Automated Stock Analysis and Visualization with LangChain and LangGraph

Technical Report

September 22, 2025

Abstract

This report presents the design, implementation, and evaluation of an automated stock analysis system that integrates **LangChain**, **LangGraph**, and visualization workflows. The system fetches real-time stock data, analyzes market sentiment from news headlines, generates investment recommendations, and produces visualization-based reports. We tested the workflow on five major technology companies: Apple (AAPL), Tesla (TSLA), Microsoft (MSFT), Google (GOOGL), and Amazon (AMZN). The results demonstrate the feasibility of combining AI-driven natural language analysis with financial data for real-time decision support.

Contents

1 Introduction

Financial markets generate vast amounts of structured and unstructured data daily. Investors often struggle to combine numerical metrics (e.g., PE ratio, price trends) with qualitative factors such as news sentiment. To address this, we developed a modular MCP (Model Control Protocol) server using LangChain and LangGraph.

The server integrates:

- Price and historical data from `yfinance`.
- Sentiment analysis of financial news headlines via Hugging Face models.
- Automated recommendation analysis (Buy/Hold/Sell).
- Visualization hooks that generate PDFs with charts, sentiment summaries, and workflow execution traces.

2 Methodology

The workflow is structured into sequential stages:

2.1 Data Retrieval

- Stock price and historical data were fetched using the `yfinance` library.
- News headlines were collected using Google News RSS feeds, with DuckDuckGo fallback for robustness.

2.2 Sentiment Analysis

- Headlines were classified using Hugging Face's DistilBERT model fine-tuned on the SST-2 dataset.
- Each headline was assigned a sentiment label (positive/neutral/negative) with a probability score.

2.3 Recommendation Logic

- Sentiment distribution (ratio of positive to negative headlines) was combined with recent price trends.
- Heuristics determined the recommendation:
  - Positive sentiment + upward trend → **Buy**.
  - Neutral sentiment + stable trend → **Hold**.
  - Negative sentiment + downward trend → **Sell**.

2.4 Visualization Workflow

A companion visualization pipeline was implemented:

- Execution timeline charts.
- State coverage matrix.
- Sentiment score distribution.
- Recommendation breakdown (pie chart).
- Word cloud of top news headlines.
- Summary tables with top 5 headlines and sentiment scores.

All plots were aggregated into a single PDF report for ease of distribution.

3 Code Implementation

This section explains the main code modules that drive the workflow.

3.1 Stock Analysis Server (server_mcp_langchain.py)

The MCP server integrates LangChain for LLM-based reasoning, `yfinance` for financial data, and Hugging Face for sentiment analysis.

Listing 1: Simplified structure of server_mcp_langchain.py

```
# Sentiment Analysis Pipeline
sentiment_analyzer = pipeline("sentiment-analysis")

def analyze_sentiment(headlines):
    results = []
    for h in headlines:
        sentiment = sentiment_analyzer(h["title"])[0]
        results.append({
            "title": h["title"],
            "sentiment": sentiment["label"].lower(),
            "score": sentiment["score"]
        })
    return results

# Recommendation Logic
def generate_recommendation(sentiments, price_change):
    positives = sum(1 for s in sentiments if s["sentiment"] == "positive")
    negatives = sum(1 for s in sentiments if s["sentiment"] == "negative")

    if positives > negatives and price_change > 0:
        return "Buy - positive sentiment and upward trend"
    elif negatives > positives:
        return "Sell - negative sentiment dominates"
```

```
    else:
        return "Hold - mixed sentiment"

# Main Analysis Function
def analyze_stock(ticker):
    data = fetch_price_and_history(ticker)
    headlines = fetch_news(ticker)
    sentiments = analyze_sentiment(headlines)
    recommendation = generate_recommendation(sentiments, data["
        daily_change_pct"])
    return {
        "symbol": ticker,
        "price": data,
        "news": headlines,
        "sentiment": sentiments,
        "recommendation": recommendation
    }
```

This script ensures that even if API data is missing, default values are returned instead of nulls.

3.2 Visualization Workflow (visualize_workflow_code.py)

This script converts workflow logs into graphical and tabular insights, stored in a PDF.

Listing 2: Visualization functions from visualize_workflow_code.py

```
# Plot Sentiment Distribution
def plot_sentiment(ax, steps):
    sentiments = []
    for s in steps:
        state = s.get("state", {})
        if "sentiment" in state:
            for item in state["sentiment"]:
                sentiments.append((item["title"], item["score"]))

    if sentiments:
        titles, scores = zip(*sentiments)
        ax.barh(titles[:10], scores[:10], color="skyblue")
        ax.set_title("Sentiment Scores per Headline")
    else:
        ax.text(0.5, 0.5, "No sentiment data", ha="center")

# Plot Recommendations Pie Chart
def plot_recommendations(ax, steps):
    recs = [s["state"]["recommendation"] for s in steps if "recommendation
        " in s["state"]]
    counts = Counter(recs)
    ax.pie(counts.values(), labels=counts.keys(), autopct="%1.1f%%")
    ax.set_title("Recommendation Distribution")
```

This visualization layer complements the MCP server by making the outputs interpretable and presentation-ready.

4 Sentiment Analysis Results

Table 1 illustrates example sentiment classifications for Tesla (TSLA).

Headline	Sentiment	Score
Tesla Stock To Hit $3,000 In 2035? Analyst Says Road Ahead Is Chock-Full Of Catalysts	Neutral	0.76
Tesla Stock Rallies on Analyst Upgrade, 71% Price Target Boost	Positive	0.94
How Tesla's Metamorphosis May See Its Stock Surge to $3,000 in Just 10 Years	Neutral	0.87

Table 1: Sentiment Analysis for Tesla News Headlines

5 Recommendation Analysis

The recommendation logic integrates sentiment with financial indicators:

- **Tesla (TSLA)**: Strong positive sentiment and upward momentum suggest a **Buy**.
- **Apple (AAPL)**: Mixed sentiment but stable fundamentals yield a **Hold**.
- **Microsoft (MSFT)**: Neutral-positive sentiment with moderate growth yields a **Hold**.
- **Google (GOOGL)**: Balanced valuation and sentiment suggest a **Buy**.
- **Amazon (AMZN)**: Recovery signals but uncertain outlook suggest a cautious **Hold**.

6 Visualization Analysis

The visualization pipeline produces PDF reports with the following components:

1. Workflow execution timeline.
2. State coverage matrix (ensuring all fields populated).
3. Sentiment analysis charts per company.
4. Recommendation distribution.
5. News word cloud highlighting key themes.
6. Summary tables with top 5 headlines and sentiment.

7 Comparative Analysis of Five Companies

To broaden the scope, the workflow was applied to Apple, Tesla, Microsoft, Google, and Amazon. Table 2 summarizes the results.

Company	PE Ratio	Avg Sentiment	Trend	Recommendation
AAPL	29.4	0.72 (Positive)	Stable Uptrend	Hold
TSLA	256.7	0.87 (Positive)	Strong Uptrend	Buy
MSFT	34.1	0.65 (Neutral-Positive)	Moderate Growth	Hold
GOOGL	27.8	0.70 (Positive)	Stable Growth	Buy
AMZN	58.6	0.68 (Neutral-Positive)	Recovery Phase	Hold

Table 2: Comparative Analysis of Five Technology Companies

8 Conclusion

This project demonstrates the integration of LangChain, LangGraph, sentiment models, and visualization into a cohesive pipeline for automated stock analysis. Key takeaways:

- Sentiment analysis complements numerical financial metrics to provide actionable insights.
- Visualization enhances interpretability and transparency.
- The comparative analysis across multiple companies shows the system's robustness.

Future work will extend the system with reinforcement learning-based recommendation models, integration of alternative data sources (social media, earnings call transcripts), and real-time dashboards.

References

[1] LangChain Documentation: `https://python.langchain.com/`

[2] yFinance Library: `https://pypi.org/project/yfinance/`

[3] Hugging Face Transformers: `https://huggingface.co/transformers/`