PATHWAY ENDTERM REPORT

Team 84

ABSTRACT

This report presents our significant progress in developing a Multi-Agent Retrieval-Augmented Generation (MARAG) system designed for efficient financial data processing. Our architecture uniquely balances retrieval and contextual understanding with modules optimized for dynamic data changes, addressing key challenges in financial data processing. We introduce a hybrid RAG model that integrates LightRAG and GraphRAG for enhanced data access, complemented by FinGPT, a language model fine-tuned for financial terminology. Our system features advanced schedulers, guardrails, and fallback mechanisms to ensure robustness and reliability.

Notable achievements include the development of a novel TableVision mode for the Unstructured parser, which significantly improves table retrieval accuracy. We also implemented a Supervisor based on the LLM Compiler, enhancing dynamic task management and human-in-the-loop (HITL) capabilities. The Financial Analyst Group, Math Tool, and Report Generation Tool further augment the system's analytical prowess. Our Responsible AI framework ensures ethical and reliable performance through sophisticated input guardrails and content validation.

Initial tests on the challenging FinanceBench dataset demonstrate enhanced data retrieval and processing accuracy, validating the system's potential for reliable financial data analysis. This comprehensive MARAG system represents a significant advancement in financial technology, offering a robust and automated solution for real-time data analysis and decision-making.

Keywords: *financial data; RAG; FinGPT; LightRAG; multi-agent system; TableVision; LLM Compiler; Responsible AI.*

1. INTRODUCTION

The financial sector is increasingly relying on advanced technologies to manage and interpret complex data. Traditional methods often fall short in handling the intricacies and volume of financial data, necessitating more sophisticated systems. In response, we have developed a Multi-Agent Retrieval-Augmented Generation (MARAG) system designed to handle complex financial data processing tasks.

Our Multi-Agentic RAG system integrates retrieval, reasoning, and generation capabilities, leveraging state-of-the-art techniques such as specialized language models (e.g., FinGPT) and adaptive retrieval mechanisms. This system addresses key challenges in financial data processing, including complex data handling, dynamic adaptability, robust reasoning, and responsible AI practices.

2. USE CASE

Our Multi-Agent Retrieval-Augmented Generation (MARAG) system is designed to address complex financial data processing tasks. Below, we present a specific use case that demonstrates the system's capabilities in a real-world scenario.

2.1. Scenario: Financial Report Analysis

2.1.1. Background

A mid-sized investment firm requires a comprehensive analysis of a publicly traded company's financial health. The firm needs detailed, accurate, and timely insights to make informed investment decisions.

2.1.2. System Workflow

1. **User Query Input:** The firm submits a query to the MARAG system, requesting an analysis of the company's financial health. The query includes specific details such as the company's name, financial statements, and relevant market data.
2. **Guardrails and Query Processing:** The system applies guardrails to ensure the query is safe and appropriate. The query is then processed by the Supervisor, which decomposes it into manageable tasks.
3. **Agent Activation:** The Supervisor activates specialized agents, including:
    - **Financial Analyst:** Examines financial statements, market trends, and company performance.
    - **Risk Assessment Analyst:** Identifies and quantifies risks associated with the company.
    - **Market Sentiment Analyst:** Assesses market trends and investor sentiment.
4. **Data Retrieval:** The system retrieves relevant data from internal document repositories and external sources using the Adaptive RAG framework. The retrieved data includes financial statements, market reports, and historical data.
5. **Analysis and Reasoning:** The activated agents perform detailed analysis and reasoning on the retrieved data. The Financial Analyst examines the financial statements, the Risk Assessment Analyst identifies potential risks, and the Market Sentiment Analyst assesses market trends.
6. **Report Generation:** The system synthesizes the analysis results into a comprehensive financial report. The report includes detailed financial metrics, risk assessments, market trends, and actionable recommendations.
7. **Human-in-the-Loop (HITL):** If the system encounters ambiguous or complex queries, it transitions to the HITL mechanism. The system prompts the user for additional input or clarification, ensuring accurate and reliable results.

8. **Output and Delivery:** The final report is delivered to the investment firm in a professional format, including data visualizations and charts. The report provides actionable insights to support informed investment decisions.

2.1.3. Outcome

The MARAG system successfully generates a comprehensive financial report that meets the investment firm's requirements. The report provides detailed insights into the company's financial health, risk factors, and market trends, enabling the firm to make informed investment decisions. The system's robust architecture, specialized agents, and adaptive retrieval mechanisms ensure accurate and timely analysis, demonstrating the system's effectiveness in real-world financial data processing tasks.

3. RELATED WORKS

High-quality Retrieval-Augmented Generation (RAG) can be the distinguishing factor between an AI system that provides valuable insights and one that fails to understand context. Recognizing the critical importance of RAG, we have focused extensively on this aspect in our framework.

Our framework builds upon several existing works, addressing their limitations to enhance robustness, efficiency, and accuracy in financial data processing and analysis. Below, we review these works and highlight their key features and limitations.

3.1. FinRobot

- **Key Features:** FinRobot is an AI agent platform for financial applications. Its framework is divided into four distinct layers, each targeting specific aspects of financial AI processing and application. The workflow incorporates modules for perception, brain processing using large language models (LLMs) and chain-of-thought (CoT) reasoning, and action execution. Additionally, a scheduler is employed for model selection and task allocation, with components for task direction, agent registration, adaptation, and management.
- **Limitations:** FinRobot lacks parallel execution capabilities, resulting in slower performance. Furthermore, it does not include task replanning when incorrect responses are generated, requiring manual retries by the user. Our framework addresses these issues with dynamic task replanning and optimized parallel execution.

3.2. Open Parse

- **Key Features:** Open Parse is designed to chunk complex documents for RAG systems, adopting a visually-driven approach with markdown support. It offers several modes for parsing tables and document extraction, including Unitable, GPT, PyMuPDF, and Table Transformer.
- **Limitations:**
  - **PyMuPDF:** While fast at processing PDFs, its chunking strategy fails to associate tables with their relevant contexts during retrieval.
  - **GPT:** GPT occasionally generates hallucinations when it fails to recognize text accurately, particularly problematic when extracting numerical data from tables.
  - **Unitable:** This deep learning-based table detection and extraction method is computationally expensive. Open Parse documentation acknowledges its limitations in table detection and cropping accuracy.
  - **Table Transformer:** This method is also computationally expensive and lacks a robust strategy for processing non-tabular text. Its bounding-box strategy often fails to capture table contexts, leading to poor results during testing.

3.3. Unstructured

- **Key Features:** The Unstructured library provides open-source tools for document ingestion and pre-processing. It includes partitioning modes such as `hi_res` and `fast`. The `fast` mode processes text as quickly as PyMuPDF but does not support table recognition.
- **Limitations:** Modes like `element`, `paged`, and `single` have notable drawbacks. The `element` mode creates chunks that are too small, the `single` mode captures excessive information, and the `paged` mode fails due to flawed logic.
- **Enhancements in Our Framework:** We extend unstructured data processing by introducing a custom mode, which improves query retrieval performance.

3.4. FinVerse

- **Key Features:** FinVerse is an autonomous agent system tailored for financial analysis, integrating over 600 financial APIs to provide extensive and accurate data. It features an embedded code interpreter for complex data analysis tasks, supported by a pipeline that includes task scheduling, API selection, code execution, and reflection for continuous adaptation.
- **Limitations:** FinVerse lacks dynamic replanning and human-in-the-loop capabilities, reducing its adaptability. Moreover, having numerous data sources without sufficient reasoning mechanisms does not significantly improve performance.

3.5. Multi-Agent Collaboration

- **Key Features:** This framework employs a multi-agent system where intelligent generative agents (IGAs) collaborate to handle complex tasks. Each agent has unique attributes and roles, enhancing adaptability and versatility. The framework also supports dynamic addition of agents for flexible workload management and parallel processing.
- **Limitations:** It does not include a built-in replanner, reducing the correctness probability of outputs on the first attempt. Users must often refine queries manually. Additionally, it lacks parallel code execution, unlike our enhanced framework.

3.6. MathChat

- **Key Features:** MathChat focuses on synthetic dialogue-based math datasets for fine-tuning LLMs. It aims to improve multi-turn mathematical reasoning abilities and conversational interaction in real-world applications.
- **Limitations:** It lacks dynamic replanning and parallel code execution, resulting in slower and less flexible task handling.

- **Enhancements in Our Framework:** Our framework addresses these limitations by incorporating dynamic replanning and parallel execution, enabling faster and more efficient task management.

4. OVERVIEW OF ARCHITECTURE

Our multi-agentic framework, built on LangGraph, is designed to specialize in financial Question and Answer (Q&A) tasks and comprehensive analysis of 10-K financial reports. It also generates detailed financial reports, complete with data-driven charts, providing actionable insights to users. Below, we describe the architectural components and their roles:

4.1. User Interaction and Guardrails

- **User Query Input:** Users interact with the framework by providing a query that initiates the processing pipeline.
- **Guardrails Integration:** Guardrails are applied to block unethical, jail-breaking, or inappropriate queries. Instead of outright rejection, queries are modified to align with ethical standards, and users are cautioned to refrain from such inputs. This ensures the system operates responsibly while maintaining user engagement.

4.2. Brain: The Supervisor

At the core of the architecture lies the Supervisor, a dynamic task manager inspired by the LLM Compiler paper. It orchestrates the system's operations by:

- **Planning and Scheduling:** Tasks are decomposed into a Directed Acyclic Graph (DAG) of tool calls. Independent tasks are executed first, with results propagated to dependent tasks.
- **Execution and Replanning:** Results from tools are aggregated, validated, and, if necessary, used to replan tasks. The Supervisor ensures the system adapts dynamically based on tool outputs and potential user feedback.
- **Human-in-the-Loop:** For ambiguous queries or unexpected issues, the system prompts users for clarification or input, enabling seamless collaboration.

4.3. Core Tools

The Supervisor coordinates the following specialized tools to accomplish various tasks:

- **Data Retrieval Tool:**
  - **Adaptive RAG Framework:** Queries from the Supervisor are routed to either internal document repositories or external sources via a query router.
  - **Pathway Query Handling:** If internal documents are used, the query is enhanced using a query reconstructor. Relevant document chunks are retrieved and graded for relevance. If relevance fails twice, the query is escalated to external web search.
  - The final retrieved data is returned to the Supervisor for integration into the task.
- **Financial Analysis Group:**
  - A hierarchical team comprising a leader and two analysts, dynamically assigned roles by the Supervisor.
  - Analysts collaborate on tasks such as Risk, Fundamental, or Sentiment Analysis.
  - The leader oversees the process and compiles results for submission to the Supervisor.
- **Math Tool:**
  - Equipped for advanced mathematical reasoning and computations.
  - Powered by the LLM Compiler, it leverages code-writing capabilities for execution and parallel processing, significantly enhancing inference time.
  - Final results are returned to the Supervisor for further use.
- **Report Generation Tool:**
  - This tool synthesizes financial data and insights into comprehensive PDF reports.
  - It includes financial analysis, data visualizations, and charts, ensuring professional-grade outputs for the user.

4.4. Workflow Overview

- **Query Flow:** User queries pass through guardrails, followed by the Supervisor, which orchestrates the tasks.
- **Dynamic Execution:** The Supervisor dynamically calls tools, integrates outputs, and replans as necessary.
- **Result Compilation:** Final outputs, including financial analyses or reports, are presented to the user.

4.5. Responsible AI Implementation

To ensure ethical and reliable performance, the framework incorporates guardrails and quality checks at every stage of the pipeline. This ensures adherence to responsible AI principles and minimizes risks of misuse or incorrect outputs.

5. SYSTEM ARCHITECTURE

5.1. Dynamic Retrieval

To address the dynamic nature of document retrieval, we integrated **Pathway's Drive Connector**, which enables real-time synchronization with Google Drive. Any updates to documents in the drive are reflected immediately, as the connector updates the index in memory dynamically. This eliminates the need for manual reindexing and ensures that retrieval operations always leverage the latest data.

The indexed data is processed using **Pathway's VectorStoreServer** with **OpenAIEmbedder**, which generates high-quality embeddings for vector-based search. This combination allows for efficient and precise information retrieval, particularly for complex financial queries.

5.1.1. Testing available retrievers

We evaluated multiple retrievers available within Pathway, as discussed in section on related works. Each retriever had some limitations to it. Among these, the **Unstructured** retriever offered the best overall performance.

The Unstructured retriever provides three operational modes:

- **Single Mode:** Parses each document as a single long text string.
- **Element Mode:** Splits the document into unstructured elements for fine-grained retrieval.
- **Paged Mode:** Extracts text on a page-by-page basis.

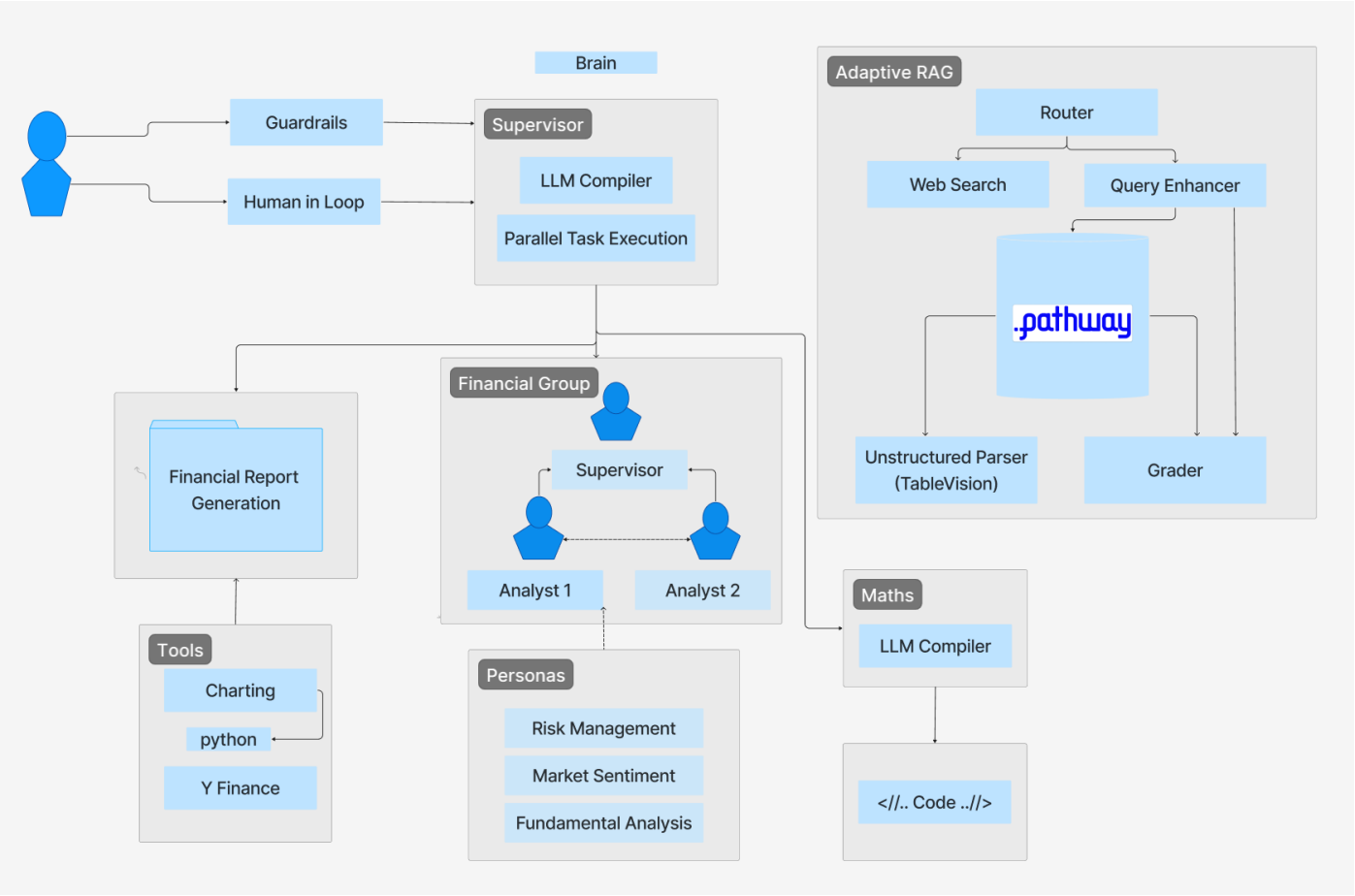


Figure 1. Pathway-Enhanced Adaptive RAG

However, retrieving tables presented a unique challenge, as tables often lacked sufficient context, making them harder to index and retrieve effectively. So we developed a Custom mode which we call **TableVision Mode**

5.1.2. TableVision: A Novel Mode Proposed for the Unstructured Parser

To address the limitations in table retrieval, we introduced a **TableVision mode** that enriches the context for tables. This mode generates:

1. A detailed description of the table.
2. The table in **Markdown format** for enhanced usability.
3. **Relevant search queries** derived from the table and its context, ensuring effective retrieval.

This enriched information is stored alongside the table in the index, significantly improving its discoverability during retrieval operations.

5.1.3. Advantages of the TableVision

The custom mode offers the following benefits:

- **Improved Retrieval and Generation Accuracy:** Tables are enriched with descriptive context and relevant queries, making them easier to locate and analyze.

Context: Organic/Core (non-GAAP)1 First-Quarter 2023 Results

Description of the Table

The table presents key financial metrics from the first quarter of 2023, focusing on organic revenue growth, core earnings per share (EPS), and the change in core constant currency EPS. Organic Revenue Growth: The first-quarter 2023 organic revenue increased by 14.3%, excluding acquisitions, divestitures, and currency effects. Core EPS: The adjusted earnings per share (EPS) for Q1 2023 is $1.50, excluding certain items. Core Constant Currency EPS Change: The core EPS increased by 18% when excluding the impact of currency fluctuations.

Table in Markdown Format

Financial Metric	Value
First-Quarter Organic Revenue Growth	14.3%
Core EPS	$1.50
Core Constant Currency EPS Change	18%

Search Queries Answerable from the Table

1. What was the organic revenue growth for the first quarter of 2023?
2. What is the core EPS for the first quarter of 2023?
3. How much did the core constant currency EPS change in the first quarter of 2023?

Figure 2. Table Element retrieved using TableVision

5.2. Generation

5.2.1. Supervisor based on LLM compiler

The **supervisor** we implemented advances the LLM Compiler concept by introducing powerful features for managing complex workflows involving large language models (LLMs). Tasks are organized into a **Directed Acyclic Graph (DAG)**, with each node representing a distinct operation or sub-task.

This structure facilitates efficient planning and execution of multi-step processes such as data extraction, reasoning, and problem-solving. Customized prompts enhance the robustness of task planning, while the **Planner** dynamically generates initial plans using tools, prompts, and observations. When necessary, replanning is triggered by system messages, leveraging historical tasks and observations to seamlessly update plans.

Functioning as a delegation brain, the supervisor coordinates access to specialized tools, including the **Finance Analyst Group, Math Tool, Data Retrieval Tool, and Report Generation Tool**. It breaks down user queries into a network of interdependent tasks, executing them dynamically using a threading-based scheduler that resolves dependencies within the DAG. Outputs from these tools are stored as observations, which the supervisor evaluates to determine if the responses are sufficient. Based on this evaluation, the system either replans tasks or generates the final output, ensuring adaptability and reliability in workflow execution.

A standout innovation of the framework is the **integration of human-in-the-loop (HITL)** functionality. When human feedback is required, the workflow pauses at a HITL node, enabling human input to refine subsequent planning. This mechanism bridges the gap between automation and human judgment, ensuring better decision-making. The updated **Joiner** Node further enhances this process by handling human feedback effectively, overcoming earlier limitations in task completion and replanning. Combined with real-time adjustments enabled by stored observations, the framework dynamically adapts workflows to new information or feedback.

To optimize memory and execution efficiency, the system creates checkpoints following human feedback. These **checkpoints summarize preceding states**, enabling context-aware workflow resumption while conserving resources. This HITL-enhanced framework achieves a seamless integration of automation and human oversight, improving decision quality, adaptability, and overall efficiency. By combining advanced planning, dynamic execution, and human feedback, the supervisor delivers a robust, context-sensitive task execution process.

5.2.2. Data Retrieval Tool (Adaptive RAG)

Adaptive RAG (Retrieval-Augmented Generation) is a sophisticated strategy that integrates query analysis with active and self-corrective RAG mechanisms. This approach allows for dynamic routing of queries based on their relevance to indexed data, enabling more accurate and contextually appropriate responses.

Query Analysis and Routing

The first step in our Adaptive RAG implementation is **Query Analysis**. This process determines whether the user's question is related to the indexed data or requires a web search for recent or external information.

1. **Unrelated to Index**: If the query is deemed unrelated to the indexed data, it is immediately routed to a **Web Search** node. Here, the query is processed using a web search tool to gather relevant information. The results are then passed to a **Generation** node, which synthesizes an answer based on the web search results.
2. **Related to Index**: If the query is related to the indexed data, it proceeds to the **Retrieve** node. However, our implementation includes an advanced step before retrieval: **Query Enhancement**.

Query Enhancement

It is a critical component of our implementation. It involves constructing multiple queries from the original user question. Specifically, the system generates five queries:

- **Original Query**: The user's original question.
- **Enhanced Queries**: Four rephrased versions of the original query, with at least one query specifically enhanced to request relevant data from indexed tables.

Retrieval and Grading

The enhanced queries are then passed to the **Retriever** node, which retrieves data and tables from the **Pathway vector store** server. For each query, the system fetches the top-k results, ensuring a broad and diverse set of data is considered. The retrieved data is then sent to the **Grader** node. In our implementation, the Grader not only grades the relevance of the retrieved data but also selects the most relevant data, ensuring that at least one table is extracted. This guarantees that the response is grounded in structured, indexed data.

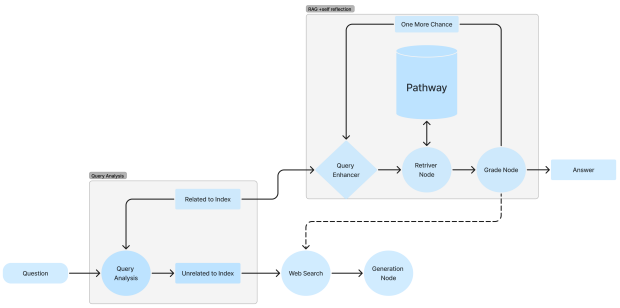


Figure 3. Pathway-Enhanced Adaptive RAG

RAG + Self-Reflection

Our implementation of RAG with self-reflection introduces a fallback mechanism to handle cases where the Grader does not find sufficient relevant data. If the Grader determines that the retrieved data is insufficient, it triggers a **One More Chance** mechanism:

1. **Query Enhancement Redux**: The Grader sends the original query back to the Query Enhancer, which again constructs five new queries.
2. **Retrieval and Grading Redux**: The new queries are processed through the Retriever and Grader nodes as before.

If, after this second attempt, the Grader still finds insufficient data, the query is routed to the **Web Search** node, and the process continues as described earlier.

5.2.3. Financial Analyst Group

The Financial Analyst Group simulates expert financial discussions, where the Supervisor agent refines and directs simplified queries to specialized financial agents. These agents,

each focusing on distinct financial topics, engage in a structured conversation led by a Financial Lead agent, who oversees the discussion and delivers a comprehensive final analysis.
Financial Lead:

- ***Purpose:*** The Financial Lead is the overarching controller of the conversation. It consolidates the findings from Agent 1 and Agent 2 and ensures the conversation flows logically.
- ***Functions:***
  - Summarizes the analysis done by the other agents.
  - Decides whether the conversation should continue or conclude based on the progress of the analysis.
  - Provides a final assessment of the financial situation, including action points and further suggestions.

Agent 1 (Financial Analyst):

- ***Purpose:*** Responsible for the core analysis of specific financial metrics and data points.
- ***Functions:***
  - Examines financial statements, market trends, and company performance.
  - Starts the analysis and collaborates with Agent 2 to refine findings.
  - Decides if the analysis requires more data, further discussions, or if it's complete and can be handed off to the Supervisor.

Agent 2 (Complementary Analyst):

- ***Purpose:*** Complements Agent 1's analysis by adding different perspectives, especially on risk or external market factors.
- ***Functions:***
  - Provides additional insights into data that Agent 1 may have overlooked, such as macroeconomic factors or alternative risk assessments.
  - Responds to Agent 1's findings and may introduce new considerations or signal when the conversation has reached its logical conclusion.
  - Identifies when further investigation is necessary or if the Supervisor should take over.

Interaction and Conversation Flow

- **Initiation:** Agent 1 begins the discussion by analyzing a specific financial topic using provided data.
- **Collaboration:** Agent 2 contributes by challenging assumptions, adding insights, or supporting data.
- **Evaluation:** Agents use flags (`YES`/`NO`) to signal readiness for escalation or the need for further analysis.
- **Financial Lead:** The Financial Lead synthesizes insights to ensure critical points are addressed.
- **Output:** A structured summary with actionable recommendations is generated.

The Financial Lead ensures the discussion concludes with actionable and well-rounded results.

Built-In Agent Personas

- **Market Sentiment Analyst:** Assesses trends and sentiments influencing the market.
- **Risk Assessment Analyst:** Identifies vulnerabilities and quantifies risks.
- **Fundamental Analyst:** Examines core financial data, such as balance sheets and valuations.

The selection of these personas is adaptive, driven by the Supervisor's logic to align the analysis with the topic's demands.

The system integrates GPT-4 or MiniGPT-4 for interpreting financial data and generating coherent, context-aware responses and producing concise, actionable summaries that facilitate informed decision-making.

5.2.4. Math Tool

The Computational Math Tool is a sophisticated solution designed to tackle complex mathematical challenges in financial analysis. By seamlessly integrating natural language processing (NLP) with high-performance numerical computation, this tool offers an intuitive and efficient approach to solving intricate mathematical problems. Below, we delve deeper into its key features and functionalities, particularly focusing on the coding executor and other advanced components.

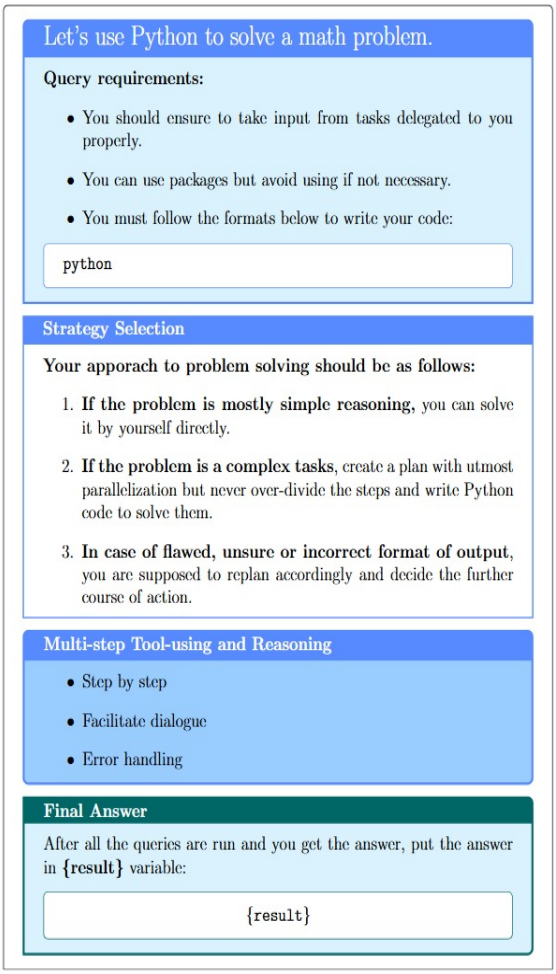


Figure 4. Math Agent Framework

- **Natural Language Processing for Mathematical Queries:** The Math Tool leverages *OpenAI's* language models to process natural language inputs and convert them into valid mathematical expressions. For instance, queries like "*What is 37593 * 67?*" or "*37593^(1/5)*" are

translated into computable forms. This feature makes it highly accessible, particularly for non-technical users, by allowing intuitive interaction with the system.

- **High-Performance Computational Framework:** Mathematical queries are broken into independent sub-queries, enabling parallel evaluation using Python's *numexpr* library. Each sub-query is treated as a self-contained task, allowing simultaneous computation which ensures efficient processing.
- **Function-Level Implementation:** The tool's architecture includes two key functions:
  - **Evaluator:** Responsible for evaluating the final mathematical expressions.
  - **Generator:** Processes user inputs and contextual information to generate the appropriate expressions for evaluation.

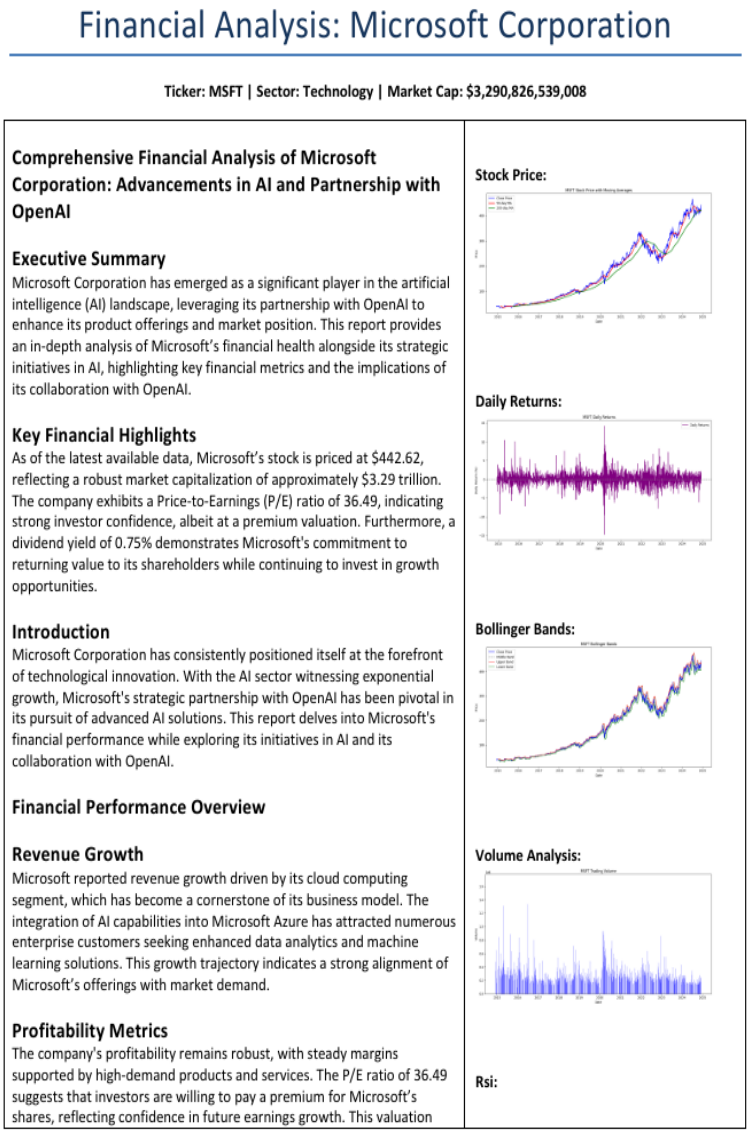
  Together, these functions enable accurate, efficient, and user-friendly computation.

5.2.5. Report Generation Tool

We implemented an advanced financial data analysis and report generation system. It utilize various Python libraries, including $yfinance$, $pandas$, $matplotlib$, and $openai$, to fetch, analyze, and visualize financial data. The system also integrates with Microsoft Word for report generation and conversion to PDF format.

The system utilizes $yfinance$ to fetch historical stock data for specified ticker symbols. It employs sophisticated visualization techniques to present key financial metrics, including stock prices, daily returns, Bollinger Bands, RSI, MACD, moving averages, volume analysis, and other technical indicators. The 'EnhancedFinancialDataAnalyzer' class encapsulates methods for plotting these financial metrics. The system utilizes $gpt-4o-mini$ to generate detailed, user-specific financial analysis based on user queries. The output comprises a well-structured Word document report that seamlessly integrates financial metrics, visualizations, and LLM-generated insights. This report offers a robust and automated solution for real-time data analysis.

The integration with Microsoft Word ensures a professional report format, while the conversion to PDF format using $comtypes$ guarantees accessibility and ease of distribution. By providing accurate and tailored reports, this system significantly enhances decision-making processes in financial analysis. Robust error handling mechanisms are in place to manage exceptions, and the system interacts with the user via a command-line interface.


Financial Analysis: Microsoft Corporation

Ticker: MSFT | Sector: Technology | Market Cap: $3,290,826,539,008

Comprehensive Financial Analysis of Microsoft Corporation: Advancements in AI and Partnership with OpenAI

Executive Summary

Microsoft Corporation has emerged as a significant player in the artificial intelligence (AI) landscape, leveraging its partnership with OpenAI to enhance its product offerings and market position. This report provides an in-depth analysis of Microsoft's financial health alongside its strategic initiatives in AI, highlighting key financial metrics and the implications of its collaboration with OpenAI.

Key Financial Highlights

As of the latest available data, Microsoft's stock is priced at $442.62, reflecting a robust market capitalization of approximately $3.29 trillion. The company exhibits a Price-to-Earnings (P/E) ratio of 36.49, indicating strong investor confidence, albeit at a premium valuation. Furthermore, a dividend yield of 0.75% demonstrates Microsoft's commitment to returning value to its shareholders while continuing to invest in growth opportunities.

Introduction

Microsoft Corporation has consistently positioned itself at the forefront of technological innovation. With the AI sector witnessing exponential growth, Microsoft's strategic partnership with OpenAI has been pivotal in its pursuit of advanced AI solutions. This report delves into Microsoft's financial performance while exploring its initiatives in AI and its collaboration with OpenAI.

Financial Performance Overview

Revenue Growth

Microsoft reported revenue growth driven by its cloud computing segment, which has become a cornerstone of its business model. The integration of AI capabilities into Microsoft Azure has attracted numerous enterprise customers seeking enhanced data analytics and machine learning solutions. This growth trajectory indicates a strong alignment of Microsoft's offerings with market demand.

Profitability Metrics

The company's profitability remains robust, with steady margins supported by high-demand products and services. The P/E ratio of 36.49 suggests that investors are willing to pay a premium for Microsoft's shares, reflecting confidence in future earnings growth. This valuation

Stock Price:

Daily Returns:

Bollinger Bands:

Volume Analysis:

Rsi:

Figure 5. Financial Analysis Report

5.3. Responsible AI

- **Input Guardrails Mechanism** implements a sophisticated self-developed language filtering system that proactively prevents malicious or inappropriate content from entering the multi-agent RAG system. The mechanism employs advanced natural language processing techniques to detect and block potentially harmful or false input while maintaining a nuanced approach to genuine queries.
- **Intelligent Query Reconstruction Protocol** enables the system to handle complex input scenarios where user communication contains inappropriate language mixed with legitimate inquiries. When offensive or potentially false language is detected, the system generates a multi-step response:
  1. Issue a clear warning to the user about inappropriate language
  2. Intelligently reconstruct the core query, stripping away offensive content
  3. Provide a comprehensive and accurate solution to the underlying genuine question
- **Content Validation Framework** rigorously screens retrieved information using advanced fact-checking algorithms, cross-referencing techniques, and contextual analysis to prevent false or misleading content from reaching the user, ensuring high standards of information integrity.
- **Guardrails AI Integration** leverages the GuardrailsAI platform to implement comprehensive safety and content moderation mechanisms. This advanced framework provides:
  - Dynamic content filtering

- Toxic language prevention
- Contextual safety checks
- Real-time input and output monitoring

5.4. Fallback Mechanism

5.4.1. Implementation of Human-in-the-Loop (HITL)

When the Supervisor encounters a task it cannot complete, the system seamlessly transitions to the Human-in-the-Loop (HITL) mechanism. This transition is facilitated through the supervisor that identifies the need for human intervention based on predefined criteria such as task complexity, multiple failures, or ambiguous queries. The HITL tool then prompts the user with clear, actionable questions, seeking additional input or clarification. Once the user inputs the required information, the system resumes processing from the last successful state using the Reviver Tool, ensuring that the workflow is not disrupted.

5.4.2. Replanner

The Replanner mechanism is a critical component of our fallback system, designed to dynamically adjust the execution flow when errors are encountered. When an error occurs, the system first passes the task to the Supervisor for initial assessment. If the Supervisor identifies the need for further adjustments, it triggers the Replanner to analyze the tools and potentially modify the query to mitigate the error. This process involves a detailed evaluation of the current state and the tools involved, ensuring that any changes are contextually relevant and operationally feasible.

5.4.3. Guardrails

Guardrails in fallback mechanisms refer to pre-defined rules or boundaries that prevent AI systems from making unsafe or incorrect decisions. These constraints ensure responsible behavior even when the AI model behaves unpredictably. In our system, guardrails act as checks or limits that the AI cannot override. Our guardrails mechanism is designed to handle unsafe or abusive queries before passing them for further processing. This approach ensures a baseline of safety, compliance, and user trust while reducing the risk of AI-induced harm.

Input Filtering Demonstration

User Input: Yo, wtf is quantum computing? Explain that sh*t to me!

System Response:

- [Warning: Positive thoughts only, please! Let's keep the conversation safe and supportive.]
- Reconstructed Query: Could you please explain quantum computing in a clear and professional manner?

5.4.4. Web Search

The Web Search component is an integral part of our fallback mechanism, designed to enhance the system's ability to gather relevant information when internal data sources are insufficient. When the system encounters a query that requires external data or when the internal data retrieval fails, the Web Search tool is activated. This tool leverages advanced search algorithms to query the internet for the most relevant and up-to-date information. The retrieved data is then processed and integrated into the system's workflow, providing the necessary context or additional insights to resolve the query. By incorporating Web Search, the system ensures that it can dynamically adapt to a wide range of scenarios, enhancing its overall robustness and accuracy in handling complex financial data processing tasks.

5.4.5. LLMOps

The LLMOps and DataOps layers have streamlined the deployment and management of large language models (LLMs) and data pipelines. These layers ensure that the models are optimized for performance, reliability, and scalability.

6. EXPERIMENTAL SETUP

We evaluated our Retrieval-Augmented Generation (RAG) solution on the challenging **FinanceBench** dataset. Each entry consists of a question, an answer, evidence with page references, and optional annotations.

- **Types of Questions:**
  1. **Domain-Relevant:** General financial analysis questions.
  2. **Novel-Generated:** Company or report-specific questions.
  3. **Metrics-Generated:** Questions requiring financial metric computation.
- **Evaluation Metrics:**
  1. **Answered %:** Correct answers.
  2. **Refusal %:** Declined answers to minimize hallucinations.
  3. **Incorrect %:** Incorrect answers.

  These metrics were validated using the **LLM-as-a-Judge** technique by comparing answers with ground truth.
- **Experimental Protocol:** For each question:
  1. The system autonomously determined the retrieval and generation process.
  2. Responses were generated through the framework's modules.
  3. Performance was compared against FinanceBench baselines: GPT-4-Turbo with a shared vector store and GPT-4-Turbo with direct page content access.
  4. Three system configurations were tested:
     (a) With Human-in-the-Loop (HITL).
     (b) Without HITL.
     (c) Passing page content as context.

     All configurations used **GPT-4o** as the main supervisor and **GPT-4o-Mini** for secondary tasks.

7. RESULTS

The results presented in Table 1 reveal several important trends:

- **Proposed System**:
  - The proposed system without human-in-the-loop (HITL) intervention achieved a balanced performance with a significant proportion of refusals,

Method	Correct (%)	Refusals (%)	Incorrect (%)
Llama2 Shared Vector Store	19%	70%	11%
GPT-4-Turbo Shared Vector Store	19%	13%	68%
GPT-4-Turbo Oracle	85%	15%	0%
Proposed System (No HITL)	42%	51%	9%
Proposed System (With HITL)	56%	37%	7%
Proposed System (Oracle Mode)	92%	4%	4%

Table 1. Performance Comparison of Different Methods on Finance Bench Dataset (Percentages Only)

showing the system's cautious approach to avoiding hallucinations.
- When human-in-the-loop intervention was incorporated, the system saw a noticeable improvement in the correct answer rate, while refusals decreased. It should be noted that the human annotator was allowed only one brief, one-liner correction per query, ensuring minimal intervention.
- In oracle mode, the system approached the performance of GPT-4-Turbo Oracle, demonstrating the effectiveness of the proposed framework when provided with direct content access.

Overall, these results highlight the effectiveness of the proposed system in improving upon the baseline methods, especially with human-in-the-loop intervention. The cautious refusal strategy and the ability to make minimal yet impactful corrections allowed the system to improve accuracy, particularly in the absence of full access to the content.

8. USER INTERFACE

8.1. Overview

To facilitate better understanding and transparency of the complex MARAG pipeline, we have developed an interactive User Interface (UI). This UI is designed to visualize the real-time operations of the system, providing an intuitive representation of its internal processes, agents and tools.

8.2. Key Features

- **Real-Time Operations Sync**
  The UI provides a dynamic overview of ongoing processes within the multi-RAG pipeline, offering users the ability to see which agents or modules are currently active and what tasks they are performing.
- **State Execution Graph**
  The UI has a step-by-step execution graph, that highlights the sequence of actions taken by the pipeline and the path taken by the framework.
- **Interactive User-Agentic Setup**
  The UI enables a human-in-the-loop approach, allowing seamless communication between the user and the pipeline. It provides mechanisms for the system to query the user as a potential data source, request clarification, or incorporate user-provided inputs dynamically.
- **Transparency in Decision-Making**
  Users can trace back the pipeline's outputs to specific retrievals, computations, or agent actions, ensuring accountability and insight into decision-making mechanisms.

8.3. Technical Implementation

1. **Frontend**
   Built using Next.js and Tailwind CSS for a responsive, scalable, and visually appealing experience. It leverages Socket.IO for real-time communication with the server, enabling instant updates and interactions without refreshing the page.
2. **Backend**
   Powered by Flask, the backend handles all data processing, integrates seamlessly with the MARAG pipeline, and provides endpoints through REST APIs and real-time updates via Socket.IO. The pipeline itself is built with LangGraph, LangChain, and LlamaIndex, ensuring modularity and efficient agent communication.
3. **VectorStore Server**
   Hosted using Pathway, the vector indexing server is optimized for high-performance dynamic data retrieval, supporting rapid embedding queries and real-time updates for changing datasets.
4. **Database**
   A NoSQL database, Firebase, is used for storing user data, including chat history, progress, and session metadata, enabling efficient retrieval and scalability for multi-user environments.

9. LESSONS & FUTURE WORKS

In this section we discuss the insights we have drawn while building our proposed system and the future works that are possible with this system.

9.1. Key Lessons

- **Dynamic Task Management:** The integration of dynamic task management systems, such as the Supervisor and Replanner inspired by the LLMCompiler, has proven to be superior to other methods of Self-Reflection RAG, which display robust error handling. In addition, the parallel calling reduces the time overhead of such systems that traditional systems suffer from.
- **Human-in-the-Loop (HITL):** The HITL mechanism has been crucial in enhancing the system's reliability and accuracy. We observed that often times , the system is capable of recognizing when is needs some input but traditional agentic systems do not take advantage of this fact, we incorporated HITL in our system that leverages this to reduce false positives as can be seen using our refusal rates vs incorrect answers from the results section.
- **Data Enrichment:** While handling traditionally difficult data for RAG such as financial documents, which include key information in every section of the document, this information becomes even more difficult to retrieve when tables are involved. By enriching the table chunks with additional data from the surrounding texts and a description of the table, we saw a significant boost in the overall accuracy of the retriever and as a result the whole system.

9.2. Future Works

Building on these lessons, we have identified several areas for future development and improvement:

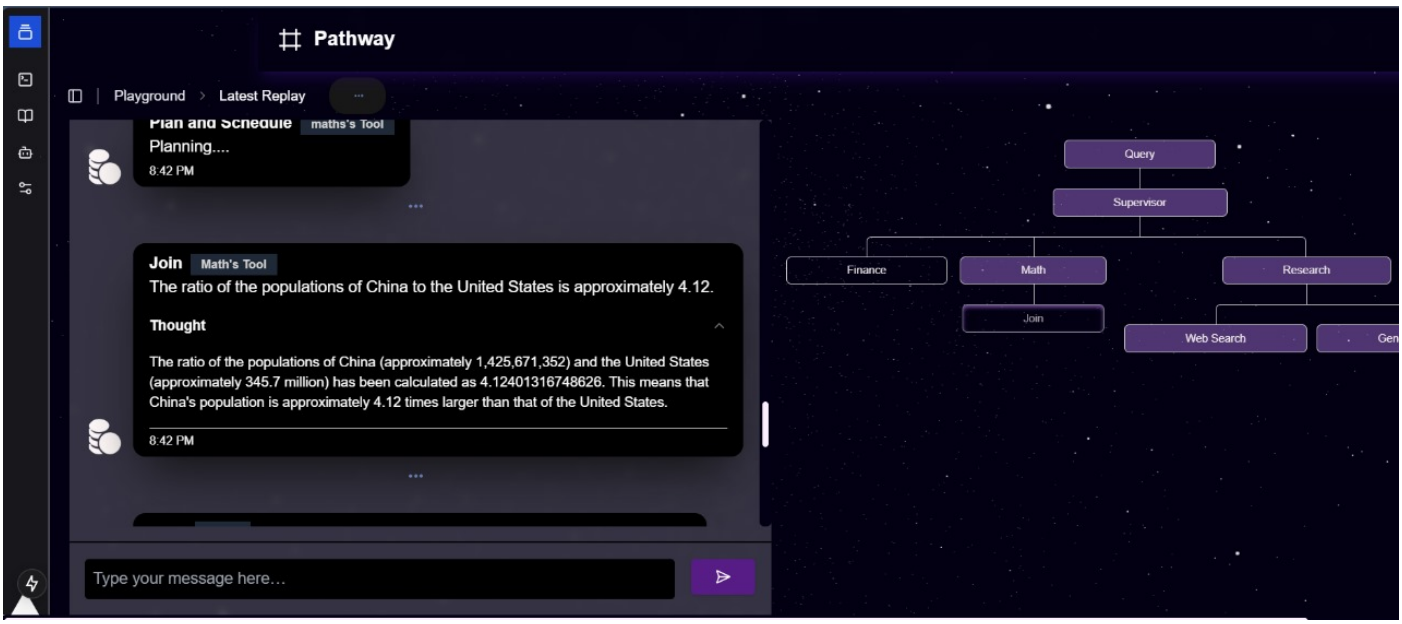


Figure 6. A query in action

- **Enhanced Agent Capabilities:** We plan to develop additional specialized agents to handle a broader range of financial data processing tasks. These agents will enhance the system's ability to manage complex and diverse financial scenarios.
- **Advanced Data Processing:** We aim to further optimize data processing techniques, including the integration of more advanced data retrieval and analysis tools. This will improve the system's efficiency and accuracy in handling large-scale financial datasets.
- **Expanded Testing:** We will conduct more comprehensive testing to evaluate the system's performance in various operational contexts. This includes testing with different types of financial data and scenarios to ensure robustness and reliability.

10. CONCLUSION

In conclusion, the development of our Multi-Agent Retrieval-Augmented Generation (MARAG) system has demonstrated significant progress in creating an efficient and reliable financial data processing system. By integrating advanced techniques such as dynamic task management, human-in-the-loop mechanisms, guardrails, and web search capabilities, we have built a robust and adaptable system capable of handling complex financial data processing tasks.

The system's architecture, which includes specialized agents, adaptive retrieval mechanisms, and responsible AI practices, ensures high standards of accuracy, reliability, and ethical operation. Initial tests using financial datasets have shown promising results, highlighting the system's potential for reliable financial data analysis.

As we continue to refine and expand the system, we remain committed to enhancing its capabilities and addressing the evolving needs of the financial sector. Our future works will focus on developing additional agents, optimizing data processing, conducting comprehensive testing, improving the user interface, and ensuring compliance with legal and regulatory requirements.

References

Talebirad, Yashar, and Amirhossein Nadiri. "Multi-agent collaboration: Harnessing the power of intelligent LLM agents." arXiv preprint arXiv:2306.03314 (2023).

Yang, Hongyang, et al. "FinRobot: An Open-Source AI Agent Platform for Financial Applications using Large Language Models." arXiv preprint arXiv:2405.14767 (2024).

An, Siyu, et al. "FinVerse: An Autonomous Agent System for Versatile Financial Analysis." arXiv preprint arXiv:2406.06379 (2024).

Yang, Hongyang, Xiao-Yang Liu, and Christina Dan Wang. "FinGPT: Open-source financial large language models." arXiv preprint arXiv:2306.06031 (2023).

Guo, Zirui, et al. "LightRAG: Simple and Fast Retrieval-Augmented Generation." arXiv preprint arXiv:2410.05779 (2024).

Jeong, Soyeong, et al. "Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity." arXiv preprint arXiv:2403.14403 (2024).

Yan, Shi-Qi, et al. "Corrective retrieval augmented generation." arXiv preprint arXiv:2401.15884 (2024).

Sarmah, Bhaskarjit, et al. "HybridRAG: Integrating Knowledge Graphs and Vector Retrieval Augmented Generation for Efficient Information Extraction." arXiv preprint arXiv:2408.04948 (2024).