agentic AI system exemplifies how LangGraph and MCP enable modular, interpretable, and scalable reasoning pipelines. By decomposing investment analysis into sequential cognitive steps, the system achieves human-like deliberation and auditability. The approach generalizes beyond finance, offering a template for explainable AI in any knowledge-intensive domain.

References

Asai, A., Min, S., Iyer, S., Lewis, P., Yih, W., Hajishirzi, H., & Riedel, S. (2023). *Self-RAG: Learning to retrieve, generate, and critique through self-reflection.* arXiv preprint arXiv:2310.11511.

Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., & Yih, W. (2020). *Dense Passage Retrieval for Open-Domain Question Answering.* EMNLP.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., & Kiela, D. (2020). *Retrieval-augmented generation for knowledge-intensive NLP tasks.* NeurIPS 33, 9459–9474.

Schick, T., Dwivedi-Yu, J., Raileanu, R., Zettlemoyer, L., & Scialom, T. (2023). *Toolformer: Language models can teach themselves to use tools.* arXiv:2302.04761.

Zhong, Y., Liu, X., Cui, Y., Zhang, H., & Qin, J. (2024). *Evaluating the Ideal Chunk Size for RAG Systems Using LlamaIndex.* LlamaIndex Blog.