THE UNIVERSITY OF HONG KONG

COMP4801 - Final Year Project

Final Report

Changjin Lee (3035435840)

TradeInbox: LLM-based Financial News Notification System

20 April 2025

Abstract

Today's volatile financial market and the overwhelming volume of data make it challenging for individual investors to receive timely, relevant updates on their stock portfolios, as manually sifting through vast amounts of information may lead to missed opportunities or losses. The project aims to tackle these challenges by processing user-provided stock information, enhancing it with a keyword generator LLM, and utilizing embedding search to filter relevant news. The summary generator LLM will produce personalized news summaries, delivered in real-time, empowering investors to make more informed investment decisions. Initial results include the implementation of the keyword generator LLM that could improve article relevance filtering by producing pertinent keywords and sentences that are likely to appear in financial news relevant to user inputs. By providing timely summaries, the system aims to help investors respond swiftly to volatile market events. The immediate next step is front-end development using React to build an interactive user interface for the system.

Acknowledgments

We extend our gratitude to Dr. Chow Ka Ho, our final-year project advisor, for his exceptional guidance and continuous support throughout the project. His insightful feedback and mentorship have been pivotal in refining our project and enriching our learning experience. We also want to recognize the contributions of our dedicated team members, whose collaboration and commitment were instrumental in the project's success. A special thanks goes to Mr. Gagandeep Singh, our CAES lecturer, for his invaluable lessons on technical writing, which significantly improved the quality of our documentation. Finally, we are grateful to the faculty, staff, and colleagues at The University of Hong Kong, as well as friends who generously shared their insights and feedback, for their support and encouragement along the way.

Table of Contents

List of Figures

List of Tables

1. Introduction

In today's highly dynamic financial markets, access to relevant and up-to-date information is critical for informed decision-making. A prime example of this is the collapse of FTX. On 8 November 2022, after news articles were published that FTX announced to halt all withdrawals, Bitcoin's price plummeted from 20,600 USD to 15,883 USD within just a few days [1]. This incident demonstrates the importance of promptly reacting to market events to minimize losses.

However, individual traders often struggle to navigate the overwhelming volume and pace of financial news. With thousands of articles published daily across various platforms at unpredictable times, identifying relevant information quickly becomes a significant challenge. A web traffic study by FinText shows that readers spend an average of 30 to 60 seconds per page on financial articles and read 3 to 4 articles daily [2], underscoring the difficulty of navigating news with limited time and resources. While platforms like Yahoo Finance provide real-time news alerts, they lack personalized summaries and stock impact analysis tailored to user custom inputs [3], requiring traders to manually analyze articles for relevance and potential impact—a time-intensive process.

The project ‘LLM-based real-time and personalized financial news notification system’ is a web-based system aiming to tackle this challenge by delivering real-time summary and impact analysis of news articles relevant to user’s stock portfolio by leveraging Large Language Models (LLM), focusing specifically on the USA stock market. Recent achievements in LLMs have enabled new ways of processing text data with its capability to understand unstructured natural language texts and produce highly customizable output utilizing prompt engineering. Within the project, LLMs specifically fine-tuned for financial news analysis, combined with user input such as stock names or custom keywords, allow the

system to filter and select only the most relevant news articles. The system then generates article summaries and stock impact analyses, delivered to users through various SNS channels, including Whatsapp and Discord, to provide convenient access to critical market information. To ensure the most up-to-date information, multiple agents continuously poll target finance APIs and news websites at short intervals, extensively utilizing multi-threading and asynchronous message queues to minimize latency and ensure real-time delivery. The system also incorporates a dashboard user interface displaying stock graphs alongside the history of the generated news article summaries and impact analyses to provide users with an intuitive view of investment trends.

The project's objective is to help investors make more informed investment decisions by providing tailored news summaries related to their stock portfolios, ensuring they can stay informed without being overwhelmed by manually exploring vast amounts of information. Also, the project aims to enable investors to act promptly on potential investment opportunities or minimize losses through a real-time notification system.

The remainder of the report is structured as follows. First, Chapter 2 discusses the details of the methodologies adopted throughout the project. Next, Chapter 3 presents the key achievements and the potential future improvements for the project. Lastly, Chapter 4 provides a summary and conclusion of the work conducted.

2. Methodology

This section introduces the methodologies utilized in the project. It details the data source and APIs, system architecture, and various system components to achieve personalization and real-time news summary and analysis delivery.

2.1 Data Source and APIs

The project will utilize Refinitiv API [4] to retrieve stock information, including stock name, industry classification, and relevant keywords and tags describing the company and its activities. The decisive advantage of Refinitiv API, compared to other finance information providers, is that it provides endpoints to retrieve real-time news updates along with headlines and full content. A custom web scraper will be implemented to poll extra sources, such as the Yahoo Finance website as a supplementary data source, as Refinitiv API does not fully cover every financial news website.

2.2 System Architecture

The system will have various components (see Figure 1 below) to construct a personalized and real-time financial news notification system.

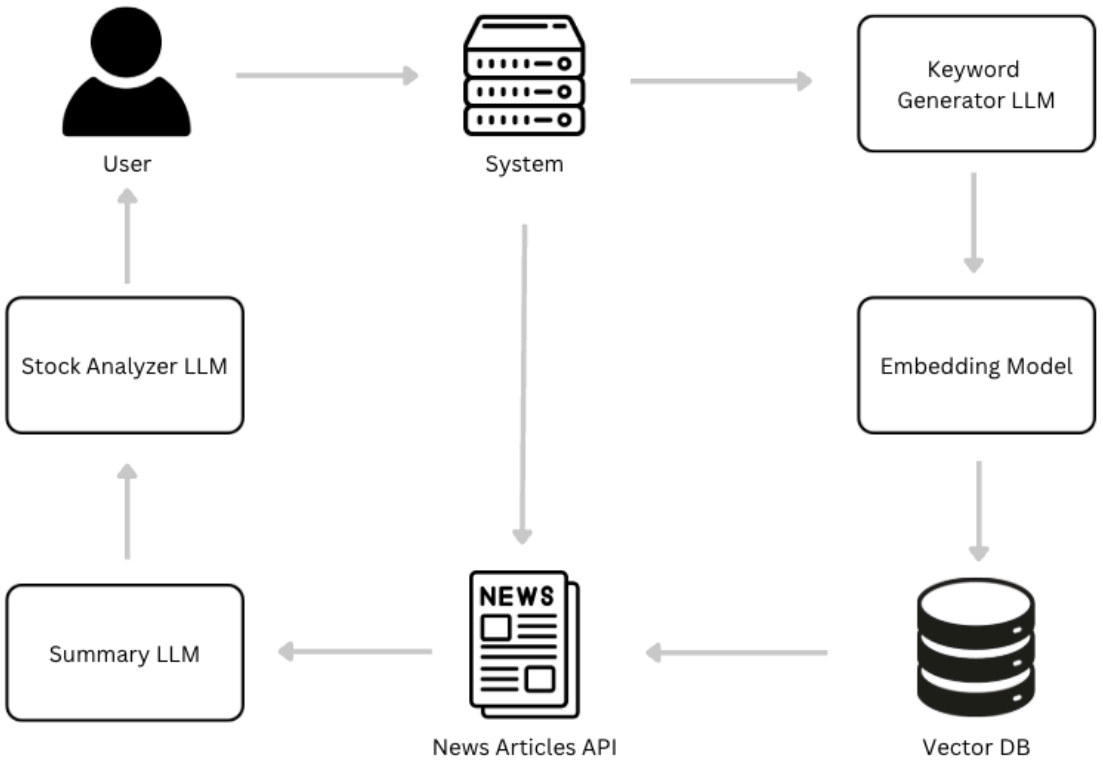

Figure 1. *System Architecture* - The system generates the embeddings from the user-provided stock information and stores them in the vector database. A polling agent extracts news articles, and LLMs analyze them to create personalized insights, delivering real-time updates to users through SNS channels.

The system's personalization begins with user input processing. Upon registration, users can provide stock names or custom keywords such as 'beverage industry' or 'GenAI,' which will be used for personalized news article selection. If stock names are provided, the system will utilize Refinitiv API to fetch stock information, including industry classification and keywords. Once the system has stock names or keywords, the Keyword Generator LLM will expand the data by generating additional related keywords and example sentences that reflect potential news narratives. The generated keywords and sentences will then be converted into embeddings and saved into a vector database. The vector database will be utilized for efficient embedding search against news articles to filter only the relevant ones.

To achieve real-time delivery, a polling agent, a background batch process running asynchronously from the main backend, will continuously poll Refinitiv API and financial news websites at short intervals (about 10 seconds) to extract news articles. The Document Analyzer will compare the embeddings of each article with the stored embeddings of user-specific keywords and example sentences. This process selects only the users whose stock data is relevant to the extracted article. Finally, the fine-tuned Stock Analyzer LLM will generate a summary and user-specific stock impact analysis and deliver them to users through SNS channels.

2.3 Authentication with JSON Web Token (JWT)

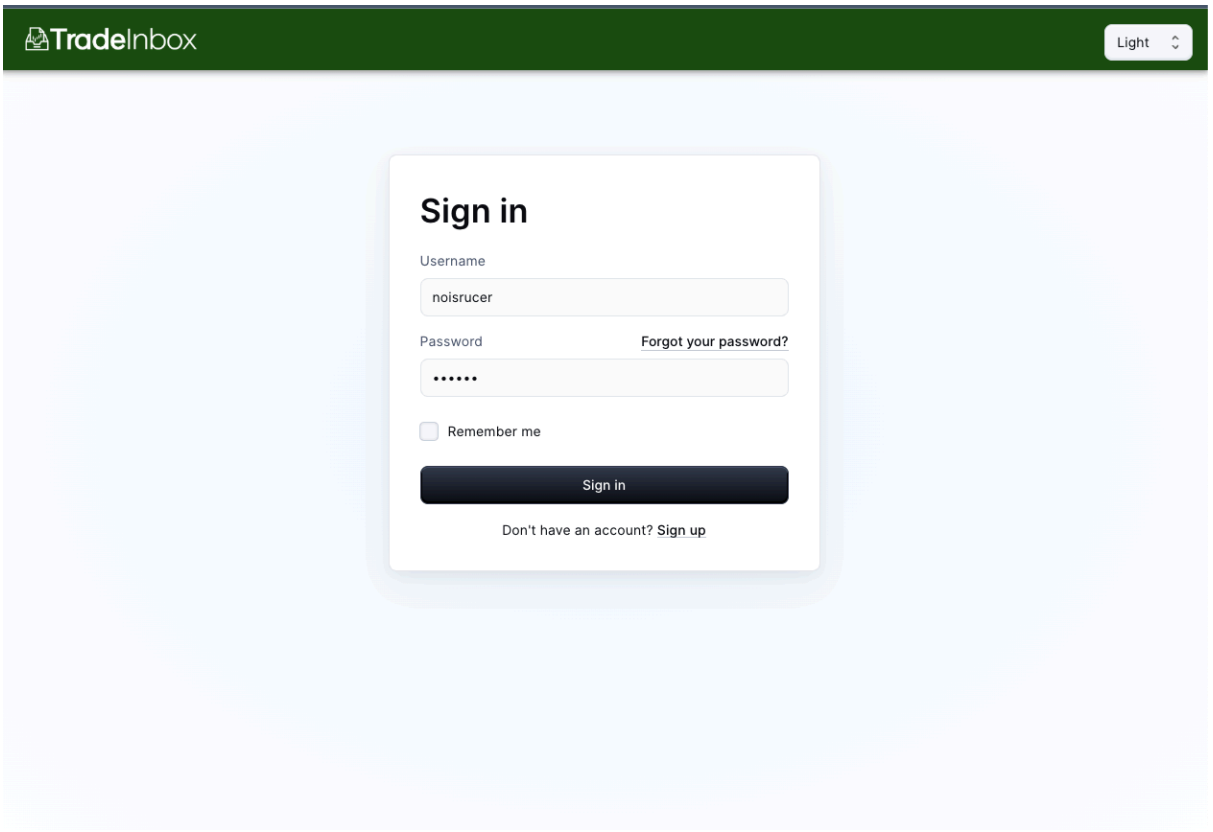

Figure 2. *Sign-In Page*

Strong authentication is crucial, especially because the service handles user personalized financial data. TradeInbox leverages JWT(JSON Web Token) as a primary authentication mechanism. JWT acts as a stateless access token consisting of a header, payload, and signature. A user signs in with a username and password, and the server responds with a JWT signed with the secret key. Then, in subsequent requests, the user attaches the access token to the ‘Authorization’ HTTP header for verification.

Our team chose JWT primarily due to its stateless characteristic in a distributed and scaled-out environment. The server is expected to adopt horizontal scale-out as the user traffic increases, and JWT allows each server to share only the secret key to seamlessly scale

out without maintaining a separate session storage such as Redis. Moreover, JWT token is self-contained, meaning it contains all necessary information about the user (i.e., user ID), reducing additional database queries.

2.4 Development Tools and Infrastructure

Figure 3. *Development Tools and Infrastructure*

The primary backend system of the project is implemented using FastAPI. FastAPI is a modern back-end framework written in Python, offering high performance and scalability. FastAPI is built on ASGI(Asynchronous Server Gateway Interface operating on a single-threaded model, providing high performance, especially in a network I/O heavy environment. This ability significantly saves resources and boosts the overall performance of the app, as TradeInbox features heavily rely on external API providers such as OpenAI and Zilliz Cloud. Another consideration was the language compatibility with libraries. The backend system heavily utilizes LLM APIs and libraries such as OpenAI, LangChain, and Milvus, and most of them best support Python as their primary SDK language.

The project's infrastructure will be hosted on Amazon Web Services (AWS) for its first-class scalability and reliability. Moreover, Milvus will be used as the vector database due to its superior performance metrics. It provides the highest 2,406 queries-per-second rates and offers the lowest computation latency(1 ms), outperforming alternatives such as Pinecone and Qdrant [6].

The front-end will be developed with React, chosen for its efficiency, reusability of components, and strong ecosystem for building dynamic web applications. React also provides a powerful component-based architecture, breaking down complex UIs into reusable components.

2.5 Features and UI & UX

This section provides an overview of the project features, user journey flow, as well as the descriptions of the user interface (UI).

2.5.1 Multi-Stock Input

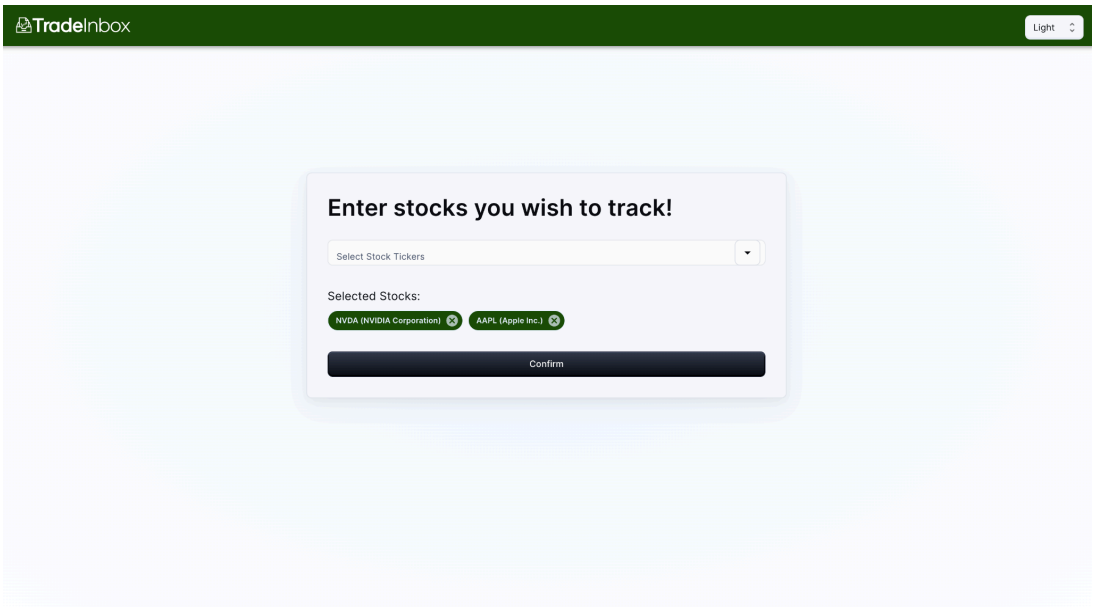

Figure 4. *Stock Input Page*

Upon a successful sign-in, user could provide a list of stocks to track. TradeInbox maintains a database of 4793 publicly-listed companies in NASDAQ. To simplify the search process, user can search by either a stock name or a stock ticker.

2.5.2 Dashboard

After successfully providing stocks, the user will be directed to the main dashboard, where each stock will have its own dedicated dashboard. The left portion of the dashboard shows a overview of stock information including the exchange, industry sector, current stock price, as well as previous close price. Additionally, the dashboard displays the past month's price history as well as the trading volume history.

The left side of the dashboard provides a personalized news curation. Once the user has provided the stocks, the system automatically scrapes and collects the news articles relevant to the stock and curate them with further analysis. Since there might be an excessive amount of the captured news articles, the app provides the 'Quick Summary' where the user can easily glimpse at what positive and negative events have happened recently related to the stock. The curation also provides a list of the captured news articles, offering the title, a short summary, and the sentiment analysis result such as 'Positive', 'Negative', or 'Strong Positive'. To allow users to quickly identify the strong indicators, the news articles with a sentiment 'Strong Positive' and 'Strong Negative' are highlighted in background.

2.5.3 Stock Impact Analysis Report

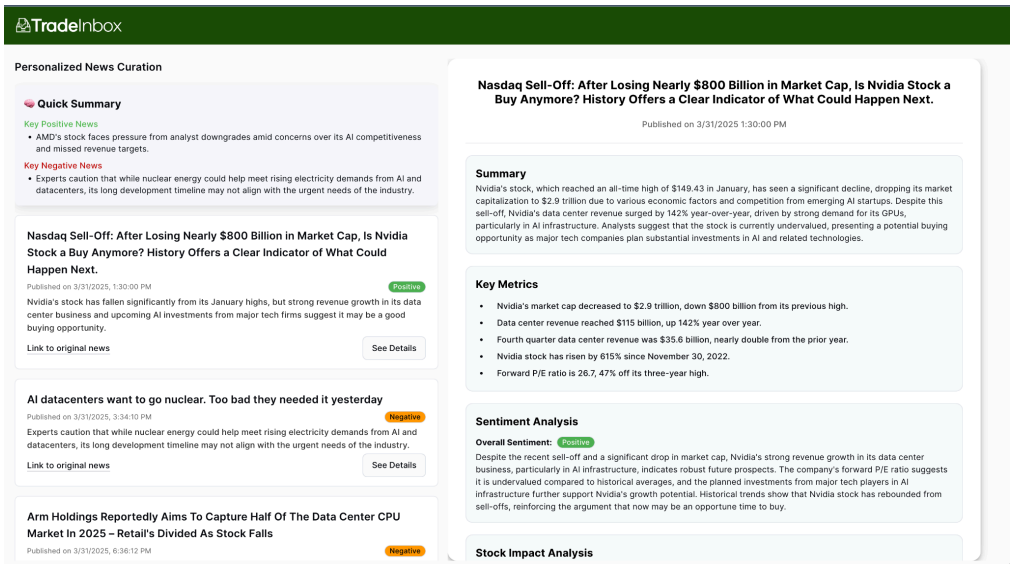

Figure 5. *Stock Analysis Report Page*

When clicking on a news article, the system shows a detailed stock analysis report including a summary, key metrics extracted from the article, sentiment analysis, and the stock impact analysis predicting how the article is likely to impact the stock price.

2.6 Embedding Search & Keyword Generator LLM

Embeddings are numerical representations of words or sentences in vector space, capturing the semantic relationships between text data [5]. Due to their ability to distinguish semantic nuances, embeddings are frequently utilized in embedding search, also known as semantic search, to determine the similarity between sentences or whole documents.

Upon receiving the stock from the user, the system will generate the embedding vectors from the stock name and store them in the Milvus vector database. Meanwhile, the polling agent continuously scrapes the news articles and converts the news title and the body into vector embeddings and performs embedding search against the user-provided stock embedding to calculate the cosine similarity score. If the similarity score exceeds a pre-defined threshold, then the article is considered a match and stored in the database. The cosine similarity score threshold was set to 0.4 in the system based on the experiments. The news articles that clearly have no relation to the stock tend to show an extremely low similarity score. For example, the news article ‘Spurs take on the Grizzlies on 3-game losing streak’ showed the score of 0.016 when comparing against the embedding of ‘NVIDIA Corporation’. Also, the system converts the whole news article body into a single embedding. Hence, our team assumed that due to the lengthy text, detailed semantics are diluted, resulting in a relatively lower similarity score. On the other hand, relevant articles such as ‘Nasdaq Sell-Off: After Losing Nearly $800 Billion in Market Cap, Is Nvidia Stock a Buy Anymore?’ consistently show above 0.4 similarity score.

Directly converting the stock name, such as ‘Nvidia’, into the vector embedding might work in scenarios where the stock name is explicitly stated in the news content. However, it suffers when a news article is indirectly related to the given stock, where neither the news title nor the content explicitly contains the stock name. An example is the news where the title is ‘Arm Holdings Reportedly Aims To Capture Half of the Data Center CPU Market in 2025 - Retail’s Divided As Stock Falls’ and the content doesn’t directly include the word ‘Nvidia.’ This article is reasonably related to Nvidia, as it describes that Arm aims to capture a significant share of the data center CPU market, particularly in the AI sector, which could pose a threat to Nvidia. However, simply calculating the similarity score from the stock name’s embedding gives only 0.26.

To tackle this challenge of capturing indirectly related news articles, the Keyword Generator LLM is devised. The Keyword Generator LLM will expand the user input to a richer dataset for a more robust embedding search. For instance, if a user is interested in Coca-Cola, the LLM might generate additional phrases like ‘beverage industry trends’ or ‘the introduction of sugar taxes has led to a decrease in demand for traditional sugary beverages,’ enabling a more thorough analysis.

Integrating with this approach, the system generates 10 example keywords from the user-provided stock name. The system then generates a vector embedding for each of the keywords as well as from the stock name itself, resulting in a total of 11 vector embeddings stored in the vector database. When the polling agent captures a fresh news article, the embedding search is performed against each of the stored keywords. Next, it will select the one with the maximum cosine similarity score and determine whether the article is relevant if the maximum score exceeds the threshold. This approach allows the system to obtain a diverse set of perspectives to determine relevancy. If the news article explicitly contains the

stock name, then the vector embedding extracted from the stock name is likely to be matched, while the indirectly related news articles will be captured by those of the generated keywords.

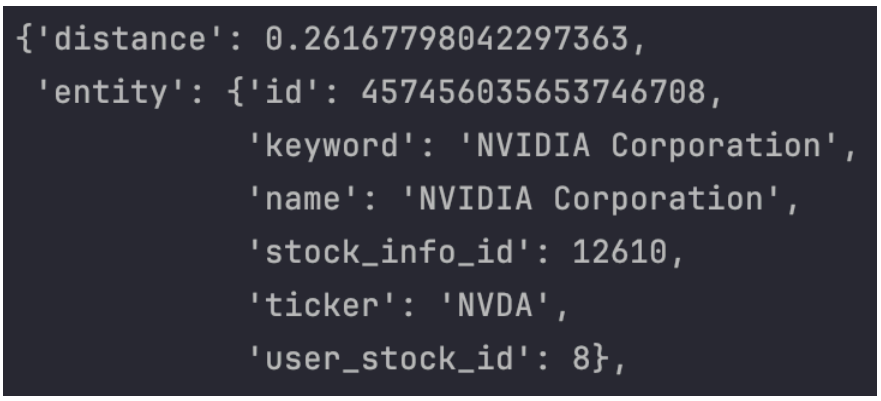

Figure 6. *Embedding Search Result 1*

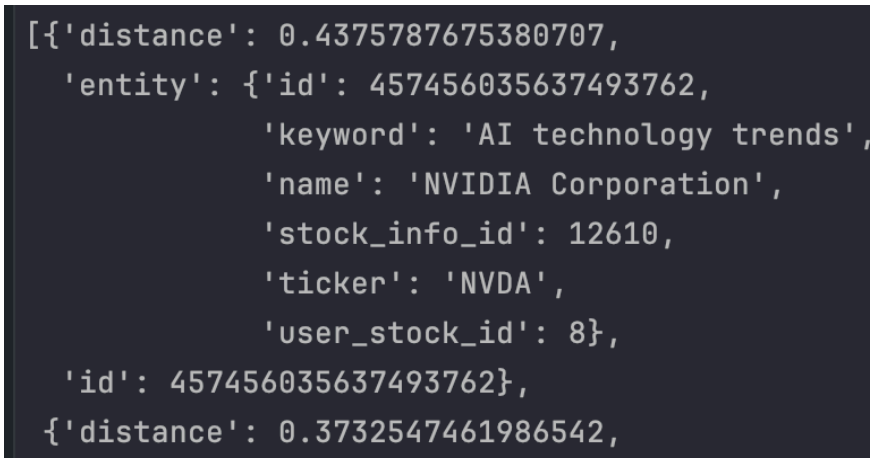

Figure 7. *Embedding Search Result 2*

The following experiments demonstrate the effectiveness of this strategy. With the 'Nvidia' as user input, the system generated keywords such as 'graphics processing units', 'AI technology trends', 'gaming industry growth', 'data center demand', and 'major competitor AMD'. The cosine similarity score between the stock name 'Nvidia' itself and the news article 'Arm Holdings Reportedly Aims To Capture Half of the Data Center CPU Market in 2025 - Retail's Divided As Stock Falls' was 0.26, failing to exceed the threshold of

0.4. On the other hand, one of the generated keywords, ‘AI technology trends,’ resulted in 0.44, which exceeds the threshold.

On the other hand, the news article ‘Nasdaq Sell-Off: After Losing Nearly $800 Billion in Market Cap, Is Nvidia Stock a Buy Anymore?’ is matched with the stock name ‘Nvidia’, showing a 0.52 similarity score. The experiments show that the system can capture both directly and indirectly related news articles, backed by the diverse generated keywords.

2.7 Polling Agent

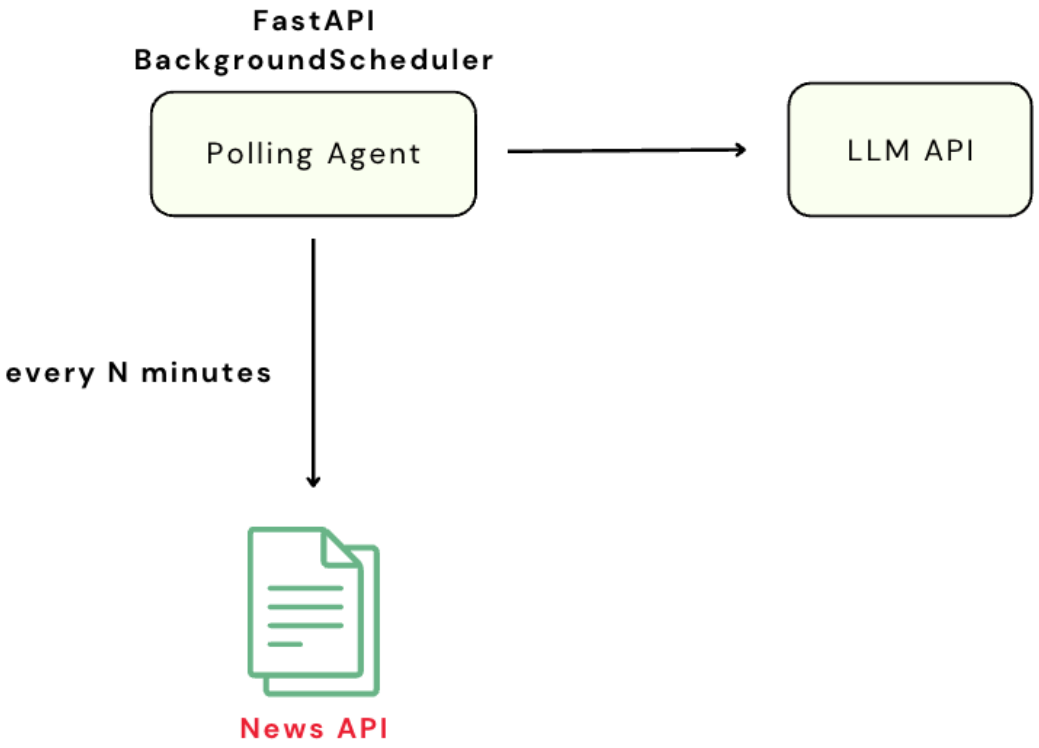

Figure 8. *Polling Agent*

The polling agent continuously polls every 10 minutes from the news API to fetch fresh news articles. The polling agent runs on the same backend server and is implemented using the APScheduler library’s BackgroundScheduler to implement recurring tasks without blocking the main thread. After the polling agent receives a news article, the agent performs a

series of tasks, including the embedding search, news summary generation, sentiment analysis, and stock impact analysis.

However, the above tasks leverage LLM APIs, which involve heavy network I/O operations. Unlike traditional APIs, whose latency usually spans from 50ms ~ 500ms, the external LLM APIs such as OpenAI take a considerable amount of time, often surpassing 5~10 seconds. If those tasks are executed serially, then the total latency of each news processing is expected to take more than 30 seconds, which is too long for real-time processing and unscalable when heavy traffic comes. To optimize, the system leverages Python's asynchronous programming capabilities through asyncio and aiohttp to enable concurrent processing of those heavy network I/O bound tasks. The below is a brief summary of the polling codebase.

```
async def start_polling(threshold=0.4):
    session = next(get_session())
    news_data = get_next_news_article()

    search_results = await client.search(
        collection_name="dummy_demo1",
        anns_field="vector",
        data=query_vectors,
        limit=500,
        output_fields=["id", "stock_info_id", "ticker", "name", "keyword", "user_s
        search_params={
            "metric_type": "COSINE",
            "params": {"nprobe": 10},
        }
    )

    results = filter_results(results, threshold)

    if not results:
        return

    # Run tasks in parallel
    summary_task = asyncio.create_task(generate_summary(news_data['summary']))
    metrics_task = asyncio.create_task(analyze_key_metrics(news_data['summary']))

    summary_dto, key_metrics = await asyncio.gather(summary_task, metrics_task)

    tasks = [
        process_stock_match(result, news_data, summary_dto, key_metrics, session)
        for result in results
    ]

    await asyncio.gather(*tasks)
```

Figure 9. *Polling Agent Asynchronous Processing*

The `asyncio.create_task()` is a non-blocking and asynchronous function that schedules a coroutine to run as an asynchronous task. The task gets scheduled for execution right away and does not block the caller, enabling multiple tasks to run in parallel. Moreover, FastAPI utilizes a single-threaded model to minimize thread usage by handling multiple network I/O operations with a single thread in parallel.

2.8 Feedback Loop

One of the important features of TradeInbox is personalization. The system provides personalization through the following features.

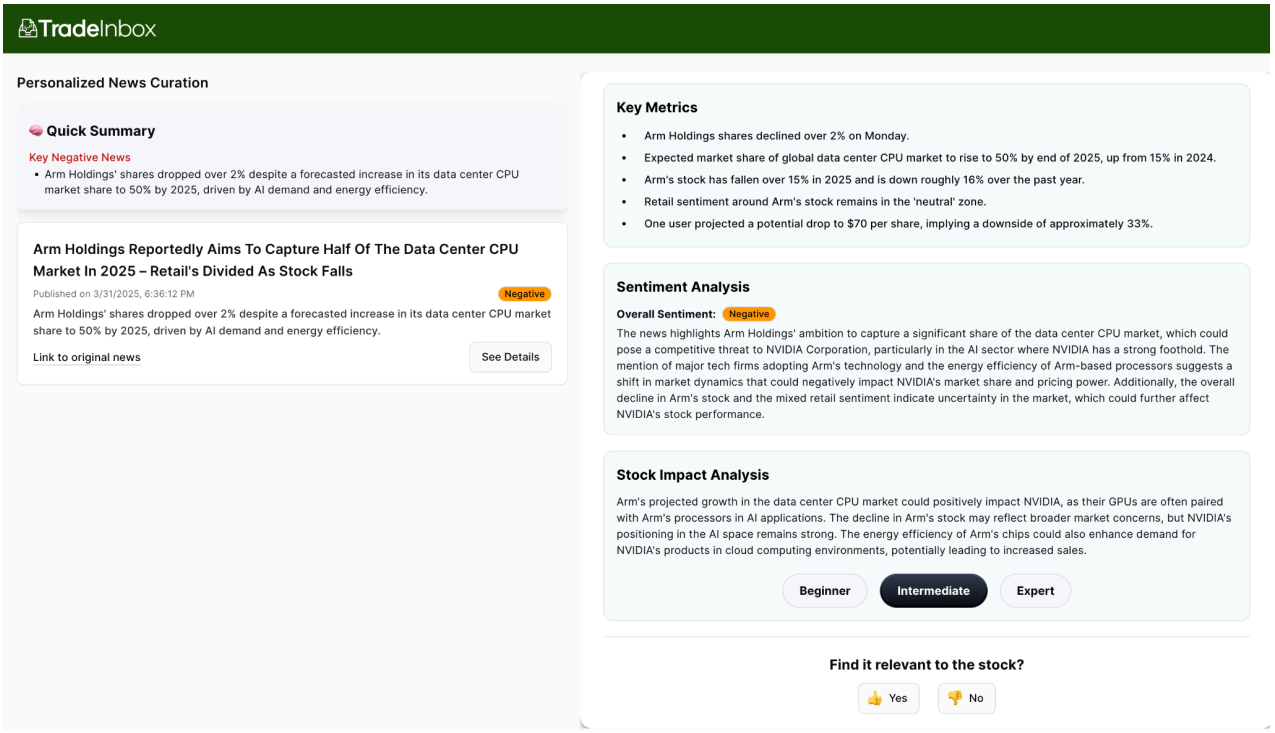

Figure 10. *Feedback Loop page*

The system captures news articles relevant to the stock provided by the user. Some of the captured articles are evidently related to the stock when the article explicitly describes the company. However, some articles do not directly mention the company name but are still reasonably related to the provided stock. For example, if a news article describes the

increasing GPU demands in the gaming industry or a strong competitor of Nvidia that is quickly taking over the GPU market, then it's reasonable to judge it relevant to Nvidia. However, one of the challenges to incorporate this relevancy is that the concept of relevancy is rather a subjective terminology than an objective one. For example, one can argue that a news article about the gaming industry is definitely related to Nvidia, while others can dispute that not all game companies rely on GPUs, as there are light-weight Indie games. In this conflicting situation, neither party is completely wrong. Hence, our team assumed that there does not exist an axiomatic metric that correctly determines the relevancy.

Instead, the system integrated a feedback loop that allows users to provide feedback on the news articles that appear on the dashboard, as the articles in the dashboard are already marked as relevant by the system. The feedback question is 'Find it relevant to the stock?' and users can respond with either 'Yes' or 'No'.

The first approach our team implemented for the feedback loop was adjusting the similarity score threshold for each user. If the user finds it relevant, the system remains the same. If the user responds to it as irrelevant, then the system increases the similarity score threshold by a small number, such as 0.5, to incorporate the user feedback. However, this approach had a limitation. Suppose the Keyword Generator LLM generated two keywords about Nvidia - 'gaming industry' and 'data center demands'. The 'gaming industry' keyword showed a 0.5 similarity score, and the 'data center demands' keyword showed a 0.45 similarity score. Hence, the news article is matched with the keyword 'gaming industry' with the maximum score. However, the user might find the 'gaming industry' tag irrelevant and 'data center demands' highly relevant. In such cases, simply raising the overall similarity score threshold is ineffective, as it doesn't allow the user to specifically reject the incorrect 'gaming industry' keyword while potentially keeping the relevant 'data center demands' one.

To tackle this limitation, our team devised an alternative feedback loop system that effectively incorporates the user feedback. If the user responds it as irrelevant, then the system will first look for the generated keyword that matched with the news article and remove the keyword from the Milvus vector database. In this way, the keyword will not be used in embedding search for incoming news articles, while keeping other keywords as intact. Additionally, the system will generate an additional example keyword to backfill the keyword pool.

2.9 Flexible LLM Prompt Engineering Code Design

The system heavily utilizes LLMs and stores prompts for each LLM task. To effectively manage multiple prompts and seamlessly add new prompts without code duplication, our team designed a scalable and maintainable custom LLM prompt executor following OOP(Object-Oriented Programming) principles.

2.9.1 LLM Client Implementation

```
@dataclass  8 usages  noisrucer
class ChatMessage:
    role: Literal["system", "user", "assistant"]
    content: str
```

Figure 11. *LLM Client ChatMessage Implementation*

The prompts are categorized into one of the ‘system’, ‘user’, and ‘assistant’ prompts and abstracted in a `ChatMessage` object.

```python
class OpenAIChatLLM:  12 usages  noisrucer *
    def __init__(self, api_key: str, model: str = "gpt-4o-mini", temperature: float = 0.0):
        self.async_client = AsyncOpenAI(api_key=api_key)
        self.sync_client = OpenAI(api_key=api_key)

        self.model_kwargs = {"model": model, "temperature": temperature}

    def predict_json(self, messages: list[ChatMessage]) -> dict:  5 usages  noisrucer
        extra_params = {}
        if self.model_kwargs["model"] == "gpt-4o-mini":
            extra_params["response_format"] = {"type": "json_object"}

        resp = self.sync_client.chat.completions.create(
            messages=[asdict(message) for message in messages], **self.model_kwargs
        )
        resp_content = resp.choices[0].message.content
        resp_content = self.response_to_dict(text=resp_content)
        return resp_content

    def response_to_dict(self, text: str) -> dict:  2 usages  noisrucer
        try:
            return json.loads(text)
        except json.JSONDecodeError as e:
            raise e
```

Figure 12. *LLM Client Implementation*

The OpenAIChatLLM class serves as a clean abstraction over the OpenAI API, providing easy-to-use interfaces. It offers `predict_json` method that accepts a list of `ChatMessage` and returns the LLM response as a JSON using an internal `response_to_dict` method. This class also offers `apredict_json` method, which offers an asynchronous processing to maximize performance.

2.9.2 Prompt Management System

One core feature is file-based prompt storage. The prompt management utilities in `utils.py` provide a flexible approach to handling prompts. The design allows developers to maintain LLM prompts solely in external text files, separating prompt content from

application logic. Also, a custom prompt parsing logic is implemented, using a content format with `[%role%]` delimiter markers to define message boundaries and roles, as well as `{{$placeholder}}` format to effectively insert parameters into the prompts.

```
[%system%]
You're a professional journalist specialized in ...

[%user%]
Content: Once upon a time, Jessy took on a journey ...

[%assistant%]
{
    'sentiment': 'Positive',
    'score': 0.92
}

[%user%]
{{$article_content}}
```

Figure 13. *Prompt Storage Example*

The above image shows a prompt example where `[%system%]` section indicates a system prompt. The `[%user%]` and `[%assistant%]` sections are utilized for few-shot learning. After the prompt is ready, a developer can easily integrate it with OpenAILLM class leveraging custom utility functions.

```python
def load_prompt_messages(prompt_path: str) -> list[ChatMessage]:  10 usages  noisrucer
    with open(prompt_path) as f:
        content = f.read().strip()

    parts = content.split("[%")
    messages = []

    for part in parts[1:]:
        split_part = part.split( sep: "%]",  maxsplit: 1)
        if len(split_part) == 2:
            role, message_content = split_part
            role = role.strip()
            message_content = message_content.strip()
            messages.append(ChatMessage(role=role, content=message_content))

    return messages


def fill_message_placeholders(messages: list[ChatMessage], placeholders: dict[str, str]) -> list[ChatMessage]:
    messages = [ChatMessage(role=message.role, content=message.content) for message in messages]

    for message in messages:
        for placeholder_name, value in placeholders.items():
            if "{{$%s}}" % placeholder_name in message.content:
                message.content = message.content.replace("{{$%s}}" % placeholder_name, str(value))

    return messages
```

Figure 14. *Prompt Management Utility Class*

`load_prompt_messages` utility function with the prompt file path as a parameter, parses the role delimiters and converts them to a list of `ChatMessage` objects. Then, a developer can pass this list to the `fill_message_placeholders` function with placeholder values to complete the prompt loading process. Finally, the processed list of `ChatMessage` objects will be passed to the `OpenAIChatLLM` object to obtain the LLM response. Utilizing the components, one can easily create a new LLM use case as below.

```python
class ExampleKeywordsGenerator:  4 usages  noisrucer
    def __init__(self, llm: OpenAIChatLLM):  noisrucer
        self.llm = llm
        self.msg = load_prompt_messages("src/shared/llm/prompts/generate_keyword.txt")

    def generate_keywords(self, stock_name: str) -> KeywordGenerationResultDto:  1 usage  noisrucer
        messages = fill_message_placeholders(self.msg, placeholders={
            'stock_name': stock_name})

        resp = self.llm.predict_json(messages)
        return KeywordGenerationResultDto(resp['keywords'])
```

Figure 15. *Prompt Executor Example*

This prompt management design allows seamless integration and scalability, enabling developers to rapidly build and deploy diverse LLM applications.

2.10 Stock Analysis Report

After a news article is successfully matched as relevant to the user-provided stock, it goes through a series of tasks to deliver a detailed report to the user. The report includes the news summary, key metrics, sentiment analysis, and stock impact analysis.

2.10.1 News Summary

After the polling agent captures a relevant news article, it generates a brief news summary by leveraging Large Language Models (LLMs).

```
[%system%]

You're a professional journalist and an expert in summarizing a news article
into a concise and high-quality summary.

Your Task: Generate a 3-4 sentences summary from the given news article
content from any topic. In addition, generate a short ONE-SENTENCE summary.



[IMPORTANT]

Note that the generated keywords and examples MUST NOT BE too general. IT
MUST BE RELATED TO the provided user input.

{

 "summary": "xxx",

 "one_sentence_summary": "yyy"

}



YOU MUST RETURN ONLY JSON. DO NOT INCLUDE ANYTHING ELSE IN YOUR RESPONSE.

News Article Content: {{$article_content}}
```

```
Your JSON Output:
```

Figure 16. *News Summary Prompt*

The above is the prompt used for the news summary generation. The prompt includes a system prompt emphasizing on providing a clear summary in regards of the user-provided stock.

2.10.2 Key Metrics

Another component of the stock analysis report is the Key Metrics. Experienced retail traders frequently analyze the stock impact with the factual data, such as Earnings Per Share(EPS), Return on Equity (ROE), P/E Ratio, or PEG Ratio. Although the news summary provides an overview glimpse of the news article, it does not fully deliver the list of facts stated in the news article. Hence, the Key Metrics section curates all the facts from the article, especially the quantitative metrics. This feature could help experienced traders to quickly grasp the current status of the market and promptly react to market events.

```
[%system%]

You're a professional financial data extraction assistant and an expert in
analyzing news articles and generating key financial metrics in the news
article. Focus on numerical values related to the company's financial
performance, stock movement, and market reactions.



Your Task: Extract key financial metrics from the provided financial news
article. Focus on numerical values and financial indicators relevant to the
company or market mentioned.



[IMPORTANT]

Note that the key metrics MUST NOT BE generated on your own. It must be
consolidated from the user input. Restrict your each response into one short
concise sentence. Return 1 to 5 key metrics, ensuring each is short and
concise while covering all critical financial indicators. If there's no key
metrics, return empty list in JSON format given.
```

```
// Few-shot learning examples hidden due to the length of the text.


Extraction Criteria:

1.  Company Performance: Revenue, Net Income, Earnings Per Share (EPS),
Year-over-Year (YoY) or Quarter-over-Quarter (QoQ) changes, Operating Profit,
Gross Margin, etc.

2.  Stock Market Metrics: Stock price changes (e.g., % increase or decrease),
pre-market or after-hours movement, analyst target price updates, trading
volume, market capitalization, etc.

3.  Financial Ratios: Price-to-Earnings (P/E) ratio, Debt-to-Equity (D/E)
ratio, Dividend Yield, Free Cash Flow, etc.

4.  Macroeconomic Indicators (if applicable): Interest rate impact, inflation
rate, GDP growth, unemployment rate.

[%user%]

YOU MUST RETURN ONLY JSON. DO NOT INCLUDE ANYTHING ELSE IN YOUR RESPONSE.

Return the key metrics as concise, well-formatted short sentences.


Article: {{$content}}

Your JSON Output:
```

Figure 17. *Key Metrics Prompt*

The above figure shows the prompt used to extract key metrics from the article. The system prompt emphasizes the definitive criteria of ‘key metrics’. A key metric is one of the following categories: company performance, such as EPS, YoY, QoQ; stock market metrics such as trading volume; financial ratios such as P/E ratio, D/E ratio, or Dividend Yield; macroeconomic indicators including GDP growth and unemployment rate. Additionally, the prompt follows one-shot learning, providing an example article and expected key metrics to make the model more aligned with the expected results.

2.10.3 Sentiment Analysis

A core part of the stock analysis report is the sentiment analysis, providing investors with valuable insights into market perception and emotional trends surrounding a stock. The sentiment analysis provides the overall sentiment as five levels of ‘Strongly Negative’, ‘Negative’, ‘Neutral’, ‘Positive ’, and ‘Strong Positive.’ Alongside the sentiment tag, the system provides a detailed explanation, rationalizing why the model tagged the news article as such. Lastly, the LLM also returns the ‘sentiment score’ ranging from -5 to 5, representing the positivity of the news article towards the stock price. This score is utilized in the ‘Quick Summary’ section in the dashboard to select only the most impactful news articles for users.

```
[%system%]

You're a professional financial expert who specializes in sentiment analysis
of a financial news article related to a provided stock name.



The sentiment analysis includes the following items.



1) One of [Strong Negative, Negative, Neutral, Positive, Strong Positive].

  - where strong negative means the given news article or the contents of the
news article are high indicators of the given stock's price will fall down in
a future.

  - where strong positive means the given news article or the contents of the
news article are high indicators of the given stock's price will go up in a
future

  - Neutral means that it's not reasonable or there's no significant
indicator of the stock price from the news article.

2) Insightful, factual, reasonable, logical, detailed analysis & rationale &
proofs & evidence of your claim from item (1).

  Do not merely give meaningless, verbose, no-depth, abstract reasons. Give
DEFINITIVE PROOFS OR EVIDENCE OR RATIONALE to your claim in a structured
manner.

3) Also, provider a "SENTIMENT SCORE" ranging from [-5 ~ 5] (inclusive) to
indicate how positive/negative the given news is.
```

```
  - 5 means Strong positive, and -5 means Strong Negative, and 0 means
Neutral.

  - Give it as a floating-point number.



Make sure that your sentiment analysis output is CONCISE AND INSIGHTFUL. It
should not be verbose and long. Must be around 3~4 sentences depending on the
situation.



In addition, you will be provided a specific stock name that you must analyze
the impact with.

Your sentiment analysis of the news article must be done in consideration of
this given stock name.



YOU MUST RETURN ONLY JSON. DO NOT INCLUDE ANYTHING ELSE IN YOUR RESPONSE.



You must return response in the following "JSON" format (only JSON)

{

   "sentiment": "xxx", // item (1) - either Strong_Negative, Negative,
Neutral, Positive, Strong_Positive

   "sentiment_score": xx,

   "analysis": "..." // item (2)

}



Given stock name: {{$stock_name}}

News Title: {{$title}}

News Content: {{$content}}



Your JSON Output:
```

Figure 18. *Sentiment Analysis Prompt*

The prompt includes clear definitions of positive and negative tags and emphasizes the factual, insightful, and logical rationale of the predicted sentiment.

2.10.4 Stock Impact Analysis

The last part of the report is the stock impact analysis, describing how the news article is likely to impact the stock price in the near future. While the sentiment analysis focuses on the emotional tone or attitude expressed in text, the stock impact analysis explicitly examines how the news might affect the price, trading volume, or market position.

```
[%system%]

You are a stock market news analysis agent that evaluates how news impacts
stock prices, catering to retail investors with three expertise levels (easy,
intermediate, expert).



The news could be discussing any kind of topic, but related to the stock. The
key task is to find an impact to stock price of this news, and possibly
provide a logical explaination behind the stock price prediction in English.



You will also be given a stock name. Create your stock impact analysis report
towards the given stock.



<Requirements>



1. Summary of the news is not compulsory, mainly discuss the implication to
the stock price and logical explanation behind it.

2. If the news has negligible impact to stock price then you can just give
some logical explanation of why it is not important.

3. For the analysis, please find below instructions for your reference. It
would be better if you can provide industry-specific viewpoints, as well.

4. **Easy**: Explain all industry-specific/technical/financial terms (e.g.,
P/E ratio, EBITDA) in simple language. Give full defintion of the terms below
the analysis. better
```

```
5. **Intermediate**: Assume basic financial knowledge; skip explanations for
common terms (e.g., dividends, market capitalization). Better to give some
explaination of complex technicals below the analysis.

6. **Expert**: Use advanced knowledge (e.g., discounted cash flow, beta
volatility and many others) and financial ratios without much explanations.



<Output Format>

Please return a JSON format like the following:



{

"easy": "..."

"intermediate": "..."

"expert": "..."

}



[%user%]

Example 1: Biotech FDA Approval



Stock: BioPharma Inc.



News Content:

BioPharma Inc. receives FDA approval for its Alzheimer’s drug, projects $1.2B
in peak annual sales, and sets a 12-month price target of $85. A short seller
report warns of trial data inconsistencies.



[%assistant%]

{

"easy": "BioPharma Inc. got approval from the FDA (U.S. drug regulators) for
its Alzheimer’s treatment, which it expects to generate $1.2 billion per
year. Analysts predict the stock could reach $85 within a year, but a
critical report claims some test results might be unreliable. Risks include
competition from larger drugmakers and high R&D costs (money spent developing
new drugs).",

"intermediate": "BioPharma’s FDA approval supports a bullish $85 PT (35x
P/E), but a $300M per quarter cash burn raises dilution risks. Pipeline
catalysts include a Parkinson’s drug entering Phase 2 trials.",
```

```
"expert": "BioPharma's Alzheimer's drug approval (FDA label includes broad
indication) drives PT to $85 (DCF: WACC 12%, $1.2B peak sales, 55%
probability-adjusted). Short seller claims on trial data heterogeneity
(p=0.07 in subgroup) may limit near-term upside. With a cash runway of six
quarters at the current burn rate, an equity offering (15-20% dilution) is
likely. EV/sales at 5x vs. sector 7x reflects pipeline overhang."

}

[%user%]



YOU MUST RETURN ONLY JSON. DO NOT INCLUDE ANYTHING ELSE IN YOUR RESPONSE.



Stock: {{$stock}}

News Content: {{$content}}



Your JSON Output:
```

Figure 19. *Stock Analysis Prompt*

The prompt includes instructions to provide the analysis in terms of three difficulty levels. In the 'Easy' level, the analysis removes jargon and industry-specific vocabulary such as P/E ratio or EBITDA. The 'Intermediate' level assumes users possess basic financial knowledge and skip explanations for common terms such as dividends and market capitalization. The 'Expert' level assumes the full knowledge and include every jargon and industry specific vocabulary. This feature enables users from different financial backgrounds to easily navigate the report to make informed investment decisions.

2.11 Stock Analysis Report Notification

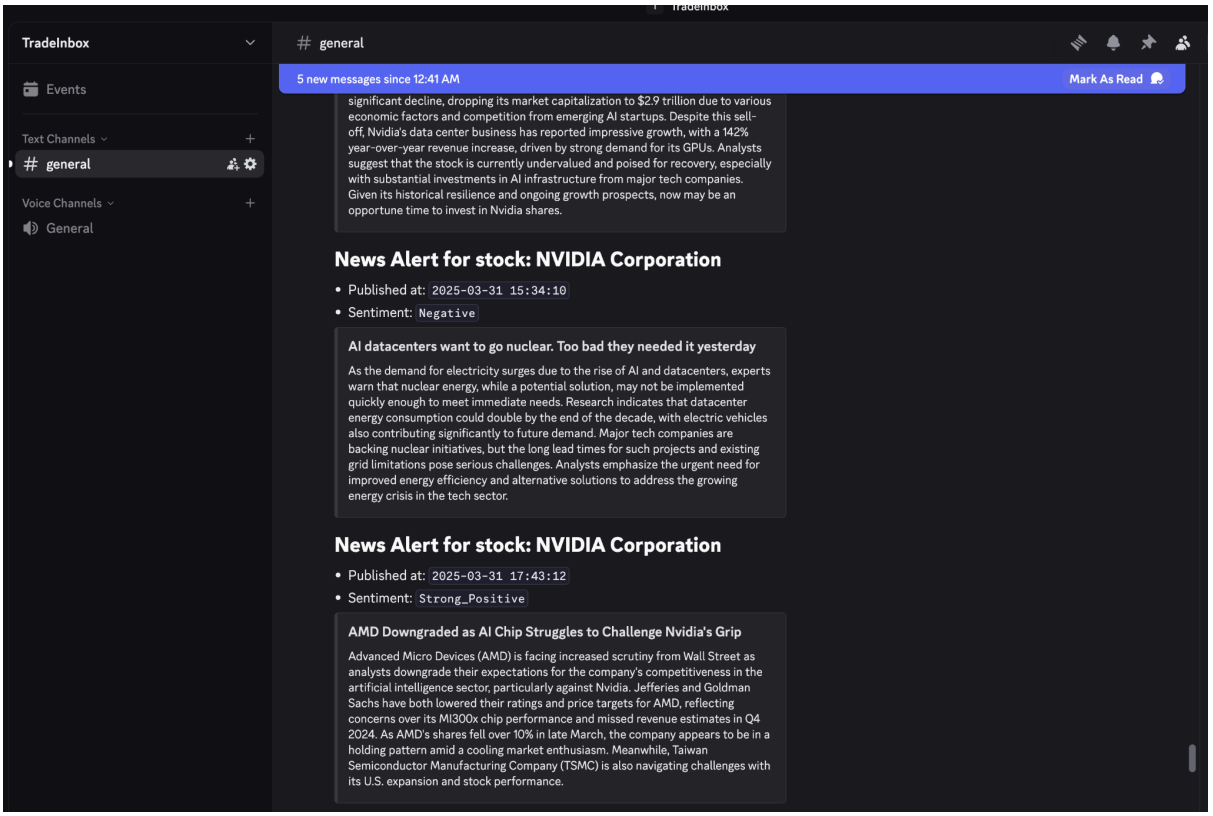

Figure 20. *Discord Stock Analysis Report Notification*

The system delivers the summary of the generated stock analysis report through Discord, providing users with timely and accessible financial insights. The notification system leverages Discord's robust Webhook API. After the polling agent successfully geneates the report, it immediately sends HTTP request to the Discord Webhook server which then delivers the report to the pre-selected Discord channel. The delivered summary includes the news title, sentiment, published datetime, and the news summary.

3. Results and Discussion

This section provides an overview of the accomplishments achieved, as well as the difficulties encountered and the possible resolutions.

3.1 Example Keyword Generator LLM

One notable achievement to date is the successful implementation of the example keywords generator LLM. This module expands the user input data by generating additional keywords and sentences that are highly relevant to the given input (see figure below), enriching the input for embedding search and thereby improving its performance and relevance.

For instance, when a user provides a stock name “Nvidia,” the generator produces 5-10 relevant keywords and sentences, including examples such as “AI technology trends,” “Data center demand,” and “CEO Jensen Huang statements”. These outputs are crafted to reflect the language and topics commonly associated with financial news articles about “Nvidia.”

```
Stock Name: NVIDIA Corporation
Example Keywords: [
    'graphics processing units',
    'AI technology trends',
    'gaming industry growth',
    'data center demand',
    'autonomous vehicle partnerships',
    'semiconductor supply chain',
    'cloud computing expansion',
    'major competitor AMD',
    'machine learning applications',
    'CEO Jensen Huang statements'
]

Stock Name: Apple Inc.
Example Keywords: [
    'iPhone sales',
    'Apple Watch market',
    'MacBook performance',
    'App Store revenue',
    'smartphone competition',
    'supply chain disruptions',
    '5G technology impact',
    'CEO Tim Cook statements',
    'consumer electronics trends',
    'global chip shortage'
]
```

Figure 21. *Example Keyword Generator LLM Output* - The keyword generator produces a variety of relevant keywords and sentences to enrich the user input.

Notably, the generated examples strongly suggest that the model has the capability to capture both direct and indirect relevancy. While keywords such as “graphics processing units” directly relate to Nvidia’s core GPU business, the model also identifies broader and indirectly related topics, such as “gaming industry growth” and “autonomous vehicle partnerships,” that heavily leverage AI chip technology. This ability to generate versatile and contextually aligned examples could significantly enhance the embedding search process by enabling more precise article filtering, ensuring that highly relevant and diverse articles are identified and analyzed for summary generation.

News Article Title	Matched Keyword	Similarity Score
Arm Holdings Reportedly Aims To Capture Half Of The Data Center CPU Market In 2025 – Retail's Divided As Stock Falls	AI technology trends	0.437
Nasdaq Sell-Off: After Losing Nearly $800 Billion in Market Cap, Is Nvidia Stock a Buy Anymore? History Offers a Clear Indicator of What Could Happen Next.	NVIDIA Corporation	0.521
AI datacenters want to go nuclear. Too bad they needed it yesterday	data center demand	0.494
AMD Downgraded as AI Chip Struggles to Challenge Nvidia's Grip	major competitor AMD	0.616

Table 1. *Keyword Generator LLM Result*

The above table shows examples of matched news article for ‘Nvidia’ as well as its matched keyword and similarity score. It’s observed that each a diverse set of keywords

successfully capture various news articles, demonstrating the effectiveness of the keyword generation embedding search approach.

3.2 Stock Dashboard UI

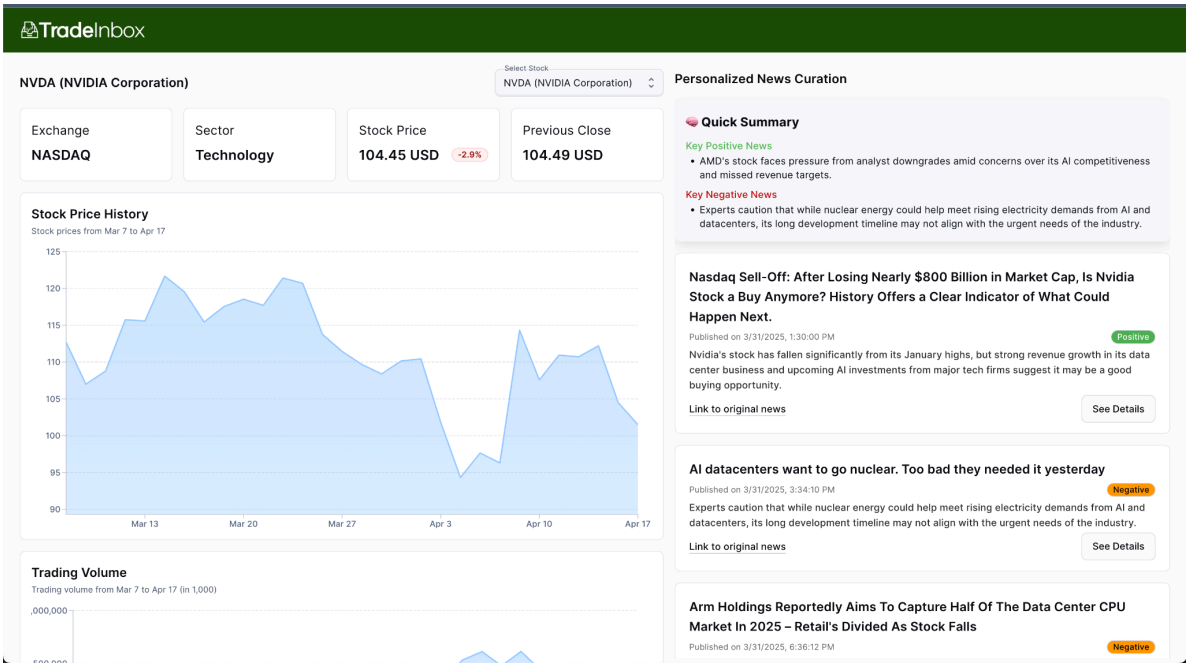

Figure 22. *Stock Dashboard Page*

The stock dashboard displays a variety of stock information including the current stock price, previous close price, and stock price history. Also, the dashboard displays the "Quick Summary" where the system aggregates the recent news articles and selects the highest similarity score for each positive and negative to allow users to quickly grasp the recent trend. Lastly, a list of relevant news articles along with a one-sentence summary and sentiment are shown at the right panel.

3.3 Stock Analysis Report UI

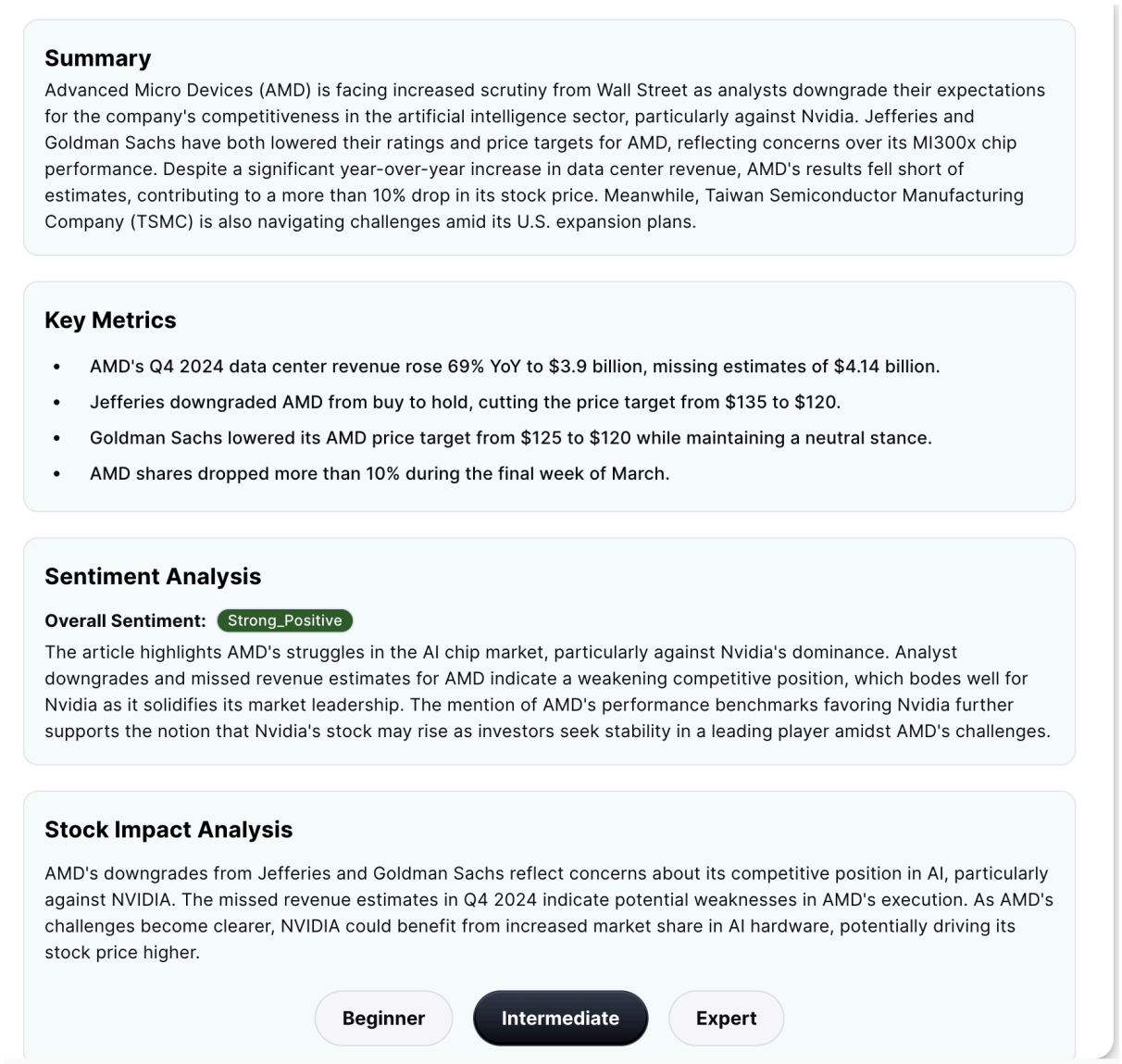

Figure 23. *Stock Analysis Report Page*

The above figure displays the components of the stock analysis report. The news summary provides a brief overview of the article. The Key Metrics section displays a list of quantitative fact data extracted from the text. The system also provides the sentiment analysis describing the emotional tone and attitude surrounding the article. Lastly, the stock impact analysis is provided describing how the news article is likely to impact the stock price in the near future.

3.4 Difficulties and Possible Resolutions

One significant difficulty encountered is the detection of the newly updated articles from web-scraped data. Since the scraping agent continuously scrapes a news website at short intervals of 1 to 5 seconds, there may be instances where no new articles are published within that time frame. As a result, the agent must efficiently determine whether a scraped article is new or has already been processed, to avoid redundant processing and waste of resources.

To address this issue, an improved scraping mechanism is proposed. Rather than scraping the full content of each article at every interval, the system first retrieves only the published date and time of the article. If the date and time indicate that it's indeed new, then the agent proceeds to process its main content. The scraping agent also tracks the latest scraped published date and time. This allows it to compare the timestamp of newly scraped articles: if the published date and time are equal to or earlier than the stored latest timestamp, the agent can safely skip processing the article further.

This optimized approach could reduce unnecessary data processing and ensure that the system focuses only on genuinely new content, enhancing efficiency in handling real-time article updates.

5. Conclusion

The primary objective of this project is to streamline the financial news processing for retail investors to mitigate the risks of delayed news processing and aid better investment decisions. This paper described the implementation of the LLM-based real-time financial news notification system, to help retail investors make informed decisions by delivering relevant news quickly and efficiently. The significance of this system lies in its potential

ability to bridge the gap between processing raw financial data and generating actionable insights for retail investors. By automating the identification, analysis, and delivery of relevant news, the system could save time and reduce the burden of manual research for investors. It is likely to enhance the decision-making process with predictive analysis performed by the system's LLMs, particularly for retail investors who lack the time or expertise to perform such analyses by themselves.

The project’s result highlights the significance of its potential to transform how investors interact with financial news by providing personalized information updates. The successful implementation of the example generator LLM, which produces highly pertinent example keywords and sentences tailored to user inputs, has been a critical step to achieving personalization. These outputs integrate with the embedding search module, a core component to filter out only the most relevant news articles, ensuring effective personalization of financial content.

Nevertheless, further refinement in the Keyword Generator LLM model via fine-tuning is required to improve the relevancy of keywords generated to the user inputs and the accuracy of the relevant news article selected for that user for personalization based on quantitiative evaluation process. Therefore, future works include fine-tuning the Keyword Generator LLM with a manually curated dataset and implementing the embedding search model to quantitatively evaluate the relevance between user inputs and the actual financial news chosen. These steps are crucial for improving the system's accuracy and achieving reliable personalization for the user. The successful implementation of these steps will keep significant progress toward achieving the project's objectives in personalization and addressing the needs of retail investors.

Moreover, as the system currently relies on a limited news dataset, it's difficult to generalize the system's performance in a more real-world setting with diverse streams of news websites such as New York Times, Bloomberg, and Wall Street Journal. Hence, an important future work would be constructing a pipeline to ingest from diverse sources of news articles to better reflect the real-world settings.

References

[1] T. Wang, "FTX exchange halts all crypto withdrawals," coindesk.com, 2022. [Online]. Available: https://www.coindesk.com/business/2022/11/08/ftx-exchange-halts-all-crypto-withdrawals. [Accessed: Oct. 15, 2024].

[2] "Who Reads Finance News? Traffic and User Behavior," fintext.io. [Online]. Available: https://www.fintext.io/case-studies/benchmarking/who-reads-financial-news-web-traffic-and-user-behaviour. [Accessed: Oct. 15, 2024].

[3] "Yahoo Finance," finance.yahoo.com. [Online]. Available: https://finance.yahoo.com. [Accessed: Oct. 15, 2024].

[4] "Refinitiv Data Platforms APIs," developers.lseg.com. [Online]. Available: https://developers.lseg.com/en/api-catalog/refinitiv-data-platform/refinitiv-data-platform-apis . [Accessed: Oct. 15, 2024].

[5] J.-T. Huang et al., "Embedding-based Retrieval in Facebook Search," in *Proc. 26th ACM SIGKDD Int. Conf. Knowledge Discovery & Data Mining (KDD '20)*, pp. 2553–2561, Aug. 2020. doi: 10.1145/3394486.3403305.

[6] "Picking a vector database: a comparison and guide for 2023," benchmark.vectorview.ai. [Online]. Available: https://benchmark.vectorview.ai/vectordbs.html. [Accessed: Oct. 15, 2024].