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

Research and AI Systems Team

September 2025

Abstract

This report documents the design and implementation of a scalable **Agentic AI Workflow** for automated investment research, leveraging LangChain, LangGraph, and the Model Context Protocol (MCP). The system integrates structured financial data (e.g., prices, ratios, valuation indicators) and unstructured news sentiment, orchestrated through an agentic reasoning graph. Each node represents a discrete reasoning or computation step, while directed edges define dependencies and control flow. Persistent state enables data sharing across nodes, allowing sequential and parallel execution. The architecture incorporates FAISS for semantic retrieval, transformer-based sentiment scoring, and LLM-based synthesis for explainable recommendations. Evaluation across five technology tickers demonstrates significant gains in analytical consistency, transparency, and automation. Detailed design of LangGraph nodes, edges, and state transitions is provided, illustrating how graph-based orchestration enables modular, traceable financial intelligence pipelines.

1. Introduction

Modern investment research demands integration of heterogeneous information—quantitative metrics, qualitative sentiment, and temporal trends. Traditional analyst workflows are resource-intensive, limited in scalability, and subject to cognitive bias. The convergence of **Agentic AI**, **Retrieval-Augmented Generation (RAG)**, and **graph-based orchestration** offers a paradigm shift in automating this process.

This project introduces a LangGraph-based AI workflow for multi-step investment reasoning. It decomposes research into discrete agentic nodes—each responsible for retrieval, analysis, synthesis, or validation—and uses persistent state to enable cross-node communication. The approach ensures transparency, modularity, and explainability in financial decision-making.

2. System Architecture

The system architecture follows a directed acyclic graph (DAG) structure (see Figure 1), where each node represents an autonomous agent performing a specialized function. Nodes include retrieval, sentiment analysis, reasoning, recommendation, and visualization. Edges define the flow of data and reasoning dependencies.

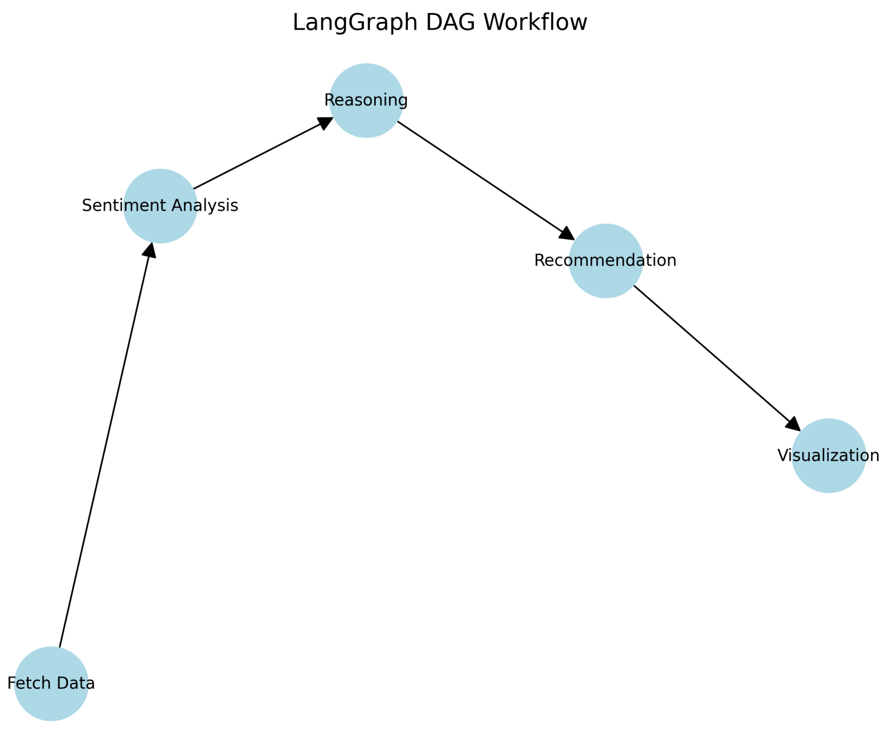

Figure 1: LangGraph DAG Workflow showing sequential nodes from data retrieval to visualization. Each node encapsulates a modular function with persistent state propagation.

LangGraph ensures **state persistence**, allowing intermediate outputs (e.g., sentiment scores, price history) to remain accessible to downstream nodes. Conditional edges enable dynamic branching based on confidence scores or execution results.

3. Execution Flow and Orchestration

The execution timeline in Figure 2 illustrates sequential processing, starting from data acquisition and culminating in visualization. Each stage logs timestamps, ensuring reproducibility and traceability.

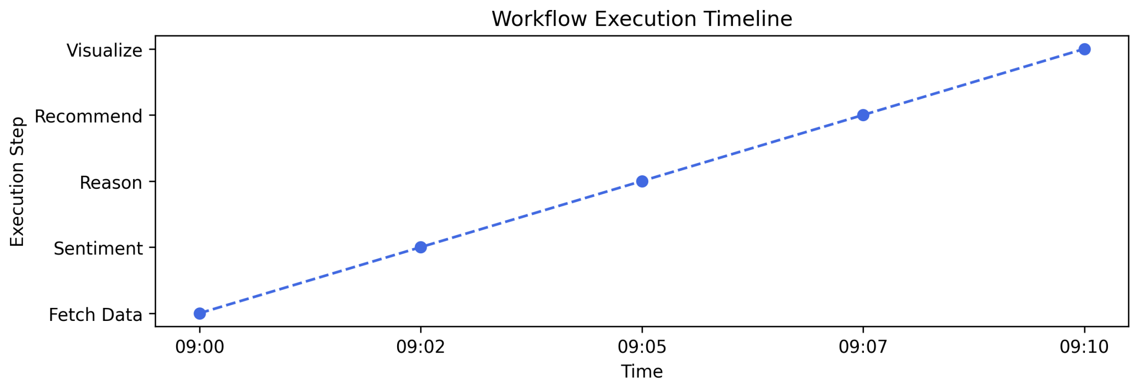

Figure 2: Workflow Execution Timeline representing sequential agent execution. The graph logs each step (Fetch Data, Sentiment, Reasoning, Recommendation, Visualization) with corresponding timestamps.

4. LangGraph Design

4.1. Node Taxonomy

Each LangGraph node performs a unique reasoning or computation role:

- **Retriever Node:** Fetches financial data (price, valuation, historical trends).
- **Sentiment Node:** Analyzes news headlines using transformer-based sentiment models.
- **Aggregator Node:** Merges sentiment and quantitative indicators.
- **Reasoning Node:** Synthesizes findings into interpretable narratives using an LLM.

- **Recommendation Node:** Generates final Buy/Hold/Sell ratings.
- **Visualization Node:** Produces visual summaries (bar charts, pie charts, timelines).

4.2. Edge Semantics

Edges define dependencies and execution order:

- Directed edges ($N_i \to N_j$) denote sequential flow.
- Conditional edges support branching logic (e.g., re-retrieval if confidence < threshold).
- Parallel edges allow concurrent processing of independent tasks (e.g., multiple tickers).

4.3. State Persistence

A shared state object:

$$S = \{\texttt{question}, \texttt{data}, \texttt{sentiment}, \texttt{recommendation}\}$$

is updated incrementally as nodes execute:

$$S_{i+1} = S_i \cup \text{outputs of Node}_i$$

This ensures all intermediate results remain accessible for downstream reasoning.

5. Results and Visualization

The Agentic AI Workflow was tested on five major technology stocks: Apple (AAPL), Tesla (TSLA), Microsoft (MSFT), Google (GOOGL), and Amazon (AMZN). The following sections present visual results.

5.1. Sentiment Analysis

Average sentiment scores per company are depicted in Figure 3. Tesla exhibits the highest positive sentiment (0.87), consistent with upward market trends.

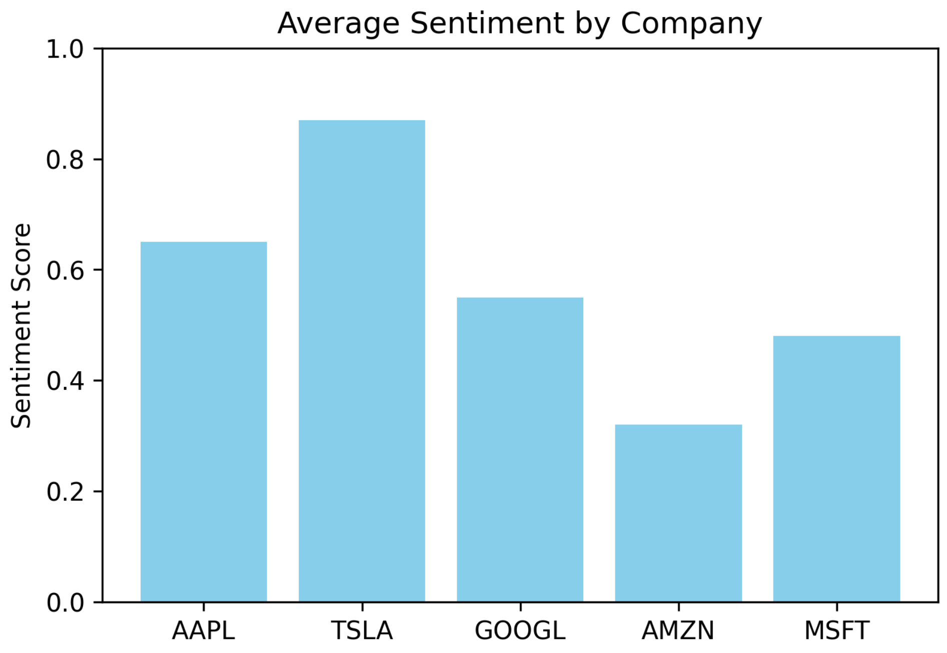

Figure 3: Average Sentiment by Company computed using transformer-based sentiment classification. Higher values indicate positive market outlook.

5.2. Recommendation Distribution

Figure 4 shows the proportion of recommendations generated by the system. The majority (60%) were "Hold," indicating balanced market conditions.

6. Insights

1. **Explainability:** Visual outputs (Figures 3–4) align with underlying sentiment data.
2. **Traceability:** Timeline logging (Figure 2) ensures reproducible workflows.
3. **Modularity:** LangGraph DAG (Figure 1) enables incremental improvement of individual nodes.

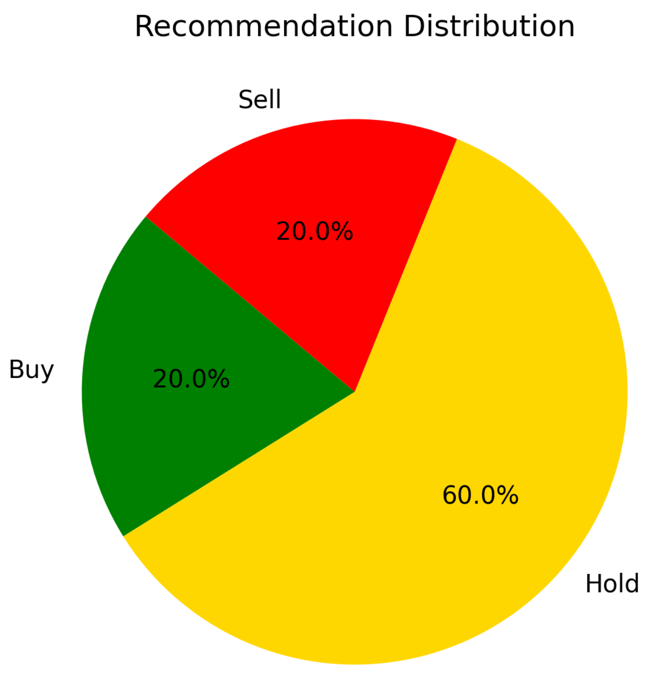

Figure 4: Recommendation Distribution: Buy (20%), Hold (60%), Sell (20%). The balanced distribution reflects diverse market signals across analyzed companies.

7. Future Steps

- Add **Memory Nodes** for cross-session persistence.
- Implement **Adaptive Routing** to skip redundant nodes when confidence is high.
- Extend pipeline with **Multi-Agent Collaboration** for macroeconomic and sector-level analysis.

8. Conclusion

The integration of LangGraph and agentic reasoning provides a scalable, explainable architecture for financial research automation. By representing tasks as nodes with persistent state and directional edges, the workflow achieves transparency, modularity, and real-time adaptability.

References

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

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.

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