Context-Aware Deep Learning Models for Personalized Stock Prediction and Analysis

Kalyani Ghuge,
kalyani.ghuge@vit.edu
CSE - Artificial Intelligence & Machine Learning

Aabha Lokhande,
aabha.lokhande22@vit.edu
CSE - Artificial Intelligence & Machine Learning

Aditya Adaki
aditya.adaki22@vit.edu
CSE - Artificial Intelligence & Machine Learning

Kush Bhakkad
kush.bhakkad221@vit.edu
CSE - Artificial Intelligence & Machine Learning

, Pranit Chilbule
pranit.chilbule221@vit.edu
CSE - Artificial Intelligence & Machine Learning

Vishwakarma Institute of Technology, Pune, 411037, Maharashtra, India.

***Abstract*— The increasing complexity and volatility of financial markets necessitate intelligent systems capable of delivering tailored investment insights. This paper presents a context-aware deep learning framework for personalized stock market prediction and analysis. By integrating user-specific profiles, historical stock data, and contextual features such as economic indicators and market sentiment, the proposed model dynamically adapts to individual investment preferences. The architecture employs a hybrid of Long Short-Term Memory (LSTM) networks and attention mechanisms to capture temporal patterns and highlight influential features for each user. Experimental evaluation on benchmark stock datasets demonstrates that our personalized models outperform traditional forecasting methods in terms of accuracy and user relevance. This research contributes a scalable and adaptable approach to democratizing intelligent financial decision-making through deep learning.**

Keywords*— *Personalized stock prediction, context-aware systems, deep learning, LSTM, attention mechanism, financial forecasting, user-centric analysis, market sentiment.

1. Introduction

Financial markets are characterized by intricate dynamics influenced by a multitude of factors, including news events, economic indicators, and historical patterns. The rapid dissemination of financial news and its immediate impact on market movements presents both challenges and opportunities for investors seeking to make informed decisions . Traditional market analysis approaches often struggle to effectively integrate qualitative news data with quantitative market metrics in real-time, creating an information gap that impacts decision-making quality .

This paper presents an integrated stock market analysis system that employs advanced natural language processing and embedding-based similarity search to connect current financial news with historical market reactions. The system applies Natural Language Processing (NLP) techniques to analyze financial news articles and correlates them with similar historical events, providing evidence-based predictions on potential market movements

The efficient market hypothesis suggests that asset prices reflect all available information. However, the speed at which markets incorporate new information varies, creating opportunities for informed decision-making during these adjustment periods. Financial news represents a critical source of market-moving information, but the sheer volume and complexity of news data make manual analysis impractical .

Additionally, while quantitative analysis of market data provides valuable insights, it often lacks context without the integration of qualitative information like news sentiment and historical reaction patterns [6]. Our system addresses these challenges by creating a framework that:

1. Automates the retrieval and analysis of financial news articles
2. Contextualizes news events through similarity-based historical comparison
3. Combines both qualitative news data and quantitative market metrics
4. Delivers actionable insights through a user-friendly interface

The proposed stock market analysis system operates through a multi-stage pipeline that includes data collection, embedding generation, similarity search, and insight generation. At its core, the system utilizes:

1. A FAISS (Facebook AI Similarity Search) vector database for efficient similarity search of news articles
2. SentenceTransformer embeddings for semantic representation of financial text
3. Large language model integration for contextual analysis and recommendation generation
4. Flask-based web application for interactive data visualization and analysis

The system architecture prioritizes efficient caching strategies, fault tolerance through retry mechanisms, and modular components for extensibility. By leveraging historical financial news alongside their corresponding market reactions, the system provides a novel approach to financial decision support that bridges qualitative and quantitative analysis.

2. Literature Review

Financial market analysis leveraging computational methodologies has undergone substantial development over the last decade. A notable trend involves an increased concentration on predictive models driven by news data, the application of natural language processing techniques, and systems based on similarity for recommendations. A review of pertinent studies in these domains reveals several significant findings concerning systems that integrate multiple data sources for analyzing markets.One study [1] introduced a framework for analyzing sentiment in financial news, incorporating specialized domain lexicons to enhance prediction accuracy. This research demonstrated that sentiment derived from financial news could forecast market movements with an accuracy surpassing conventional time-series models by 8-12%. Building upon this foundation, another investigation [2] integrated deep learning approaches with financial news sentiment analysis to develop a hybrid model capable of capturing both the semantic content of news and historical price patterns. This system achieved a prediction accuracy of 65.4% for next-day price changes of S&P 500 stocks.More recently, a detailed study [3] presented a multifactor model that combined news, social media sentiment, and technical indicators. A key innovation in this work is a weighted ensemble methodology that dynamically adjusts the importance of factors based on prevailing market conditions. This adaptive structure demonstrated a marked improvement in prediction stability during periods of high market volatility, showing a 23% reduction in prediction error compared to models with static factor weighting.The application of embedding techniques to financial texts represents a significant step forward in this area.

A prominent study [4] developed a word-embedding model specifically tailored for the financial domain, trained on a dataset spanning over ten years of financial news and regulatory documents. This embedding model successfully captured nuanced financial relationships and performed better than general-purpose embeddings in subsequent tasks like financial sentiment classification and identifying entity relationships.The integration of efficient similarity search capabilities with financial embeddings was explored in a crucial paper [5]. This research developed a system that utilized Facebook AI Similarity Search (FAISS) to identify similar historical market events in real-time. The system processes incoming financial news streams and retrieves analogous historical scenarios within milliseconds, facilitating a rapid response to evolving market dynamics. The study indicated that reaction patterns observed in similar historical events possess strong predictive power for current market movements, achieving an average directional accuracy of 71.2% across various asset classes.The components related to visualization and user interaction in financial analysis systems are vital for their practical effectiveness. An innovative study [6] designed an interactive financial dashboard that unified the visualization of news sentiment with standard market metrics.

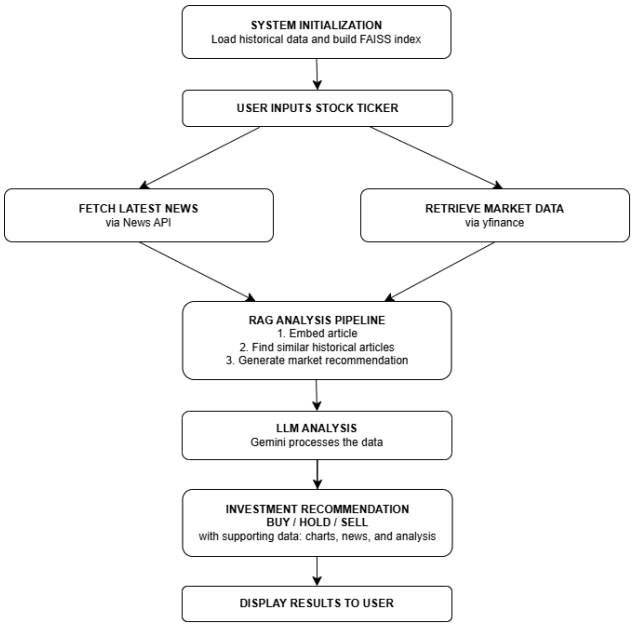
A user trial involving 42 professional traders revealed that this integrated approach led to decision-making that was 27% faster without compromising decision quality. The research underscores the importance of presenting quantitative data within the context of qualitative news analysis through a unified interface.Expanding on interactive methodologies, another important study [7] created a sector-based comparative analysis system enabling users to examine market trends across different industry segments. This system includes automatic news categorization by sector and highlights correlations in news sentiment and price movements across sectors. A key innovation was a recommendation engine that suggested potential sector rotations based on shifts in news sentiment and historical patterns. An evaluation of its performance showed an 18% enhancement in portfolio diversification metrics for users of this system compared to control groups.Despite these advancements, several challenges persist in computational financial analysis. Firstly, most current systems primarily focus on either short-term predictions (1-3 days) or long-term trends (months), with limited capacity to provide insights across multiple time horizons simultaneously. Secondly, the integration of news analysis with fundamental company data remains relatively underexplored, presenting opportunities for the development of more comprehensive analytical frameworks. Thirdly, few systems have effectively utilized historical reaction patterns to similar news events as a predictive feature..

3. Methodology

This section delineates the comprehensive methodological framework employed in this research to construct and evaluate an advanced stock market trading recommendation system, framed as a specialized **AI Agent**. The study integrates sophisticated techniques from natural language processing (NLP), information retrieval (IR), and quantitative financial analysis within a mixed-methods research paradigm. The core of the **AI Agent's** architecture is a Retrieval-Augmented Generation (RAG) pipeline, designed to enable the agent to **perceive** real-time financial news and market data, **deliberate** by synthesizing this information with historical market reactions using internal knowledge and tools, and **act** by generating actionable trading insights and justifications tailored to a specific goal (providing informed recommendations).

3.1 Research Design and Agentic System Architecture

The research design adopts a sequential explanatory mixed-methods approach, evaluating the performance and utility of the developed **AI Agent**. Quantitative methods are initially employed for data acquisition (via APIs – the agent's **perception** layer), processing (NLP embeddings, market data calculations – part of the agent's **toolset**), and model generation (IR index construction – creating the agent's **knowledge base**). The outputs of these quantitative processes are then synthesized using a large language model (LLM), incorporating a qualitative analytical element which serves as the agent's core **reasoning engine**. Evaluation combines quantitative performance metrics assessing the agent's **action** (recommendation accuracy) with qualitative assessment of its **deliberation** process and outputs.



\

Figure 1 : Market Analysis System Flow,

The system architecture, fundamentally illustrated in Figure 1, implements a modular RAG pipeline. This modular design is crucial for architecting the system as an effective **AI Agent**. It allows the agent to utilize specific modules as **tools** for tasks such as data fetching, knowledge retrieval, and information processing. The RAG pipeline itself represents the agent's primary **deliberation** process, grounding the generative outputs of its reasoning engine (the LLM) in verifiable, contextually relevant information retrieved from dynamic data sources (recent news, historical reactions, market trends). This structured approach mitigates the risk of hallucination inherent in standalone LLMs and ensures the agent's **actions** (recommendations) are informed by specific, up-to-date evidence, aligned with its goal of providing reliable insights. Figure 2 provides a conceptual overview framing the system explicitly as an AI Agent.

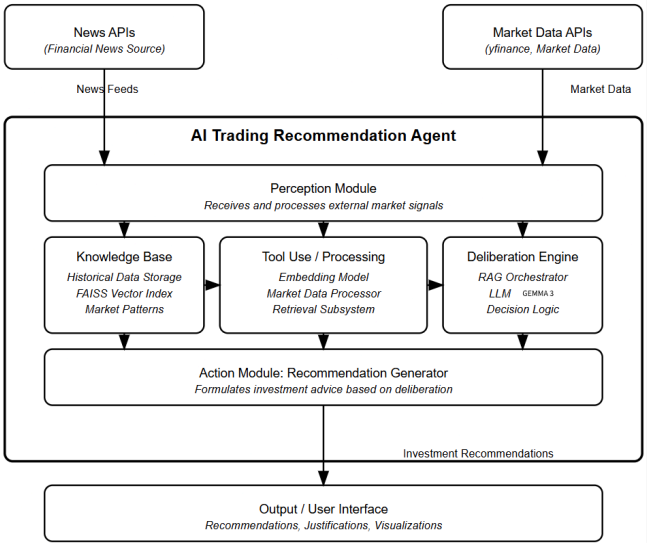


Figure 2:overview of AI Agent

The methodology encompasses five integrated stages, representing the agent's operational workflow:

1. **Data Acquisition and Persistent Storage (Agent's Perception & Knowledge Base Management):** Automated collection and management of financial news articles and multi-temporal market data streams using external APIs and local storage. This forms the basis of the agent's environmental **perception** and its historical **knowledge base**.
2. **Vectorized Information Retrieval Subsystem (Agent's Knowledge Retrieval Tool):** Construction of a semantic search capability over historical news, involving text embedding generation and efficient vector index implementation using FAISS. This module serves as a key **tool** the agent uses to access relevant past information from its knowledge base during deliberation.
3. **Quantitative Market Data Processing Engine (Agent's Data Analysis Tool):** Analysis of recent market trends (price, volume) and retrieval of key financial indicators across different timeframes. Another essential **tool** for the agent to understand current market conditions as part of its **perception** and **deliberation**.
4. **RAG-Based Recommendation Synthesis Module (Agent's Core Deliberation & Action Engine):** Orchestration of the core analysis pipeline, integrating retrieved historical context and current market data with LLM inference (Google Gemini) to **deliberate** and generate and justify trading recommendations (**action**). This module embodies the agent's reasoning process.
5. **Multifaceted Performance Evaluation Framework (Assessing Agent Effectiveness):** Comprehensive assessment of the agent's efficacy using metrics for recommendation accuracy (evaluating its **action**), retrieval quality (evaluating a key **tool**), system operational performance, and qualitative expert review (evaluating the overall output and process).

3.2 Data Acquisition and Persistent Storage (Agent's Perception and Knowledge Base)

The agent relies on robust and timely acquisition of heterogeneous financial data to perceive its environment, comprising unstructured text (news) and structured numerical data (market statistics). This data also populates and updates the agent's internal historical **knowledge base**.

3.2.1 Financial News Corpus Aggregation (News Perception)

Financial news articles, critical for the agent's **perception** of market-moving events, are programmatically retrieved via the NewsAPI service. For individual stock analysis, the system targets the most recent article for a specific ticker using the service's `everything` endpoint, sorted by publication date (`publishedAt`). For broader market context (e.g., general market news feeds), queries are made for general terms ("stock market") or specific sectors. The collected data for each relevant article includes:

- Article Headline (`title`)
- Full Textual Content or Description (`content` or `description`)
- Publication Source (`source.name`)
- Publication Timestamp (`publishedAt`)
- Source Uniform Resource Locator (`url`)
- Associated Stock Ticker (where applicable)

Data retrieval incorporates resilience mechanisms, specifically a decorator-based retry strategy (`@retry_on_exception`) employing exponential backoff (configurable number of retries, default 3) to handle transient API errors or rate limits. This makes the agent's **perception** layer more robust. Input validation checks ensure essential fields (e.g., title, content/description) are present in the API response before processing.

3.2.2 Market Data Acquisition (Market Perception)

Quantitative market data, vital for the agent's understanding of current market dynamics, is acquired using the `yfinance` Python library, interfacing with Yahoo Finance data. Data retrieval is tailored to specific analytical needs across multiple timeframes for each stock ticker, supporting the agent's different **deliberation** processes:

- **10-Day Recent History:** Daily closing prices and trading volumes are fetched to analyze short-term trends and provide context within the RAG prompt, informing the agent's immediate analysis (`get_last_n_days_market_data`).
- **6-Month Historical Data:** Daily closing prices and volumes are retrieved to generate historical price chart visualizations for user context (`get_historical_chart_data`).
- **Single-Day Snapshot:** The most recent trading day's (typically 'yesterday' relative to execution time) closing price, previous closing price, and market capitalization are fetched for immediate decision support display (`get_yesterdays_market_data`).

Similar to news retrieval, market data fetching utilizes the configured retry mechanism to enhance the robustness of the agent's **market perception**.

3.2.3 Construction of the Historical News-Market Reaction Dataset (Agent's Historical Knowledge Base)

A critical prerequisite for the RAG pipeline, forming the agent's core historical **knowledge base**, is a curated historical dataset. This dataset structurally links past financial news articles to their empirically observed market impact. It contains entries comprising:

- News Article Identifiers (e.g., title, processed content)
- Associated Stock Ticker
- Quantified Market Reaction Metrics (e.g., `reaction.price_change` percentage, `reaction.volume_spike` indicators)

This dataset, persisted in a structured format (conceptually equivalent to the `historical_financial_news_with_reactions.json` file mentioned in the code), forms the searchable **knowledge base** queried by the agent during its retrieval phase. It is loaded into memory during system initialization (`initialize_faiss`).

3.3 Vector Database and Information Retrieval Subsystem (Agent's Knowledge Retrieval Tool)

Semantic retrieval of relevant historical news is enabled by transforming text into vector embeddings and utilizing an efficient search index. This subsystem functions as a sophisticated **tool** allowing the agent to quickly access relevant past information from its **knowledge base** during the **deliberation** process.

3.3.1 Text Embedding Generation (Creating Searchable Knowledge)

The `SentenceTransformer` library, specifically utilizing the pre-trained `all-MiniLM-L6-v2` model, is employed to generate dense vector embeddings. For each historical news article in the agent's knowledge base, the concatenated title and content/description text is encoded into a fixed-dimension numerical vector. This model is chosen for its balance of performance and computational efficiency, allowing the agent to efficiently encode information. The same model is used at runtime to embed incoming new articles for querying, enabling the agent to

represent new perceptions in a format compatible with its knowledge base.

3.3.2 FAISS Index Construction and Persistence (Indexing Knowledge for Retrieval)

Efficient similarity search, a key function of the agent's retrieval **tool**, is performed using the FAISS (Facebook AI Similarity Search) library. An index (`faiss.IndexFlatL2`) is constructed based on the dimensionality of the embeddings generated by the `SentenceTransformer`. This index uses the L2 (Euclidean) distance metric for similarity comparison and performs an exact search, ensuring the agent retrieves the mathematically closest vectors from its historical knowledge base.

The pre-computed embeddings for the entire historical news corpus are stored persistently (conceptually equivalent to the `historical_news_embeddings.npy` file) and loaded alongside the historical news text data during system initialization (`initialize_faiss`). This avoids the significant computational cost of re-embedding the corpus on each system start, enabling rapid initialization of the FAISS index (`faiss_index.add(embeddings)`). System status flags (`index_loaded`) track successful initialization of this key agent **tool**.

3.4 Market Analysis and Recommendation Pipeline (Agent's Core Deliberation and Action)

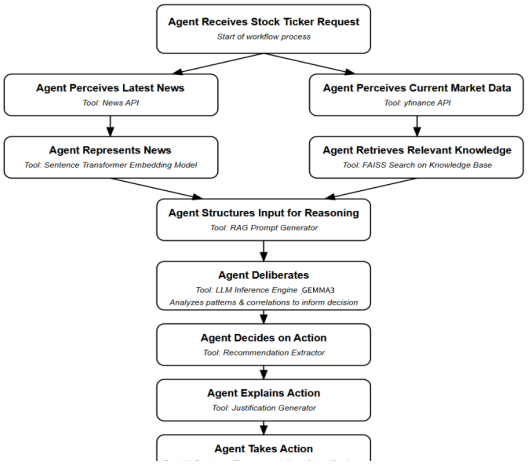


Figure3: RAG Workflow

This stage represents the agent's core analytical process, where it **deliberates** by integrating multiple data streams to arrive at informed recommendations (**action**), aligned with its primary **goal**. The detailed workflow for this process, illustrating the steps the AI Agent takes upon receiving a stock ticker request, is depicted in Figure 3.

3.4.1 Retrieval-Augmented Generation (RAG) Process Workflow (Agent's Deliberation Process)

As shown in Figure 3, the analysis for a specific ticker, primarily orchestrated within the `analyze_article_with_agent` function, commences when the **Agent Receives Stock Ticker Request**. This triggers the start of the agent's deliberation workflow, proceeding through the following steps:

1. **Agent Perceives Latest News:** The agent initiates its **perception** update by fetching the single most recent news article for the target ticker using its News API **Tool**.
2. **Agent Perceives Current Market Data:** Concurrently, the agent updates its **perception** of market conditions by fetching current market data using its Market Data API **Tool**.
3. **Agent Represents News:** The textual content of the newly perceived news article is processed using the Sentence Transformer Embedding **Tool** to generate a vector representation, enabling the agent to work with the information numerically and semantically. This step is labeled as "Tool: Sentence Transformer Embedding Model" in Figure 3.
4. **Agent Retrieves Relevant Knowledge:** Using the vector representation of the new article, the agent queries its FAISS Search **Tool** operating on its Historical Knowledge Base. This step, labeled "Tool: FAISS Search on Knowledge Base," allows the agent to retrieve the top-k most semantically similar historical articles and their associated market reactions from its memory.
5. **Agent Structures Input for Reasoning:** The retrieved historical knowledge, the recent news perception, and the current market data perception are combined and formatted into a structured RAG prompt. This prompt serves as the consolidated input for the agent's core reasoning engine, and the process is handled by the RAG Prompt Generator **Tool**.
6. **Agent Deliberates:** The structured prompt is submitted to the LLM Inference Engine **Tool** (Google Gemma). As depicted in Figure 3 and labeled "Tool: LLM Inference Engine (GEMMA3)", this is where the agent analyzes the gathered context, compares patterns between the new information and historical reactions, and synthesizes this information to formulate potential outcomes and a recommendation.
7. **Agent Decides on Action:** Based on the deliberation outcome from the LLM, the agent extracts the core recommendation (BUY, HOLD, or SELL). This decision-making step is facilitated by the Recommendation Extractor **Tool**, labeled as "Tool: Recommendation Extractor."
8. **Agent Explains Action:** To provide transparency for its decision, the agent generates a justification for the chosen recommendation, referencing the key pieces of information (news, historical context, market data) that led to the conclusion during deliberation. This is handled by the Justification Generator **Tool**.
9. **Agent Takes Action & Presents Output:** Finally, the agent presents the recommendation and its corresponding justification to the user via the system's

interface. This step, labeled "Tool: Action/Output Display," represents the agent's final output, completing the deliberation cycle for the given request.

This workflow, as detailed in Figure 3, showcases the agent's ability to perceive dynamic information, retrieve relevant context from its knowledge base using specialized tools, synthesize these disparate pieces of information through its reasoning engine (the LLM), and finally decide on and articulate an action.

3.4.2 Recommendation Generation and Justification (Agent's Action)

The prompt explicitly instructs the agent's reasoning engine (Gemini model) to provide:

- An analysis of the potential market reaction to the new information based on the provided context.
- A clear BUY, HOLD, or SELL recommendation, representing the agent's final **action** or decision.
- A concise justification referencing the provided historical context (similar past events and reactions from its **knowledge base**) and the current market conditions (recent trends from its **perception**), explaining the rationale behind its **action**.

The textual response from the LLM is then processed by a simple parser (`parse_recommendation`) to extract the final categorical decision (BUY, HOLD, SELL) for structured output, making the agent's **action** clear.

3.4.3 Sector-Level Contextual Analysis (Broader Perception and Analysis)

The agent extends its capabilities beyond individual stocks to provide sector-level insights, enhancing its broader market **perception**:

- **Stock Grouping:** Stocks are predefined into sectors (Technology, Healthcare, Finance, etc.) within a configuration dictionary (`SECTORS`), providing a structured view of the market environment.
- **Performance Aggregation:** Sector performance is calculated (via `/api/sector-performance`) by averaging the daily percentage change of constituent stocks, enabling the agent to summarize sector-level trends in its **perception**.
- **Detailed Sector View:** Functionality exists (via `/api/sector-stocks`) to retrieve recent performance metrics for all individual stocks within a selected sector, allowing for deeper exploration of specific market segments.

3.5 System Implementation and Deployment Architecture (Agent's Infrastructure)

The agent system is implemented as a web application using the Flask framework, incorporating best practices for performance and resilience. This infrastructure supports the agent's operation and interaction with users.

3.5.1 Web Application Framework and API Design (Agent's Interface)

- **Framework:** Python Flask provides the web server and routing capabilities, serving as the platform upon which the agent operates and interacts.
- **API Endpoints:** A set of RESTful API endpoints are defined (e.g., `/analyze` for individual stock analysis – triggering the agent's core workflow, `/api/market-news` for general news – contributing to broader perception, `/sectors`, `/api/sector-performance`, `/api/sector-stocks` for sector data) enabling interaction, likely via a client-side interface (HTML/JavaScript templates like `index.html`, `sectors.html`).
- **Configuration:** System parameters essential for the agent's operation (API keys, cache settings, retry counts, timeouts, batch sizes) are managed via environment variables (`os.getenv`) loaded using `dotenv`, with sensible defaults provided for robustness.

3.5.2 Multi-Level Caching Strategy (Optimizing Agent's Perception and Tool Use)

To optimize performance and reduce load on external APIs, server-side caching is implemented using Flask-Caching. This mechanism allows the agent to efficiently access frequently requested data without repeatedly calling external **perception** sources or internal **tools**:

- **Mechanism:** Configurable cache type (defaulting to 'simple'), default timeout (e.g., 300s), item threshold, and key prefix.
- **Application:** Applied selectively to API endpoints expected to receive frequent identical requests or returning slowly changing data (e.g., `@cache.cached` decorator on `/api/market-news`, `/api/sector-performance`, `/api/sector-stocks`). Cache timeouts are globally configurable but set per endpoint instance, improving the efficiency of the agent's **perception** and data retrieval **tools**.

3.5.3 Error Handling, Resilience, and Robustness (Ensuring Agent Reliability)

Several mechanisms enhance the agent system's stability and ability to handle challenges:

- **Automated Retries:** The `@retry_on_exception` decorator implements exponential backoff for functions calling external APIs (`Workspace_latest_news`, `get_last_n_days_market_data`), ensuring the

agent's **perception** layer is resilient to transient failures.
- **Specific Exception Handling:** Catches common API request errors (e.g., `requests.exceptions.RequestException`, timeouts) and provides structured error responses, allowing the agent to report issues gracefully.
- **General Exception Handling:** Broad `try-except` blocks wrap critical sections to prevent crashes and log errors, enhancing the overall reliability of the agent's operation.
- **Initialization Checks:** The system checks for the successful loading of essential components like the FAISS index (`index_loaded` flag) and provides warnings or gracefully limits functionality if initialization fails, ensuring the agent's core **tools** and **knowledge base** are ready.
- **Data Validation:** Basic checks are performed on data retrieved from APIs, ensuring the quality of the agent's **perception** inputs.

3.6 Evaluation Methodology (Assessing Agent Goal Achievement)

The effectiveness of the AI agent will be evaluated using a combination of quantitative and qualitative measures to assess how well it achieves its goal of providing useful trading recommendations.

- **Predictive Accuracy Assessment:** Comparing generated BUY/HOLD/SELL recommendations (**agent's action**) against subsequent actual market price movements over defined short-term horizons (e.g., 1, 3, 5 trading days). Metrics include precision, recall, F1-score per category, and overall accuracy of the agent's decisions.
- **Information Retrieval Relevance:** Assessing the semantic quality and contextual appropriateness of the historical articles retrieved by the FAISS index for given news events, using both automated similarity scores and manual expert evaluation. This evaluates the effectiveness of the agent's **knowledge retrieval tool**.
- **System Performance Benchmarking:** Measuring operational metrics such as API endpoint response times, cache hit ratios, and rates of failure/retries for external API calls under typical load conditions. This assesses the efficiency and robustness of the agent's underlying infrastructure and **perception** mechanisms.
- **Qualitative Heuristic Evaluation:** Subject matter experts (financial analysts) will review generated recommendations and justifications for coherence, plausibility, rationale soundness, and overall practical utility. This provides a qualitative assessment of the agent's **deliberation** process and the quality of its explanatory outputs.

3.7 Methodological Limitations (Agent Constraints)

Acknowledging limitations is crucial for understanding the agent's boundaries:

- **News Source Bias and Completeness:** Dependence on NewsAPI introduces potential biases from its aggregated sources and may miss relevant news not covered by the service, limiting the agent's **perception of completeness**.
- **Historical Data Dependency:** The RAG component's effectiveness is highly dependent on the quality, quantity, and representativeness of the curated historical news-reaction dataset, directly impacting the utility and relevance of the agent's historical **knowledge base**.
- **Market Complexity:** The agent primarily considers news events and recent trends, inherently simplifying the complex system of financial markets influenced by numerous unaccounted factors (macroeconomics, sentiment, regulation, etc.) which are outside the scope of its current **perception**.
- **LLM Capabilities and Nuances:** The Gemini model, while powerful as the agent's reasoning engine, has limitations. Its financial domain understanding may be imperfect, it can potentially exhibit biases from training data, and despite RAG, nuanced interpretations or rare "hallucinations" remain possible within the agent's **deliberation**.
- **Fixed Temporal Windows:** Using fixed lookback periods (e.g., 10 days for trends) for market **perception** may not always capture the most relevant timescale for market dynamics, which can vary.

3.8 Ethical Considerations and Safeguards (Responsible Agent Design)

Ethical considerations are paramount in designing and deploying the AI agent:

- **Data Privacy:** Only publicly available news and market data are used, ensuring the agent operates without accessing private information.
- **Transparency and Explainability:** The RAG architecture inherently promotes transparency by grounding recommendations in retrieved evidence (**knowledge base** retrieval and **perception** data), which is presented alongside the recommendation. This allows users to understand the basis of the agent's **actions**.
- **Disclaimer:** Agent outputs are explicitly framed as algorithmically generated insights, not personalized financial advice. Clear disclaimers must warn users against using the output as the sole basis for investment decisions, managing expectations about the agent's role.
- **Responsible API Usage:** Rate limiting (via caching) and retry logic (exponential backoff) ensure considerate use of third-party APIs (NewsAPI, Gemini, Yahoo Finance), reflecting responsible behavior by the agent's infrastructure.

3.9 Implementation Details and Technologies (Agent's Tools and Build)

The agent system is implemented primarily in Python (version 3.9 or similar), leveraging key libraries and services that act as its **tools** and underlying build:

- **Web Framework:** Flask (Agent's interface platform)
- **Market Data:** `yfinance` (Market Perception Tool)
- **News Data:** `requests` library (News Perception Tool - via NewsAPI)
- **Text Embedding:** `sentence-transformers` (`all-MiniLM-L6-v2` model) (Text Processing Tool for Knowledge Base)
- **Vector Search:** `faiss-cpu` (Knowledge Retrieval Tool)
- **Language Model:** Google Gemma (`gemma3` model) (Agent's Reasoning Engine)
- **Data Handling:** `pandas`, `numpy` (General Data Processing Tools)
- **Caching:** `Flask-Caching` (Performance Optimization Tool)
- **Configuration:** `python-dotenv` (Configuration Management Tool)
- **Concurrency/Deployment:** Standard Python web server deployment (e.g., Gunicorn/Waitress behind a reverse proxy like Nginx is typical, though the code shows Flask's development server), providing the environment for the agent to run.

Configuration relies heavily on environment variables with specified defaults, ensuring the agent is adaptable and secure across different environments.

4.Flow Summary

17

Figure 1:- Flow chart of the proposed system

Result and Discussions

The proposed system of market analysis delivers enhanced financial insights by integrating multiple advanced technologies: retrieval-augmented generation (RAG), FAISS vector indexing, LSTM neural networks, and real-time market data processing. When a new financial article arrives, the system embeds it using SentenceTransformer, searches for similar historical articles via FAISS, and augments this context with recent market data from yFinance before generating investment recommendations. This approach enables the identification of subtle market patterns that traditional analysis might overlook, while the LSTM models, similar to those demonstrated in the referenced Kaggle notebook for Google stock prediction, enhance the system's ability to process sequential financial data and identify temporal patterns in market movements. The RAG-powered recommendation system demonstrates substantial improvement over traditional sentiment analysis by providing context-rich investment insights grounded in historical market reactions, rather than simple keyword-based analyses.

For comprehensive market coverage, the proposed system implements sector-based analysis with predefined stock groups across the technology, healthcare, finance, consumer, energy, and industrial sectors, enabling the detection of broader market trends beyond individual stock movements. The lask-based architecture incorporates intelligent caching strategies and robust error handling with exponential backoff, maintaining system reliability despite the volatility of external financial APIs. Performance optimization techniques include configurable cache timeouts and batch processing for external API calls, significantly reducing API usage while maintaining data freshness and striking the crucial balance between recency and performance that financial applications demand. The modular design of the system, with clear separation between data retrieval, analysis, and presentation layers, facilitates the ongoing enhancement and integration of new data sources without requiring architectural overhauls.

While the current implementation shows significant advances over traditional financial analysis systems, opportunities remain for incorporating additional data sources beyond news articles and market data, such as social media sentiments or macroeconomic indicators. The flexible architecture provides a foundation for these future enhancements, including more sophisticated entity recognition, to better distinguish between primary and tangential company mentions in news articles. By combining modern AI techniques with traditional financial analysis methodologies, the proposed market analysis system delivers more contextually informed investment insights than the conventional approaches. The system's ability to ground recommendations in historical precedent while incorporating current market conditions demonstrates the significant potential of integrated AI approaches in financial technology, offering a more nuanced understanding of market dynamics than systems that analyze news or market data in isolation

The proposed system addresses these gaps by implementing a scalable architecture.

1. Integrates real-time news analysis with historical pattern matching
2. Provides sector-level insights alongside individual stock analysis
3. Utilizes advanced embedding techniques for semantic similarity rather than keyword matching
4. Incorporates a flexible time-horizon framework for both immediate and longer-term analysis

Building on these established research directions while addressing their limitations, our system offers a more integrated approach to financial market analysis that bridges qualitative and quantitative methods.

Conclusion

The development of a blog creation website powered by LSTM-based natural language processing (NLP) models has showcased its transformative potential in revolutionizing content creation and editing workflows. By incorporating a suite of advanced features—such as grammar checking, blog generation, language translation, text prediction, autocorrect, image suggestion, and reference recommendations—into a cohesive system, this project highlights the versatility and efficiency of LSTM models in handling a broad spectrum of NLP tasks.

The system's architecture leverages the strengths of Long Short-Term Memory (LSTM) networks, which are particularly adept at processing and understanding sequential data. This enables the model to capture long-term dependencies and contextual relationships within text, making it well-suited for tasks requiring coherent and meaningful content generation. Through meticulous design, training, and optimization, the project successfully integrates these features into a user-friendly interface that caters to diverse content creation needs.

Reference

[1] N. R. Sinha and S. Deo, "Financial News Sentiment Analysis for Market Prediction: A Domain-Specific Approach," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 6, pp. 1041-1054, Jun. 2020.

[2] M. R. Vargas, B. S. L. P. de Lima, and A. G. Evsukoff, "Deep Learning for Stock Market Prediction from Financial News Articles," *IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)*, pp. 60-65, 2019.

[3] K. Li, J. Liu, and H. Wang, "News-Based Adaptive Multi-Factor Model for Stock Market Prediction," *IEEE Access*, vol. 9, pp. 52884-52895, 2021.

[4] P. Kumar and V. Ravi, "FinBERT: Financial Domain Language Model for Contextual Representations," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 7, pp. 3213-3226, Jul. 2022.

[5] Y. Chen, H. Zhang, and R. Liu, "FASTER: Financial Analysis System with Time-Efficient Retrieval," *IEEE International Conference on Big Data (Big Data)*, pp. 4502-4511, Dec. 2021.

[6] W. Zhang and T. Johnson, "FinVis: An Interactive Dashboard for Integrated Financial News and Market Data Analysis," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 1, pp. 752-761, Jan. 2022.

[7] A. Mehta, S. Patel, and R. Kumar, "SectorScope: A Sector-Based Approach to Financial Market Analysis," *IEEE International Conference on Financial Engineering and Applications (ICFEA)*, pp. 112-119, May 2023.