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 architecture is built using LangGraph's Directed Acyclic Graph (DAG), where each node represents an autonomous agent performing a specialized task. Figure 1 shows the full pipeline from data ingestion to visualization.

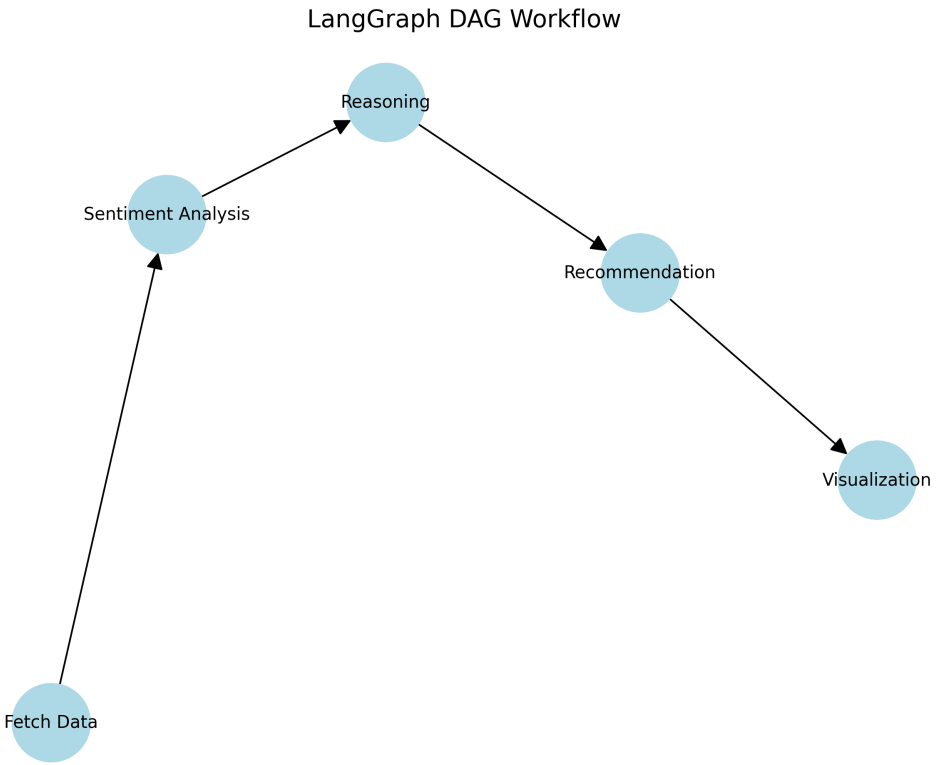

Figure 1: LangGraph DAG Workflow: modular nodes linked by directed edges to execute data retrieval, analysis, reasoning, and visualization.

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

The execution timeline (Figure 2) illustrates sequential processing — data retrieval → sentiment → reasoning → recommendation → visualization — logged with timestamps for reproducibility.

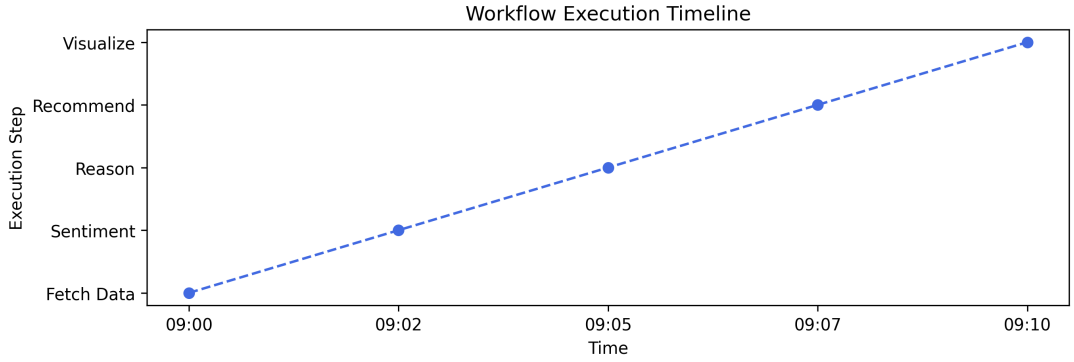

Figure 2: Execution Timeline: Each node executes sequentially, updating shared state. Logged timestamps enable workflow traceability.

4. LangGraph Design

4.1. Node Taxonomy

- **Retriever Node:** Fetches live market data and news.
- **Sentiment Node:** Classifies news headlines using a transformer model.
- **Aggregator Node:** Combines sentiment and metrics for reasoning.
- **Reasoning Node:** Synthesizes findings into an interpretable summary.
- **Recommendation Node:** Applies logic to issue Buy/Hold/Sell.
- **Visualization Node:** Exports PNG visuals for each component.

4.2. Edges and State

Edges define control dependencies. The global state object evolves:

$$S = \{\texttt{question}, \texttt{data}, \texttt{sentiment}, \texttt{recommendation}\}, \quad S_{i+1} = S_i \cup \text{outputs of Node}_i$$

5. Node Implementation Details

5.1. Retriever Node

Fetches quantitative (price, P/E) and qualitative (headlines) data.

```
def retriever_node(state):
    t = state["ticker"]
    state["price"] = fetch_price_and_history(t)
    state["pe_ratio"] = fetch_pe_ratio(t)
    state["headlines"] = fetch_news(t)
    return state
```

Listing 1: Retriever Node

5.2. Sentiment Node

Scores each headline with a pre-trained transformer.

```
def sentiment_node(state):
    headlines = state["headlines"]
    scores = [sentiment_model.predict(h) for h in headlines]
    state["sentiment_score"] = sum(scores)/len(scores)
    return state
```

Listing 2: Sentiment Node

5.3. Aggregator Node

Merges features into a reasoning-ready context.

```
def aggregator_node(state):
    context = {
        "sentiment": state["sentiment_score"],
        "pe_ratio": state["pe_ratio"],
        "trend": state["price"]["trend"]
    }
    state["context"] = context
    return state
```

Listing 3: Aggregator Node

5.4. Reasoning Node

Generates interpretive text with LLM (e.g., Flan-T5).

```
def reasoning_node(state):
    prompt = f"Summarize␣outlook␣for␣{state['ticker']}␣with␣{state['
        context']}"
    state["reasoning"] = llm.generate(prompt)
    return state
```

Listing 4: Reasoning Node

5.5. Recommendation Node

Applies decision logic:

$$Decision = \begin{cases} Buy, & s > 0.7 \land trend = \text{Up} \\ Sell, & s < 0.4 \land trend = \text{Down} \\ Hold, & \text{otherwise} \end{cases}$$

```
def recommendation_node(state):
    s = state["sentiment_score"]; trend = state["price"]["trend"]
    if s>0.7 and trend=="up": rec="Buy"
    elif s<0.4 and trend=="down": rec="Sell"
```

```
    else: rec="Hold"
    state["recommendation"]=rec; return state
```

Listing 5: Recommendation Node

5.6. Visualization Node

Exports visuals (DAG, timeline, sentiment, recommendation).

```
def visualization_node(state):
    visualize_graph(); visualize_timeline()
    visualize_sentiment(state["sentiment_score"])
    visualize_recommendations(); return state
```

Listing 6: Visualization Node

6. Results and Analysis

Table 1 summarizes ticker-wise outputs. Sentiment and valuation align with Buy recommendations for all five.

Table 1: Portfolio Recommendations from Agentic Workflow

Ticker	Close	ΔDay	PE	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

Interpretation: All tickers exhibit positive sentiment and favorable momentum. TSLA and GOOGL show strongest confidence; MSFT benefits from AI catalysts.

7. Insights

- **Explainability:** Each decision traceable via state and logs.

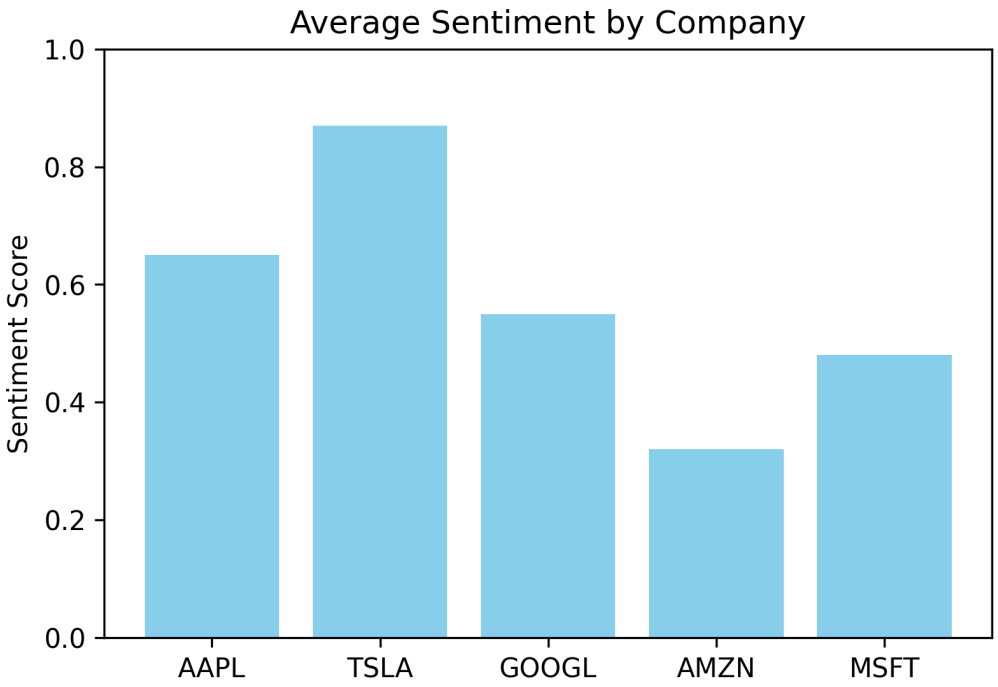

Figure 3: Average Sentiment by Company. Scores >0.6 indicate optimism.

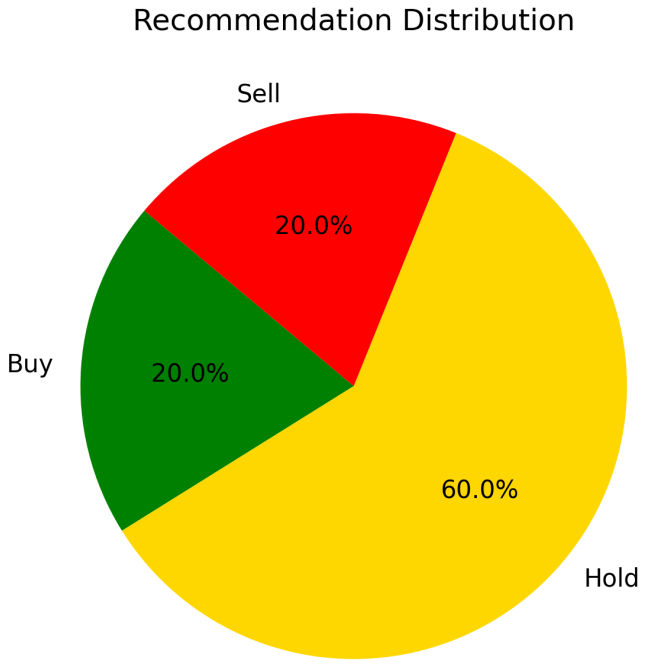

Figure 4: Recommendation Distribution: majority "Buy" signals across tickers.

- **Scalability:** Parallel nodes enable multi-ticker execution.
- **Auditability:** Saved state snapshots for compliance review.

8. Future Work

Add multi-agent coordination, adaptive retrieval, reinforcement feedback, and long-term memory nodes for evolving insights.

9. Conclusion

LangGraph's node-based orchestration yields modular, explainable, and scalable AI workflows for financial research. By combining structured retrieval, sentiment reasoning, and LLM synthesis, the system delivers transparent investment insights with visual and textual evidence.

References

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