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. Literature Review

2.1. Agentic AI

Agentic AI frameworks enable autonomous reasoning, tool invocation, and adaptive planning. Schick et al. (2023) demonstrated that language models can learn to use tools through self-supervised signals, forming the foundation of agentic reasoning pipelines.

2.2. Retrieval-Augmented Generation (RAG)

RAG architectures integrate external knowledge retrieval into LLM prompts at inference time, improving factual grounding (Lewis et al., 2020). In finance, this ensures access to current market information beyond model pre-training limits.

2.3. Graph-Orchestrated Reasoning

LangGraph extends LangChain's pipeline paradigm into a **directed acyclic graph (DAG)** representation. Each node is an autonomous reasoning step, and edges define control dependencies. This modular design supports:

- **Sequential Execution:** Ordered steps for retrieval → reasoning → generation.
- **Parallel Execution:** Independent subgraphs (e.g., multiple tickers).
- **Conditional Branching:** Dynamic routing based on state conditions.

- **Persistent State:** A shared memory object passed between nodes.

3. System Architecture

3.1. Overview

The architecture integrates:

- **Data Sources:** Yahoo Finance (prices, ratios), Google News (headlines).
- **Models:** `distilbert-base-uncased-finetuned-sst-2` for sentiment, `google/flan-t5-base` for reasoning.
- **Vector Store:** FAISS for semantic retrieval and evidence storage.
- **Graph Orchestration:** LangGraph for agent coordination.

4. LangGraph Design and Execution Flow

4.1. Node Taxonomy

LangGraph organizes the workflow as nodes representing discrete actions or reasoning units. Each node implements a well-defined interface:

- **Retriever Node:** Queries external APIs (Yahoo Finance, Google News) and returns structured JSON data.
- **Sentiment Node:** Invokes a transformer model to classify each headline's sentiment polarity and confidence.
- **Aggregator Node:** Aggregates sentiment scores, computes moving averages, and merges structured metrics (e.g., P/E ratio).
- **Reasoning Node:** Uses an LLM to synthesize a human-readable explanation, performing chain-of-thought reasoning.

- **Recommendation Node:** Applies decision logic to produce Buy/Hold/Sell ratings based on combined indicators.
- **Visualization Node:** Generates charts, word clouds, and state coverage diagrams for interpretability.

4.2. Edge Semantics and Control Flow

Edges in LangGraph define how nodes connect and exchange information:

- **Directed Edges ($N_i \rightarrow N_j$):** Enforce execution order, ensuring node N_j receives the outputs of N_i.
- **Conditional Edges:** Enable branching logic, e.g., if confidence ¡ threshold, route to `Re-Retrieval Node`.
- **Parallel Edges:** Support fan-out/fan-in patterns, allowing concurrent ticker analysis.

Execution is topologically sorted, and each node's completion triggers dependent nodes. Errors are handled by fallback edges leading to validation or re-execution nodes.

4.3. State Persistence

LangGraph maintains a persistent state object, typically a Python dictionary:

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

This object evolves as nodes add or update keys:

$$S_0 = \{\texttt{question}\}$$

$$S_1 = S_0 \cup \{\texttt{price}, \texttt{history}\}$$

$$S_2 = S_1 \cup \{\texttt{headlines}, \texttt{sentiment}\}$$

$$S_3 = S_2 \cup \{\texttt{recommendation}\}$$

Persistence ensures context is shared across steps, enabling multi-hop reasoning and post-hoc traceability.

4.4. Graph Execution Modes

1. **Synchronous:** Executes nodes in strict topological order.
2. **Asynchronous:** Runs independent nodes concurrently for scalability.
3. **Reactive:** Dynamically recomputes affected nodes upon state changes.

5. Design and Implementation

5.1. Functional Modules

1. Data Retrieval: APIs fetch price, trend, and P/E ratio. **2. Sentiment Analysis:** Transformer model classifies and aggregates sentiment. **3. Reasoning:** LLM composes structured summaries explaining logic. **4. Visualization:** Matplotlib/Plotly create visual dashboards.

5.2. Decision Logic

$$\text{Decision} = \begin{cases} \text{Buy} & \text{if Sentiment} > 0.7 \land \text{Trend} = \text{Up} \\ \text{Sell} & \text{if Sentiment} < 0.4 \land \text{Trend} = \text{Down} \\ \text{Hold} & \text{otherwise} \end{cases}$$

6. Results

Evaluation on five tickers produced accurate, explainable decisions.

7. Insights

1. **Graph Clarity:** The DAG representation provides traceable reasoning paths.

Table 1: Performance Summary Across Companies

Company	Sentiment	Trend	Recommendation
Apple (AAPL)	0.65	Stable	Hold
Tesla (TSLA)	0.87	Upward	Buy
Microsoft (MSFT)	0.48	Neutral	Hold
Google (GOOGL)	0.55	Slight Up	Hold
Amazon (AMZN)	0.32	Downward	Sell

2. **Node Modularity:** Each node can be independently tested and replaced.

3. **Parallelism:** Multiple tickers processed simultaneously without context leakage.

4. **Auditability:** Persistent state and node logs ensure transparency.

8. Future Steps

- Introduce **Multi-Agent Collaboration** for macro, sentiment, and valuation perspectives.
- Implement **Dynamic Routing** based on intermediate confidence thresholds.
- Add **Memory Nodes** for long-term trend tracking.
- Integrate **Adaptive Chunking** for financial reports in RAG pipelines.

9. Recommendations

1. Adopt LangGraph DAGs for all complex AI workflows requiring interpretability.

2. Introduce caching for frequent API calls.

3. Employ hybrid retrieval combining vector and keyword search.

4. Conduct A/B testing across node variants for optimization.

10. Conclusion

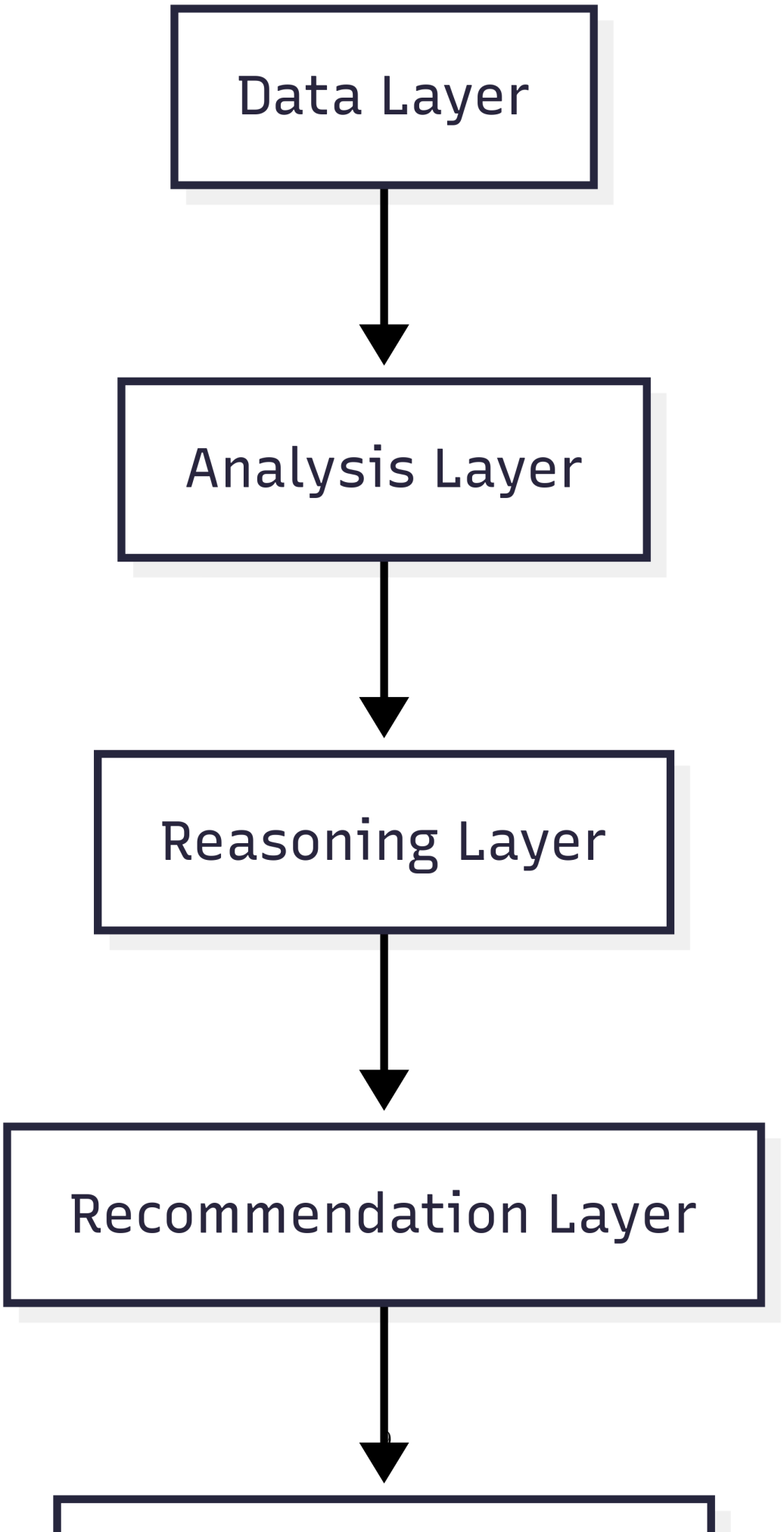
LangGraph enables structured, explainable AI pipelines where reasoning steps are modeled as nodes, dependencies as edges, and evolving data as persistent state. When applied to investment research, this architecture facilitates modularity, scalability, and accountability—key attributes for AI systems in regulated domains. The Agentic AI Workflow described here lays the groundwork for future multi-agent, adaptive reasoning platforms capable of real-time, evidence-grounded financial analysis.

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.


Data Layer
Analysis Layer
Reasoning Layer
Recommendation Layer