INTERNATIONAL UNIVERSITY OF APPLIED SCIENCES

Masters Thesis (DLMMTHE01)

University of Applied Science - Online

Masters in Data Science (MSDS60ECTS)

AI Stock Analyst: Financial Chatbot Using Large Language Models

Gummadi Sai Dheeraj

Matriculation: 3210935

gummadi.saidheeraj@iu-study.org

Advisor: Professor Broweleit, Tobias

Delivery date: January 08, 2025

Contents

I List of Figures

1 Abstract

In recent years, the rapid developments in the field of machine learning and artificial intelligence has profoundly impacted various fields, particularly the financial services sector. The growth of machine learning techniques, including data mining, neural networks, and expert systems has significantly advanced developments in financial security, risk management, and related areas. Among these advancements, the progress in Natural Language Processing (NLP) has been particularly swift, with large language models (LLMs) demonstrating powerful capabilities and potential applications in various fields.

This paper explores the application of a Retrieval-Augmented Generation (RAG) based chatbot for financial stock analysis, utilizing GPT-4o and Gemini 1.5 flash models. The integration of RAG with LangChain aims to revolutionize the analysis and interpretation of stock market data, enabling precise, data-driven investment decisions. The RAG model addresses several limitations of traditional LLMs, such as the need for additional training to adapt to new data, the resources and effort required for customization, and the potential for generating inaccurate information. The growing intricacy and amount of financial information underlines the need for an LLM-based solution for stock analysis. Traditional methods of stock analysis are often time-consuming and require a lot of expertise, making it hard for individual investors to keep pace with the rapidly changing market conditions. An LLM-based solution, for example, the RAG-based chatbot, can process vast volumes of data with great speed and accuracy, therefore providing investors with timely and relevant information that guides their investment decisions.

The benefits of using a RAG-based chatbot for financial stock analysis are manifold. Firstly, it enhances the accuracy of stock market predictions by integrating reliable data sources, such as the Yahoo Finance, which contributes to more accurate identification and prediction of stock market trends. Secondly, the combination of the RAG model and LangChain enables the chatbot to provide precise, data-driven investment decisions, taking the application of AI in finance to a new level. Thirdly, RAG-based chatbot for providing both fundamental and technical analysis of stocks. Fundamental analysis assesses a company's financial performance and market attributes, whereas technical analysis aims to forecast stock price changes based on past data and market patterns. By integrating both approaches, the RAG-based chatbot can provide a comprehensive analysis of stocks, enabling investors to make informed decisions.

The integration of RAG with LangChain and the use of GPT-4o and Gemini 1.5 flash models offer a promising solution for financial stock analysis. The RAG-based chatbot enhances the accuracy and efficiency of stock market predictions, provides precise, data-driven investment decisions, and addresses the limitations of traditional LLMs. Through the integration of fundamental and technical analysis, the chatbot provides a thorough method for analyzing stocks, enabling investors to make educated choices in a financial market that is becoming more intricate and dynamic.

2 Introduction

2.1 The Evolving Landscape of Financial Markets

The financial markets have undergone significant transformations over the past few decades, driven by technological advancements, globalization, and the increasing complexity of financial instruments. This evolving landscape has brought about new challenges and opportunities, particularly in the context of information overload, the limitations of traditional methods, and the promise of Artificial Intelligence (AI) and Large Language Models (LLMs). This paper explores these three critical aspects in detail, emphasizing the impact of LLMs in financial services. In the digital age, the sheer volume of information available to stock market participants has grown exponentially. Financial news, market data, economic indicators, social media sentiment, and corporate disclosures flood the information channels daily. This phenomenon, known as information overload, presents both opportunities and challenges for investors, analysts, and financial institutions.

Information overload, a phenomenon arising from this development, is one that opens avenues as well as challenges investors, analysts, and financial institutions. The scenario takes place when the available information exceeds the capability of an individual to process the same with the intention of making appropriate decisions. Some of the resultant problems in the financial markets have been mentioned below.

- Decision paralysis: The sheer volume of data may overwhelm investors, making it difficult to identify the right information, thus causing indecision or delayed action.
- Noise vs Signal: Distinguishing between valuable insights (signal) and irrelevant data (noise) becomes increasingly difficult, potentially leading to poor investment decisions.
- Cognitive Biases: The abundance of information can exacerbate cognitive biases, such as confirmation bias [7], where individuals favor information that confirms their preexisting beliefs.

The rise of information overload has profound implications for financial markets.

- Market Volatility: Rapid dissemination of news and data can lead to increased market volatility [20] as investors react to new information, often without fully understanding its implications.
- Short-Termism: The steady flow of information promotes short-term trading strategies, as investors look to take advantage of quick market fluctuations instead of focusing on long-term fundamentals. [55].
- Inefficiencies: Information overload can create inefficiencies in the market, as not all participants have equal access to or the ability to process information effectively.

Traditional methods of financial analysis and decision making, face significant limitations in the context of modern financial markets. These methods include fundamental analysis, technical analysis, and quantitative models, each with its own set of challenges.

Fundamental analysis entails assessing a company's financial stability, the effectiveness of its management, its standing within the industry, and the overall economic environment to establish its intrinsic value [18]. However, ordinary people face the following challenges for fundamental analysis:

- Data Intensity: Fundamental analysis requires extensive data collection and analysis, which can be time-consuming and prone to human error [9].
- Subjectivity: Analyst interpretations of financial statements and economic indicators can vary, leading to subjective conclusions [72].
- Lagging Indicators: Financial statements and economic reports are often lagging indicators that provide a historical view rather than real-time insights [2].

Technical analysis emphasizes the analysis of past price trends and trading volumes to forecast future market behavior [27]. While widely used, it has its drawbacks:

- Historical Bias: Technical analysis depends on past data, which may not consistently forecast future trends accurately.
- Market Anomalies: Market anomalies and unexpected events can disrupt established patterns, rendering technical analysis less effective.
- Overfitting: Complex technical models can overfit historical data, leading to false signals in real-time trading.

Quantitative models use mathematical and statistical techniques to analyze financial data and make predictions. Despite their sophistication, they face limitations:

- Model Risk: Quantitative models are only as good as their underlying assumptions and data inputs. Incorrect assumptions or poor data quality can lead to significant errors.
- Black Box Nature: Numerous quantitative models function as ”black boxes,” which complicates the understanding of the decision-making process for users [40].
- Adaptability: Financial markets are dynamic, and models may struggle to adapt to changing conditions or new types of data.

Artificial Intelligence (AI) and Large Language Models (LLMs) signify a significant transformation in the realm of financial analysis and decision-making. These technologies have the capacity to tackle numerous issues related to information overload and the constraints of conventional approaches.

AI and LLMs excel at processing vast amounts of data quickly and accurately. This capability is particularly valuable in financial markets, where timely and accurate information is crucial. AI and LLMs can enhance decision-making processes by providing more accurate, data-driven insights and reducing cognitive biases. AI and LLMs can democratize financial analysis by making sophisticated tools accessible to a broader audience. While AI and LLMs offer significant promise, they also present challenges and ethical considerations that must be addressed

2.2 The Problem Statement: Bridging the Gap in Financial Stock Analysis

In the realm of financial markets, the ability to predict stock movements accurately is a highly coveted skill. However, the complexity of financial data and the myriad factors influencing stock prices make this task exceedingly challenging. Traditional models often fall short in providing clear, understandable explanations for their predictions, leaving investors in the dark about the rationale behind the recommendations. This lack of transparency can erode trust and confidence in the predictive models, making it imperative to develop systems that not only predict stock movements but also explain the reasoning behind these predictions in a comprehensible manner.

Explainable stock recommendations are crucial for several reasons. Firstly, they enhance the credibility of the predictive models. When investors understand the factors driving a recommendation, they are more likely to trust and act on it. Secondly, explainable recommendations facilitate better decision-making. Investors can weigh the provided explanations against their own knowledge and insights, leading to more informed and confident investment decisions. Lastly, regulatory bodies and compliance requirements increasingly demand transparency in financial advisories, making explainable recommendations not just a preference but a necessity.

The financial industry is rife with uncertainty and volatility. Investors, ranging from individual retail investors to large institutional players, rely heavily on stock recommendations to guide their investment strategies. However, the current state of stock analysis often leaves much to be desired. Many existing models and tools provide recommendations without sufficient context or explanation, forcing investors to take a leap of faith based on opaque algorithms.

This lack of transparency poses a significant business problem. Financial advisors and analysts are under constant pressure to deliver accurate and reliable recommendations. When these recommendations are not accompanied by clear explanations, it undermines the advisor's credibility and can lead to a loss of client trust. Moreover, in a highly competitive market, the ability to provide explainable recommendations can be a key differentiator, setting a firm apart from its competitors.

The absence of explainable stock recommendations leads to numerous missed opportunities. Investors may overlook potentially lucrative investments simply because they do not understand the rationale behind a recommendation. Conversely, they might invest in stocks that appear promising based on a recommendation but lack a solid underlying justification, leading to potential losses. For instance, consider a scenario where a predictive model identifies a stock as a strong buy due to its recent performance metrics. Without an explanation, investors might miss the fact that the stock's performance is driven by a temporary market anomaly rather than sustainable growth factors. This lack of insight can result in missed opportunities for profit and, more critically, missed opportunities to avoid losses.

Suboptimal investment decisions are a direct consequence of the lack of explainable stock recommendations. When investors lack a solid grasp of the reasons behind a stock recommendation, they are more inclined to make choices based on partial or misconstrued information. This may result in various adverse consequences, including poor portfolio performance and considerable financial setbacks. For example, an investor might receive a recommendation to sell a particular stock. Without an explanation, the investor might not realize that the recommendation is based on short-term market fluctuations rather than long-term fundamentals. As a result, they might sell the stock prematurely, missing out on potential future gains. Conversely, they might hold onto a stock that is recommended for sale without understanding the underlying risks, leading to potential losses.

The financial markets are inherently risky, and the lack of explainable stock recommendations exacerbates this risk. When investors do not understand the reasoning behind a recommendation, they are more likely to make decisions that expose them to unnecessary risk. This is particularly problematic in volatile markets, where the ability to quickly and accurately assess the factors driving stock movements is crucial.

Increased risk can manifest in several ways. Investors might overexpose themselves to certain sectors or stocks based on recommendations they do not fully understand. They might also fail to diversify their portfolios adequately, leading to increased vulnerability to market fluctuations. Furthermore, the lack of transparency can lead to a false sense of security, where investors believe they are making informed decisions when, in reality, they are operating with incomplete information.

Given the challenges outlined above, there is a clear and pressing need for a solution that bridges the gap in financial stock analysis. This solution must address the need for explainable stock recommendations, providing investors with clear and comprehensible justifications for the forecasts generated by predictive models. Such a solution would enhance trust, facilitate better decision-making, and reduce the risks associated with opaque recommendations.

One promising approach to addressing this need is the development of self-reflective large language models (LLMs) that can generate explainable stock predictions. These models leverage advanced natural language processing (NLP) techniques to analyze vast amounts of unstructured text data, such as news articles, social media posts, and financial reports. By summarizing this data and generating human-readable explanations, these models can provide investors with the context and insights they need to make informed decisions.

In summary, the demand for transparent stock suggestions has become increasingly important in the intricate and fluctuating landscape of modern financial markets. The lack of transparency in traditional predictive models leads to missed opportunities, suboptimal decisions, and increased risk for investors. By developing solutions that provide clear and understandable explanations for stock predictions, this paper can bridge the gap in financial stock analysis, enhancing trust, facilitating better decision-making, and ultimately improving investment outcomes. The SEP framework represents a promising approach to achieving this goal, leveraging the power of self-reflective large language models to generate explainable stock predictions that meet the needs of modern investors.

2.3 Introducing Retrieval-Augmented Generation (RAG) for Financial Stock Analysis

In the fast-moving world of finance, real-time data integration is crucial for accurate and timely analysis. Market situations can shift quickly,, and investors need access to the latest information to make informed decisions. RAG's ability to integrate real-time data is a game-changer in this regard.

Retrieval-Augmented Generation (RAG) [43] is a sophisticated approach that improves the functionality of conventional Large Language Models (LLMs) by incorporating a phase for information retrieval into the process of generating answers. Traditional LLMs, while powerful, have limitations in real-world applications, such as the need for additional training to adapt to new data, high resource requirements for customization, and the potential to produce inaccurate information [56]. RAG addresses these issues by combining the strengths of information retrieval systems with the generative capabilities of LLMs.

RAG operates in retrieval and generation phases. In the retrieval phase, the model first identifies and retrieves relevant data from a document database or knowledge base in response to a user's query. This ensures that the model can access the most pertinent and current information. In the generation phase, the LLM uses the retrieved documents to generate contextualized and accurate answers. By leveraging external data sources, RAG can provide more precise and detailed responses than a standalone LLM.

The integration of RAG into financial stock analysis offers several significant benefits.

- Improved Accuracy: By utilizing a separate database that is not solely dependent on the LLMs training data, RAG can deliver more precise and trustworthy information. This aspect is particularly important in financial analysis, where precision is crucial for well informed investment choices.

- Expanded Knowledge Base: RAG allows the model to search for and incorporate external data in real-time, leveraging a broader body of knowledge. This capability is essential in the dynamic and fast-paced world of finance, where new information constantly emerges.

- Real-Time Data Integration: The capability to incorporate real-time data ensures that the analysis reflects the latest market conditions and trends. This is vital for investors who must make prompt decisions based on the latest information.

- Enhanced Decision-Making: By providing comprehensive and up-to-date analysis, RAG empowers investors to make more informed and data-driven decisions. This can lead to better investment outcomes and reduced risk.

- Efficiency and Scalability: RAG streamlines the process of data collection and analysis, making it more efficient and scalable. This is particularly beneficial for financial institutions and analysts who need to process large volumes of data quickly.

The retrieval phase of RAG involves querying a curated dataset to find relevant financial data and historical trends that align with the user's query context. This dataset can include real-time stock prices, trading volumes, news articles, and additional relevant information. By incorporating this real-time data into the analysis, RAG ensures that the generated reports represent the latest market circumstances.

For example, if an investor queries the model for an analysis of a specific stock, RAG can retrieve the latest earnings reports, news articles, and market trends related to that stock. The LLM combines this information with its existing knowledge to produce a thorough and current analysis. This real-time integration allows investors to respond swiftly to stock market volatility and make more informed decisions.

LLMs play a crucial role in the RAG framework by providing the generative capabilities needed to produce detailed and contextually relevant answers. LLMs like GPT-3 and newer versions, have shown exceptional proficiency in natural language understanding and generation. They can interpret complex queries, synthesize information from multiple sources, and generate coherent and informative responses.

In the RAG framework, the LLM is responsible for the generation phase, where it uses the retrieved documents to produce contextualized answers. The model attempts to leverage the information from the retrieved documents to provide more accurate and detailed responses. This process involves several key steps:

- Query Interpretation: The LLM interprets the user's query and identifies the key information needed to generate a relevant response.

- Document Synthesis: The model synthesizes the information from the retrieved documents, combining it with its pre-trained knowledge to generate a comprehensive answer.

- Contextualization: The LLM contextualizes the answer by incorporating relevant details from the retrieved documents, ensuring that the response is accurate and informative.

- Generation: Finally, the model generates the answer in a coherent and readable format, providing the user with a detailed and contextually rich response.

The incorporation of LLMs in the RAG framework improves the model's capability to provide accurate and relevant financial analysis. Utilizing the generative nature of LLMs, RAG can produce detailed and informative reports that help investors make better decisions.

2.4 LLMs: A Primer

The GPT-4o [29] and Gemini 1.5 Flash-001 [69] architectures represent significant advancements in the field of LLMs, each with unique methodologies and design principles that cater to different aspects of natural language processing and understanding.

GPT-4o, developed by OpenAI, is an evolution of the Generative Pre-trained Transformer series. It builds upon the architecture of its predecessors with enhancements in scale, training data, and fine-tuning techniques. The core of GPT-4's architecture is the transformer model, which utilizes self-attention mechanisms to process and generate text. This model has undergone pre-training on a diverse corpus of text data, allowing it to grasp a broad spectrum of language patterns, facts, and reasoning abilities. The pre-training phase involves supervised learning, where the model predicts the next word in a sentence, thereby learning contextual relationships and language structure.

One of the key features of GPT-4o is its ability to perform zero-shot and few-shot learning [25]. This means that the model can understand and generate responses to tasks it has not explicitly been trained on, simply by being given a few examples or even just the task description. This capability is largely attributed to the extensive and diverse pre-training data, which includes books, articles, websites, and other text sources. Additionally, GPT-4o incorporates advanced fine-tuning techniques, where the model is further trained on specific datasets to enhance its performance on particular tasks, such as financial analysis or sentiment detection.

On the other hand, Gemini 1.5 Flash, developed by Google, represents a different approach to LLM architecture. While it also employs a transformer-based model, Gemini 1.5 Flash focuses on optimizing the efficiency and speed of the model [60]. This is achieved through innovations in model architecture and training processes. One of the notable features of Gemini 1.5 Flash is its use of attention mechanisms, which allow the model to focus on the most relevant parts of the input data, thereby reducing computational overhead and improving processing speed.

Gemini 1.5 Flash also incorporates advanced techniques for model compression and Knowledge Distillation [81]. These techniques reduce the size of the model and the amount of computational resources required for inference, making it more suitable for deployment in resource-constrained environments. Furthermore, Gemini 1.5 Flash leverages a combination of supervised and unsupervised learning during its training phase. This hybrid approach allows the model to benefit from the structured learning of specific tasks while also gaining the broad contextual understanding from large-scale unsupervised pre-training [28].

When comparing GPT-4o and Gemini 1.5 Flash, several key differences and similarities emerge. Both models utilize transformer architectures and self-attention mechanisms, which are fundamental to their ability to process and generate natural language. However, GPT-4o emphasizes versatility and generalization through extensive pre-training on diverse datasets, enabling it to perform well on a wide range of tasks with minimal task-specific training. In contrast, Gemini 1.5 Flash prioritizes efficiency and speed, incorporating attention and model compression techniques to optimize performance in real-time applications.

In terms of practical applications, GPT-4o's strength lies in its ability to handle complex and diverse language tasks with high accuracy, making it suitable for applications that require deep understanding and generation of text, such as content creation, customer support, and financial analysis. Gemini 1.5 Flash, with its optimized architecture, is better suited for applications that require fast and efficient processing, such as real-time language translation, conversational agents, and mobile applications. When comparing GPT-4o and Gemini 1.5 Flash

2.5 Fundamental and Technical Analysis: A Primer

Investing in financial markets requires a comprehensive understanding of various analytical methods to make informed decisions. Fundamental analysis and technical analysis are the two primary schools of thought dominate the stock market landscape. Each method provides distinct insights and resources, suites to various investment strategies and time horizons. This primer delves into the intricacies of both fundamental and technical analysis, highlighting their methodologies, strengths, weaknesses, and practical applications.

Fundamental analysis aims to determine the intrinsic value of a security by examining economic, financial, and qualitative factors [21]. This method is rooted in the belief that a security's true worth is reflected in its underlying fundamentals, which include financial reports, sector developments, and macroeconomic indicators. The key components of the fundamental analysis of stock are listed below.

- Financial Statements: The cornerstone of fundamental analysis involves closely examining a company's financial statements [61], iwhich consist of the income statement, balance sheet, and cash flow statement. These reports offer an overview of the company's financial condition, earnings potential, and cash management.
- Income Statement: This document shows the organization's earnings, costs, and profit [68] for a designated timeframe. Analysts utilize it to evaluate profitability and operational effectiveness.
- Balance Sheet: The balance sheet offers a summary of the company's assets, liabilities, and shareholders' equity at a specific moment [66]. It aids analysts in assessing the financial stability and leverage of the company.
- Cash Flow Statement:This statement monitors the movement of cash entering and leaving the organization [67], emphasizing its capacity to produce cash from operations, invest in expansion, and fulfill financial responsibilities.
- Economic Indicators: Fundamental analysts examine macroeconomic elements like GDP growth, inflation levels, interest rates, and employment statistics. These indicators offer understanding of the overall economic landscape and its possible influence on the company's performance.
- Industry Analysis: Understanding the industry in which a company operates is crucial. Analysts examine industry trends, competitive dynamics, regulatory changes, and technological advancements to gauge the company's position and growth prospects.
- Qualitative Factors: Beyond quantitative data, fundamental analysis also considers qualitative aspects such as management quality, corporate governance, brand value, and competitive advantages. These factors can significantly influence a company's long-term success.

The strength and weakness of fundamental analysis are given below:

- Long-Term Focus: Fundamental analysis is ideal for long-term investors who want to discover undervalued assets that have the potential for growth.
- Comprehensive Evaluation: By examining a wide range of factors, fundamental analysis offers a comprehensive perspective of a company's intrinsic value.
- Investment Decisions: It helps investors make informed decisions about buying, holding, or selling securities based on their intrinsic value.
- Time-Consuming: Conducting thorough fundamental analysis requires extensive research and data collection, making it labor-intensive.
- Subjectivity: Qualitative factors can be subjective and vary from one analyst to another, leading to differing interpretations.
- Information Lag: Financial reports and economic data are often released with a delay, potentially making the analysis outdated.

- Market Sentiment: Fundamental analysis may overlook short-term market sentiment and investor psychology, which can influence price movements.

Technical analysis, on the other hand, emphasizes on past price trends and volume data to predict the price changes in the future [46]. This method operates on the premise that market psychology and historical trends that can offer valuable insights into future trends. Key components of technical analysis are mentioned below.

- Price Trends: Technical analysts use various types of price trends, such as line charts, bar charts, and candlestick charts, to visualize historical price movements. These charts help identify patterns and trends.

- Technical Indicators: These calculations involve price, volume, or open interest metrics. Popular indicators consist of moving averages, the relative strength index (RSI), and moving average convergence divergence (MACD). These tools assist analysts in recognizing momentum, identifying overbought or oversold situations, and spotting possible trend reversals.

- Volume Analysis: Volume analysis examines the number of shares or contracts traded in a security or market during a specific period. It provides insights into the strength of price movements and potential reversals.

- Support and Resistance Levels: Support and resistance levels are price points in a market where traders consider an asset to be overvalued or undervalued. These levels can be used to identify potential buy or sell signals, and to confirm the strength of a trend.

- Chart Patterns: Technical analysts study price patterns such as head and shoulders, double tops and bottoms, and triangles. These patterns can suggest potential future price movements based on historical behavior.

- Candlestick Patterns: Candlestick charts display price movements using rectangular "bodies" and thin vertical lines called "wicks" or "shadows." These patterns can indicate investor sentiment, market trends, and potential reversals.

Many investors and traders opt for a blended approach, leveraging the strengths of both fundamental and technical analysis. This combination provides a more comprehensive strategy, addressing both the true value of assets and sentiment of stock. The practical applications of fundamental and technical analysis are given below:

- Long-Term Investing: Fundamental analysis is well-suited for long-term investors looking to spot undervalued assets that have the potential for growth. By analyzing a company's financial stability and its position within the industry, investors can make educated choices about whether to retain or sell their securities.

- Short-Term Trading: Technical analysis is well-suited for short-term traders looking to capitalize on price movements. By analyzing historical price patterns and volume data, traders can identify entry and exit points for profitable trades.

- Blended Approach: Integrating both approaches allows investors to make well-rounded decisions. For example, an investor might apply fundamental analysis to identify a promising stock and technical analysis to pinpoint the optimal time to buy or sell.

Fundamental and technical analyses are essential tools for navigating the financial markets. While fundamental analysis focuses on uncovering the true value of assets through financial statements, economic indicators, and qualitative factors, technical analysis looks at past price trend and volume information to predict future trends. Each approach has its own advantages and disadvantages, , and many investors find value in combining both methods to inform their decisions. Ultimately, the choice between fundamental and technical analysis depends on an investor's trading style, investment horizon, and financial objectives. By understanding the principles behind these analytical methods, investors can improve their capability to make informed and profitable investment decisions.

2.6 Research objectives and Outline

The primary objective of this research is to develop and evaluate a web-based RAG chatbot that leverages advanced LLMs, specifically GPT-4o and Gemini 1.5 flash models, for financial stock analysis. The study aims to compare the responses of these two models to determine their effectiveness and reliability in providing accurate and insightful financial analysis. The benefits of using a RAG chatbot for financial stock analysis are highlighted, including improved accessibility of information, real-time updates, and the integration of data from various sources to provide a comprehensive view of the stock market. However, the paper also addresses the limitations of using a RAG chatbot, such as dependency on data quality, model limitations in understanding complex financial data, and potential user experience issues. The research proposes the use of the RAG chatbot for both fundamental and technical analysis of stocks, aiming to enhance the decision-making tools available to investors. The thesis is structured as follows: the Introduction section provides an overview of the stock market's data generation and the importance of analysts' reports, setting the stage for the need for an automated solution. The Literature Review section delves into previous research on machine learning, natural language processing, and their applications in financial markets, highlighting the advancements and gaps that this study aims to address. The Methodology section outlines the development of the RAG chatbot, detailing the integration of LangChain and the use of APIs to gather relevant data. It also describes the prompt design and the role of agents in generating comprehensive stock analysis reports. The Results section presents the findings of the study, comparing the performance of GPT-4o and Gemini 1.5 flash models in generating accurate and useful financial analysis. It includes an analysis of the chatbot's effectiveness in real-time stock analysis and its potential impact on investment decisions. The Conclusion section summarizes the key achievements of the research, emphasizing the improved accessibility and accuracy of stock analysis reports generated by the RAG chatbot. It also discusses the significance of the study in democratizing access to sophisticated financial analysis and outlines future research directions, such as integrating more data sources, improving models, and enhancing real-time data processing capabilities. Overall, this section provides a detailed roadmap of the research, highlighting its objectives, methodology, and anticipated contributions to the field of financial stock analysis.

3 Literature Review

The finance industry has been transformed by the introduction of artificial intelligence (AI). The field of AI-powered stock analysis has shown great promise, using sophisticated algorithms to examine large datasets, spot trends, and produce insights that can guide investment choices. The current body of research on AI stock analysis is examined in this review of the literature, along with the different approaches used, their advantages and disadvantages, and their effects on the financial environment. This paper will look at the use of machine learning methods in sentiment analysis, risk assessment, portfolio optimization, and stock price prediction, including deep learning, natural language processing (NLP), and reinforcement learning. Additionally, this review will look into how to improve investment decision-making processes by combining AI with conventional financial analysis methods like technical and fundamental analysis. In order to identify important trends, knowledge gaps, and possible directions for further research in the field of AI stock analysis, this review will synthesize the current body of research. Building on this framework, this review will examine in greater detail the particular uses of AI in stock analysis. This paper will examine the application of AI algorithms to the analysis of news stories, sentiment on social media, and economic indicators in order to forecast changes in stock prices. The application of AI to the creation of algorithmic trading strategies will also be covered in the review.

3.1 Machine Learning in Finance

The paper by Vo. et al. [73] explores leveraging deep learning's power to enhance Socially Responsible Investing (SRI) and portfolio construction. Traditional SRI, reliant on often inconsistent and subjective ESG ratings from third-party agencies, faces challenges like greenwashing, data sparsity, and lagging indicators. Deep learning offers a solution by processing diverse data sources, including unstructured data like news and social media, to uncover complex relationships between ESG factors and financial performance. This data-driven approach allows for more nuanced company assessments, going beyond traditional metrics. The paper examines various deep learning architectures, such as Recurrent Neural Networks (RNNs) for analyzing time-series data like financial performance and news sentiment, Convolutional Neural Networks (CNNs) for processing image data like satellite imagery for environmental impact assessment, and Graph Neural Networks (GNNs) for understanding relationships within supply chains and industries. These architectures enable more accurate ESG scoring, enhanced portfolio optimization by predicting future performance based on ESG factors and investor preferences, improved risk management by identifying ESG-related risks like environmental disasters, and more effective impact measurement. Deep learning can create more adaptable and robust SRI strategies by continuously learning and adapting to new data, reducing human bias in the process. However, challenges remain, including data availability and quality, the need for explainable AI to build trust and transparency, computational resource requirements, and ethical considerations surrounding data privacy and potential misuse. The paper concludes that while these challenges require further research, deep learning holds immense potential to revolutionize SRI, paving the way for a more sustainable and impactful investment landscape by offering a more data-driven and objective approach to ESG integration.

The paper Goncalves et al. [23] explores the burgeoning applications of deep learning techniques within the complex landscape of financial exchange markets. Traditional trading strategies often rely on simplified models that struggle to capture the intricate dynamics and non-linear relationships present in market data. Deep learning, with its ability to learn complex patterns from vast datasets, offers the potential to significantly improve prediction accuracy and trading performance. The paper examines various deep learning architectures, including Recurrent Neural Networks (RNNs), particularly LSTMs and GRUs, for their ability to process sequential data like price time series and news sentiment, and Convolutional Neural Networks (CNNs) for analyzing image-based data like candlestick charts and order book visualizations. These models are applied to a range of tasks, including price prediction, volatility forecasting, market regime detection, and algorithmic trading strategy development. Deep reinforcement learning is also discussed, enabling agents to learn optimal trading strategies through interaction with simulated or real market environments. The paper highlights the advantages of deep learning, such as its ability to handle high-frequency data, incorporate diverse data sources including news and social media sentiment, and automatically adapt to changing market conditions. However, it also acknowledges the challenges, including the need for large, high-quality datasets, the ”black box” nature of some deep learning models, the risk of overfitting, and the computational demands of training complex architectures. Furthermore, the paper emphasizes the importance of careful model validation and risk management strategies to mitigate potential losses. Overall, the paper concludes that deep learning offers a powerful set of tools for analyzing and predicting exchange market dynamics, with the potential to transform trading strategies and improve market efficiency, but further research is needed to address the existing challenges and unlock the full potential of these techniques in this complex and rapidly evolving domain.

The paper by Wang et al. [75] investigates the application of deep learning for enhancing portfolio construction by incorporating a preselection stage based on long-term financial data analysis. Traditional portfolio optimization methods often struggle with the curse of dimensionality when dealing with a large universe of assets, leading to unstable and overfitted portfolios. This paper proposes a two-stage approach: first, a deep learning model preselects a subset of promising assets based on their long-term historical performance and other relevant features; second, a traditional portfolio optimization technique, such as mean-variance optimization or Black-Litterman allocation, is applied to the preselected subset. The deep learning model, typically a Recurrent Neural Network (RNN) like LSTM or GRU, learns complex temporal patterns from long-term financial data, including price history, fundamental indicators, and macroeconomic variables, to predict future asset returns or rank assets based on their expected performance. This preselection stage significantly reduces the dimensionality of the optimization problem, leading to more stable and robust portfolios. The paper explores various deep learning architectures and preselection strategies, comparing their performance against traditional portfolio construction methods. It demonstrates that the proposed approach can achieve superior risk-adjusted returns and improve portfolio diversification. Furthermore, the paper investigates the impact of different input features, data preprocessing techniques, and hyperparameter tuning on the performance of the deep learning model. While acknowledging the benefits of deep learning for preselection, the paper also addresses the challenges, including the need for large, high-quality datasets, the risk of overfitting, and the computational cost of training complex models. It emphasizes the importance of careful model validation and robust risk management practices. Overall, the paper concludes that incorporating deep learning-based preselection into portfolio formation can significantly enhance investment performance by leveraging the information embedded in long-term financial data and mitigating the challenges associated with high-dimensional optimization problems.

The paper Lei et al. [42] introduces a novel approach to algorithmic trading that combines deep reinforcement learning with a time-driven, feature-aware financial signal representation. Traditional algorithmic trading strategies often rely on hand-crafted features and predefined rules, which may not capture the complex and dynamic nature of financial markets. This paper proposes a jointly trained deep reinforcement learning framework that simultaneously learns an effective representation of financial signals and an optimal trading policy. The framework consists of two key components: a feature extraction module and a reinforcement learning agent. The feature extraction module, typically a deep neural network, learns to extract relevant features from raw market data, such as price and volume, while incorporating time dependencies and feature importance. This time-driven, feature-aware representation captures the temporal dynamics and hierarchical relationships among different features, providing a more informative input for the reinforcement learning agent. The agent, typically a deep Q-network or policy gradient algorithm, learns an optimal trading policy by interacting with the market environment and receiving rewards based on its trading performance. The joint training process allows the feature extraction module and the reinforcement learning agent to co-adapt and optimize their respective functionalities, leading to a more effective and robust trading strategy. The paper evaluates the proposed framework on various financial datasets and demonstrates its superior performance compared to traditional trading strategies and other deep learning-based approaches. It highlights the advantages of the time-driven, feature-aware representation and the joint training paradigm in capturing market dynamics and improving trading performance. However, the paper also acknowledges the challenges, including the need for large, high-quality datasets, the computational cost of training complex models, and the risk of overfitting. It emphasizes the importance of careful model validation and robust risk management practices. Overall, the paper concludes that the proposed time-driven, feature-aware jointly deep reinforcement learning framework offers a promising new direction for algorithmic trading, enabling the development of more adaptive and profitable trading strategies in complex and dynamic financial markets.

3.2 LLMs in Finance

The integration of Large Language Models (LLMs) into financial analysis has garnered significant attention in recent years [54], with numerous studies highlighting their transformative potential.

Kirtac and Germano [38] delve into the application of LLMs in predicting stock market returns through sentiment analysis of financial news. They compare the performance of three LLMs such as OPT, BERT, and FinBERT against the traditional Loughran-McDonald dictionary, using a dataset of 965,375 U.S. financial news articles spanning from 2010 to 2023. Their findings underscore the superior performance of the GPT-3-based OPT model, which achieves a prediction accuracy of 74.4% for stock market returns. The study employs a long-short trading strategy based on the OPT model, which, after accounting for transaction costs, yields an exceptional Sharpe ratio of 3.05 and a 355% gain from August 2021 to July 2023. This research highlights the necessity of sophisticated language models for developing effective investment strategies based on news sentiment, marking a pivotal shift from traditional sentiment analysis methods. The authors emphasize the challenges of integrating text mining into financial models due to the complexity of handling and interpreting unstructured text data, advocating for ongoing research and innovation driven by artificial intelligence to enhance financial market prediction and portfolio management.

In a broader context, Zhao et al. [80] explores the extensive impact of LLMs like ChatGPT in the financial sector. They highlight the advancements in natural language processing (NLP) that enable LLMs to effectively understand and generate human language, facilitating applications such as automating the creation of financial reports, predicting market trends, evaluating investor sentiment, and providing tailored financial guidance. The authors conducted holistic tests on multiple financial tasks using natural language instructions, showing that GPT-4o efficiently adheres to prompt instructions in a variety of financial contexts. Their study aims to enhance the comprehension of LLMs present functions in finance, uncover new opportunities for research and application, and demonstrate how these technologies can address real-world issues within the finance sector. Despite the challenges of applying LLMs to the financial sector, such as the complexity of financial data and the need for high accuracy and reliability, the paper emphasizes the ongoing refinement of algorithms and the combination of expert systems and manual review mechanisms to improve model performance. The authors conclude that LLMs are becoming powerful tools for dealing with financial problems, capable of processing and analyzing vast quantities of data while offering detailed insights and suggestions.

Dolphin et al. [15] in their paper ”Extracting Structured Insights from Financial News: An Augmented LLM Driven Approach” present a novel method for processing financial news using LLMs. They address the challenge of converting unstructured financial news into structured data, which is crucial for decision-making in the financial sector. Traditional methods rely on pre-structured data feeds, which are often inconsistent and limited by the inclusion of pre-tagged instrument identifiers like tickers. The proposed system identifies relevant company tickers directly from unstructured news articles, conducts sentiment analysis at the company level, and produces summaries without depending on pre-structured data feeds. The methodology merges the generative capabilities of LLMs with modern prompting techniques and a strong validation framework that utilizes a customized string similarity method. The system collects financial news articles from various providers through a live aggregate news feed from Google News, parses the articles, and uses LLMs to extract key details such as titles, summaries, keywords, associated companies, and sentiment details. An evaluation using a dataset of 5,530 financial news articles shows the system's efficacy, with 90% of articles accurately identifying tickers compared to existing data providers and 22% of articles uncovering additional relevant tickers. This method improves the depth and quality of the extracted information and expands the variety of news sources, representing a noteworthy leap forward in delivering comprehensive, high-quality news data in a structured format.

A significant research effort by Kim et al [34] explores the capabilities of GPT-4 Turbo in performing financial statement analysis similar to professional human analysts. Anonymizing and standardizing corporate financial reports, the researchers ensured that the LLMs predictions were not influenced by prior knowledge or memory of specific companies. Their findings revealed that GPT-4 Turbo could generate state-of-the-art inferences about a company's future performance, often outperforming human analysts and specialized machine learning models. The study highlighted the model's ability to generate useful narrative insights and its effectiveness in predicting the direction of future earnings through a sophisticated Chain-of-Thought (CoT) prompt. This prompt mimicked the thought process of financial analysts, involving the identification of trends, computation of financial ratios, and economic interpretations. The results suggested that LLMs could play a central role in decision-making, challenging the traditional view of financial analysts as the backbone of informed decision-making in financial markets.

In a related vein, Guo and Hauptmann [26] explored the application of LLMs in forecasting stock returns using financial news. Their study emphasized the importance of return forecasting in quantitative investing, crucial for tasks such as stock picking and portfolio optimization. They proposed a model that integrates text representation and forecasting modules, comparing encoder-only and decoder-only LLMs to understand their impact on forecasting performance. The study found that decoder LLMs-based prediction models performed better in larger investment universes, while no consistent winner emerged in smaller universes. Among the LLMs studied—DeBERTa, Mistral, and Llama—Mistral showed robust performance across different universes. The authors contrasted the conventional multi-step feature extraction-and-validation process with a direct news-to-return prediction approach using fine-tuned LLMs. Their research involving actual financial news and different investment areas revealed that LLM-generated text representations were strong signals for portfolio construction, outperforming conventional sentiment scores. This research contributes to the field by offering insights into appropriate text representations for various investment strategies and markets, showcasing the effectiveness of fine-tuning LLMs for predicting stock returns based on news flow.

Furthermore, Ding et al [14] present an extensive overview of the existing studies on the application of LLMs as agents in financial trading. Their survey systematically analyzes 27 papers that explore the application of

LLMs in financial trading, identifying common architectures, data inputs, and performance metrics. The survey categorizes LLM-based trading agents into two categories: LLM as a Trader and LLM as an Alpha Miner. LLM Trader agents are responsible for making direct trading decisions like BUY, HOLD, and SELL, while Alpha Miner agents use LLMs to produce high-quality alpha factors for integration into downstream trading systems. The authors highlight the importance of architecture in designing LLM-based agents, emphasizing that the primary objective is to optimize returns through trading decisions.about the kinds of data utilized by LLMs to make well-informed trading choices, including financial news, market data, and financial statements. The paper reviews various trading strategies employed by LLM agents, such as ranking-based strategies and sentiment analysis, and evaluates their performance using metrics like cumulative return, annualized return, Sharpe ratio, and maximum drawdown. The authors observe that although metrics for both risk and profit are frequently utilized, there are limited studies that take trading costs into account in their assessments. Additionally, they emphasize the importance of monitoring the predictive power of generated signals using metrics like F1 score, accuracy, and win rate. The survey identifies challenges in the current research, such as the need for more robust evaluation methods and the combination of LLMs with various machine learning methods. The authors conclude by outlining future research directions, including the development of more sophisticated LLM architectures, the exploration of new data sources, and the improvement of evaluation frameworks. This survey provides valuable insights into the current state of LLM-based financial trading agents and serves as a foundation for future research in this emerging field.

These studies collectively underscore the significant impact that LLMs can have on financial analysis and trading. They highlight the ability of LLMs to process and analyze large volumes of data, generate insightful predictions, and enhance decision-making processes traditionally reliant on human expertise. The findings from Kim et al [34] and Guo and Hauptmann [26] demonstrate the effectiveness of LLMs in financial statement analysis and stock return forecasting, respectively, while Ding et al. [14] present a thorough review of the current landscape of financial trading agents based on LLMs. Together, these studies contribute to the growing body of literature on the application of LLMs in finance, offering valuable insights into their capabilities, limitations, and potential future directions. As the field progresses, further research is needed to investigate the combination of LLMs with other machine learning methodologies, the development of more sophisticated architectures, and the improvement of evaluation frameworks to fully tap into the capabilities of LLMs in financial analysis and trading.

The study titled ”Harnessing Earnings Reports for Stock Predictions: A QLoRA-Enhanced LLM Approach” by Haowei Ni et al [53] addresses these limitations by leveraging LLMs enhanced with Quantized Low-Rank Adaptation (QLoRA) compression. This method combines ’base factors’ such as growth in financial metrics and earnings call transcripts with ’external factors’ like recent performances of market indices and analyst ratings to construct a detailed, supervised dataset. The extensive dataset enables their models to achieve enhanced predictive accuracy, winning out in metrics like weighted F1 and Matthews correlation coefficient (MCC), outperforming benchmarks like GPT-4. The authors emphasize the effectiveness of the llama-3-8b-Instruct-4bit model, which demonstrates notable advancements compared to baseline models. They also discuss explore the potential for broadening the output functionalities to incorporate a ’Hold’ option and extending the prediction timeframe to suit various investment strategies and durations. This research meticulously details the data collection and preprocessing steps, including the conversion of raw financial data into descriptive sentences, ensuring a robust and multifaceted analysis. The dataset covers 501 firms within the S&P 500, accounting for periodic fluctuations in the index. The authors utilized the API from Financial Modeling Prep to acquire extensive financial data, including market performance metrics, analyst grades, and earnings surprises. They transformed this data into textualized forms to enhance its context and processing ease for the model. The paper’s methodology section outlines the framework, comprising dataset configuration, instruction-based fine-tuning, QLoRA compression, and evaluation utilizing specific prompts and outputs. The authors demonstrate the model's robustness by handling a wide range of text lengths, ensuring comprehensive training. They prepared two separate datasets, Base and Full, to evaluate the effects of different features on model performance, assessing the relative influence of internal versus external data on predictive accuracy. The paper concludes by validating the model's effectiveness through experimental results, showcasing its ability to provide actionable insights for investors navigating the challenges associated with post-earnings stock fluctuations.

In a similar vein, Kuldeep Singh, Simerjot Kaur, and Charese Smiley [62] presents an innovative solution to streamline financial decision-making by leveraging LLMs. The authors developed FinQAPT, a comprehensive pipeline designed to handle the vast volume of financial documents and extract relevant information efficiently. The pipeline comprises main components: FinPrimary, FinContext, and FinReader. FinPrimary understands queries and locates relevant financial documents, FinContext extracts detailed contexts from these reports, and FinReader utilizes LLMs to perform numerical reasoning and generate answers. The authors introduced novel techniques such as clustering-based negative sampling to enhance context retrieval and Dynamic N-shot Prompting to improve numerical reasoning capabilities. They evaluated the pipeline using the FinQA dataset, achieving an impressive module-level accuracy of 80.6%. However, they noticed a decline in performance at the pipeline level attributed to challenges in integrating pertinent contexts from financial documents. The paper highlights the limitations of existing open-domain question-answering methods in addressing the specific challenges of the financial domain and emphasizes the necessity for tailored retriever models. The authors conducted a detailed error analysis, identifying issues such as contextual complexity, question complexity, and calculation complexity. They also noted the impact of noisy and implicit information on the model's performance. Despite these challenges, the study underscores the potential of FinQAPT to enhance financial analysis through innovative context integration techniques. The authors propose future work to develop advanced methodologies to merge relevant contexts from multiple pages of financial documents and explore other models to improve the pipeline's accuracy and robustness. They also emphasize the need of more extensive datasets that reflect the complexities and subtleties of financial analysis tasks. The research concludes by highlighting the importance of addressing these challenges to formulate a robust solution for managing intricate financial tasks and enhancing the performance of each module as well as the overall pipeline for financial analysis.

The paper by Rithesh Harish Bhat and Bhanu Jain [5] at the University of Texas at Arlington investigates a novel approach to predicting stock price movements by analyzing the emotional tone of financial news headlines. The authors address the challenge of restricted web scraping by utilizing APIs to retrieve financial news headlines, thereby eliminating the need of direct web scraping of financial information. They propose leveraging a light and computationally efficient Distilled LLM (Large Language Model) to capture the emotional tone and strength of these headlines. The study involves training the Distilled LLM model to predict emotions embedded in the headlines and then using this emotional data with machine learning classification algorithms such as Logistic Regression, Artificial Neural Networks (ANN), and Random Forest to predict stock price direction. The authors demonstrate that emotion analysis-based attributes can predict stock price direction with accuracy comparable to using financial data alone. They selected 32 mega-cap companies from the United States for their dataset, which included financial news and stock price attributes like open price, close price, volume, and daily high and low prices. The researchers used newsapi.org to gather news headlines and Alpha Vantage to fetch financial data, overcoming the limitations of web scraping by using these aggregators. They fine-tuned the Distilled LLM model with manually labeled news headlines to enhance the accuracy of emotion analysis. The study's results show that emotion analysis alone can provide accurate stock price predictions, suggesting an alternative to traditional financial data-based prediction methods. The authors also discuss the limitations and challenges faced, such as the restricted access to historical news data and the need for larger datasets to improve accuracy. They propose future enhancements, including incorporating news from social media platforms like Twitter and Reddit and analyzing the full content of news articles beyond headlines. The paper concludes that integrating Distilled LLM models, emotion analysis, and machine learning classification algorithms offers a promising approach to stock price trend prediction, paving the way for further research in this domain.

The paper ”Optimized Financial Planning: Integrating Individual and Cooperative Budgeting Models with LLM Recommendations” by de Zarzà et al [79] introduces innovative methodologies for financial planning tailored to individuals and households, leveraging large language models (LLMs) to provide economically sound and goal-aligned budget plans. This research underscores the potential of LLMs to distill complex financial data into actionable insights, promoting fiscal resilience and stability. By building upon traditional financial optimization theories such as the life-cycle hypothesis and modern portfolio theory, and incorporating behavioral aspects of financial decision-making, the authors demonstrate the advantages of LLM-integrated models over traditional methods. This approach not only makes financial planning more accessible and effective for a wider audience but also highlights the importance of AI-driven recommendations in enhancing financial decision-making processes. The study's detailed methodology, simulation framework, and comparative analysis provide a robust foundation for understanding the potential of LLMs in financial planning, paving the way for future research to expand utility functions, analyze model robustness, integrate real-time data, incorporate human expertise, and address ethical considerations.

The paper by John Morgan and Phillip C. Stocken [51] and delves into the information content of stock reports when investors are uncertain about a financial analyst's incentives. The authors explore how the alignment or misalignment of these incentives affects the credibility and informativeness of stock recommendations. They find that any uncertainty about an analyst's incentives makes full revelation of information impossible, with categorical ranking systems naturally arising as equilibria in such environments. Analysts with aligned incentives can credibly convey unfavorable information but struggle to credibly convey favorable information. The study provides a robust theoretical framework supported by statistical tests using published data, highlighting the importance of understanding the incentives behind analyst's recommendations. This lack of transparency is particularly evident in the interaction between investors and financial research analysts employed by securities firms, which offer both investment banking and brokerage services. The separation of these services is crucial because research analysts may face pressure from the investment banking division to issue favorable stock reports. The paper emphasizes the importance of understanding the incentives behind analysts' recommendations, as these incentives significantly impact the information conveyed to investors. The authors extend the model of Crawford and Sobel [12] to include uncertainty about the degree of divergence in preferences between the sender (analyst) and the receiver (investor), showing that analysts are unable to fully reveal their private information due to this uncertainty. The study concludes by offering empirical implications and suggesting that the results can be tested using real-world data, yielding useful perspectives on the behaviors surrounding stock recommendations and the role of analysts' incentives in shaping the information available to investors. Wu's [77] research on the Chinese A-share market exemplifies this trend by leveraging LLMs to analyze how news affects stock prices. By scraping news briefings from November 2022 to October 2023 and scoring stocks daily, Wu's study follows the methodology proposed by Lopez-Lira and Tang [47] to assess the correlations between news scores and stock returns. The research categorizes news into fundamental and market types, comparing different sources and backtesting strategies for each category. The findings reveal that the market reacts more strongly to negative news, challenging the Efficient Market Hypothesis (EMH) in the short term. Additionally, the study compares the stock prediction capabilities of ChatGPT, Tongyi Qianwen, and Baichuan Intelligence, demonstrating the significant impact of LLM-based news scores on trading strategies. This research emphasizes the potential of

LLMs to enhance stock forecasting and portfolio performance, particularly in markets where traditional financial theories like EMH may not fully apply.

In a similar vein, Kim and Oh [35] introduce an innovative approach to stock market analysis by combining advanced Natural Language Processing (NLP) techniques with dynamic data retrieval systems integrated with LLMs. Their method utilizes the Retrieval-Augmented Generation (RAG) framework and LangChain to produce comprehensive stock analysis reports. This approach addresses the limitations of traditional financial analysis, which often fails to deliver targeted, accessible information for the average investor. By providing real-time updates and reducing reliance on manual data interpretation, the proposed method democratizes access to sophisticated financial analysis typically reserved for experts. The authors emphasize the importance of analyst reports in the stock market, noting that these reports help investors improve their investment portfolios, make decisions, reduce potential losses, and increase returns. The LLM-based chatbot solution developed by Kim and Oh aims to improve information accessibility for ordinary investors, enhance the consistency and quality of reports, and instantly reflect rapidly changing market data. This research highlights the potential for future work to incorporate broader data types like real-time news sentiment and global economic indicators, potentially expanding the reach and accuracy of predictive models.

Bhat and Jain [5] explore a novel method for forecasting stock price movements by examining the sentiment present in financial news headlines. Their study addresses the challenge posed by financial portals restricting web scraping by utilizing API-based strategies to obtain financial news headlines. They propose that emotion analysis of these headlines can be as effective in predicting stock price direction as traditional methods that rely on financial data. The study employs a light and computationally efficient Distilled LLM Model to capture the sentiment and intensity of the headlines. The sentiment data is then used with machine learning classification algorithms, including Logistic Regression, Artificial Neural Networks (ANN), and Random Forest, to predict stock price movements. The results demonstrate that emotion analysis-based attributes alone can yield prediction accuracy comparable to that achieved using financial data. The authors highlight the limitations of their approach, such as restricted access to historical news data and incomplete text of news articles from free API tiers. They suggest future enhancements, including expanding the dataset, incorporating news from social media platforms, and analyzing the full content of news articles to improve prediction accuracy. This study concludes that integrating Distilled LLM Models with emotion analysis and machine learning classification algorithms offers a promising alternative to traditional stock price prediction methods, paving the way for further research in this domain.

4 Methodology

This section details the methodology employed in the research. The approach integrates advanced Natural Language Processing (NLP) techniques, dynamic data retrieval systems, and Large Language Models (LLMs) to provide comprehensive and reliable stock analysis. The methodology is divided into several key steps, each contributing to the overall systems effectiveness and accuracy.

4.1 Data and Prompting

In this research, the primary objective is to enhance stock analysis by leveraging Retrieval-Augmented Generation (RAG) pipelines, integrating both fundamental and technical evaluation with the trustworthiness of Large Language Models (LLMs). The data collection process is a essential part of the methodology, guaranteeing that the information is comprehensive and accurate to feed into the analytical models. This section outlines the detailed steps involved in collecting the necessary data, including stock information and related news articles, and the subsequent sentiment analysis.

The first step in the data collection process involves identifying the companies for which the stock analysis need to performed. All companies listed and detailed on the Yahoo Finance website are included, ensuring comprehensive coverage across various industries. This approach eliminates any market capitalization constraints, enabling the analysis of companies of all sizes. The selection leverages the availability of extensive financial data and news articles on Yahoo Finance, making it a reliable source for conducting analysis. This inclusive methodology ensures that the approach is applicable to a wide range of companies, from small-cap to mega-cap, across diverse industries.

Once the target company is identified, the system proceeds to collect detailed stock information from Yahoo Finance. This platform is chosen due to its extensive coverage of financial data and its reliability. The stock information collected includes key metrics such as

- Earnings Per Share (90 Days Ago): EPS (Earnings Per Share) Trend over 90 days ago indicates the company's earnings performance per share three months prior. This metric helps in understanding the historical earnings trajectory and provides a baseline for comparing more recent trends.

- Earnings Per Share (60 Days Ago): Similar to the 90-day EPS trend, this metric shows the earnings per share performance two months ago. It helps in identifying any changes or patterns in the company's profit over a shorter timeframe compared to the 90-day trend.

- Earnings Per Share (30 Days Ago): This key represents the EPS trend one month ago. It gives recent snapshot of the company's earnings performance, allowing analysts to detect any significant changes or trends in the short term.

- EPS Trend (Current): The current EPS trend indicates the most recent earnings per share data. It is essential for evaluating the current financial status and performance of the company accurately.

- Growth Estimate: The growth estimate indicates the projected rate of increase in the company's earnings or revenue. This forward-looking metric is vital for investors and analysts to gauge the company's capacity for future expansion and profitability.

- Volatility: Volatility measures the degree of variation in the company's stock price over a specific period. It is expressed as a percentage and indicates the risk associated with the stock's price movements. Increased volatility indicates a higher level of risk and the possibility of significant price fluctuations.
- PE Ratio: The Price-to-Earnings (PE) ratio serves as a valuation metric that evaluates a company's current stock price in relation to its earnings per share. This ratio assists investors in assessing whether a stock is priced too high or too low compared to its earnings.
- Beta measure: Beta assesses how much a stock's price fluctuates in relation to market changes. A beta above 1 signifies that the stock experiences greater volatility than the overall market, whereas a beta below 1 indicates reduced volatility. This measure is utilized to evaluate the stock's risk in the market.
- Alpha: Alpha represents the stock's performance relative to a benchmark index. A positive alpha indicates that the stock has outperformed the benchmark, while a negative alpha suggests underperformance. It is a measure of the stock's excess return.
- R-squared: R-squared measures the proportion of the stock's movements that can be explained by movements in the benchmark index. It ranges from 0 to 1, with higher values indicating a stronger correlation with the benchmark.
- Standard Deviation: Standard deviation quantifies the amount of variation or dispersion in the stock's returns. It is a measure of risk, with higher standard deviation indicating greater variability in returns.
- Sharpe Ratio: The Sharpe ratio assesses the return of a stock relative to its risk. It is determined by dividing the excess return, which is above the risk-free rate, by the returns' standard deviation. A greater Sharpe ratio signifies superior risk-adjusted performance.
- Stock Price Open History: This refers to the historical data of the opening prices of a stock over a specified period.The opening price refers to the value at which a stock begins trading when the exchange opens for the day. Analyzing the open price history helps in understanding the initial market trend and investor behavior at the start of trading sessions.
- Stock Price Close History: This is the historical data of the closing prices of a stock over a specified period. The closing price refers to the final trading price of a stock during a standard trading session. It is a critical indicator of a stock's daily performance and is often used in technical analysis to identify trends and patterns.
- Benchmark Returns: Benchmark returns represent the performance of a standard or benchmark index, such as the S&P 500, against which the performance of a stock or portfolio is measured. Comparing stock returns to benchmark returns helps in assessing the relative performance and identifying whether a stock is outperforming or underperforming the market.
- Stock Returns: Stock returns indicate the profit or loss generated by a stock over a specific period, expressed as a percentage of the initial investment. It includes capital gains and dividends. Analyzing stock returns is essential for evaluating the investment's profitability and risk.
- EBITDA (Earnings Before Interest, Taxes, Depreciation, and Amortization): EBITDA serves as an indicator of a company's total financial performance and is often considered a substitute for net income. It provides insights into the company's operational efficiency by excluding non-operational expenses. EBITDA is often used in valuation and comparison of companies within the same industry.

- EBIT (Earnings Before Interest and Taxes): EBIT is a metric that assesses a company's profitability by accounting for all expenses except for interest and income tax expenses. It is used to analyze the profitability of core operations without the impact of taxation and financial structure. EBIT is crucial for understanding the company's operational performance.

- Net Income: Net income represents the complete earnings of a business once all costs, such as taxes and interest, have been subtracted from total revenue. It serves as an essential measure of a business's profitability and overall financial well-being. Net income is often used to calculate earnings per share (EPS) and to assess the company's ability to generate profit.

- Net Debt: Net debt refers to a company's overall debt subtracted by its cash and cash equivalents. This metric offers a more transparent view of a company's financial leverage and its capacity to fulfill debt responsibilities. Evaluating net debt is crucial for assessing financial risk and stability.

- Total Debt: Total debt encompasses all of a company's short-term and long-term debt responsibilities. It is a important factor in assessing the company's financial leverage and risk. High levels of debt can indicate potential financial distress, while manageable debt levels suggest financial stability.

- Total Assets: Total assets represent the sum of all assets owned by a company, including cash, investments, property, and equipment. It is a measure of the company's total resources and is used in various financial ratios to assess the company's financial health and operational efficiency.

- Payables: Payables refer to the amounts a company owes to its suppliers and creditors for goods and services received. It is a component of current liabilities and is used to assess the company's short-term financial obligations and liquidity.

- Change in Payables: This metric indicates the variation in the amount of payables over a specific period. An increase in payables may suggest that the company is delaying payments to conserve cash, while a decrease may indicate improved cash flow management.

- Cash Flow: Free cash flow refers to the cash a company produces once it has accounted for its capital expenditures. It is a key indicator of a company's financial flexibility and its ability to generate cash to fund operations, pay dividends, and pursue growth opportunities.

- Current Analyst Price Target: This is the average price target set by financial analysts for a stock, based on their research and projections. It provides an estimate of the stock's future price and is used by investors to gauge potential upside or downside.

- Max Analyst Price Target: The maximum price target is the highest price estimate given by analysts for a stock. It represents the most optimistic view of the stock's potential future price.

- Min Analyst Price Target: The minimum price target is the lowest price estimate given by analysts for a stock. It represents the most conservative view of the stock's potential future price.

- Analyst Earnings Estimate: This is the projected earnings per share (EPS) for a company, as estimated by financial analysts. It is used to gauge the company's expected profitability and to compare actual earnings against expectations.

- Current Analyst target Price: In a given time period, usually 12 months, this is the average price or consensus that analysts believe a stock will reach. It is determined by taking the average of all the price targets provided by various analysts who follow the stock. This number gives a broad indication of the anticipated trading position of the stock based on the opinions of analysts.

- Maximum Analyst Target Price: Out of all the analysts polled, this is the most optimistic price forecast for the stock. It displays the maximum possible gain that at least one analyst thinks the stock is capable of.

- Minimum Analyst Target Price: Out of all the analysts polled, this is the most negative price forecast for the investment. It displays the possible decline in value that at least one analyst thinks the stock might encounter. The range of opinions and uncertainty surrounding the stock's future performance can be gauged by comparing the maximum and minimum price targets. More variation indicates more analyst disagreement. One way to estimate the possible upside or downside is to compare the current price target with the stock's current market price.

- Normalized EBITDA: Adjusted EBITDA, or normalized EBITDA, goes one step further by eliminating the influence of irregular or non-recurring items that can skew EBITDA. Once-time gains or losses from asset sales; restructuring charges; litigation settlements; impairment charges; and unusual items are a few examples of these items. Normalized EBITDA eliminates these items in order to reflect what a company's earnings would have been under typical operating circumstances.

- Reconciled Depreciation: The process of reconciling depreciation in financial statements basically entails making sure that the expense of depreciation recorded in the income statement and the change in the balance sheet's accumulated depreciation are correct and in line with the business's accounting guidelines. This is the amount of money that is set aside for expenses over the course of an asset's useful life. According to the income statement, it lowers net income.

- Reconciled Cost Of Revenue: The process of making sure that the costs directly related to producing goods or services are appropriately documented and equal the corresponding revenue generated is known as reconciling cost of revenue, or COGS. Maintaining accurate financial records and figuring out a company's profitability depend on this process.

- Net Interest Income: One important performance indicator for financial institutions, especially banks, is net interest income (NII). It stands for the difference between the money received from assets that bear interest and the costs incurred by liabilities that bear interest. It is, to put it simply, the profit a bank makes from lending and borrowing.

- Interest Expense: The cost a business bears when borrowing money is known as interest expense. It stands for the interest paid on a variety of debts, including bonds, credit lines, and loans. It significantly affects a company's profitability and is a crucial line item on its income statement.

- Normalized Income: A company's reported net income is adjusted to eliminate the impact of unusual or non-recurring items using a financial metric called normalized income, sometimes referred to as normalized earnings or adjusted earnings. Comparing performance over time or with other businesses is made simpler by this, which also gives a clearer picture of the company's core, continuous profitability.

- Diluted Average Shares: A key indicator in determining diluted earnings per share (EPS) is the diluted average number of outstanding shares. In the event that all possible sources of stock dilution were used up, it represents the total number of shares that would remain outstanding.

- Net Income Continuous Operations: In financial analysis, net income from continuing operations is a critical metric that separates the profitability of a business's main, continuous operations. A clearer picture of the company's sustainable earnings power is provided by removing the impact of discontinued operations.

- Net Income Discontinuous Operations: The profit or loss made by a portion of a company that has been sold, spun off, or is currently being disposed of is known as net income from discontinued operations. To differentiate it from the business's continuing operations, it is shown separately on the income statement.
- Total Other Finance Cost: The total of all costs related to raising money for a project or business agreement, minus the principal interest payments on loans, is known as the total other finance cost.
- Operating Income: Operating Income, sometimes referred to as Operating Profit or Earnings Before Interest and Taxes (EBIT), is a crucial financial indicator that shows how profitable a company's core business operations are.
- Operating Expense: Operating expenses (OpEx) are the continuous costs that a business incurs in order to make money on a daily basis. These costs are necessary for the operation of the company and are separate from the costs directly related to the production of goods or services (COGS, or cost of goods sold).
- Common Stock Equity: A company's common stock equity is the ownership stake held by common stockholders. It represents their remaining claim to the company's profits and assets following the payment of all other debts and dividends on preferred stock.
- Total Capitalization: The sum of a business's debt and equity is known as total capitalization. It offers a thorough understanding of the business's overall funding structure and financial well-being.
- Treasury Shares: Shares that a business has previously issued to the public but has since bought back from shareholders are referred to as treasury shares, or reacquired stock. The business itself then keeps these shares in its treasury.
- Ordinary Shares: Common stock, sometimes referred to as ordinary shares, is the most prevalent kind of stock that a business issues. They grant shareholders rights and serve as a symbol of ownership in the business.
- Invested Capital: The total sum of funds that debt and equity holders have invested in a business. It stands for the total amount of money available to run and expand the business.
- Working Capital: The sum of money that a business has on hand to finance its ongoing operations. A business's short-term liquidity is represented by it.
- Current Liabilities: Current Liabilities are the short-term financial commitments of a business that are due within a year or within a typical business cycle, whichever comes first.
- Accounts Payable: The money that a business owes suppliers or vendors for goods or services that have been received but not yet paid for is known as accounts payable. These are temporary commitments that normally have a 30- to 60-day payback period. Since accounts payable have a direct impact on cash flow and liquidity, they are an essential part of a business's financial health.
- Total Assets: A company's total assets are the culmination of all of its resources, including both short-term and long-term assets. This number offers a thorough assessment of the business's overall size and financial stability.

- Total Non Current Assets: A company's long-term investments, including intangible assets (patents, trademarks) and property, plant, and equipment (land, buildings, and machinery), are represented by its total non-current assets. These assets, which are not anticipated to be readily turned into cash within a year, are essential to the long-term operations of the business.

- Investments in Financial Assets: A company's holdings of securities, such as stocks, bonds, and other financial instruments, are referred to as investments in financial assets. The goal of these investments is to produce income in the form of interest, dividends, or capital gains. Businesses with substantial financial asset investments might aim to produce sizable returns from their holdings. This might suggest that they are concentrating on financial market activity in an effort to diversify their revenue sources or profit from market trends.

- Net Property Plant Equipment: Net Property, Plant, and Equipment (Net PPE) on a company's financial statement represents the total value of a company's long-term physical assets, such as land, buildings, machinery, and equipment, after deducting accumulated depreciation. Depreciation reflects the gradual wear and tear of these assets over time. Net PPE provides valuable insights into a company's long-term investment in its operations and its capacity for future growth.

- Gross Property Plant Equipment: Gross Property, Plant, and Equipment (Gross PPE) on a company's financial statement represents the original cost of all the company's long-term physical assets, such as land, buildings, machinery, and equipment. This figure reflects the initial investment made in these assets without accounting for any depreciation or wear and tear.

- Inventory: Inventory is a company's stock of products that are on the market. This comprises finished goods, work-in-progress products, and raw materials. Due to its direct correlation with revenue generation, inventory is an essential current asset for businesses.

- Account Receivables: Accounts Receivable represents the money owed to a company by its customers for goods or services that have already been delivered but not yet paid for. It's essentially an IOU from the customer to the company. Accounts Receivable is a crucial current asset, as it represents a significant portion of a company's expected future cash inflows.

- Cash and Cash Equivalants: Cash, Cash Equivalents, and Short-Term Investments represent a company's most liquid assets. Cash includes readily available funds like currency, bank deposits, and demand deposits. Cash equivalents are highly liquid investments with short maturities (usually within 90 days), easily convertible into cash with minimal risk of value fluctuation. Short-term investments include more diverse securities with maturities exceeding 90 days, offering potential for higher returns but with slightly greater risk compared to cash equivalents. This category reflects the company's immediate financial flexibility and its ability to meet short-term obligations.

These metrics are crucial for conducting both fundamental and technical analysis of the company's stock performance.

The primary source for extracting stock information is Yahoo Finance, a widely recognized platform that provides comprehensive financial data. The extraction process involves scraping the Yahoo Finance website to gather the required financial metrics. This includes parsing the HTML content of the web pages to locate and extract specific data points such as EPS, EBITDA, the 50-day moving average, and the current share price. The scraping process is automated using Python, which facilitates efficient and accurate data extraction.

In addition to financial metrics, the application also retrieves related news articles for the specified company. These news articles are sourced from Yahoo News, which offers a vast repository of financial news and updates. The extraction process involves scraping the Yahoo News website to collect articles that mention the company. This is achieved by querying the website's search functionality and parsing the resulting web pages to extract the headlines, publication dates, and full text of the articles.

Once the news articles are collected, the next step is to perform sentiment analysis on the content. Sentiment analysis is a crucial component of methodology as it provides insights into the market sentiment surrounding the company. The sentiment scores are generated using advanced Natural Language Processing (NLP) techniques. Specifically, pre-trained sentiment analysis models DistilBERT (Bidirectional Encoder Representations from Transformers) [59] is employed to analyze the sentiment of each news article. These models assign sentiment scores to the articles, indicating whether the sentiment is positive, negative, or neutral.

The scraped news articles and the corresponding sentiment scores, along with the extracted financial data, form the core dataset for the study. This dataset is then fed into the LLM model to answer the user's query. The LLM model, leverages the RAG framework to dynamically retrieve and integrate external data during the generation process. This ensures that the model's responses are not only accurate but also contextually relevant and up-to-date.

Furthermore, the dataset is continuously updated to reflect the latest market conditions and news. This is achieved by scheduling regular scraping intervals, ensuring that the financial metrics and news articles are always current. The dynamic nature of the RAG framework allows the LLM model to incorporate these updates in real-time, providing users with the most relevant and timely stock analysis.

In summary, the data collection and preprocessing steps in this research are designed to create a comprehensive and reliable dataset for enhancing stock analysis through RAG pipelines. By integrating fundamental and technical evaluation with the trustworthiness of LLMs, methodology provides complex insights into market trends and potential investment opportunities. This approach democratizes access to sophisticated financial analysis, making it accessible and actionable for a broader audience, ranging from novice investors to seasoned analysts.

4.2 Natural Language Processing

Natural Language Processing (NLP) is a branch of artificial intelligence dedicated to the communication between computers and humans using natural language [48]. The aim of NLP is to allow computers to comprehend, interpret, and produce human language in a manner that is both meaningful and beneficial. NLP encompasses a variety of tasks including question answering, machine translation, sentiment analysis, and text classification. In the context of stock analysis, NLP techniques are employed to process and analyze vast amounts of textual data such as financial reports, news articles, and social media posts to extract valuable insights and trends.

NLP techniques can be broadly categorized into rule-based approaches, statistical methods, and machine learning models. Rule-based approaches rely on predefined linguistic rules, while statistical methods use probabilistic models to infer patterns from data. In a variety of NLP tasks, machine learning models especially those built on deep learning have demonstrated impressive performance. These models learn to represent and process language data through training on large corpora, enabling them to capture complex linguistic patterns and nuances.

- Semantic Analysis: A crucial aspect of natural language processing (NLP) is semantic analysis, which aims to comprehend the meaning of words, phrases, and sentences in a particular context [50]. It involves the extraction of meaning from text and the representation of this meaning in a structured form that machines can process. Semantic analysis can be divided into two main tasks: lexical semantics [30] and compositional semantics [45]. Lexical semantics studies the relationships between words and their meanings.It includes exercises like semantic similarity, which gauges how similar two words or phrases are, and word sense disambiguation, which seeks to ascertain a word's true meaning based on its context. Compositional semantics, on the other hand, focuses on understanding how the meanings of individual words combine to form the meaning of larger units such as phrases and sentences. This involves tasks such as semantic role labeling, which identifies the roles played by different entities in a sentence, and semantic parsing, which maps sentences to their corresponding meaning representations. In the context of stock analysis, semantic analysis can be used to extract relevant information from financial news and reports, such as identifying key events, entities, and relationships that may impact stock prices.

- Pragmatic Analysis [65]: Pragmatic analysis is concerned with understanding the intended meaning of a text by considering the context in which it is used. It goes beyond the literal meaning of words and sentences to interpret the speaker's or writer's intentions, beliefs, and goals. Pragmatic analysis involves tasks such as reference resolution, which identifies the entities referred to by pronouns and other referring expressions, and speech act recognition, which determines the communicative function of a sentence (e.g., statement, question, command). In the context of stock analysis, pragmatic analysis can help in understanding the implications of financial news and reports. For example, it can be used to determine whether a news article is expressing a positive or negative sentiment towards a particular stock, or to identify the underlying assumptions and motivations behind an analyst's report.

- Discourse Integration [44]: Discourse integration involves understanding how individual sentences and phrases are connected to form a coherent and meaningful text. It focuses on the relationships between different parts of a text and how they contribute to the overall meaning. Discourse integration involves tasks such as anaphora resolution, which identifies the antecedents of pronouns and other referring expressions, and discourse parsing, which identifies the structure and organization of a text. In the context of stock analysis, discourse integration can help in understanding the overall narrative and context of financial news and reports. For example, it can be used to identify the main topics and themes discussed in a news article, and to understand how different pieces of information are related to each other.

- Syntactic Analysis [58]: Analyzing a sentence's grammatical structure to identify its syntactic constituents and their relationships is known as syntactic analysis, or parsing. It includes activities like part-of-speech tagging, which allocates grammatical classifications (e.g. G. dependency parsing, which determines the syntactic dependencies between words in a sentence, and noun, verb, and adjective) to words. Syntactic analysis is a fundamental step in NLP as it provides the structural foundation for higher-level tasks such as semantic and pragmatic analysis. In the context of stock analysis, syntactic analysis can be used to extract structured information from unstructured text, such as identifying the subject, object, and predicate of a sentence, and to identify key entities and events mentioned in financial news and reports.

- Parsing [31]: Parsing is the process of analyzing the syntactic structure of a sentence to determine its grammatical constituents and their relationships. It involves constructing a parse tree or a dependency graph that represents the hierarchical structure of a sentence. Parsing can be divided into two main types: constituency parsing and dependency parsing. Constituency parsing involves breaking down a sentence into its constituent parts, such as noun phrases, verb phrases, and prepositional phrases, and representing these parts in a hierarchical tree structure. Dependency parsing, on the other hand, focuses on identifying the syntactic dependencies between words in a sentence and representing these dependencies in a graph structure. In the context of stock analysis, parsing can be used to extract structured information from unstructured text, such as identifying the main entities and events mentioned in financial news and reports, and to understand the relationships between different pieces of information.

In summary, Natural Language Processing (NLP) encompasses a wide range of tasks and techniques that enable machines to understand, interpret, and generate human language. Key components of NLP, such as semantic analysis, pragmatic analysis, discourse integration, syntactic analysis, and parsing, play a crucial role in the processing and analyzing of textual data from various sources. In the context of enhancing stock analysis through RAG pipelines, these NLP techniques can be used to extract relevant information from financial news and reports, understand the implications of this information, and provide actionable insights for investors. By leveraging the capabilities of NLP and LLMs, robust and flexible systems can be developed for stock analysis that can process raw news content into a consistent format, regardless of its source or pre-existing structure, and provide deeper and more accurate insights into market trends and potential investment opportunities.

The Transformer architecture, introduced by Vaswani et al. [71] has revolutionized the field of Natural Language Processing (NLP). This architecture is designed to handle sequential data and has proven to be highly effective in understanding and generating human language. The Transformer model's core innovation lies in its use of self-attention mechanisms, which allow it to weigh the importance of different words in a sentence relative to each other, thereby capturing context more effectively than previous models like RNNs and LSTMs.

A multilayer perceptron, transformer blocks, positional encoding, tokenization, embedding, and attention mechanisms are some of the essential elements that make up the Transformer architecture [52]. Tokenization, which turns words into unique tokens, is the first step in the process. An embedding layer is then used to convert these tokens into numerical vectors, which represent them in vector form. By adding the order to the words, positional encoding is used to preserve the text's word sequence. The transformer block, which predicts the next word, is the central component of the model. An attention mechanism that adds context to the text and a feedforward neural network that helps with word guessing are the two primary components of this block. In order to make it easier to choose the following word in the sequence, a Softmax layer is used to convert the scores into probabilities. These step repetitions give the text coherence, and the large number of parameters allows models to capture a broader context of the text.

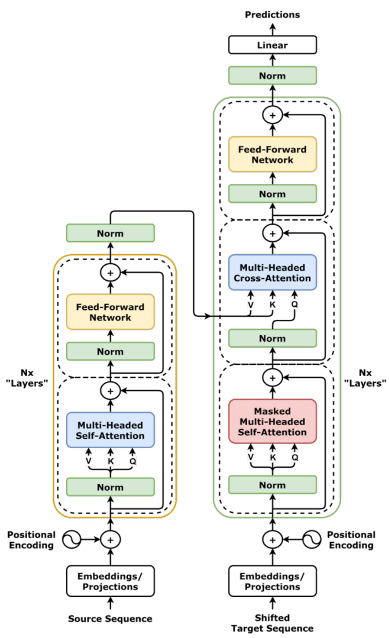


Figure 1: Transformer Architecture [71]

4.2.1 Self attention

According to [71], self-attention is a key component of Transformer models that helps with understanding the meaning of words in sentences. Imagine a word like ”bank” that, depending on the words around it, has multiple meanings. By taking into account the relationships between words, the self-attention mechanism enables the model to differentiate between these meanings. The model uses vectors to represent words and decides how much each word should emphasize the others. In the sentences ”The bank of the river” and ”Money in the bank,” for example, the word ”bank” denotes distinct ideas. The model matches the words ”bank” and ”river” or ”money” according to context. Through a process known as Multi-Head Attention, it then creates new vectors that capture these relationships while accounting for a number of factors. Essentially, the model can perceive words in their particular context thanks to self-attention, which is similar to how people concentrate on conversational cues. The efficacy of the transformers in text generation and comprehension has been greatly enhanced by this.

The utilization of queries (Q), keys (K), and values (V), which are obtained from the input data and represented as three vectors, is what defines self-attention. Through a linear transformation, each element in the input sequence is transformed into these three vectors. The query is then multiplied by all keys by the self-attention mechanism to determine the attention scores. The outcome is a number that represents how much attention or focus each component of the sequence should receive in relation to the others. A probability distribution is produced by normalizing these values using a softmax function to make sure they add up to one. A sum of the value vectors, each weighted by the softmax normalized attention scores, is the final output of the self-attention layer. With the weights indicating the degree of attention paid to each input element, this procedure allows each output element of the self-attention layer to be a composition of the inputs. The self-attention mechanism is particularly effective for tasks that require an understanding of context and sequence because it can assign different weights to inputs, enabling LLMs to identify complex relationships in the data, such as long-range dependencies.

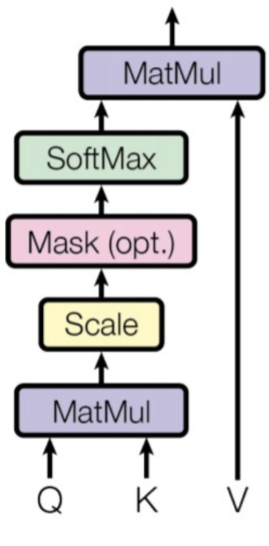


Figure 2: Self Attention mechanism [71]

4.2.2 Multi Headed Attention

Multi-Headed Attention [10] is an extension of the self-attention mechanism that allows the model to focus on different parts of the input sequence simultaneously. Instead of computing a single set of attention scores, the model computes multiple sets, each with its own set of queries, keys, and values. These multiple sets of attention scores are then concatenated and linearly transformed to produce the final output. This process enables the model to capture different aspects of the input sequence, such as syntactic and semantic relationships, making it more robust and effective in understanding complex language structures.

In the context of financial analysis, Multi-Headed Attention can be particularly useful for capturing the nuanced relationships between different financial indicators and textual data. For example, when analyzing a company's annual report, the model can simultaneously focus on different sections of the report, such as the Management Discussion and Analysis (MD&A) and the financial statements, to gain a comprehensive understanding of the company's performance and potential risks.

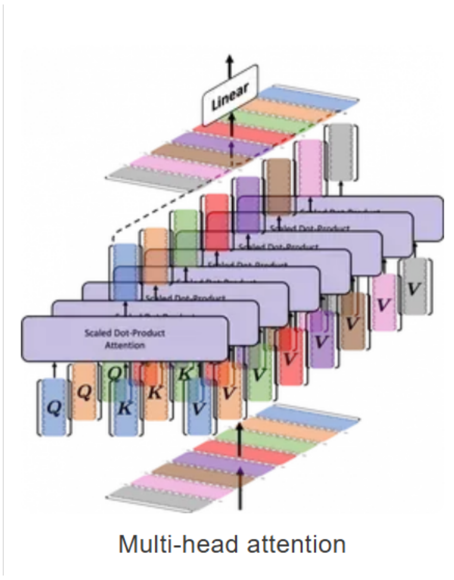


Figure 3: Multi headed self attention mechanism [16]

4.2.3 Encoder

The encoder is a crucial component of the Transformer architecture, responsible for processing the input sequence and generating a set of encoded representations. The encoder consists of multiple layers, each containing a self-attention mechanism and a feedforward neural network. The self-attention mechanism allows the encoder to weigh the importance of different words in the input sequence relative to each other, while the feedforward neural network helps to capture more complex patterns and relationships.

In each layer of the encoder, the input sequence is first passed through the self-attention mechanism, which computes the attention scores and generates a weighted sum of the value vectors. This output is then passed through a feedforward neural network, which applies a series of linear transformations and non linear activations to capture more complex patterns. The output of each layer is then passed to the next layer, allowing the model to build increasingly sophisticated representations of the input sequence.

4.2.4 Decoder

The Decoder is another essential component of the Transformer architecture, responsible for generating the output sequence based on the encoded representations produced by the Encoder. Like the Encoder, the Decoder consists of multiple layers, each containing a self-attention mechanism, an encoder-decoder attention mechanism, and a feed-forward neural network.

The self-attention mechanism in the decoder allows the model to focus on different parts of the output sequence generated so far, while the encoder-decoder attention mechanism enables the model to incorporate information

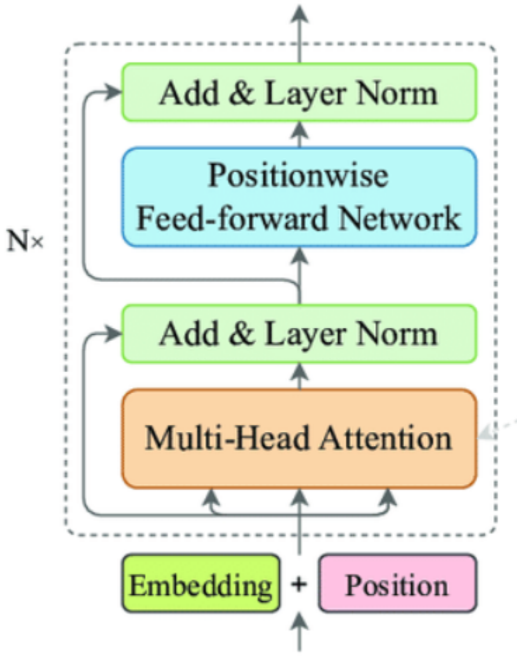


Figure 4: Encoder Block in Transformer [6]

from the encoded representations produced by the encoder. The feed-forward neural network helps to capture more complex patterns and relationships in the data.

In each layer of the decoder, the input sequence is first passed through the self-attention mechanism, which computes the attention scores and generates a weighted sum of the value vectors. This output is then passed through the encoder-decoder attention mechanism, which incorporates information from the encoded representations produced by the encoder. Finally, the output is passed through a feedforward neural network, which applies a series of linear transformations and non-linear activations to capture more complex patterns.

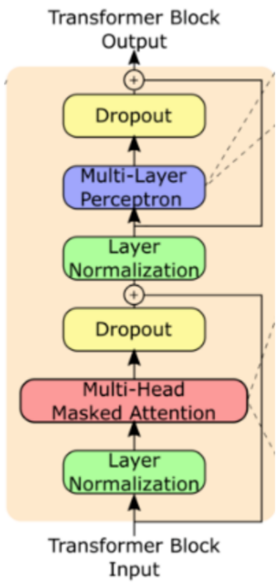


Figure 5: Decoder Block in Transformer [78]

4.2.5 Masked Self Attention

Masked self-attention [36] is a crucial mechanism in transformer models, particularly in the context of pre-training encoder models. This technique is primarily based on masked-language modeling, where a training text sequence is prepared by randomly masking some tokens. These masked tokens are replaced with a special token, often denoted as x_{mask}, which serves as a placeholder without concrete meaning. The objective of masked self-attention is to predict these masked tokens by maximizing the likelihood of the masked tokens given the surrounding context.

The model actually turns the input sequence into three vectors: values (V), keys (K), and queries (Q). Each element in the sequence should have a certain amount of focus or attention in relation to all other elements, which is determined by the self-attention mechanism using the . product of the query with all keys. After that, a softmax function is used to normalize these scores and create a probability distribution. The weights are the softmax normalized attention scores, and the weighted sum of the value vectors is the final result.

Information leakage from subsequent tokens is effectively prevented by the masked self-attention mechanism, which makes sure the model can only focus on the positions before the masked token. This makes it possible for the model to predict the masked token in both left and right contexts, improving its comprehension of the text's context and sequence. The model becomes incredibly powerful for tasks requiring a profound comprehension of context and sequence as a result of iterating through this process and learning to capture intricate relationships and dependencies within the data.

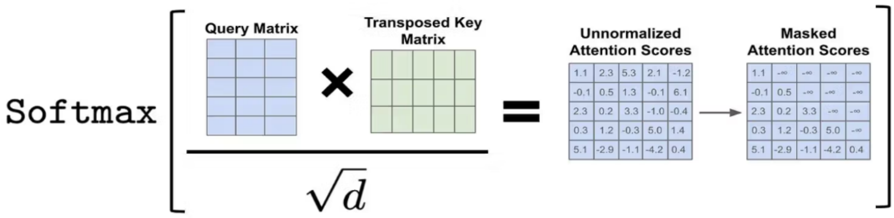


Figure 6: Masked Self Attention in Transformer [57]

4.3 GPT - 4o vs Gemini 1.5 Flash

In my thesis, I will be comparing two prominent LLMs, namely GPT-4o and Gemini 1.5 Flash in the context of financial analysis. These models represent the cutting edge of natural language NLP technology and have shown significant potential in various applications, including financial statement analysis, market trend prediction, and investment decision making.

GPT-4o, an evolution of OpenAI's Generative Pre-trained Transformer series [70], builds upon the foundational principles of its predecessors with significant enhancements in both architecture and training methodologies. The model leverages a transformer-based architecture, characterized by its multilayered self-attention mechanisms that enable the processing of vast amounts of textual data. The architecture of GPT-4o includes several key components:

- Enhanced Attention Mechanisms: GPT-4o incorporates advanced attention mechanisms [39] that allow for more efficient handling of long-range dependencies in text. This is achieved through modifications in the self-attention layers, which improve the model's ability to focus on relevant parts of the input sequence, thereby enhancing contextual understanding.

- Increased Model Depth and Width: The model features a larger number of layers, wider layers and multimodality training compared to its predecessors [11]. This increase in depth and width allows GPT-4o to capture more complex patterns and relationships within the data, leading to improved performance across a variety of tasks.
- Optimized Training Regime: GPT-4o benefits from a more sophisticated training regime that includes techniques such as curriculum learning and dynamic data augmentation [24], [19]. These methods help the model to gradually learn from simpler tasks to more complex tasks, improving its generalization capabilities.
- Chain-of-Thought (CoT) Prompting [76]: A notable feature of GPT-4o is its use of CoT prompting, which guides the model through a step-by-step reasoning process [17]. This approach enhances the model's ability to generate coherent and logically consistent output, particularly in tasks that require complex reasoning, such as financial analysis.

Gemini 1.5 has shown decent results in financial statement analysis and market prediction tasks. It's high accuracy and ability to process large datasets make it a powerful tool for financial analysts. The model's performance in predicting earnings changes and its ability to generate detailed financial insights highlight its potential in the financial sector. The key features of Gemini 1.5 flash are listed as follows.

- Hybrid Attention Mechanisms: Gemini 1.5 Flash employs a hybrid architecture mechanism [63] that combines self-attention with cross-attention layers. This hybrid approach allows the model to effectively integrate information from multiple sources, improving its ability to generate accurate and contextually relevant responses.
- Sparse Mixture-of-Experts (MoE): The model utilizes sparse mixture-of-experts [69] approach, which enable it to selectively activate only a subset of neurons during inference. This technique reduces computational overhead and improves efficiency without compromising performance.
- Enhanced Embedding Techniques: The model features advanced embedding techniques that improve its ability to represent complex linguistic structures. These techniques include the use of contextual embeddings and positional encodings, which enhance the model's understanding of the relationships between different parts of the input sequence.

By highlighting the key features and potential of GPT-4o and Gemini 1.5 flash, this thesis aims to provide a comprehensive understanding of how these advanced LLMs can be leveraged for financial analysis and decision-making. The key differences and similarities are listed as follows.

- Attention Mechanisms: Both models employ advanced attention mechanisms, but GPT-4o focuses on enhanced self-attention, while Gemini 1.5 Flash uses a hybrid approach that combines self-attention and cross-attention. This difference in attention mechanisms reflects their distinct strategies for handling contextual information.
- Efficiency and Computational Overhead: Gemini 1.5 Flash's use of sparse activation patterns and parallel computation contributes to its efficiency, making it more computationally efficient compared to GPT-4o. This efficiency is particularly beneficial in scenarios where computational resources are limited.
- Training Methodologies: The optimized training regime of GPT-4o, including curriculum learning and dynamic data augmentation, contrasts with Gemini 1.5 Flash's layer-wise adaptive learning rates. Both approaches aim to improve model performance, but they do so through different training strategies.

In summary, GPT-4o and Gemini 1.5 Flash represent two advanced LLM architectures with distinct methodologies and design principles. While GPT-4o emphasizes enhanced self-attention and optimized training regimens, Gemini 1.5 Flash focuses on hybrid attention mechanisms and efficiency. Both models offer significant advancements in the field of LLMs, with unique strengths that make them suitable for a variety of applications, including financial analysis.

4.4 Retrieval-Augmented Generation (RAG)

In the realm of AI and NLP, the advent of LLMs has revolutionized the way humans interact with machines. These models, such as GPT-3 [49] and BERT [13], have demonstrated remarkable capabilities in understanding and generating human-like text. However, despite their impressive performance, traditional LLMs have several limitations, particularly when it comes to real-world applications. One innovative approach to address these limitations is RAG [43]. This method enhances the performance of LLMs by integrating an up=to-date information retrieval phase into the answer generation process, thereby improving accuracy and expanding the knowledge base. In this comprehensive overview, this section will delve into the intricacies of RAG, its components, benefits, and applications. Traditional LLMs have gained significant traction due to their ability to perform a wide range of tasks, from text summarization to question answering. However, they are not without drawbacks. Firstly, these models require extensive training on large datasets, which is both time-consuming and costly [37]. Secondly, they need additional training to adapt to new data, making them less flexible in dynamic environments. Thirdly, LLMs can sometimes produce information that is not true, a phenomenon known as hallucination [3]. These limitations highlight the need for a more robust and efficient approach to leverage the full potential of LLMs. Retrieval-Augmented Generation (RAG) is a novel approach designed to enhance the performance of LLMs by incorporating an information retrieval phase into the answer generation process. Unlike traditional LLMs that rely solely on pre-trained knowledge, RAG models can search for and incorporate external data in real time, thereby leveraging a broader body of knowledge. This integration of retrieval and generation phases allows RAG models to provide more accurate and contextually relevant answers.

RAG consists of two main phases: the retrieval phase and the generation phase.

- Retrieval Phase: In this phase, given a user's query, the RAG model first retrieves relevant data from a document database or knowledge base. This is typically done using information retrieval techniques such as K nearest neighbors or dense retrieval methods. The goal is to select documents that are highly relevant to the user's query.
- Generation Phase: Based on the selected documents, the LLM generates contextualized answers. During this process, the model attempts to leverage information from the retrieved documents to provide more accurate and detailed responses. This phase involves synthesizing the retrieved information with the model's pre-trained knowledge to generate a coherent and contextually appropriate answer.

One of the key benefits of RAG is its ability to improve the accuracy of language models. By utilizing a separate database that is not part of the LLMs training data, RAG can cross-reference and validate information, reducing the likelihood of generating false or misleading content. This approach is particularly valuable in fields such as finance, healthcare, and legal services, where the accuracy and reliability of information are paramount. Additionally, RAG's ability to incorporate external data in real-time allows it to stay current with new developments and trends, making it a powerful tool for dynamic and rapidly changing environments.

In conclusion, Retrieval-Augmented Generation (RAG) offers a robust solution to the limitations of traditional large language models by integrating an information retrieval phase into the answer generation process. This methodology enhances the accuracy and reliability of the model's outputs by grounding them in verifiable data. The benefits of RAG, including improved accuracy, expanded knowledge base, and real-time data integration, make it a valuable tool for various applications, particularly in fields where information precision is critical. The integration of RAG with frameworks like LlamaIndex further amplifies its potential, enabling the development of advanced AI systems capable of delivering accurate and contextually relevant responses.

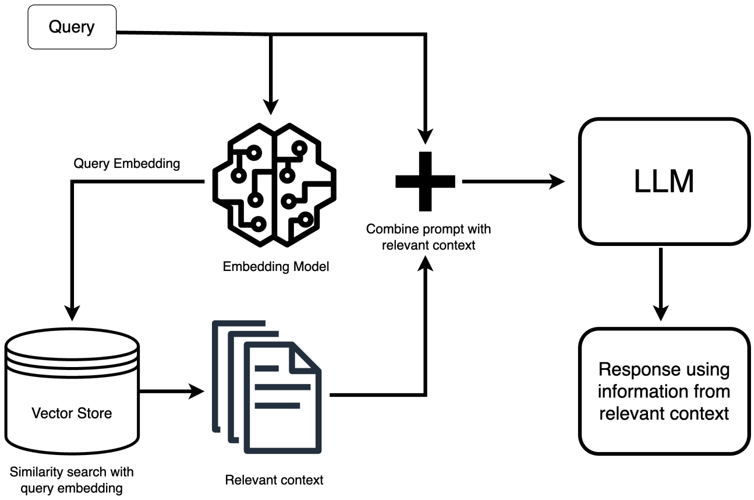


Figure 7: RAG Architecture [33]

4.5 Evaluation

In the realm of NLP, LLMs have demonstrated remarkable capabilities across various tasks. However, their propensity to generate plausible yet incorrect responses has raised concerns about their trustworthiness, especially in closed-book question-answering tasks. To address these challenges, the concept of trustworthiness scores has been introduced, particularly in the context of Retrieval-Augmented Generation (RAG) models. This paper aims to explain the trustworthiness score for RAG evaluation, including the underlying formulas and methodologies.

Bad Speculative Detector (BSDetector) [8] is a novel evaluation framework that assesses the trustworthiness of LLM responses. The BSDetector framework is based on the concept of self-reflection consistency, which evaluates the confidence of LLMs on the generated response using it's intrinsic parametric knowledge and observed consistency which evaluates the reliability of the generated response. The key component of TrustScore is the Self-reflection Certainty, which assesses the confidence of the LLM in it's generated response with its intrinsic knowledge. This evaluator functions independently, making it particularly useful in situations where external knowledge bases are not accessible or there is insufficient information available.

Self-Reflection Certainty is a critical component of the BSDetector framework, designed to enhance the reliability of outputs generated by Large Language Models (LLMs). This method enables LLMs to internally assess the accuracy of their responses through a structured evaluation process. The LLM is prompted with follow-up questions regarding its original answer, typically framed in a multiple-choice format: A) Correct, B) Incorrect, and C) I am not sure. Each response is assigned a numerical score—1.0 for correct, 0.0 for incorrect, and 0.5 for uncertainty. The Self-Reflection Certainty score is calculated as the average of the scores obtained from multiple rounds of questioning, allowing the model to reflect on its confidence level regarding the original answer. This intrinsic evaluation mechanism serves to identify potential flaws in the LLMs responses, even when the model consistently produces the same output across different prompts. By incorporating Self-Reflection Certainty, the BSD ETECTOR framework provides a nuanced understanding of an LLMs confidence in its answers, complementing extrinsic measures such as Observed Consistency. This dual approach enhances the overall trustworthiness of LLM output, making it particularly valuable in high-stakes applications where accuracy and reliability are essential. Ultimately, Self-Reflection Certainty contributes to more informed decision-making in the deployment of LLMs across various domains.

$$\mathrm{S} = \frac{1}{N}\sum_{i=1}^{N}\mathrm{Score}(Q_i)] \tag{4.1}$$

where:

- (S) = Self-Reflection Certainty score.
- (N) = Number of follow-up questions asked.
- (Score(Q_i) is score assigned based on the LLMs response to the (i^{th}) follow-up question.

Observed consistency is a fundamental aspect of the BSDectector framework, aimed at quantifying the reliability of answers generated by LLMs. It evaluates how consistently an LLM produces similar responses when queried multiple times with the same or slightly varied prompts. High observed consistency indicates that the model is likely confident in its answer, while significant variation suggests uncertainty or unreliability. To assess observed consistency, the LLM is prompted multiple times (typically five) to generate diverse outputs. The responses are then analyzed for contradictions by measuring semantic similarity between each generated answer and the original response. This is often accomplished using Natural Language Inference (NLI) models, which classify pairs of responses as entailment, neutral, or contradiction.

$$\mathrm{O} = \frac{1}{k}\sum_{i=1}^{k} o_i \tag{4.2}$$

where

- (O) = Observed Consistency score.
- (k) = Number of sampled outputs.
- (o_i) is Similarity score for each response (y_i) compared to the original answer (y).

The overall trustworthiness score integrates both Self Reflection Consistency and Observed Consistency to provide a holistic evaluation of the LLMs response.

The formula for calculating $Trust_{OV}$ is as follows

$$Trust_{OV} = \alpha \cdot \mathrm{O} + \beta \cdot \mathrm{S} \tag{4.3}$$

where: - (α) and (β) are weighting factors that determine the relative importance of Observed consistency and Self reflection consistency.

The trustworthiness score for RAG evaluation provides a robust and comprehensive metric for assessing the reliability of LLM responses. By integrating Behavioral Consistency and Factual Consistency, the trustworthiness score offers a nuanced evaluation that aligns closely with human judgments. This framework not only enhances the evaluation of LLMs but also provides valuable insights for improving their performance in real-world applications. Future work may explore the integration of trustworthiness scores with model editing

4.6 Implementation

This section details the implementation of the proposed methodology for enhancing stock analysis through Retrieval-Augmented Generation (RAG) pipelines, focusing on both fundamental and technical evaluation with the trustworthiness of LLMs. Approach integrates dynamic data retrieval systems with advanced NLP techniques to provide comprehensive and reliable stock analysis reports.

The implementation begins with the collection of relevant stock information and news articles. The application is designed to accept a yahoo finance company ticker and a specific query as input. Upon receiving these inputs, the system initiates the data retrieval process. Stock information such as earnings per share (EPS), EBITDA, 50-day moving average, and current share price is scraped from Yahoo Finance using web scraping techniques. Concurrently, related news articles about the company are also scraped from Yahoo Finance to capture the latest market sentiment and news context. To ensure the robustness and reliability of the data, the approach employs APIs and web scraping libraries that facilitate efficient and accurate data extraction. The scraped data undergo a preprocessing phase in which the data is normalized and converted into json structure for further analysis. This preprocessing step is crucial to maintain data quality and ensuring that the subsequent analysis is based on accurate and relevant information.

Once the data is collected and preprocessed, the next step involves performing sentiment analysis on the news articles. Transformer based DistilBERT model was used to analyze the sentiment of each news article to generate sentiment scores. These scores provide a quantitative measure of the market sentiment towards the company, which is essential to make informed investment decisions. The sentiment scores, along with the fundamental and technical information of the stock, are then fed into the LLM model. The LLM, enhanced with the RAG framework, dynamically integrates external data during the generation process to produce contextually relevant and accurate stock analysis reports. The RAG framework allows the LLM to retrieve and incorporate the most recent and pertinent information, thereby enhancing the trustworthiness and reliability of the generated reports.

The final step in the implementation involves generating comprehensive analytical reports based on the integrated data. The LLM synthesizes the fundamental and technical stock information with the sentiment analysis results to provide nuanced insights into the company's performance and market outlook. These reports are designed to be accessible and actionable, catering to both novice investors and seasoned analysts. By leveraging the capabilities of LLMs and the RAG framework, the methodology significantly improves the accuracy and relevance of stock analysis. The integration of real-time data retrieval and advanced NLP techniques ensures that the generated reports are timely, contextually relevant, and trustworthy. This approach democratizes access to sophisticated financial analysis, making it more accessible and actionable for a broader audience.

In conclusion, the implementation of the RAG pipeline for stock analysis represents a significant advancement in the field of financial analysis. By combining fundamental and technical evaluation with the trustworthiness of LLMs, this paper provide a powerful tool for investors to make informed decisions based on comprehensive and reliable stock analysis reports.

4.7 Software Design using AWS

The response journey of the chatbot that uses AWS architecture components to process company-specific stock information and related news articles can be described as follows. The application begins by accepting the company name and query as input from the user. This input is processed by an AWS Lambda function, which serves as the entry point for the application. The Lambda function triggers a series of AWS services to gather the required data. Firstly, it invokes an AWS API Gateway to interact with the Yahoo Finance API, fetching stock

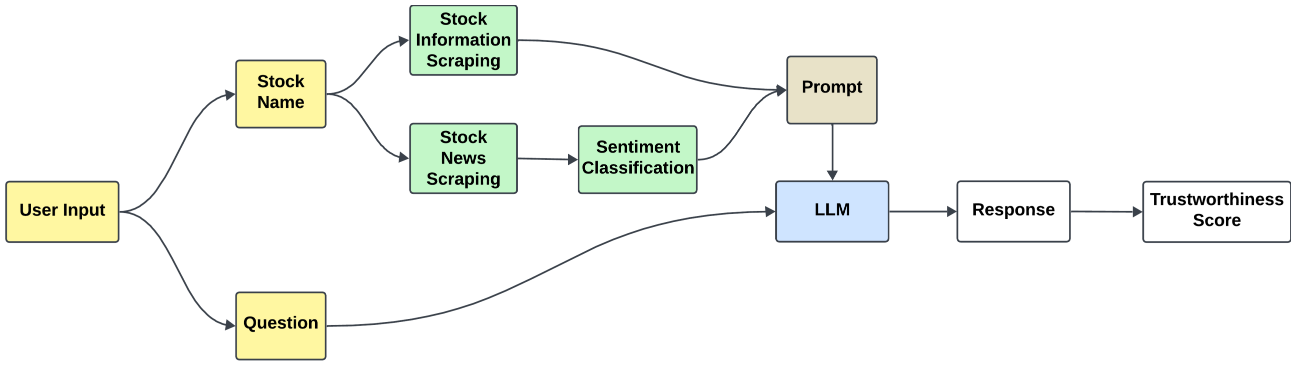


Figure 8: Proposed Pipeline

information such as earnings per share, EBITDA, 50-day moving average, and current share price. Concurrently, another API Gateway call is made to scrape related news articles from Yahoo News. These API calls are managed efficiently using AWS Step Functions, which orchestrate the sequence of tasks and handle any retries or errors.

Once the stock information and news articles are retrieved, The sentiment analysis is conducted using an NLP service that extracts sentiment scores from the news articles These sentiment scores, along with the stock information, are then fed into a pre-trained LLMs, which is responsible for generating the final response.

The LLM model processes the input data and generates a comprehensive response which is then sent back to the Lambda function. The Lambda function formats the response and sends it to the user through the API Gateway. Throughout this process, AWS CloudWatch monitors the performance and logs any issues, ensuring the system operates smoothly. This architecture leverages the scalability, reliability, and efficiency of AWS services to provide accurate and timely responses to user queries about company stock information and related news sentiments.

The described system design involves a comprehensive architecture that uses various AWS services to handle requests, process data, and provide responses to users. This architecture is designed to be robust, scalable, and efficient, ensuring that user queries are processed quickly and accurately. Below is a detailed explanation of the system design in 5-6 paragraphs.

The first step in a user request's journey is DNS resolution via AWS Route 53. The Domain Name System (DNS) web service Route 53 is scalable and highly available, and it is intended to direct end-user requests to the relevant AWS resources. AWS CloudFront, a Content Delivery Network (CDN) service, receives the request after the DNS resolution is finished. Through global content caching at edge locations, CloudFront facilitates the delivery of content with low latency and high transfer speeds. This stage lessens the strain on the origin servers by guaranteeing that the request is processed promptly and effectively. After passing through the CDN, the request is sent to the AWS Web Application Firewall (WAF). One security tool that helps shield web apps from frequent online threats and vulnerabilities is AWS WAF. It makes sure that only valid requests make it to the backend services by enabling the creation of custom rules that either permit or prohibit particular traffic patterns. AWS Network Load Balancer (NLB) receives the request after it has passed through the WAF. With extremely low latencies, the NLB can process millions of requests per second. Incoming application traffic is split up among several targets, including EC2 instances, in one or more Availability Zones.

The AWS Application Load Balancer (ALB) is the next step in the request flow. The ALB offers sophisticated routing features like host-based or path-based routing and functions at the application layer (Layer 7). Based on the request's content, it guarantees that the request is forwarded to the relevant backend service. The request is sent to the AWS API Gateway after it has passed the ALB. Applications can access information, business logic, or functionality from backend services through the API Gateway. It manages traffic, authorization and access control, monitoring, and API version management, among other tasks related to receiving and processing hundreds of thousands of concurrent API calls.

The request is routed from the API Gateway to AWS Lambda, a serverless compute service that automatically maintains the underlying compute resources and executes code in response to events. Lambda scales automatically from a few daily requests to thousands per second, running the code only when necessary. In addition to processing the request, the Lambda function communicates with the backend microservices, which comprise AWS EKS (Elastic Kubernetes Service), AWS ECS (Elastic Container Service), and other Lambda functions. A Large Language Model (LLM) is used to generate responses, analyze sentiment, and scrape stock information, among other tasks, all handled by these microservices.

Finally, the response generated by the backend microservices is sent back to the user through the same path, ensuring that the response is delivered efficiently and securely. The user query, response, and feedback are stored in a PostgreSQL database hosted on AWS RDS (Relational Database Service). This database service provides a scalable and highly available database solution, ensuring that all data is stored reliably and can be accessed quickly when needed.

The microservice architecture includes several key services, such as Scraper Service, Sentiment Analysis Service, LLM Service, and Database Service. The Scraper Service is responsible for scraping stock information and related news articles from sources like Yahoo Finance and Yahoo News. The Sentiment Analysis Service analyzes the sentiment of the scraped news articles and generates sentiment scores, which are crucial for understanding the market sentiment. The LLM Service feeds the scraped news articles and stock information into the LLM model to generate responses to user queries. Finally, the Database Service stores all user queries, responses, and other relevant data in the PostgreSQL database, ensuring that the system can learn and improve over time based on user interactions.

In summary, this system design leverages a combination of AWS services to create a robust, scalable, and efficient architecture for handling user requests, processing data, and providing accurate responses. By utilizing services like Route 53, CloudFront, WAF, NLB, ALB, API Gateway, Lambda, ECS, EKS, and RDS, the system ensures high availability, security, and performance, making it well-suited for handling the complex task of stock price trend prediction using emotion analysis of financial headlines.

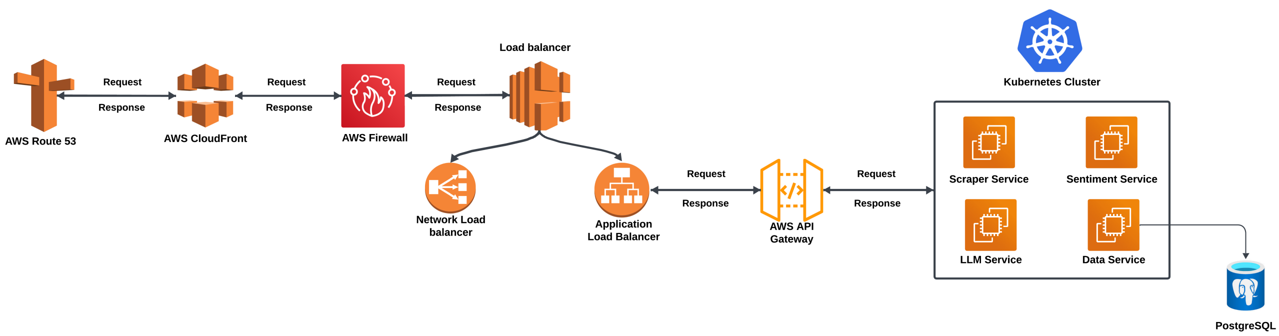


Figure 9: Scalable and Secured System design of the Chatbot

4.8 System Architecture

4.8.1 Event-Driven Architecture

Event-Driven Architecture (EDA) [1] is a design paradigm in which the flow of the program is determined by events such as user actions, sensor outputs, or messages from other programs or threads. In an EDA, components of the system communicate with each other through events, which are messages that signify that something has happened. This architecture is particularly useful for building scalable and resilient systems as it allows for the decoupling of components and asynchronous communication.

In the context of a web scraper service, an event-driven architecture can be highly effective. The scraper service can be designed as an event-driven microservice that reacts to specific requests for scraping data. This means that instead of continuously polling for new tasks, the scraper service listens for events (requests) and processes them as they arrive. This approach can lead to more efficient resource utilization and better scalability.

The architecture of the scraper service can be broken down into several key components.

- Request Queue: This component is responsible for queuing scraping requests. A message broker like Apache Kafka can be used to manage the queue. When a new scraping request is generated, it is placed in the request queue. This decouples the generation of requests from their processing, allowing the system to handle bursts of requests more gracefully.

- Worker Nodes: These are distributed workers that process the scraping requests in parallel. Each worker node listens to the request queue, picks up a request, and performs the scraping task. By distributing the workload across multiple worker nodes, the system can achieve high throughput and fault tolerance. If one worker node fails, others can continue processing requests.

- Data Validator: After the data is scraped, it is passed to the data validator. This component ensures that the scraped data adheres to the required formats and standards before it is stored. This step is crucial for maintaining data quality and consistency.

- Scaling and Error Handling: To handle high scraping loads, the system can use horizontal scaling with container orchestration tools like Kubernetes. This allows the system to dynamically adjust the number of worker nodes based on the current load. Additionally, the system should implement retry logic and error handling to deal with websites that use anti-scraping measures. This can include techniques like rotating proxies and respecting rate limits to avoid getting blocked.

- Communication Protocol: For efficient communication with other services that need the scraped data, the system can use gRPC, which provides high-performance, low-latency communication. For asynchronous communication with downstream services like the Stock Analyzer, a message broker can be used. This ensures that the data flows smoothly through the system without bottlenecks.

In summary, an event-driven architecture for a web scraper service involves using a request queue to manage scraping requests, distributed worker nodes to process the requests in parallel, a data validator to ensure data quality, and robust scaling and error handling mechanisms. By using efficient communication protocols like gRPC and message brokers, the system can achieve high performance and resilience. This architecture not only improves the efficiency and scalability of the web scraper service but also ensures that it can handle the dynamic and unpredictable nature of web scraping tasks.

4.8.2 Stateless Processing Service

A Stateless Processing Service [32] is a type of computing service architecture where each request from a client is treated as an independent transaction, without any dependency on previous requests. This means that the service does not retain session information or state between different requests. Stateless processing is a fundamental concept in distributed systems and cloud computing, offering several advantages in terms of scalability, reliability, and simplicity.

In a stateless processing service, each request contains all the information needed for the server to understand and process it [4]. This approach contrasts with stateful services, where the server maintains context or state information across multiple requests from the same client. Stateless services are inherently more scalable because they can handle a large number of requests independently [64]. Since there is no need to maintain session state, the server can easily distribute requests across multiple instances, making it easier to scale out horizontally. This scalability is particularly beneficial in cloud environments, where resources can be dynamically allocated based on demand.

Another significant advantage of stateless processing services is their reliability and fault tolerance. In a stateful system [74], if a server fails, the state information stored on that server may be lost, potentially disrupting the service. However, in a stateless system, any server can handle any request because there is no dependency on previous interactions. This means that if a server fails, another server can seamlessly take over without any loss of information or service continuity. This redundancy and fault tolerance are crucial for building robust and highly available systems.

Stateless processing services also simplify the development and maintenance of distributed systems. Since there is no need to manage session state, developers can focus on implementing the core functionality of the service without worrying about state management complexities. This simplicity reduces the likelihood of bugs and errors related to state handling, leading to more reliable and maintainable code. Additionally, stateless services are easier to test and debug because each request can be treated as an isolated unit of work. The key benifits of the stateless processing service is given below:

- Ease of Scaling: Stateless services can scale horizontally (by adding more instances) without requiring synchronization between instances.
- High Availability: Since no state is stored, instances can be replaced or restarted without affecting ongoing operations.
- Fault tolerance: Stateless services can recover quickly from failures because they don't depend on previous state.
- Resource Efficiency:Statelessness eliminates the need for in-memory storage of session data, reducing resource usage.

In summary, stateless processing services offer a scalable, reliable, and simple approach to building distributed systems. By treating each request as an independent transaction, these services can efficiently handle large volumes of requests, provide high availability, and simplify development and maintenance. This architecture is particularly well-suited for fetching or processing data, where dynamic resource allocation and fault tolerance are essential.

4.8.3 Model as a Service

Model as a Service (MaaS) [22] is an innovative paradigm in the field of AI and machine learning (ML) that allows users to access and utilize sophisticated predictive models over the internet without the need to develop, train, or maintain these models themselves. This service-oriented approach democratizes access to advanced analytics and AI capabilities, making it possible for businesses and individuals to leverage the power of machine learning without requiring deep technical expertise or significant infrastructure investments.

At its core, MaaS operates on a cloud-based infrastructure where pre-trained models are hosted and made available through APIs (Application Programming Interfaces). Users can send data to these APIs and receive predictions or insights in return. This model abstracts away the complexities involved in the development and deployment of machine learning models, such as data preprocessing, feature engineering, model selection, training, and tuning. By providing a ready-to-use solution, MaaS enables organizations to quickly integrate AI capabilities into their applications and workflows, thereby accelerating innovation and improving decision-making processes.

One of the key advantages of MaaS is its scalability and flexibility. Since the models are hosted in the cloud, they can handle varying loads and can be scaled up or down based on demand. This is particularly beneficial for businesses that experience fluctuating workloads or need to process large volumes of data intermittently. Additionally, MaaS providers often offer a range of models tailored to different use cases, such as image recognition, natural language processing, and predictive analytics, allowing users to select the most appropriate model for their specific needs. This flexibility ensures that businesses can adopt AI solutions that are aligned with their strategic objectives and operational requirements.

Furthermore, MaaS significantly reduces the barrier to entry for AI adoption. Traditional model development requires a team of data scientists, access to large datasets, and substantial computational resources, which can be cost-prohibitive for many organizations. MaaS, on the other hand, offers a cost-effective alternative by providing access to state-of-the-art models on a subscription or pay-per-use basis. This not only lowers the initial investment but also allows businesses to experiment with AI technologies and iterate on their solutions without incurring significant costs. As a result, even small and medium-sized enterprises can harness the power of AI to drive growth and competitiveness.

In conclusion, Model as a Service represents a transformative approach to AI and machine learning, enabling widespread access to advanced predictive models through a cloud-based delivery model. By simplifying the complexities of model development and offering scalable, flexible, and cost-effective solutions, MaaS empowers organizations of all sizes to integrate AI into their operations and unlock new opportunities for innovation and efficiency. As the demand for AI-driven insights continues to grow, MaaS is poised to play a crucial role in shaping the future of intelligent applications and data-driven decision-making.

4.8.4 Database as a Service

Database as a Service (DBaaS) [41] is a cloud service model that provides users with access to a database without the need for setting up physical hardware, installing software, or managing the database infrastructure. This service allows businesses and developers to focus on application development and other core activities while the service provider handles the database management tasks. DBaaS offers several advantages, including scalability, cost-efficiency, and ease of use, making it an attractive option for organizations of all sizes.

One of the primary benefits of DBaaS is its scalability. Traditional database management systems often require significant upfront investment in hardware and software, and scaling up can be a complex and costly process. With DBaaS, users can easily scale their database resources up or down based on their needs, without worrying about the underlying infrastructure. This flexibility is particularly beneficial for businesses with fluctuating workloads or those experiencing rapid growth. Additionally, DBaaS providers typically offer automated backups, updates, and maintenance, ensuring that the database is always up-to-date and secure.

Cost-efficiency is another significant advantage of DBaaS. By leveraging a cloud-based service, organizations can reduce their capital expenditures on hardware and software, as well as operational costs associated with database management. DBaaS operates on a subscription or pay-as-you-go model, allowing businesses to pay only for the resources they use. This model can lead to substantial cost savings, especially for small and medium-sized enterprises that may not have the budget for extensive IT infrastructure. Furthermore, the reduced need for in-house database management expertise can result in lower personnel costs.

Ease of use is a key feature of DBaaS that makes it appealing to a wide range of users. Setting up and managing a traditional database can be a complex and time-consuming process, requiring specialized knowledge and skills. DBaaS simplifies this process by providing a user-friendly interface and automated management tools. Users can quickly deploy and configure databases, perform backups, and monitor performance through a centralized dashboard. This ease of use enables developers to focus on building and optimizing their applications, rather than dealing with the intricacies of database management.

In conclusion, Database as a Service (DBaaS) offers a scalable, cost-efficient, and user-friendly solution for managing databases in the cloud. By offloading the responsibilities of database setup, maintenance, and scaling to the service provider, organizations can save time and resources, allowing them to concentrate on their core business activities. As the demand for cloud-based services continues to grow, DBaaS is likely to become an increasingly popular choice for businesses seeking to streamline their database management processes.

4.8.5 Scraper Service

The Event-Driven pattern is a versatile architectural approach that can be effectively applied to the design of a web scraper service. . In the event-driven mode, the scraper reacts to specific requests for scraping, which are typically triggered by certain events or user actions. This ensures that the scraper operates in real-time, providing immediate responses to data requests. The architecture of the scraper service can be broken down into several key components, each playing a crucial role in ensuring efficient and reliable data scraping. The first component is the Request Queue, which is responsible for managing the incoming scraping requests. By integrating a message broker like Apache Kafka, the system can queue these requests and ensure they are processed in an orderly manner. This not only helps in managing the load but also provides a buffer that can handle spikes in request volume without overwhelming the system.

Next, the Worker Nodes are the backbone of the scraping process. These distributed workers operate in parallel, each handling a portion of the scraping tasks. This parallel processing capability is essential for scaling the scraper service to handle large volumes of data efficiently. By distributing the workload across multiple nodes, the system can achieve higher throughput and faster data retrieval times. Additionally, the use of container orchestration tools like Kubernetes allows for horizontal scaling, enabling the system to dynamically adjust the number of worker nodes based on the current load.

Communication between the scraper service and other components of the system is facilitated through efficient protocols. gRPC is used for high-performance communication with other services that require the scraped data, ensuring low latency and high throughput. Additionally, a message broker is employed for asynchronous communication with downstream services like the Stock Analyzer. This decouples the scraping process from the data analysis, allowing each component to operate independently and efficiently.

In summary, the event driven pattern provides a flexible and scalable approach to designing a web scraper service. By leveraging components like the Request Queue, Worker Nodes, and employing best practices such as horizontal scaling, error handling, and rate limiting, the scraper service can efficiently handle high loads and ensure the quality of the scraped data. Effective communication protocols like gRPC and message brokers further enhance the system's performance and reliability, making it well-suited for applications in financial data collection and analysis.

4.8.6 Sentiment Analysis Service

The Sentiment Analyzer Service is designed to analyze the sentiment of input text using a fine-tuned language model. This service is built on a stateless processing pattern, meaning each request to the service is independent. This design choice ensures scalability and ease of maintenance, as it allows the service to handle multiple requests simultaneously without maintaining any session state between them.

The architecture of the Sentiment Analyzer Service comprises several key components. The model hosting component is responsible for deploying the sentiment analysis model. This can be achieved using frameworks such as TensorFlow Serving, TorchServe, or the Hugging Face Transformers API, which provide robust and scalable solutions for serving machine learning models. The Preprocessing Module is tasked with cleaning and tokenizing the input text before it is fed into the model. This step is crucial as it ensures that the text is in the correct format for the model to process, removing any noise or irrelevant information that could affect the accuracy of the sentiment analysis.

The NLP transformer is the core component that handles the sentiment analysis. It uses the pre-trained distilBERT model to analyze the input text and determine its sentiment. This component is optimized for performance to ensure that the sentiment analysis is conducted in real-time, providing quick and accurate results. To further enhance performance, a Caching Layer is implemented. This layer caches the results of frequently analyzed inputs, reducing the inference time for subsequent requests with the same input. This not only improves the efficiency of the service but also reduces the computational load on the Inference Engine.

For communication, the service uses REST protocols. REST is widely used and easy to implement, making it a suitable choice for the service. Although gRPC offers efficient binary serialization that reduces the overhead of data transmission and improves communication speed, REST has been selected for its simplicity and broad compatibility across various platforms. This approach ensures ease of integration while maintaining reliable performance for the service.

In summary, the Sentiment Analyzer Service is a robust and scalable solution for analyzing the sentiment of input text. Its architecture, comprising Model Hosting, Preprocessing Module, Inference Engine, and Caching Layer, ensures efficient and accurate sentiment analysis. The use of gRPC for communication further enhances its performance, making it an ideal choice for real-time applications.

4.8.7 LLM Service

The architecture of an LLM service is designed to efficiently process user queries, generate responses, and provide context-aware interactions. This service is typically implemented using a Model-as-a-Service pattern, which leverages pre-trained models to deliver inference capabilities. The service utilize APIs such as OpenAI or Vertex AI to access the necessary computational resources.

At the core of the LLM Service architecture are several key components. The Request Handler is responsible for accepting user queries, pre-processing the input to ensure it is in the correct format, and formatting the responses generated by the model. This component acts as the initial interface between the user and the service, ensuring that the input data is clean and structured appropriately for the subsequent processing stages.

To ensure the service operates efficiently and reliably, several best practices are implemented. Autoscaling is used to dynamically adjust the infrastructure based on query volume, ensuring that the service can handle varying levels of demand without performance degradation. Additionally, monitoring tools are used to track response times and accuracy metrics, helping to identify and address any bottlenecks or performance issues.

Communication with the LLM Service is typically facilitated through a REST API for simple query-response interactions. This allows for flexible integration with various applications and platforms, enabling seamless interaction with the LLM service.

In summary, the architecture of an LLM service is designed to provide efficient and context-aware natural language processing capabilities. By leveraging a Model-as-a-Service pattern, advanced inference techniques, and robust infrastructure management practices, the service can deliver high-quality responses and maintain a coherent conversational context, ensuring a superior user experience.

4.8.8 Database Service

The architecture of a Database Service, particularly in the context of a Database-as-a-Service (DBaaS) pattern, is designed to provide a centralized and efficient repository for storing and retrieving various types of data. This service is crucial for managing data such as scraped data, analyzed stock metrics, and user interactions, ensuring that the data is accessible, reliable, and secure

At the core of this architecture is the database type, which in this case is a relational database like PostgreSQL. Relational databases are well-suited for structured data, making them ideal for storing information such as prompts, LLM responses, and user ratings of those responses. The structured nature of SQL databases allows for complex queries and transactions, which are essential for maintaining the integrity and consistency of the data. PostgreSQL, in particular, is known for its robustness, scalability, and support for advanced features such as indexing, which optimizes queries for frequently accessed data, and replication, which ensures high availability by duplicating the database across multiple servers.

Best practices in managing the database service include implementing regular backups and disaster recovery plans to protect against data loss. Indexing is used to enhance the performance of queries, making data retrieval faster and more efficient. Replication is another critical practice, as it not only provides high availability but also improves the reliability of the service by ensuring that there is always a copy of the data available in case of server failure. These practices are essential for maintaining the performance and reliability of the database service, especially in environments where data integrity and availability are paramount.

Communication protocols play a vital role in the architecture of the database service. For structured queries and data fetching, gRPC or REST APIs are commonly used. These protocols facilitate efficient and secure communication between the database service and other components of the system. Direct SQL queries are used for internal services with trusted access, allowing for more direct and efficient data manipulation. Additionally, Kafka or other event streaming platforms can be employed to notify services about data updates in real-time, ensuring that all components of the system are synchronized and up-to-date.

In summary, the architecture of a Database Service in a DBaaS pattern involves using a relational database like PostgreSQL for structured data, adhering to best practices such as indexing, backups, and replication, and employing communication protocols like gRPC, REST, and Kafka to ensure efficient and reliable data management. This architecture not only supports the storage and retrieval of data but also ensures that the data is accessible, secure, and consistently available, making it a critical component of any data-driven application.

4.9 Advantages and Disadvantages of the Proposed Design

The LLM-based stock analyzer's microservice architecture, which consists of an API gateway, load balancers, queues (SQS), Lambda functions, a MaaS component (for the LLM), and a PostgreSQL database, has several benefits and drawbacks that should be examined.

- Independent Scalability: Depending on its unique load requirements, each microservice (Web Scraper, Sentiment, LLM, and Database) can be scaled separately. This makes cost optimization and effective resource use possible.

- Fault Tolerance and Resilience: A failure in one microservice does not always bring down the entire system because microservices are loosely coupled. By dividing up traffic among several instances, load balancers make sure that traffic is diverted to reliable instances in the event of an instance failure. By serving as a buffer between services, SQS makes sure that messages are not lost in the event that one of them is momentarily unavailable. As a result, the system is more resilient.

- Maintainability and Deployability: Codebases are smaller for each microservice, which facilitates comprehension, development, and maintenance. Independent Deployments: Services can be updated and deployed separately from other system components. Faster release cycles and lower risk are made possible by this.
- Improved Agility: Microservices' modular design enables quicker iteration and development. Teams can work on multiple services at once, which speeds up the development process as a whole.
- API Gateway Benefits: The API Gateway streamlines management, security, and monitoring by offering a single point of entry for all API requests. Requests are routed to the relevant backend services by the API Gateway, which also has the ability to modify requests as necessary.
- Security (Authentication/Authorization): Security policies like authorization and authentication can be centrally enforced by the API Gateway.
- Model as a Service: LLMs that would be challenging or impossible to run within Lambda functions can be used with a MaaS solution like Bedrock. The LLMs underlying infrastructure is not your responsibility. MaaS providers improve performance and scalability by optimizing their infrastructure to serve LLMs.
- Increased Complexity: It is intrinsically more difficult to manage a distributed system with several services than a monolithic application. Careful planning and execution are necessary to establish dependable inter-service communication. It can be more difficult to monitor and debug problems in a distributed system.
- Latency: The overall response time may be impacted by latency introduced by inter-service communication. This can be lessened, though, by improving service performance and utilizing asynchronous communication with queues.

5 Results

In the rapidly evolving landscape of financial markets, the integration of advanced technologies such as LLMs and RAG frameworks has opened new avenues for stock analysis. This study presents the development and implementation of a chatbot designed to perform a fundamental and technical analysis of stocks. The chatbot leverages the RAG framework to dynamically retrieve and process data, providing real-time, comprehensive stock analysis reports. By taking a company name and query as input, the chatbot scrapes relevant stock information from Yahoo Finance, including metrics such as earnings per share (EPS), EBITDA, 50-day moving average, and current share price. Additionally, it gathers related news articles and generates sentiment scores based on the content. These data points are then fed into LLM models, specifically the GPT-4o and Gemini 1.5 flash models, to generate informed responses. The chatbot not only offers buy, sell, or hold recommendations but also provides risk analysis, making it a versatile tool for both retail and institutional investors.

The LLM-based chatbot offers several key benefits that enhance the stock analysis process. Firstly, it improves accessibility to comprehensive stock information and analysis, enabling users to make informed investment decisions quickly. By consolidating data from various sources and providing real-time updates, the chatbot ensures that investors have the most current information at their fingertips. Secondly, the integration of sentiment analysis from news articles adds a layer of depth to the stock analysis, allowing users to gauge market sentiment and its potential impact on stock performance. This holistic approach to data analysis helps investors to better understand market trends and make more accurate predictions. Lastly, the chatbot's ability to provide personalized recommendations and risk analysis tailored to individual stocks makes it a versatile tool for both novice and experienced investors. The key impacts of the proposed work are listed below.

- Improved Accessibility of Information: The chatbot provides users with real-time, comprehensive stock analysis reports. This feature is particularly beneficial for retail investors who may not have the resources to conduct in-depth analysis independently. By simply entering a company name, users can access detailed financial metrics and sentiment analysis, enabling them to make more informed investment decisions.

- Data Integration: The chatbot consolidates data from various sources, including financial metrics from Yahoo Finance and sentiment scores from news articles. This integration offers a holistic view of a company's performance, allowing investors to analyze the stock market from a broader perspective rather than relying on fragmented information.

- Real-Time Updates: The chatbot generates reports that reflect the latest data, ensuring that investors are always making decisions based on the most current information. This capability is crucial in the fast-paced world of stock trading, where market conditions can change rapidly.

- User-Friendly Interface: Utilizing the Streamlit framework, the chatbot provides an intuitive and accessible interface. This design ensures that users, regardless of their technical expertise, can easily navigate the system and obtain the information they need.

- Enhanced Decision-Making: By offering buy, sell, or hold recommendations and risk analysis, the chatbot equips investors with actionable insights. This feature is valuable for both novice investors seeking guidance and seasoned analysts looking for additional data points to support their strategies.

Retail investors, institutional investors, and fund managers can leverage the LLM-based chatbot in various ways to enhance their investment strategies. Retail investors, who often lack the resources and expertise of professional analysts, can use the chatbot to gain insights into stock performance and market trends. The chatbot's user-friendly interface and real-time data updates make it an invaluable tool for making informed investment decisions. Institutional investors and fund managers, on the other hand, can use the chatbot to streamline their research process. By providing comprehensive and up-to-date stock analysis, the chatbot allows these professionals to focus on strategic decision-making rather than data collection and interpretation. Additionally, the chatbot's ability to generate sentiment scores from news articles can help institutional investors and fund managers to anticipate market movements and adjust their portfolios accordingly.

- Retail Investors: For individual investors, the chatbot serves as a powerful tool to bridge the gap between complex financial analysis and accessible information. Retail investors can use the chatbot to quickly obtain detailed reports on their stocks of interest, helping them to make more informed decisions without needing extensive financial expertise. The sentiment analysis of news articles provides additional context, allowing investors to gauge market sentiment and potential impacts on stock prices.

- Institutional Investors: Institutional investors, such as hedge funds and mutual funds, can leverage the chatbot to enhance their research capabilities. The real-time data integration and comprehensive analysis reports can supplement traditional research methods, providing a more nuanced understanding of market trends and company performance. The chatbot's ability to process large volumes of data quickly makes it an efficient tool for institutional investors managing extensive portfolios.

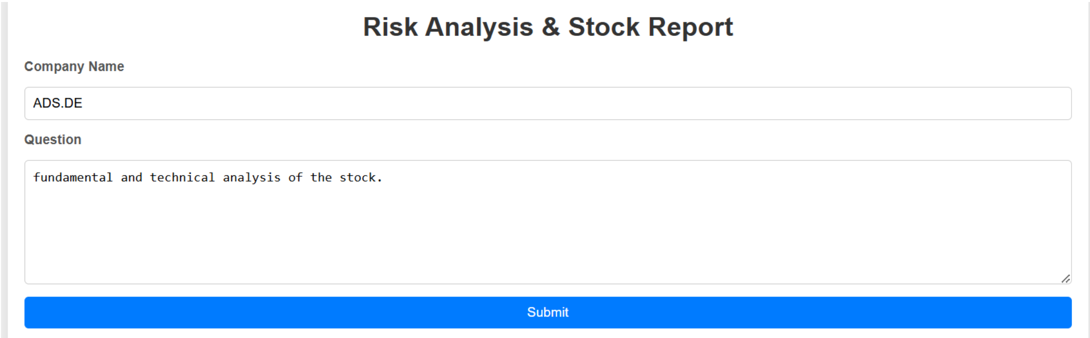
- Fund Managers: Fund managers can use the chatbot to streamline their decision-making processes. The buy, sell, or hold recommendations, along with risk analysis, offer valuable insights that can inform portfolio adjustments. The chatbot's real-time updates ensure that fund managers are always working with the latest information, enabling them to respond swiftly to market changes. Additionally, the customizable features of the chatbot allow fund managers to tailor the analysis to their specific needs, enhancing the overall effectiveness of their investment strategies.

- Financial Advisors: Advisors who provide personalized investment advice can use the chatbot to offer more informed recommendations to their clients. The chatbot's ability to perform both fundamental and technical analysis can help advisors tailor their advice to individual client needs. Firms that offer financial advisory services can integrate the chatbot into their platforms to provide clients with instant access to stock analysis and recommendations, thereby enhancing their service offerings.

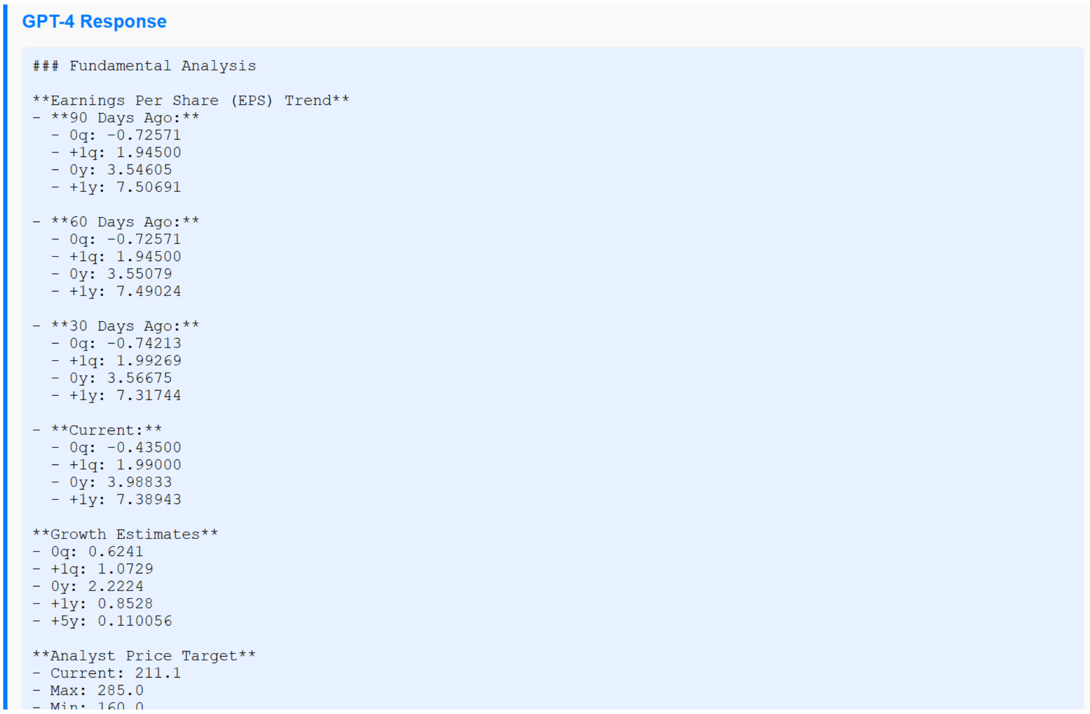
To illustrate the capabilities of the LLM-based chatbot, Fig 10 show the input and response of fundamental and technical analysis of adidas stock and Fig 11 show the input and response of fundamental and technical analysis of Aarti Industries stock.

Fig. 10a and 11a, show the sample input for the application, which takes the company code and the question as input. Fig. 10b, 10c, 10d, 10e shows the GPT-4o analysis and 10f, 10g presents the Gemini 1.5 Flash analysis of Adidas stock listed on the Frankrut Stock Exchange.

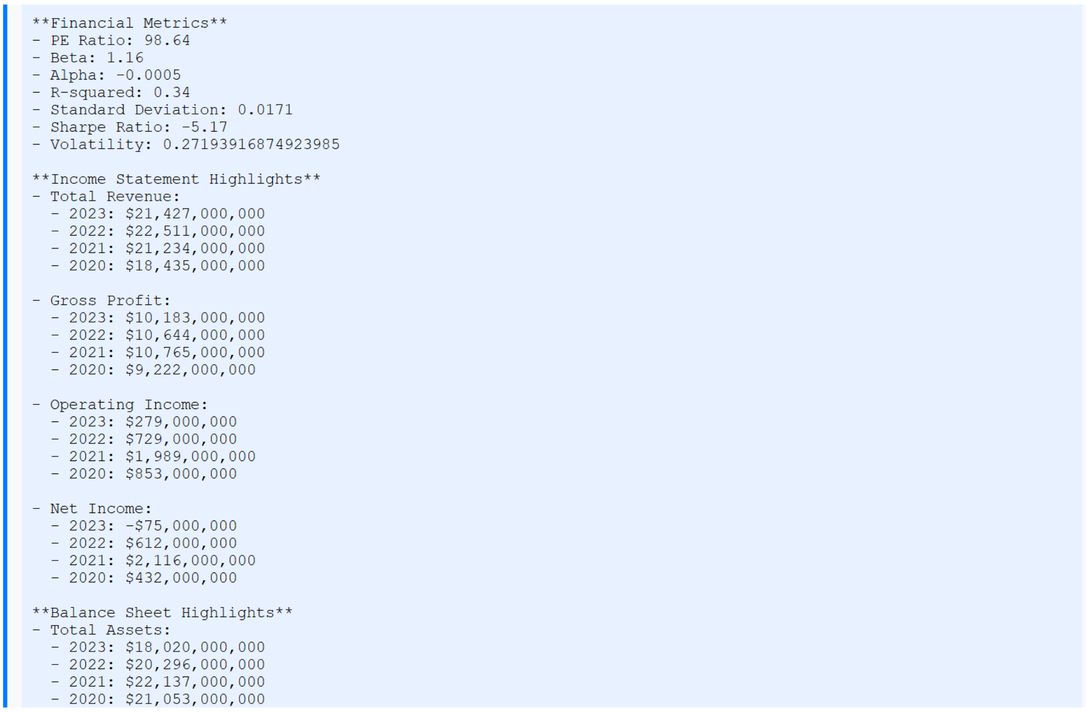
Figure 10b presents the EPS data obtained from Yahoo Finance, which includes the 90-day EPS trend, 60-day EPS trend, 30-day EPS trend, and current EPS trend for Adidas stock. Additionally, it includes growth estimates provided by analysts on Yahoo Finance for the current quarter, next quarter, current year, next year, and the next 3- and 5-year periods. Figure 10c illustrates key financial metrics, such as the PE ratio, Beta score, Alpha score, R-squared score, stock price standard deviation, Sharpe ratio, stock price volatility, and four-year historical



(a) Sample input for the System



(b) Sample GPT-4o response



(c) Sample GPT-4o response

```
- Total Liabilities:
  - 2023: $13,095,000,000
  - 2022: $14,945,000,000
  - 2021: $14,300,000,000
  - 2020: $14,362,000,000

- Total Equity:
  - 2023: $4,580,000,000
  - 2022: $4,991,000,000
  - 2021: $7,519,000,000
  - 2020: $6,454,000,000

**Cash Flow Highlights**
- Operating Cash Flow:
  - 2023: $2,630,000,000
  - 2022: -$479,000,000
  - 2021: $3,192,000,000
  - 2020: $1,486,000,000

- Investing Cash Flow:
  - 2023: -$450,000,000
  - 2022: $495,000,000
  - 2021: -$424,000,000
  - 2020: -$115,000,000

- Financing Cash Flow:
  - 2023: -$1,425,000,000
  - 2022: -$2,963,000,000
  - 2021: -$2,991,000,000
  - 2020: $479,000,000

### Technical Analysis

**Stock Price History**
- **Open Price History (May 27, 2024 - November 21, 2024)**
  - High: $230.899994 (May 31, 2024)
  - Low: $212.300003 (November 21, 2024)
  - Recent: $212.300003 (November 21, 2024)
```

(d) Sample GPT-4o response

```
- **Close Price History (May 27, 2024 - November 21, 2024)**
  - High: $231.500000 (May 30, 2024)
  - Low: $211.100006 (November 21, 2024)
  - Recent: $211.100006 (November 21, 2024)

**Stock Returns**
- Mixed returns with negative and positive fluctuations.
- Notable dips in May and June, with recovery in subsequent months.

**Benchmark Returns**
- Generally negative trends with occasional positive spikes.
- Negative correlation with stock returns.

**Volatility and Risk Metrics**
- Volatility: 0.27 (current)
- PE Ratio: 98.64 (high, indicating potential overvaluation)
- Beta: 1.16 (higher market risk)
- Alpha: -0.0005 (underperformance relative to the market)
- R-squared: 0.34 (moderate correlation with the market)
- Standard Deviation: 0.0171 (low variability)
- Sharpe Ratio: -5.17 (negative, indicating poor risk-adjusted returns)
- Sentiment Score: -0.50 (negative sentiment)

**Analyst Sentiment**
- Mixed news with both positive and negative outlooks.
- Current analyst price target of $211.1, with a range between $160.0 and $285.0.

### Conclusion
**Fundamental Analysis**
- Strong revenue and gross profit figures, but recent net income is negative.
- The company has a high PE ratio, indicating potential overvaluation.
- Cash flows are mixed, with recent negative operating and investing cash flows.
- High debt levels with significant liabilities.

**Technical Analysis**
- Stock price has seen fluctuations with significant highs and lows.
- The stock has exhibited high volatility and risk.
- Negative sentiment and poor risk-adjusted returns.

**Investment Recommendation**
- Caution is advised due to the negative net income, high debt, and poor risk-adjusted returns.
- Potential for value if the company can improve its earnings and manage its liabilities.
- Investors should monitor upcoming earnings reports and market conditions closely.
- Diversify investments to mitigate risk associated with this stock.
```

(e) Sample GPT-4o response

Gemini Response

```
## ADS.DE Fundamental and Technical Analysis:

This is a comprehensive analysis of ADS.DE based on the provided data:

**Fundamental Analysis:**

* **EPS Trend:** The company's EPS trend is positive, showcasing a consistent growth pattern over the past year. The current
quarter's EPS is expected to be -0.435, which is a slight decline, but the positive trend should continue for the following
quarters.
* **Growth Estimate:** The growth estimate is also positive across all timeframes, indicating a healthy outlook for the
company.
* **PE Ratio:** The PE ratio of 98.64 is high, suggesting the stock is currently overvalued compared to its earnings. This
might signal a potential for a price correction in the near future.
* **Beta:** The beta of 1.16 indicates that the stock is slightly more volatile than the overall market.
* **Alpha:** The negative alpha (-0.0005) implies that the stock's returns are not outperforming the market after adjusting
for risk, potentially contributing to the high PE ratio.
* **Sentiment Score:**  The negative sentiment score (-0.5) suggests that analysts and investors are generally bearish on the
stock. This aligns with some bearish indicators like the high PE ratio and negative alpha.
* **Risk:** The "High" risk classification, although not quantified, is likely due to the combination of the high PE ratio,
negative alpha, and bearish sentiment score.
* **Analyst Price Target:** The current analyst price target of 211.1 suggests a slight downside potential from the current
stock price, while the max and min price targets suggest a wide range of potential outcomes.
* **Analyst Earnings Estimate:** The analyst earnings estimate aligns with the EPS Trend data, indicating a positive growth
trajectory for the company.

**Technical Analysis:**

* **Volatility:** The volatility measure of 0.27 indicates that the stock's price fluctuates moderately. This can be
interpreted as a sign of potential volatility and risk.
* **Sharpe Ratio:** The Sharpe ratio of -5.17 is extremely negative, suggesting a poor risk-adjusted return. This is a strong
cautionary sign and might indicate a significant risk relative to the potential return.
* **Stock Price History:** The short-term price action shows some volatility but generally indicates a downward trend. The
stock opened at 223.69 and currently trades at 211.1, which is about 5.6% lower.
* **Stock Returns:** The stock returns consistently fall below the benchmark returns, reinforcing the idea of
underperformance against the market.

**Overall Analysis:**

ADS.DE presents a mixed picture with positive fundamental indicators like EPS trend and growth estimates, overshadowed by
concerning technical factors like high volatility and low risk-adjusted return. The high PE ratio and negative alpha suggest
the stock is overpriced and might be due for a correction. Additionally, the bearish sentiment score further reinforces the
potential for a price pullback.
```

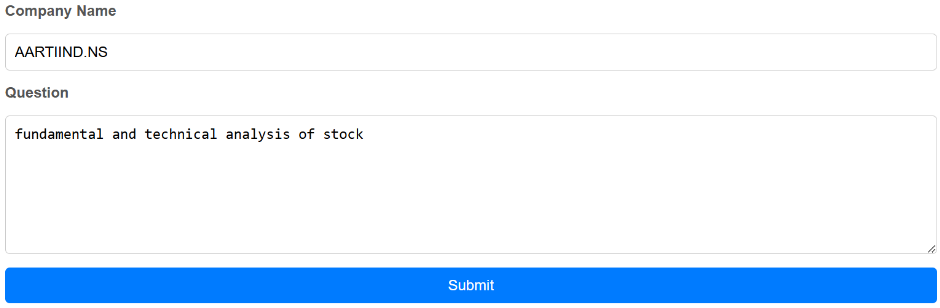
(f) Sample Gemini 1.5 Flash response

```
**Investment Recommendation:**

Given the high risk and the potential for a price correction, it is recommended to exercise caution and avoid investing in
ADS.DE at this time. Investors might consider waiting for a more favorable entry point, with a lower PE ratio, a positive
alpha, and a less negative Sharpe ratio.  It's also essential to monitor the stock's price action closely, the sentiment
score, and the company's future earnings releases for further assessment.

**Disclaimer:** This is a hypothetical analysis based on the provided data. It should not be considered financial advice.
Always conduct thorough research and consult with a qualified financial advisor before making any investment decisions.
```

(g) Sample Gemini 1.5 Flash response

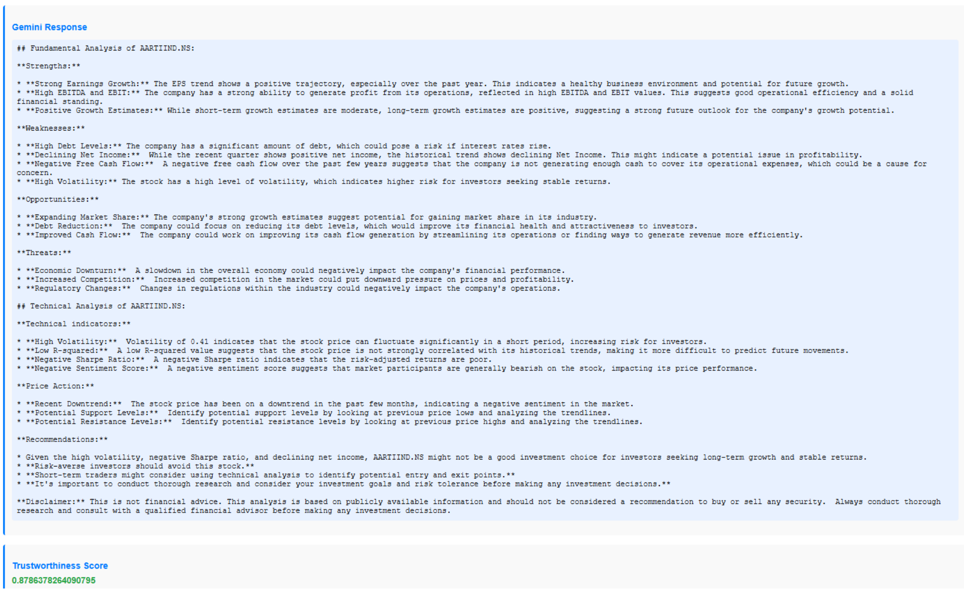
Figure 10: Analysis of Frankfurt Stock Exchange Indexed Adidas Stock

Risk Analysis & Stock Report



(a) Sample input for the System

(b) Sample GPT-4o response



(c) Sample Gemini response

Figure 11: Analysis of Indian National Stock Exchange Indexed Aarti Industries Stock

Service	Latency
Scraper Service	3.6 seconds
Sentiment Service	0.6 seconds
LLM Service	20.5 seconds
Database Service	50 ms

Table 1: Latencies of each module of Application

patterns from income statements, including trends in total revenue, gross profit, operating income, and net income.

Figure 10d depicts four-year trends in balance sheet data, such as total assets, total liabilities, and total equity, alongside four-year cash flow data, including operating cash flow, investing cash flow, and financing cash flow, which are critical for conducting a fundamental analysis of the company. Figure 10e showcases six months of stock data, including open-high prices, open-low prices, recent opening prices, high-close prices, low-close prices, and recent closing prices, as well as fluctuations in stock returns over the past six months and their correlation with benchmark returns. The conclusion section in Figure 10e highlights the fundamental and technical analysis of GPT-4o of the stock, along with its investment recommendations.

Figure 10f presents the functional and technical analysis conducted by the Gemini 1.5 Flash model. Unlike GPT-4o, which focuses on showing detailed financial data, Gemini emphasizes explaining the impact of key metrics. Figure 10g provides the investment recommendations generated by the Gemini model.

When comparing the outputs of GPT-4o and Gemini 1.5 Flash, the results suggest that the analysis provided by GPT-4o appears to be more accurate. Specifically, while Gemini highlights strong fundamentals for the company, this assessment seems inconsistent with the observed significant reductions in total revenue, gross profit, net income, and operating income, leading to concerns about the accuracy of its conclusions.

The development of an LLM-based chatbot using the RAG framework represents a significant advancement in stock market analysis. By integrating real-time data scraping, sentiment analysis, and advanced NLP techniques, this chatbot provides comprehensive and actionable insights for investors. Retail investors can benefit from the accessibility and depth of analysis, while institutional investors and fund managers can streamline their research processes and make more informed strategic decisions. The sample responses for Adidas and Aarti Industries demonstrate the chatbot's ability to provide detailed stock analysis, highlighting its potential to transform the way investors approach the stock market. As the technology continues to evolve, the LLM-based chatbot is poised to become an indispensable tool for investors seeking to navigate the complexities of the financial markets.

The end-to-end pipeline is implemented as a microservice architecture to facilitate the extraction, analysis, and utilization of financial data and news articles for stock trend prediction and user query responses. The pipeline comprises four distinct services, that are Scraper Service, Sentiment Analysis Service, LLM Service, and Database Service, each designed to perform specific tasks and communicate with each other through REST APIs and message queues. The latencies of each service are listed in Table 1

The Scraper Service is responsible for gathering essential stock information, such as EPS, EBITDA, moving averages, and current prices, from Yahoo Finance based on the company name. Additionally, it scrapes related news articles from Yahoo News. This service demonstrated a latency of 3.6 seconds, indicating its efficiency in collecting and processing the required data promptly.

The Sentiment Analysis Service analyzes the sentiment of the scraped news articles, generating sentiment scores that provide insights into the market's perception of the company. This service exhibited a latency of 0.6 seconds, showcasing its capability to quickly process and analyze the sentiment of the news articles.

The LLM Service plays a crucial role in the pipeline by feeding the scraped news articles and stock information into the LLM model to generate responses to user queries. This service has a latency of 20 seconds, within which the GPT model takes 14.9 seconds to respond, and the Gemini 1.5 Flash model takes 5.5 seconds to respond. The latency observed in this service is primarily due to the computational complexity and the time required for the LLM models to process the input data and generate accurate responses.

The Database Service is tasked with storing user queries, responses, and other relevant data in a PostgreSQL database. This service demonstrated a latency of 50 milliseconds, indicating its high efficiency in updating the database with minimal delay. The results of this study highlight the impact of using LLMs for financial analysis and potential of using microservice architecture to build scalable and efficient systems for financial analysis.

6 Conclusion

This paper introduces a novel chatbot system leveraging the RAG framework to enhance the accessibility and accuracy of stock market analysis for both novice and seasoned investors. The chatbot takes a company name and a specific query as input, dynamically retrieving pertinent stock information such as earnings per share, EBITDA, 50-day moving average, and current share price from Yahoo Finance. Additionally, it scrapes related news articles from Yahoo to generate sentiment scores, which are then fed into Large Language Models (LLMs) like GPT-4o and Gemini 1.5 flash models to provide comprehensive responses.

The chatbot is adept at performing both fundamental and technical analysis of stocks, offering buy, sell, or hold recommendations based on the synthesized data. It also provides risk analysis, helping investors make informed decisions by evaluating potential risks associated with their investments. The integration of real-time data retrieval and advanced NLP techniques ensures that the analysis is both timely and contextually relevant, democratizing access to high-quality financial insights traditionally reserved for expert analysts.

Looking ahead, the cahtbot can be integrated with the user's personal data. This enhancement aims to broaden the scope of the chatbot to include financial health analysis for businesses and individuals, which could be instrumental for banks in reducing NonPerforming Assets (NPAs) by making more informed loan approval decisions. Furthermore, this integration could significantly benefit investors by aiding in comprehensive portfolio management, thereby optimizing their investment strategies. The further use cases are listed below.

- Personalized Investment Advice: Customers can receive individualized investment advice from the stock analyzer according to their financial circumstances, investment objectives, and risk tolerance. To provide individualized recommendations, the LLM can examine market trends, customer data, and the performance of specific stocks.

- Risk Assessment: Banks can use the system to evaluate the risk of different investment options, including stocks, bonds, and derivatives. To find possible risks and issue early warnings, the LLM can examine economic indicators, market volatility, and company-specific variables.

- Fraud Detection: Financial transactions can be examined by the LLM, who can spot trends that might point to fraud, including insider trading or money laundering. Natural language processing and machine learning are combined to enable the system to identify irregularities and notify the appropriate authorities.

- Customer Service: The stock analyzer can be incorporated into virtual assistants or chatbots to give users prompt, precise answers to their financial inquiries. In addition to providing individualized answers, the LLM can comprehend client inquiries and retrieve pertinent data from the bank's systems.

- Investment Research: Investment analysts can use the stock analyzer to help them carry out in-depth research on businesses and market trends. In order to determine possible investment opportunities and evaluate the risks connected to particular stocks, the LLM can examine news articles, financial reports, and market data.

- Portfolio Management: Through market analysis, asset mispricing detection, and rebalancing strategy generation, the system can assist portfolio managers in optimizing investment portfolios. The LLM can offer data-driven portfolio management recommendations by taking into account variables like market volatility, investment goals, and risk tolerance.

- Competitive Intelligence: Through the analysis of other investment firms' performance and strategies, the system can assist investment organizations in gaining a competitive advantage. To find competitive advantages and possible risks, the LLM can examine trading activity, market share, and investment portfolios.
- Stock Selection: Portfolio managers can choose stocks that fit their risk tolerance and investing goals with the aid of the stock analyzer. To find promising investment opportunities, the LLM can examine both technical and fundamental factors, including market sentiment, valuation, and earnings growth.
- Performance Analysis: Portfolio managers can use the stock analyzer to examine the performance of their holdings and pinpoint areas in need of development. The LLM is able to examine past portfolio performance, evaluate it against benchmarks, and pinpoint the elements that have influenced portfolio returns.

In conclusion, the chatbot represents a significant advancement in the field of financial analysis, combining the power of LLMs with dynamic data retrieval to provide nuanced and actionable insights. By making sophisticated financial analysis more accessible, and aim to empower a broader audience of investors, ultimately contributing to more informed and effective investment decisions. Future work will focus on expanding the chatbot's capabilities to encompass a wider range of financial services, thereby enhancing its utility and impact in the financial sector.

References

[1] Asanka Abeysinghe. *Event-Driven Architecture: The Path to Increased Agility and High Expandability*. Tech. rep. Sept. 2016.

[2] Shashwat Agrawal. "My Stock Trading Strategy Using Technical Analysis". In: *Available at SSRN 4952407* (2024).

[3] Sourav Banerjee, Ayushi Agarwal, and Saloni Singla. "Llms will always hallucinate, and we need to live with this". In: *arXiv preprint arXiv:2409.05746* (2024).

[4] Terence Bennett. *REST API Principles — A Comprehensive Overview*. URL: https://blog.dreamfactory.com/rest-apis-an-overview-of-basic-principles.

[5] Rithesh Bhat and Bhanu Jain. "Stock Price Trend Prediction using Emotion Analysis of Financial Headlines with Distilled LLM Model". In: *Proceedings of the 17th International Conference on PErvasive Technologies Related to Assistive Environments*. 2024, pp. 67–73.

[6] Zongjing Cao, Yan Li, and Byeong-Seok Shin. "Content-Adaptive and Attention-Based Network for Hand Gesture Recognition". In: *Applied Sciences* 12 (Feb. 2022), p. 2041. DOI: 10.3390/app12042041.

[7] Gary Charness and Chetan Dave. "Confirmation bias with motivated beliefs". In: *Games and Economic Behavior* 104 (2017), pp. 1–23.

[8] Jiuhai Chen and Jonas Mueller. "Quantifying uncertainty in answers from any language model and enhancing their trustworthiness". In: *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2024, pp. 5186–5200.

[9] Comparables.ai. "The Future of Market Analysis: Exploring Emerging Technologies and Innovative Approaches". In: (). URL: https://www.comparables.ai/articles/future-of-market-analysis-exploring-emerging-technologies-and-innovative-approaches.

[10] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. "Multi-head attention: Collaborate instead of concatenate". In: *arXiv preprint arXiv:2006.16362* (2020).

[11] Lev Craig. *GPT-4o vs. GPT-4: How do they compare?* URL: https://www.techtarget.com/searchenterpriseai/feature/GPT-4o-vs-GPT-4-How-do-they-compare.

[12] Vincent P Crawford and Joel Sobel. "Strategic information transmission". In: *Econometrica: Journal of the Econometric Society* (1982), pp. 1431–1451.

[13] Jacob Devlin. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[14] Han Ding et al. "Large language model agent in financial trading: A survey". In: *arXiv preprint arXiv:2408.06361* (2024).

[15] Rian Dolphin et al. "Extracting Structured Insights from Financial News: An Augmented LLM Driven Approach". In: *arXiv preprint arXiv:2407.15788* (2024).

[16] Nibedita Dutta. *Unveiling the Inner Workings: A Deep Dive into BERT's Attention Mechanism*. URL: https://www.analyticsvidhya.com/blog/2023/12/berts-attention-mechanism/.

[17] Dario Fabiani. *Structured Output and Chain of Thought: Enhancing GPT-4o Completions*. URL: https://medium.com/@dario.fabiani/structured-output-and-chain-of-thought-revolutionizing-gpt-4o-interactions-90f60ce75186.

[18] FasterCapital. “Role of fundamental analysis”. In: (). URL: https://fastercapital.com/startup-topic/role-of-fundamental-analysis.html.

[19] Fiza Fatima. *GPT-4o Explained: Training, Features, and Performance*. URL: https://datasciencedojo.com/blog/gpt4o/.

[20] fidelity. “Market volatility: defined and explained”. In: (). URL: https://www.fidelity.com.sg/static/singapore/pdf/volatility-1-market-volatility-defined.pdf.

[21] Bajaj Finserve. *Fundamental Analysis*. URL: https://www.bajajfinserv.in/fundamental-analysis.

[22] Wensheng Gan, Shicheng Wan, and S Yu Philip. “Model-as-a-service (MaaS): A survey”. In: *2023 IEEE International Conference on Big Data (BigData)*. IEEE. 2023, pp. 4636–4645.

[23] Rui Gonçalves et al. “Deep learning in exchange markets”. In: *Information Economics and Policy* 47 (2019), pp. 38–51.

[24] Grammarly. *GPT-4o 101: What It Is and How It Works*. URL: https://www.grammarly.com/blog/ai/what-is-gpt-4o/.

[25] Bravos Consulting Group. *Understanding GPT-4o: The Power Behind OpenAI’s Latest Language Model*. URL: https://bravocg.com/understanding-gpt-4o-the-power-behind-openais-latest-language-model/.

[26] Tian Guo and Emmanuel Hauptmann. “Fine-tuning large language models for stock return prediction using newsflow”. In: *arXiv preprint arXiv:2407.18103* (2024).

[27] Yufeng Han et al. “Technical analysis in the stock market: A review”. In: *Handbook of Investment Analysis, Portfolio Management, and Financial Derivatives: In 4 Volumes* (2024), pp. 1893–1928.

[28] Erich Hellstrom. *An analysis of Google models: Gemini 1.5 Flash vs 1.5 Pro*. URL: https://blog.promptlayer.com/an-analysis-of-google-models-gemini-1-5-flash-vs-1-5-pro/.

[29] Aaron Hurst et al. “Gpt-4o system card”. In: *arXiv preprint arXiv:2410.21276* (2024).

[30] Ray Jackendoff. “Lexical Semantics”. In: (Jan. 2022). DOI: 10.1093/oxfordhb/9780198845003.013.3.

[31] Sardar Jaf and Calum Calder. “Deep learning for natural language parsing”. In: *IEEE Access* 7 (2019), pp. 131363–131373.

[32] Murad Kablan et al. “Stateless Network Functions: Breaking the Tight Coupling of State and Processing”. In: *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. Boston, MA: USENIX Association, Mar. 2017, pp. 97–112. ISBN: 978-1-931971-37-9. URL: https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/kablan.

[33] Ian Kelk. *What is RAG? (Retrieval Augmented Generation)*. URL: https://www.clarifai.com/blog/what-is-rag-retrieval-augmented-generation.

[34] Alex Kim, Maximilian Muhn, and Valeri Nikolaev. “Financial statement analysis with large language models”. In: *arXiv preprint arXiv:2407.17866* (2024).

[35] Hyunjong Kim and Hayoung Oh. "Llm Analyst: What Stocks Do You Recommend Today". In: *Available at SSRN 4899957* ().

[36] Soohyeong Kim et al. "Bidirectional Masked Self-attention and N-gram Span Attention for Constituency Parsing". In: *Findings of the Association for Computational Linguistics: EMNLP 2023*. 2023, pp. 326–338.

[37] Yungi Kim et al. "LP Data Pipeline: Lightweight, Purpose-driven Data Pipeline for Large Language Models". In: *arXiv preprint arXiv:2411.11289* (2024).

[38] Kemal Kirtac and Guido Germano. "Sentiment trading with large language models". In: *Finance Research Letters* 62 (2024), p. 105227.

[39] Acron Labs. *GPT 3 vs. GPT 4: 10 Key Differences How to Choose*. URL: https://www.acorn.io/resources/learning-center/gpt3-vs-gpt4/.

[40] A. Lakshmi and Sailaja Vedala. "Quantitative finance - Black box trading models". In: *International Journal of Applied Business and Economic Research* 15 (Jan. 2017), pp. 41–53.

[41] Wolfgang Lehner and Kai-Uwe Sattler. "Database as a service (DBaaS)". In: *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*. 2010, pp. 1216–1217. DOI: 10.1109/ICDE.2010.5447723.

[42] Kai Lei et al. "Time-driven feature-aware jointly deep reinforcement learning for financial signal representation and algorithmic trading". In: *Expert Systems with Applications* 140 (2020), p. 112872.

[43] Patrick Lewis et al. "Retrieval-augmented generation for knowledge-intensive nlp tasks". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 9459–9474.

[44] Elizabeth D Liddy. "Natural language processing". In: (2001).

[45] Zhiyuan Liu, Yankai Lin, and Maosong Sun. "Compositional Semantics". In: July 2020, pp. 43–57. ISBN: 978-981-15-5572-5. DOI: 10.1007/978-981-15-5573-2_3.

[46] R Logambal and G Kanagasabapathy. "Investment Decision Using Technical Analysis: A Study on Selected Stocks in Indian Stock Market". In: *Shanlax International Journal of Economics* 12 (Mar. 2024), pp. 62–68. DOI: 10.34293/economics.v12i2.7059.

[47] Alejandro Lopez-Lira and Yuehua Tang. "Can chatgpt forecast stock price movements? return predictability and large language models". In: *arXiv preprint arXiv:2304.07619* (2023).

[48] K Maithili et al. "A Survey (NLP) Natural Language Processing and Transactions on (NNL) Neural Networks and learning Systems". In: *E3S Web of Conferences*. Vol. 430. EDP Sciences. 2023, p. 01148.

[49] Ben Mann et al. "Language models are few-shot learners". In: *arXiv preprint arXiv:2005.14165* 1 (2020).

[50] Dastan Hussen Maulud et al. "State of art for semantic analysis of natural language processing". In: *Qubahan academic journal* 1.2 (2021), pp. 21–28.

[51] John Morgan and Phillip C Stocken. "An analysis of stock recommendations". In: *RAND Journal of economics* (2003), pp. 183–203.

[52] Fionn Murtagh. "Multilayer perceptrons for classification and regression". In: *Neurocomputing* 2.5-6 (1991), pp. 183–197.

[53] Haowei Ni et al. "Harnessing earnings reports for stock predictions: A qlora-enhanced llm approach". In: *2024 6th International Conference on Data-driven Optimization of Complex Systems (DOCS)*. IEEE. 2024, pp. 909–915.

[54] Yuqi Nie et al. “A Survey of Large Language Models for Financial Applications: Progress, Prospects and Challenges”. In: *arXiv preprint arXiv:2406.11903* (2024).

[55] Mark J Roe. “Stock market short-termism’s impact”. In: *University of Pennsylvania Law Review* (2018), pp. 71–121.

[56] Giorgio Roffo. “Exploring Advanced Large Language Models with LLMsuite”. In: *arXiv preprint arXiv:2407.12036* (2024).

[57] Cameron R.Wolfe. *Masked self-attention: How LLMs learn relationships between tokens*. Sept. 2024. URL: https://stackoverflow.blog/2024/09/26/masked-self-attention-how-llms-learn-relationships-between-tokens/.

[58] Naomi Sager. “Syntactic analysis of natural language”. In: *Advances in computers*. Vol. 8. Elsevier, 1967, pp. 153–188.

[59] V Sanh. “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter”. In: *arXiv preprint arXiv:1910.01108* (2019).

[60] scalenut. *Gemini 1.5: Flash, Pro, and Everything You Need to Know*. URL: https://www.scalenut.com/blogs/gemini-1-5-flash-pro-guide.

[61] Ventura Securities. *What is fundamental analysis in the stock market?* URL: https://www.venturasecurities.com/blog/what-is-fundamental-analysis-in-the-stock-market/.

[62] Kuldeep Singh, Simerjot Kaur, and Charese Smiley. “FinQAPT: Empowering Financial Decisions with End-to-End LLM-driven Question Answering Pipeline”. In: *Proceedings of the 5th ACM International Conference on AI in Finance*. 2024, pp. 266–273.

[63] WeeTech Solution. *GPT 4 vs GPT 4o vs Gemini 1.5 Flash: A Technical Analysis of AI Titans*. URL: https://www.weetechsolution.com/blog/gpt-4-vs-gpt-4o-vs-gemini-1-5-flash.

[64] Ion Stoica. “Stateless Core: A Scalable Approach for Quality of Service in the Internet”. In: 2979 (Jan. 2000).

[65] CS Sütçü and C Aytekin. “An example of pragmatic analysis in natural language processing: sentimental analysis of movie reviews”. In: *CTC 2019* (2019).

[66] CFI Team. *Balance Sheet*. URL: https://corporatefinanceinstitute.com/resources/accounting/balance-sheet/.

[67] CFI Team. *Cash Flow*. URL: https://corporatefinanceinstitute.com/resources/accounting/cash-flow/.

[68] CFI Team. *Income Statement*. URL: https://corporatefinanceinstitute.com/resources/accounting/income-statement/.

[69] Gemini Team et al. “Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context”. In: *arXiv preprint arXiv:2403.05530* (2024).

[70] InnoBoon Technologies. *The Evolution of GPT: From GPT-1 to GPT-4o Mini*. URL: https://www.linkedin.com/pulse/evolution-gpt-from-gpt-1-gpt-4o-mini-innoboon-gzt1f/.

[71] A Vaswani. “Attention is all you need”. In: *Advances in Neural Information Processing Systems* (2017).

[72] Anojan Vickneswaran. “Interpretation on Financial Statements”. In: Nov. 2020.

[73] Nhi NY Vo et al. “Deep learning for decision making and the optimization of socially responsible investments and portfolio”. In: *Decision Support Systems* 124 (2019), p. 113097.

[74] James Walker. *The importance of stateless architecture in authorization systems*. URL: https://www.cerbos.dev/blog/the-importance-of-stateless-architecture-in-authorization-systems.

[75] Wuyu Wang et al. "Portfolio formation with preselection using deep learning from long-term financial data". In: *Expert Systems with Applications* 143 (2020), p. 113042.

[76] Jason Wei et al. "Chain-of-thought prompting elicits reasoning in large language models". In: *Advances in neural information processing systems* 35 (2022), pp. 24824–24837.

[77] Ruoxu Wu. "Portfolio performance based on LLM news scores and related economical analysis". In: *Available at SSRN 4709617* (2024).

[78] Steve Yang, Zulfikhar Ali, and Bryan Wong. "FLUID-GPT (Fast Learning to Understand and Investigate Dynamics with a Generative Pre-Trained Transformer): Efficient Predictions of Particle Trajectories and Erosion". In: (Aug. 2023). DOI: 10.26434/chemrxiv-2023-ppk9s.

[79] I de Zarzà et al. "Optimized financial planning: integrating individual and cooperative budgeting models with LLM recommendations". In: *AI* 5.1 (2023), pp. 91–114.

[80] Huaqin Zhao et al. "Revolutionizing finance with llms: An overview of applications and insights". In: *arXiv preprint arXiv:2401.11641* (2024).

[81] Rubens Zimbres. *Multimodality with Gemini-1.5-Flash: Technical Details and Use Cases*. URL: https://medium.com/google-cloud/multimodality-with-gemini-1-5-flash-technical-details-and-use-cases-84e8440625b6.

A Appendix

A.1 Scrapper Service

StockService class Implements the following functionalities:

- Fetches stock and benchmark data.
- Calculates critical financial metrics (e.g., volatility, Sharpe ratio, beta, etc.).
- Extracts financial data from income statements, balance sheets, and cash flows.
- Retrieves news articles related to the stock for sentiment analysis or contextual insights.

The code uses following libraries:

- yfinance: For fetching stock-related historical and live financial data.
- statsmodels: For statistical modeling and regression analysis (used for beta and alpha computation).
- BeautifulSoup and requests: For web scraping of stock-related news.
- numpy: To perform numerical operations such as standard deviation and mean.
- datetime: To calculate time ranges for stock data retrieval.

Method 1: get_stock_data
It fetches stock and benchmark data, performs calculations, and compiles relevant insights.

- etches stock data using yfinance.Ticker.
- Determines an appropriate benchmark index based on the stock's market suffix (e.g., .NS for NSE India, .BO for BSE India).
- Historical data is retrieved for a time period calculated using the num_months parameter.
- Retrieves specific rows from financial statements (income statement, balance sheet, cashflow statement)

Method 2: scrape_news
Scrapes news articles related to the given stock.

- Constructs a URL using the stock's symbol to retrieve relevant articles from Yahoo Finance.
- Fetches the webpage content and parses it using BeautifulSoup.
- Extracts headlines ($< h3 >$ tags) and filters out irrelevant entries

```
import yfinance as yf
from datetime import datetime, timedelta
import numpy as np
import statsmodels.api as sm
from bs4 import BeautifulSoup
import requests

class StockService:
    RISK_FREE_RATE = 0.02  # Risk-free rate for Sharpe ratio

    def get_stock_data(self, company_name: str, num_months: int):
        stock = yf.Ticker(company_name)

        if company_name.split(".")[-1] == "NS":
           benchmark = yf.Ticker("^NSEI")
        elif company_name.split(".")[-1] == "BO":
           benchmark = yf.Ticker("^BSESN")
        elif company_name.split(".")[-1] == "AX":
            benchmark = yf.Ticker("^AXJO")
        elif company_name.split(".")[-1] == "L":
            benchmark = yf.Ticker("^FTSE")
        elif company_name.split(".")[-1] == "DE":
            benchmark = yf.Ticker("^GDAXI")
        else:
            benchmark = yf.Ticker("^IXIC")

        end_date = datetime.now()
        start_date = end_date - timedelta(days=30 * num_months)
        stock_hist = stock.history(start=start_date, end=end_date)
        benchmark_hist = benchmark.history(start=start_date, end=end_date)

        stock_returns = stock_hist['Close'].pct_change().dropna()
        benchmark_returns = benchmark_hist['Close'].pct_change().dropna()
        stock_returns, benchmark_returns =
        stock_returns.align(benchmark_returns, join='inner')

        volatility = stock_returns.std() * np.sqrt(252)
        pe_ratio = stock.info.get('trailingPE', None)

        X = sm.add_constant(benchmark_returns)
        model = sm.OLS(stock_returns, X).fit()
        beta = model.params[1]
        alpha = model.params[0]
        r_squared = model.rsquared
        std_dev = stock_returns.std()
        mean_return = stock_returns.mean() * 252
        sharpe_ratio = (mean_return - self.RISK_FREE_RATE) / std_dev
        news_texts = self.scrape_news(company_name)
```

```python
    # Keys to extract
    income_extract = list(stock.income_stmt.index)
    balance_extract = list(stock.balance_sheet.index)
    cashflow_extract = list(stock.cashflow.index)

    # Dictionary to store extracted data
    extracted_data = {}

    # Extract rows for each key
    for key in income_extract:
        if key in stock.income_stmt.index:
            extracted_data[key] = stock.income_stmt.loc[key]
    for key in balance_extract:
        if key in stock.balance_sheet.index:
            extracted_data[key] = stock.balance_sheet.loc[key]
    for key in cashflow_extract:
        if key in stock.cashflow.index:
            extracted_data[key] = stock.cashflow.loc[key]

    return_dict = {
        "volatility": volatility,
        "pe_ratio": pe_ratio,
        "beta": beta,
        "alpha": alpha,
        "r_squared": r_squared,
        "std_dev": std_dev,
        "sharpe_ratio": sharpe_ratio,
        "news_texts": news_texts,
        "eps_trend_90day": stock.eps_trend['90daysAgo'],
        "eps_trend_60day": stock.eps_trend['60daysAgo'],
        "eps_trend_30day": stock.eps_trend['30daysAgo'],
        "eps_trend_current": stock.eps_trend['current'],
        "growth_estimate": stock.growth_estimates['stock'],
        "stock_price open history": stock_hist['Open'],
        "stock_price close history": stock_hist['Close'],
        "stock returns": stock_returns,
        "benchmark returns": benchmark_returns,
        "Current Analyst Price Target": stock.analyst_price_targets['
           current'],
        "Max Analyst Price Target": stock.analyst_price_targets['high'],
        "Min Analyst Price Target": stock.analyst_price_targets['low'],
        "Analyst Earnigns Estimate": stock.earnings_estimate['avg']
    }
    return_dict.update(extracted_data)

    return return_dict

def scrape_news(self, company_name: str):
    from bs4 import BeautifulSoup
    import requests
```

```
        news_url = f"https://finance.yahoo.com/lookup/?s={company_name}"
        news_content = requests.get(news_url)
        soup = BeautifulSoup(news_content.content, 'html.parser')
        news_articles = [h3.get_text() for h3 in soup.find_all('h3')]

        return [heading for heading in news_articles if len(heading.split()) >
            4]
```

A.2 Sentiment Service

Sentiment Analysis Service implemented using the transformers library. The service processes a list of news headlines and calculates a net sentiment score that reflects the overall sentiment (positive or negative) of the provided texts.

Method 1: analyze_sentiment

- Processes each news headline in the input list and assigns a sentiment label (POSITIVE or NEGATIVE) along with a confidence score.

```
from transformers import pipeline

class SentimentService:
    def __init__(self):
        self.sentiment_pipeline = pipeline("sentiment-analysis")

    def analyze_sentiment(self, news_texts: list):

        sentiments = self.sentiment_pipeline(news_texts)
        positive_sentiment = sum([1 for s in sentiments if s['label'] == '
            POSITIVE'])
        negative_sentiment = sum([1 for s in sentiments if s['label'] == '
            NEGATIVE'])
        total = len(sentiments)
        return (positive_sentiment - negative_sentiment) / total  # Net
            sentiment score
```

A.3 GPT Service

GPTService code sets up a service for querying Azure GPT-4.0 OpenAI and generating responses based on provided context, including stock data, sentiment scores, and other inputs. It also evaluates the trustworthiness of the generated response using CleanLab Studio.

The code uses following libraries:

- numpy: Used for numerical computations. Here, it generates random trustworthiness scores.
- langchain: Used to build prompts and create a chain for interacting with the Azure GPT model.
- cleanlab_studio: Provides trustworthiness evaluation.

- AzureChatOpenAI and AzureOpenAIEmbeddings: APIs for Azure's OpenAI service to interact with chat models and generate embeddings. os and json: Handle environment variables and manage JSON data, respectively.

__init__ Method
Initializes key components required for querying Azure GPT-4 and calculating trustworthiness.
Method 1: query_gpt_4_and_tlm

Executes a query on the GPT-4 model using stock data, sentiment analysis, risk, and strategy, and evaluates the response for trustworthiness.

- Converts stock data (a dictionary) into a formatted string to create the "context."
- Constructs a prompt template using PromptTemplate from LangChain
- LLMChain: Links the prompt with the GPT model

```
import numpy as np
from langchain.prompts import PromptTemplate
from langchain.chains import LLMChain
from cleanlab_studio import Studio
from langchain.chat_models import AzureChatOpenAI
from langchain_openai import AzureOpenAIEmbeddings
import os

class GPTService:
    def __init__(self):
        self.studio = Studio(os.getenv('TLM_API_KEY'))
        self.llm = AzureChatOpenAI(
            model=os.getenv('gpt_model'),
            azure_deployment=os.getenv('azure_deployment'),
            api_key=os.getenv('API_KEY'),
            azure_endpoint=os.getenv('AZURE_ENDPOINT')
        )
        self.embedding_model = AzureOpenAIEmbeddings(
            model="text-embedding-ada-002",
            api_key=os.getenv('API_KEY'),
            azure_endpoint=os.getenv('AZURE_ENDPOINT'),
            azure_deployment=os.getenv('azure_embedding_deployment')
        )


    def query_gpt_4_and_tlm(self, stock_data, question):

        st_data = ", ".join([f"{key}: {value}\n" for key, value in stock_data.
            items()])

        context = st_data
```

```
        prompt_template = PromptTemplate(
                                        input_variables=['question'],
                                        template=context + "\nQuestion: {
                                            question}"
                                    )
        llm_chain = LLMChain(llm=self.llm, prompt=prompt_template)
        answer = llm_chain.run(question)

        # Trustworthiness score from CleanLab Studio

        tlm = self.studio.TLM()
        trustworthiness_score = tlm.get_trustworthiness_score(context,response=
            answer)

        return answer, trustworthiness_score
```

A.4 Gemini Service

Gemino Service code provides a service to generate responses using the Gemini-1.5 Flash model on Google Vertex AI. It integrates Vertex AI for model execution and CleanLab Studio for evaluating response trustworthiness. Below is a detailed explanation.

The code uses following libraries:

- langchain: Can facilitate the creation of custom prompts and chain operations. These are imported but not actively used in the current implementation.
- cleanlab_studio: A service for calculating trustworthiness scores of AI-generated responses.
- vertexai: Google Vertex AI SDK for integrating with Vertex AI services.
- vertexai.generative_models: Used to configure generation settings and interact with Google's generative models.

Method 1: gemini_response

Queries the Gemini-1.5 Flash model hosted on Vertex AI with the provided stock data and question. Evaluates the response for trustworthiness.

- Converts the stock data dictionary into a formatted string.
- Combines the stock data with the user question to create the context.
- GenerationConfig Specifies parameters for generating responses.

```
import numpy as np
from langchain.prompts import PromptTemplate
```

```
from langchain.chains import LLMChain
from cleanlab_studio import Studio
from langchain.chat_models import AzureChatOpenAI
from langchain_openai import AzureOpenAIEmbeddings
import vertexai
from vertexai.generative_models import GenerationConfig, GenerativeModel
import os


class VertexAIService:
    def __init__(self):
        self.studio = Studio(os.getenv(  TLM_API_KEY      ))
        self.project_id = os.getenv("GOOGLE_PROJECT_ID")
        self.location = os.getenv("GOOGLE_LOCATION")
        self.credentials_path = os.getenv("credential_path")
        self._init_vertex_ai()

    def _init_vertex_ai(self):
        os.environ["GOOGLE_APPLICATION_CREDENTIALS"] = self.credentials_path
        vertexai.init()


    def gemini_response(self, stock_data, question):

        st_data = ", ".join([f"{key}: {value}\n" for key, value in stock_data.
           items()])

        context = st_data + f"""Question: {question} """

        generation_config = GenerationConfig(
                                                temperature=0.9,
                                                top_p=1.0,
                                                top_k=32,
                                                candidate_count=1,
                                                max_output_tokens=8192,
                                            )

        model = GenerativeModel(
            "gemini-1.5-flash-001",
            system_instruction=[
                "You are a good financial advisor", "You are good at analysing
                   Technical and Fundamental Anlysis of Stocks",
                "You are good at answering questions."
            ]
        )

        contents = [context]

        response = model.generate_content(contents, generation_config=
           generation_config)
```

```
        tlm = self.studio.TLM()
        trustworthiness_score = tlm.get_trustworthiness_score(context,response=
            response.text)


        return response.text, trustworthiness_score
```

A.5 Calling Service

The provided FastAPI code creates a service that integrates multiple AI and analytical services to handle stock analysis requests. Below is a step-by-step explanation of the analyze stock endpoint and its supporting structure

- Analyzes a stock for a given company name.
- Performs sentiment analysis, risk analysis, and queries both GPT-4 and Gemini AI models to generate responses.
- Returns AI-generated responses along with their trustworthiness scores.

```
from fastapi import FastAPI, Request, Depends, HTTPException
from fastapi.responses import HTMLResponse, JSONResponse
from fastapi.staticfiles import StaticFiles
from fastapi.templating import Jinja2Templates
from pydantic import BaseModel
from services.stock_service import StockService
from services.sentiment_service import SentimentService
from services.gpt_service import GPTService
from services.gemini_service import VertexAIService
import logging
import ecs_logging
import time
import nest_asyncio
nest_asyncio.apply()

# Your existing imports and financial analysis functions go here...

# Configure logging
logger = logging.getLogger('stdout')
logger.setLevel(logging.INFO)

handler = logging.StreamHandler()
handler.setFormatter(ecs_logging.StdlibFormatter())
logger.addHandler(handler)

app = FastAPI()

# Mount static files for CSS
app.mount("/static", StaticFiles(directory="static"), name="static")
```

```python
# Initialize templates for Jinja2
templates = Jinja2Templates(directory="templates")

# Request model for financial analysis
class AnalyzeRequest(BaseModel):
    company_name: str
    question: str

class ScrapperRequest(BaseModel):
    company_name: str

class RiskAnalysisRequest(BaseModel):
    company_name: str
    question: str

class RiskAnalysisResponse(BaseModel):
    gpt_response: str
    trustworthiness_score: float

# Dependency Injection of services
def get_stock_service():
    return StockService()

def get_sentiment_service():
    return SentimentService()

def get_gpt_service():
    return GPTService()

def get_vertex_service():
    return VertexAIService()

# Render the UI page
@app.get("/", response_class=HTMLResponse)
async def read_index(request: Request):
    return templates.TemplateResponse("index.html", {"request": request})

@app.get("/stockinformation", response_class=HTMLResponse)
async def read_index(request: Request):
    return templates.TemplateResponse("indexscrapper.html", {"request": request
        })

# API endpoint to handle stock analysis requests
@app.post("/analyze")
async def analyze_stock(request: AnalyzeRequest,
                        stock_service: StockService = Depends(get_stock_service
                            ),
                        sentiment_service: SentimentService = Depends(
                            get_sentiment_service),
```

```
                        gpt_service: GPTService = Depends(get_gpt_service),
                        gemini_service: VertexAIService = Depends(
                            get_vertex_service)):
    try:
        st_time = time.time()
        stock_data = stock_service.get_stock_data(request.company_name, 6)
        logger.info(f'Scraper Service Time: {time.time() - st_time}')

        # Perform sentiment analysis
        st_time = time.time()
        sentiment_score = sentiment_service.analyze_sentiment(stock_data["
            news_texts"])
        logger.info(f'Sentiment Service Time: {time.time() - st_time}')


        # Query GPT-4 and get trustworthiness score
        st_time = time.time()
        gpt_response, gpt_trustworthiness_score = gpt_service.
            query_gpt_4_and_tlm(
              stock_data, request.question
        )
        logger.info(f'GPT Response Time: {time.time() - st_time}')

        gemini_st_time = time.time()
        gemini_response, gemini_trustworthiness_score = gemini_service.
            gemini_response(
              stock_data, request.question
        )
        logger.info(f'GPT Response Time: {time.time() - gemini_st_time}')
        logger.info(f"LLM Service Time: {time.time() - st_time}")

        # Call your financial analysis function with the provided data

        return JSONResponse(content={
            "gpt_response": gpt_response,
            "gpt_trustworthiness_score": gpt_trustworthiness_score, #['
                trustworthiness_score'],
            "gemini_response": gemini_response,
            "gemini_trustworthiness_score": gemini_trustworthiness_score #['
                trustworthiness_score']
        })
    except Exception as e:
         logger.error(f"Error analyzing stock: {e}")
         return JSONResponse(content={"error": str(e)}, status_code=500)

@app.post("/webscrapper")
async def webscrapper(request: ScrapperRequest,
                        stock_service: StockService = Depends(get_stock_service
                            )):
    try:
```

```
        st_time = time.time()
        stock_data = stock_service.get_stock_data(request.company_name, 6)
        logger.info(f'Scraper Service Time: {time.time() - st_time}')

        return JSONResponse(content={
            "data": str(stock_data),
        })
    except Exception as e:
        logger.error(f"Error analyzing stock: {e}")
        return JSONResponse(content={"error": str(e)}, status_code=500)
```

A.6 Front-end

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Risk Analysis & Stock Report</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            background-color: #f0f0f0;
        }

        .container {
            width: 80%;
            margin: 50px auto;
            padding: 20px;
            background-color: #fff;
            border-radius: 10px;
            box-shadow: 0px 0px 15px rgba(0, 0, 0, 0.1);
        }

        h1 {
            text-align: center;
            color: #333;
        }

        form {
            display: flex;
            flex-direction: column;
            gap: 15px;
        }

        label {
```

```
    font-weight: bold;
    color: #555;
}

input[type="text"], input[type="number"], input[type="date"], textarea
   {
    padding: 10px;
    border-radius: 5px;
    border: 1px solid #ccc;
    font-size: 16px;
    width: 100%;
    box-sizing: border-box;
}

textarea {
    resize: vertical;
    height: 150px;
}

button {
    padding: 10px 20px;
    background-color: #007BFF;
    color: white;
    font-size: 16px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}

button:hover {
    background-color: #0056b3;
}

.response-container {
    margin-top: 30px;
}

.response-card {
    margin: 20px 0;
    padding: 15px;
    background-color: #f9f9f9;
    border-left: 5px solid #007BFF;
    border-radius: 5px;
}

.response-card h3 {
    margin-bottom: 10px;
    color: #007BFF;
}
```

```
        .response-card .score {
            font-size: 18px;
            color: #28a745;
            font-weight: bold;
        }

        .gpt-response {
            font-family: "Courier New", monospace;
            background-color: #e8f1ff;
            padding: 10px;
            border-radius: 5px;
            white-space: pre-wrap;
        }

        .gemini-response {
            font-family: "Courier New", monospace;
            background-color: #e8f1ff;
            padding: 10px;
            border-radius: 5px;
            white-space: pre-wrap;
        }

    </style>
</head>
<body>

    <div class="container">
        <h1>Risk Analysis & Stock Report</h1>
        <form id="stockForm">
            <label for="company_name">Company Name</label>
            <input type="text" id="company_name" name="company_name" required>

            <label for="question">Question</label>
            <textarea id="question" name="question" required></textarea>

            <button type="submit">Submit</button>
        </form>

        <div class="response-container" id="responseContainer" style="display:
           none;">
            <div class="response-card">
                <h3>GPT-4 Response</h3>
                <pre id="gptResponse" class="gpt-response"></pre>
            </div>
            <div class="response-card">
                <h3>Trustworthiness Score</h3>
                <div class="score" id="gpt-trustworthinessScore"></div>
            </div>
            <div class="response-card">
                <h3>Gemini Response</h3>
```

```
            <pre id="geminiResponse" class="gemini-response"></pre>
        </div>
        <div class="response-card">
            <h3>Trustworthiness Score</h3>
            <div class="score" id="gemini-trustworthinessScore"></div>
        </div>
    </div>
</div>

<script>
    const form = document.getElementById("stockForm");
    const responseContainer = document.getElementById("responseContainer");
    const gptResponseElement = document.getElementById("gptResponse");
    const gpttrustworthinessScoreElement = document.getElementById("gpt-
       trustworthinessScore");
    const geminiResponseElement = document.getElementById("geminiResponse")
       ;
    const geminitrustworthinessScoreElement = document.getElementById("
       gemini-trustworthinessScore");

    form.addEventListener("submit", async (e) => {
        e.preventDefault();

        const formData = new FormData(form);
        const company_name = formData.get("company_name");
        const question = formData.get("question");

        const payload = {
            company_name: company_name,
            question: question
        };

        const response = await fetch("/analyze", {
            method: "POST",
            headers: {
                "Content-Type": "application/json"
            },
            body: JSON.stringify(payload)
        });

        const result = await response.json();

        if (result.gpt_response && result.gpt_trustworthiness_score &&
            result.gemini_response && result.gemini_trustworthiness_score) {
            gptResponseElement.textContent = result.gpt_response;
            gpttrustworthinessScoreElement.textContent = result.
                gpt_trustworthiness_score;
            geminiResponseElement.textContent = result.gemini_response;
            geminitrustworthinessScoreElement.textContent = result.
                gemini_trustworthiness_score;
```

```
                responseContainer.style.display = "block";
            }
        });
    </script>

</body>
</html>


<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Risk Analysis & Stock Report</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            background-color: #f0f0f0;
        }

        .container {
            width: 80%;
            margin: 50px auto;
            padding: 20px;
            background-color: #fff;
            border-radius: 10px;
            box-shadow: 0px 0px 15px rgba(0, 0, 0, 0.1);
        }

        h1 {
            text-align: center;
            color: #333;
        }

        form {
            display: flex;
            flex-direction: column;
            gap: 15px;
        }

        label {
            font-weight: bold;
            color: #555;
        }
```

```
input[type="text"], input[type="number"], input[type="date"], textarea
   {
    padding: 10px;
    border-radius: 5px;
    border: 1px solid #ccc;
    font-size: 16px;
    width: 100%;
    box-sizing: border-box;
}

textarea {
    resize: vertical;
    height: 150px;
}

button {
    padding: 10px 20px;
    background-color: #007BFF;
    color: white;
    font-size: 16px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}

button:hover {
    background-color: #0056b3;
}

.response-container {
    margin-top: 30px;
}

.response-card {
    margin: 20px 0;
    padding: 15px;
    background-color: #f9f9f9;
    border-left: 5px solid #007BFF;
    border-radius: 5px;
}

.response-card h3 {
    margin-bottom: 10px;
    color: #007BFF;
}

.response-card .score {
    font-size: 18px;
    color: #28a745;
    font-weight: bold;
```

```
        }

        .gpt-response {
            font-family: "Courier New", monospace;
            background-color: #e8f1ff;
            padding: 10px;
            border-radius: 5px;
            white-space: pre-wrap;
        }

        .gemini-response {
            font-family: "Courier New", monospace;
            background-color: #e8f1ff;
            padding: 10px;
            border-radius: 5px;
            white-space: pre-wrap;
        }

    </style>
</head>
<body>

    <div class="container">
        <h1>Risk Analysis & Stock Report</h1>
        <form id="stockForm">
            <label for="company_name">Company Name</label>
            <input type="text" id="company_name" name="company_name" required>

            <button type="submit">Submit</button>
        </form>

        <div class="response-container" id="responseContainer" style="display:
           none;">
            <div class="response-card">
                <h3>Stock Data</h3>
                <pre id="stockdata" class="gpt-response"></pre>
            </div>
        </div>
    </div>

    <script>
        const form = document.getElementById("stockForm");
        const responseContainer = document.getElementById("responseContainer");
        const gptResponseElement = document.getElementById("stockdata");

        form.addEventListener("submit", async (e) => {
            e.preventDefault();

            const formData = new FormData(form);
            const company_name = formData.get("company_name");
```

```
            const payload = {
                company_name: company_name
            };

            const response = await fetch("/webscrapper", {
                method: "POST",
                headers: {
                    "Content-Type": "application/json"
                },
                body: JSON.stringify(payload)
            });

            const result = await response.json();

            if (result) {
                gptResponseElement.textContent = result.data;
                responseContainer.style.display = "block";
            }
        });
    </script>

</body>
</html>
```