FinSight - Dynamic Agentic RAG with Pathway Final Report

Team 24

Contents

1. Introduction

Our project, **FinSight**, is focused on developing a dynamic multi-agent Retrieval-Augmented Generation (RAG) system, seamlessly integrated with **Pathway**—a cutting-edge dynamic vector database optimized for efficient and adaptive information retrieval.

Deploying Agentic RAG in production poses several challenges. Autonomous agents can lead to unpredictable decision-making, retrieval errors, and inconsistent responses. Effective error handling and real-time failure management are essential for reliability. Coordinating multiple agents can introduce latency, impacting response times. Additionally, ensuring transparency and explainability of agent actions is crucial for maintaining user trust. Optimizing token usage reduces costs and computational overhead. Managing security, privacy, and error propagation is critical, especially for sensitive applications all while ensuring responsible AI practices to prevent harmful or biased outputs.

The primary objective of FinSight is to create a powerful, domain-agnostic framework for retrieving and generating precise insights. Specializing in financial data, it offers finance professionals and enthusiasts accurate answers to their queries, while also ensuring adaptability for effective information retrieval across various knowledge domains. Inspired by state-of-the-art RAG systems, FinSight employs real-time adaptability and advanced vector search to ensure efficient and reliable query resolution. Its primary objectives include:

- **Finance-Centric Specialization:** Specializes in handling and analyzing financial documents, including complex tabular data, to provide accurate answers.
- **Scalable Multi-Agent Architecture:** Designed for robust agent interactions, with scalability enabled by Pathway's serve_callable method.
- **Handling Complex Queries**: Capable of managing complex queries that may require multiple retrievals to ensure comprehensive and precise answers.
- **Responsible AI**: Adheres to responsible AI practices, ensuring ethical use and transparency.

FinSight sets a new standard in intelligent query-response systems with its user-centric design and domain expertise. It provides a robust, transparent solution for both financial and general inquiries, reimagining the possibilities of retrieval-augmented AI.

2. Uniqueness

Existing financial models have advanced financial analysis but face key limitations. BloombergGPT [14], struggles with multimodal processing, limiting its analysis of charts and tables FinGPT [16] encounters data sparsity in niche topics, affecting predictions. Hybrid RAG models [18], improve explainability through symbolic reasoning but compromise speed. **FinSight** addresses these issues by integrating multimodal capabilities, dynamic retrieval pipelines, and balancing speed with transparency.

Recent advancements like FishNet [3] and FinRobot [17] have improved financial query handling but have limitations. FishNet's sub-querying and expert swarms enhance systematic analysis but struggle with coherently integrating results and lack flexibility for dynamic queries. FinRobot uses Chain-of-Thought (CoT) prompting for complex reasoning but faces slower processing and lacks persona-driven, context-sensitive insights tailored to specific financial roles or goals.

We compare FinSight with Quillai and Fintool because these platforms are widely adopted by prominent companies in the financial industry, serving as established benchmarks for financial intelligence and analysis. FinSight offers several distinct advantages over Fintool and Quillai:

1. **Multimodal Analysis:** Unlike Fintool, which lacks graph and table analysis, FinSight integrates multimodal capabilities, analyzing text, tables, and graphs in financial documents for richer insights and includes Human in the Loop for ambiguous queries.
2. **Dynamic Query Routing:** Fintool lacks dynamic query routing, while FinSight automatically directs queries to the appropriate sub-models, enhancing user experience and efficiency.
3. **Financial-Based Personas:** FinSight creates specialized financial personas for tailored responses (e.g., for traders, analysts, investors), a feature absent in Quillai, which focuses on generalized quantitative analysis.
4. **Comprehensive Query Handling:** Unlike Quillai, which focuses on specific financial queries (e.g., market predictions or trading strategies), FinSight handles both qualitative and quantitative queries for more holistic answers.

These features make FinSight a more flexible and comprehensive solution for financial analysis, addressing the limitations of existing platforms.

3. Use-Case Selection and Novelty

We selected the financial domain for its complex interplay of narrative, tabular data, and specialized terminology. Handling charts, tables, speech, and images further complicates analysis. Ensuring regulatory compliance, mitigating bias, and maintaining interpretability is particularly critical in high-stakes applications such as investment advice or risk assessment.

Financial queries can vary in how their sub-questions relate to each other, with some requiring simultaneous retrieval of multiple, independent pieces of data—such as comparing financial metrics across different companies(called **parallel**)—while others follow a step-by-step logic, where the answer to one question dictates the next area of investigation—such as identifying revenue sources before examining their influence on profit margins(called **series**). Some queries can also decompose in a **recursive** manner, where a previously parallel query may further decompose into a series of dependent steps, or vice versa, making them especially challenging to manage.

Financial analysis extends beyond mere data retrieval, requiring meaningful interpretation. In real world, multiple analysts examine the same dataset through varied lenses: some prioritize basic financial computation–based metrics, while others adopt different perspective–based strategies to explain patterns. Reflecting this duality, our system integrates both approaches into a cohesive, user-friendly framework.

By addressing these multifaceted challenges, our system surpasses a generic Agentic RAG approach, offering a domain-

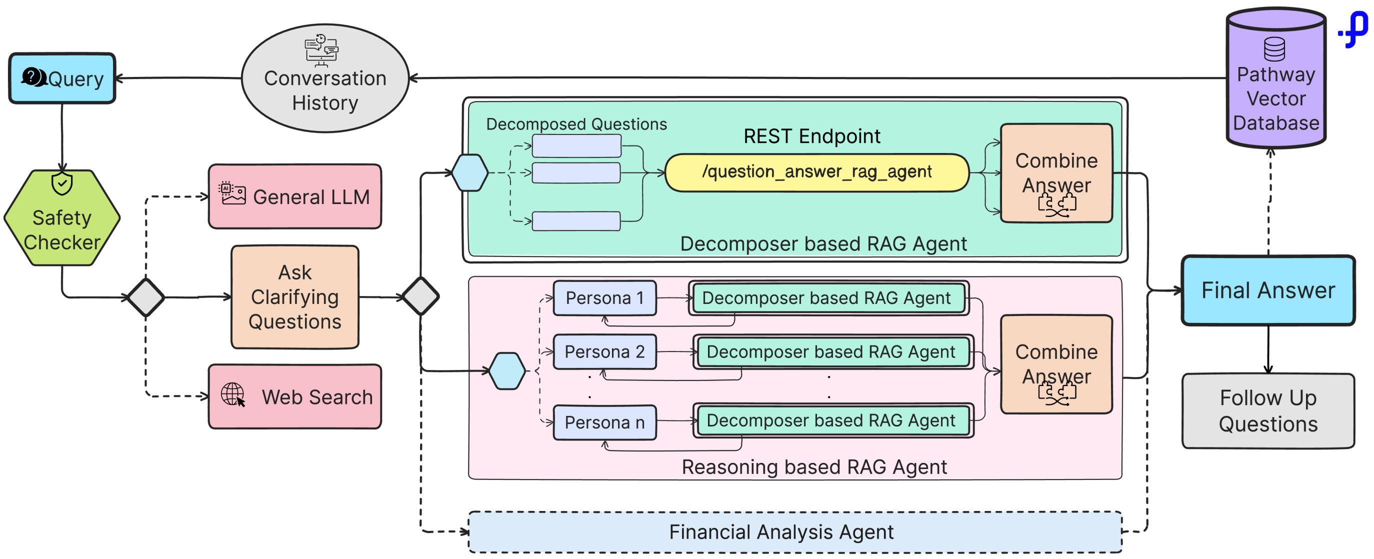


Figure 1. Final end-to-end Workflow

specific framework optimized for financial applications. It addresses long-standing gaps in existing tools[2] by effectively handling queries derived from financial documents such as SEC filings, including 10-K documents. Our system's innovative design bridges the divide between complexity and usability, setting a new benchmark for financial intelligence solutions.

4. Solution Overview

We propose an Agentic Architecture designed to handle diverse queries through specialized workflows and an advanced end to end Agentic RAG pipeline. Specialized workflows address specific query types, such as information retrieval, financial analysis, and persona-based reasoning. The system can handle a wide range of queries, from simple fact-based questions to complex financial analyses and open-ended summaries, overcoming the limitations of previous work. Our system supports two modes : Fast Mode and Precise Mode, facilitating speed-detail tradeoff[A.3]. Our end to end pipeline consists of the following:

1. Uploaded documents are parsed using OpenParse, extracting text, table and table-value maps[5.1] as well as the metadata to preserve structure and meaning.
2. Initial queries are screened for safety compliance, with prior conversation history and a human-in-the-loop mechanism used to refine them.
3. Routing agents direct queries to the specialized workflows[5.4]. Simple or out-of-domain queries are handled by the LLM, either independently or with web assistance.
4. We use three workflows: decomposed RAG for information retrieval, persona-based reasoning for multi-perspective queries, and a KPI-based financial analyzer.
5. For information retrieval the input query is decomposed by the *Query Decomposer*[5.5] and passed to the RAG agent, which processes metadata extraction, document retrieval, document relevance, answer generation, hallucination detector, and answer grading. The results are then combined to generate a combined answer.
6. In persona-based reasoning[5.4.2], financial agents, based on the input query, ask questions from their own perspective to address the original query, with the final output synthesized after all questions are answered.
7. In the financial analysis agent[5.4.3], the user selects preferred analyses, and KPIs are calculated for the requested companies and years, generating insights based on these values.
8. Once the final answer is synthesized, it is passed through the *Postprocessing Agent*[5.8] to generate actionable insights and visualizations.

5. System Architecture

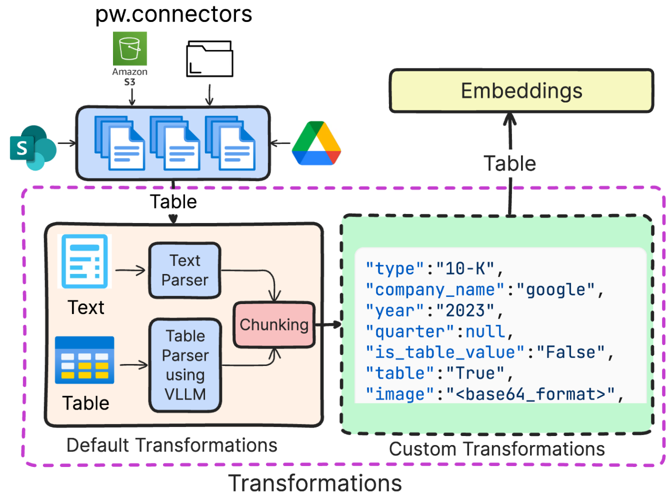
The full architecture of our Agentic RAG system is illustrated graphically in Figure 1. The system is implemented using the LangGraph framework, providing a flexible and modular structure for agent-based processing. Retrieval is powered by Pathway's *DocumentStore*, with custom modifications to optimize indexing and retrieval processes. We also leverage Pathway's User-Defined Functions (*UDFs*) for caching to enhance system performance.

5.1 Indexing

We use Pathway's dynamic data handling in three stages: Connectors, Tables, and Transformations. Documents are ingested via connectors, forming rows in a preliminary table. A parsing transformation converts raw documents into structured data, followed by chunking to break them into smaller pieces. Finally, a transformation generates indices for efficient data retrieval.

We extract key metadata from the first 10 pages of a document using LLM, identifying the document type, company name (excluding suffixes), and publishing year. For tables, Visual LLMs parse and convert data into natural language. When OpenParse struggles with tables, we generate Table Value Maps, detailing table name, row, column, and significance and integrate these into *Pathway tables*.This approach improves the system's ability to answer quantitative queries and boosted correctness by 19% in our evaluation.

To enhance the LLM's understanding of document chunks, we implemented **Contextual Retrieval** using Anthropic's prompt caching, where the full document and chunk are input with context prepended. We assign **Intradocument Topic Tags** to each chunk—35 financial topics for financial documents or dynamic topics for non-financial ones—to improve retrieval filtering. For 10-K reports, we assign the *Item Number* to aid precise retrieval. However, due to impracticality, this is excluded from the final indexing pipeline. We also applied a **compression technique**[B.1] to parsed chunks before creating the vector store, optimizing the retrieval process.



```
"type":"10-K",
"company_name":"google",
"year":"2023",
"quarter":null,
"is_table_value":"False",
"table":"True",
"image":"<base64_format>",
```

Figure 2. Indexing workflow

5.2 Conversational Awareness

Conversational context management is achieved through two mechanisms: a state-level repository storing the most recent K messages (user and system) and a distributed caching system that uses Pathway's Document Store to index messages with metadata into a `cache_server`, enabling granular retrieval of interaction histories.

Upon receiving a query, the system refines it by incorporating relevant context from the conversational history derived through both these mechnanisms. The contextualized query then traverses subsequent pipeline stages, thereby maintaining continuity and preserving interaction-level contextual nuances.

5.3 Human In The Loop

Human in the Loop (HITL) improves query clarity by generating *clarifying questions* when ambiguity or missing context is detected, refining the query through up to three rounds of user responses. It supports three types of questions: open-ended (Direct Answer), single-choice, and multiple-choice.

Users can also select to conduct *KPI-based financial analysis* for a company, with our system suggesting up to 16 relevant analyses, that runs in parallel with our RAG Workflows [5.4].

5.4 Specialized Workflows

5.4.1 Information retrieval via Decomposed RAG Agent

When only data retrieval is needed, the system utilizes our novel RRR[5.5.4] framework. It retrieves relevant documents, generates answers, combines them, and recursively decomposes unresolved questions to ensure efficient, precise output without unnecessary analysis.

5.4.2 Reasoning via Dynamic Persona Generation

This workflow addresses questions from multiple perspectives using *dynamically generated persona agents*, each specializing in a specific domain. It includes two variants: the **Parallel Persona Workflow**, where tasks are handled simultaneously by multiple agents, with answers combined for the final response and a generalist agent ensuring full coverage, and the **Supervised Persona Workflow**, where tasks are performed sequentially, with the supervisor selecting the next agent based on prior responses. Due to increased latency, the parallel workflow is preferred. Each persona has a decomposed RAG agent as a tool, which they can query to obtain grounded information.

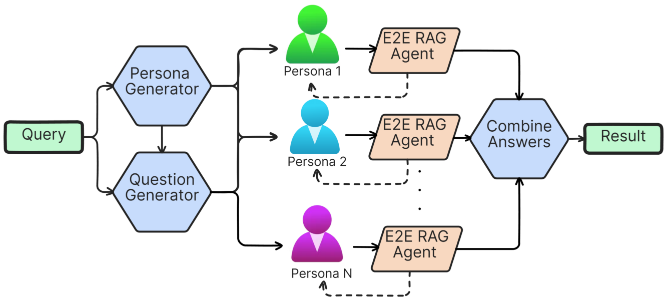


Figure 3. Persona based workflow

5.4.3 Financial Analysis via Key Performance Indicators

This workflow performs financial analysis using Key Performance Indicators (KPIs) by extracting relevant data for specified companies and years. Each analysis requires the retrieval of specific values based on a distinct set of KPIs, which are obtained using our end-to-end RAG agent. We then generate code to calculate the KPIs, and to optimize performance, responses are cached using **Pathway's UDF** wrapper for improved efficiency.

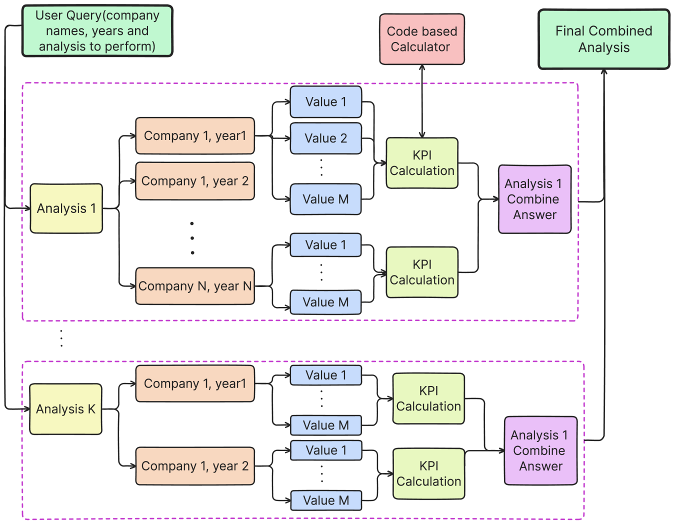


Figure 4. Financial Analysis Agent

5.5 Query Decomposition

We observed that many questions, especially financial queries, are composed of multiple, smaller sub-questions. Retrieval performance is significantly improved when queries are singular and focused, so we implemented 4 different types of question decomposition pipelines.

5.5.1 Parallel & Series-Parallel

We initially employed a *parallel* decomposition strategy, dividing the query into independent sub-questions for concurrent processing. However, this approach proved suboptimal for retrieval performance.

To address this, we adopted a *series-parallel* architecture. Here, each complex parallel stream was further broken down into a series of dependent queries, facilitating easier resolution. This hybrid approach effectively balances efficiency and query handling, optimizing overall performance.

5.5.2 Generator-Critic Framework

The Generator-Critic framework enhances decomposition by introducing a feedback loop. The critic evaluates parallel decomposed questions, providing feedback on suboptimal splits. The decomposer refines the decomposition based on this feedback. This iterative process continues until the critic is satisfied or a retry limit is reached, ensuring continuous improvement. This decomposer is illustrated in Figure 10

5.5.3 ConTReGen

Inspired by ConTReGen [8], we integrated a recursive tree-based architecture into our workflow for dynamically breaking down questions into simpler sub-questions until further decomposition is impossible or the specified tree depth is reached. The goal is to cover the full breadth of documents by simplifying the questions. The implementation of how the answers have been combined has been dealt in detail in the appendix.

5.5.4 RRR - Recursive Residual Resolution Framework

The RRR framework, our *novel* architecture, ensures comprehensive answers by iteratively decomposing and addressing unanswered aspects of the query. After an initial decomposition (max five sub-questions), unresolved parts are used in the creation of new queries, after which the same process is repeated with a maximum decomposition depth of three. The process maintains a history to avoid overlap, and once all aspects are addressed, the sub-question-answer pairs are combined into a complete response, ensuring comprehensive coverage through this iterative ”recursive resolution” process.

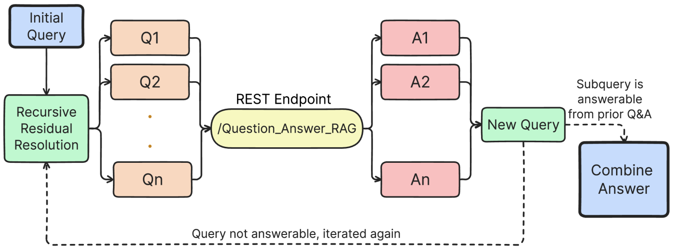

Initial Query
Recursive Residual Resolution
Q1
Q2
.
.
Qn
REST Endpoint
/Question_Answer_RAG
A1
A2
An
New Query
Subquery is answerable from prior Q&A
Combine Answer
Query not answerable, iterated again

Figure 5. RRR framework for comprehensive query answering

These decomposed questions are answered by hitting our RAG end-to-end endpoint, hosted using the **Pathway's** `serve_callable` method, making the entire answering system much more scalable. Through extensive quantitative analysis and multiple experiments detailed below, we conclude that the RRR framework is the most effective solution for our use case.

5.6 Path Decider

Efficient workflow routing directs incoming queries through the most suitable paths, enhancing our architecture's autonomy and efficiency. The system employs two main path-deciding nodes managed by the conversation history combiner. General queries are sent to a general LLM, while finance-related or factual queries go to the first path decider node. Here, queries requiring general information, news, or facts are routed to a web search, whereas finance-related document retrieval queries proceed through the RAG pipeline.

If the RAG path is chosen, a human-in-the-loop stage identifies missing documents, followed by a second path decider node that determines the route. Queries requiring decomposition are sent to the Decompose RAG path, while those needing detailed analysis and reasoning are routed to Persona RAG. This decision is applicable in research mode only.

5.7 End-to-End RAG pipeline

5.7.1 Semantic Caching

After receiving a decomposed query, the RAG pipeline triggers at each endpoint, first checking the **semantic cache**[12] for relevant question-answer pairs. A cosine similarity check is done to reuse cached answers, reducing redundant computations and improving efficiency.

5.7.2 Metadata Extraction

If no cache match is found, the RAG pipeline starts with metadata extraction, which extracts filters for company, year, and intra-document topic tags. If no documents are retrieved, the process is retried without these filters.

5.7.3 Retrieval

We use a hybrid retrieval mechanism combining BM25 and Dense KNN methods to enhance performance, along with metadata filters like company, year, and topic tags. The pipeline is optimized for different query types, with two main components: routing, which determines the retrieval method based on the query's category, and document retrieval, which applies the appropriate filters and strategies.

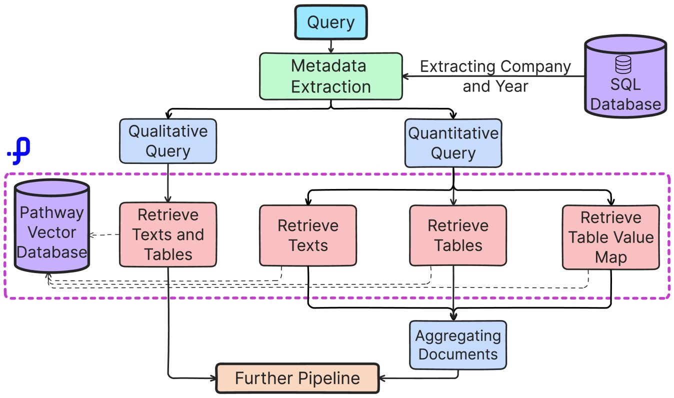


Figure 7. Workflow for Retrieving Relevant Chunks

The retrieval process starts by identifying the query's category to determine the appropriate strategy. If the query is Quantitative, a complex retrieval process is triggered, often involving specialized data like tables. If it's Qualitative, a simpler

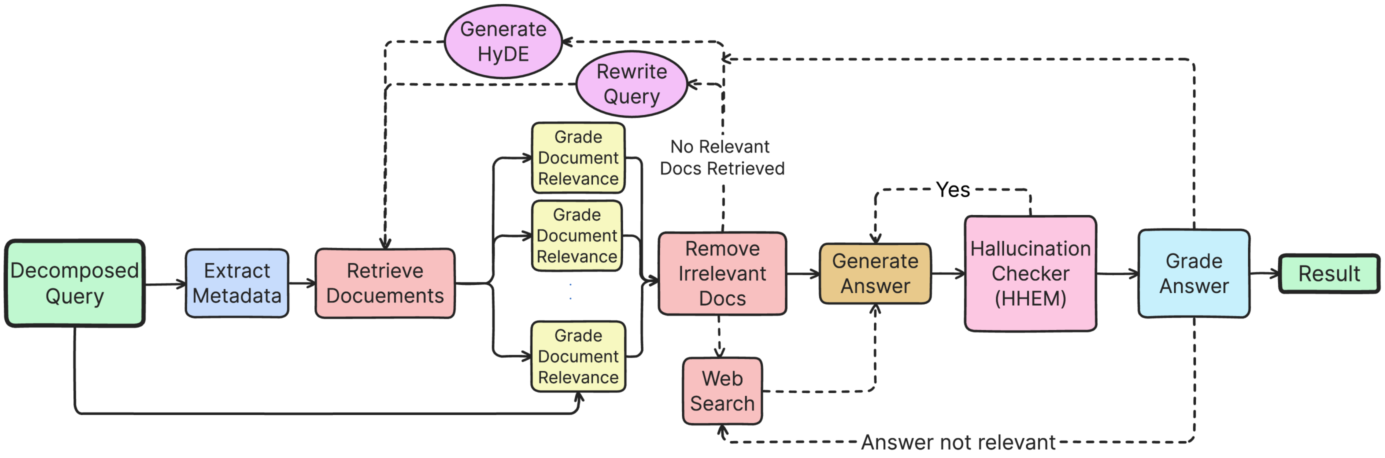


Figure 6. Overview of E2E RAG Agent

process focuses on text-based documents, similar to the midterm architecture.

5.7.4 Document Relevance

After retrieving documents from the database, the document grader node evaluates chunk relevance using a binary scoring system. Chunks deemed relevant are passed for answer generation if they meet or exceed a configurable threshold.

5.7.5 Hallucination Detection and Answer Grading

After every question-answering with the retrieved chunks, we have ensured checks for hallucination. For this, we use the HHEM [9] model which is benchmarked above the likes of `gpt-4` for hallucination detection. The answer is sent along with the query and the supporting documents to grade in binary whether the answer seems to be hallucinated.

Furthermore, an LLM answer grader is employed to assess how well the generated answer corresponds to the original query.

5.7.6 Query Rewriting

Our pipeline implements a comprehensive refinement mechanism triggered by multiple failure conditions. Query rewriting is invoked when: (1) the document grader identifies insufficient chunk relevance, (2) the hallucination grader detects a significant number of hallucinations, or (3) the answer grader determines the generated response is unsatisfactory. Along with standard query rewriting, Hypthetical Document embeddings (HyDE) are also processed in parallel with continuous re-evaluation.

5.7.7 Adding Inline Citation

The state manages a repository of citations for answers generated across various pipeline stages. Once all the sub-answers are combined into the final answer, the `append_citations` maps the cited content to relevant parts in the answer and includes corresponding in-line citations.

5.8 Post Processing

The system processes the output of the RAG agent to generate actionable insights and visualizations, empowering analysts to interpret financial trends more effectively.

The system first evaluates the quality and suitability of the input data, ensuring it meets the criteria for generating meaningful insights. Any unsuitable data is flagged for review. Once the data is validated, the system proceeds to extract key financial metrics and generate actionable insights through a structured post-processing phase. A code-based calculator agent is used to apply financial formulas, transforming raw data into meaningful figures. The system then synthesizes insights to highlight significant trends, anomalies, or opportunities. Finally, the findings are coherently summarized to facilitate interpretation by analysts and support informed decision-making.

The system also generates visual representations of financial data by selecting suitable chart types and preparing the data for visualization. This enables analysts to easily identify patterns and relationships. Together with the quantitative insights, these visual outputs support informed decision-making. By breaking down the process into clear steps, it provides a robust and scalable solution. The workflow for post-processing is provided in Appendix 11.

5.9 Server Architecture

The challenge arises when a document is added to the system, as the indexing process temporarily prevents queries from being processed. To address this, we implement a multi-server approach that combines *fast and slow indexing servers*, ensuring continuous query availability during document indexing.

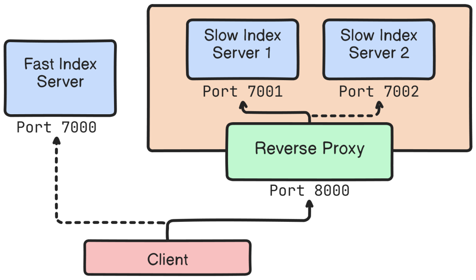


Figure 8. Fast indexing and slow indexing server architecture

Our solution involves one fast indexing server optimized with a lightweight parser for quick indexing, and two slower, resource-intensive indexing servers. When a new document is added, it is sent to one of the slow indexing servers, allowing the other to handle queries. Both slow servers are behind a reverse proxy, which directs queries appropriately. Once indexing is complete, the cache is transferred to the idle server, ensuring continuous query availability. During indexing, queries related to the new document may return inaccurate results, so we query the fast server when one slow server is unavailable. The client queries both servers based on their health status, provided by an additional endpoint at the reverse proxy: `/v1/health`.

We extended Pathway's server and client classes to implement this setup. Specifically, we enhanced the `DocumentStoreServer` to encapsulate the dual-server-with-proxy architecture and modified the Pathway client to handle switching between the fast and slow servers seamlessly.

6. Results and Metrics

6.1 Dataset and Benchmarks

Evaluating a robust RAG system for financial document analysis requires high-quality benchmarks. We explored some of the existing datasets providing foundational resources, but they also present notable challenges. The challenges encountered include limited relevance of pre-2023 data, numerical-only answers, insufficient conversation steps for effective use, lack of response for dataset access, and the unavailability of reasoning-based datasets.

We thus focused on creating high-quality datasets for Simple and Complex Question and Answering (QnA) tasks, addressing challenges related to accuracy and retrieval effectiveness.

6.1.1 Simple Dataset

The Simple Query dataset consists of a question focused on a single entity or concept, a concise answer, relevant context containing the answer, and the associated company. It was prepared using **20** documents with over **200** questions from various companies across different years.

6.1.2 Complex Query Dataset

The dataset was created by calculating cosine similarity between each pair of questions from the simple dataset. Pairs with a similarity above 0.8 were grouped, ensuring a balance between cross-company and intra-company relationships. Using GPT-4o, merged questions and answers were generated by synthesizing the grouped questions into multi-hop queries, with answers derived from relevant sections of the grouped documents. This approach resulted in a dataset capturing complex, multi-hop reasoning across **100+** questions from over **20** financial documents.

6.1.3 Benchmarking with Financial-QA-10k Dataset

For benchmarking, we utilized the "Financial-QA-10k" dataset from Kaggle, which consists of 10,000 question-answer pairs derived from company 10-K filings. The dataset covers a broad range of topics related to financial analysis, company operations, and strategic insights, making it ideal for evaluating natural language processing models in the financial domain. We utilize 500 questions based on the same 20 documents as mentioned above as our testing dataset.

6.2 Metrics for Evaluation

- **Context Precision**: This metric calculates the **accuracy of retrieved contexts**. We retrieve the top k chunks and compare each with the golden context using *Levenshtein distance*. If the similarity score (1 - Levenshtein distance) exceeds a threshold, the context is considered correct.
- **Correctness**: This metric calculates the **correctness of the generated answer** by comparing it to the reference answer. We break the reference answer into simplified facts and verify if each fact is present in the generated answer. The final score is the ratio of correctly generated facts to the total facts in the reference.
- **Faithfulness**: This metric checks for **hallucinations** in the generated answer. Each sentence in the generated answer is simplified, and facts are matched with the contexts. The final score is the ratio of correctly matched facts to the total facts.
- **Response Relevancy**: This metric checks if the **generated answer is relevant** to the question by generating new questions that the answer would address. The cosine similarity between these questions and the original question is then averaged.

In addition, we also look at **latency** (median and 99%ile), and **average token usage**.

We analyzed these metrics for the different experiments that we ran using LangSmith an observability tool. LangSmith also provides us with traces for different runs of our pipeline to debug errors we encountered during the development of our pipeline.

6.3 Results

To identify the optimal model, we conducted ablation studies on key hyperparameters and components to assess their impact on accuracy. For the RAG end-to-end evaluation, we used a simple single-hop dataset, while for end-to-end retrieval, we utilized a complex multi-hop query dataset that we prepared. Context precision is reported only for the relevant experiments in the end-to-end RAG agent evaluation.

6.3.1 Evaluating end-to-end RAG Agent

- **Retreival Techniques**: We evaluated three indexing methods: Semantic Similarity, BM25, and a Hybrid approach combining both. The Hybrid method outperformed the others in all key metrics, as it leverages BM25's term frequency and KNN's semantic similarity. Based on its superior performance, we adopted Hybrid indexing for all subsequent experiments.
- **Varying *k* in Top-K Retrieval**: Using Hybrid indexing, we conducted experiments with various values of k for top-k retrieval. Our analysis revealed that $k = 5$ strikes the optimal balance between performance and efficiency, particularly in terms of latency. Consequently, we adopted $k = 5$ as the standard for all subsequent experiments.
- **Ablation studies for Retrieval Agent**: Each row represents the addition of a component over those mentioned in previous rows. The progressive inclusion of components over the Naive RAG significantly improves both accuracy and contextual understanding. The Reranker improves response relevancy by prioritizing higher-quality

results. Metadata filtering narrows the search space, increasing relevance, while the table-value map refines results, enhancing correctness and faithfulness. The Doc Relevance module, particularly with feedback, aligns responses more closely with query intent. Hyde improves contextual understanding but increases latency due to its reasoning process. The Answer Grader provides the greatest performance boost, ensuring accuracy and relevance at the cost of higher computational overhead.

- **Table Value Map:** One of the key innovations in our system is the integration of the table-value map, designed to address gaps in the retrieval of numerical data. To validate its impact, we conducted experiments with our final RAG agent, toggling this feature on and off. The results demonstrated its effectiveness, showing a notable 19% improvement in correctness accuracy, underscoring its value in enhancing retrieval precision for numerical information.
- **Embedding Models**: We tested three embedding models: ada-002, voyage-finance-2, and bge-m3. While ada-002 and bge-m3 are general-domain models, bge-m3 (an open-source model) outperformed the others. Despite being finance-specific, voyage-finance-2 performed poorly on financial datasets, so we chose bge-m3 for final evaluations.
- **Cache Compression:** To optimize token usage, we compress the chunks during indexing in order to get rid of unnecessary information while retaining all necessary facts. The results show that compression reduces the average input tokens significantly, maintaining accuracy within a 2% margin.

6.3.2 End-to-end Retrieval ablation studies

- **Decomposer Comparisons**: We compare our decomposer architectures [1] to identify the best-performing model. Our midterm solutions, which included only parallel and basic series-parallel architectures, showed improvement over Naive RAG. However, our post-midterm implementations of the generator-critic framework and RRR produced better results, with RRR outperforming Naive RAG on multi-hop questions by 39%.
- **RRR using Different Embedding Models** : A key observation from our experiments is that using different embedding models with Naive RAG resulted in significant performance variations. However, when the same experiments were conducted using RRR, the results were independent of the embedding model used. Both BGE-M3 and Ada produced similar outcomes, showing that our final model is embedding model agnostic.
- **Varying LLMs**:Varying the LLM models shows that the correction accuracy remains consistent within a 2% variance, demonstrating that our end-to-end workflow is independent of the specific LLM model used. This ensures that switching between different LLMs, when one model encounters issues, does not compromise accuracy.

6.3.3 Finance Benchmarks

We conducted our evaluation on the Financial Q&A - 10k benchmark, a widely recognized dataset for testing systems in the finance domain. Our enhanced RAG framework demonstrated a substantial improvement across all key metrics — correctness, faithfulness, and response relevancy—when compared to the baseline Naive RAG approach. This improvement underscores the significance of incorporating advanced components such as metadata filtering, table-value mapping, and reasoning-based enhancements in creating a robust and reliable RAG system for finance applications.

7. Challenges faced and solutions

7.1 Query Handling During Precise Indexing

A key challenge was the inability to handle queries during indexing. While Pathway's dynamic indexing is fast, enhancing the indexer with our method for more efficient retrieval using metadata filtering and table value maps introduced delays. To resolve this, we implemented a dual-server approach: a fast server using Pathway's lightweight indexer for immediate querying during the initial phase, and a precise server for enhanced retrieval once detailed indexing is complete.

7.2 Token Usage Optimization

- **Leveraging Microsoft's LLM Lingua for Prompt Compression:** We utilized **LLM Lingua** [5, 7] to compress prompts and text chunks before indexing. This reduced token usage while preserving critical information. Details are provided in Appendix B.1
- **Implementing Compressed Document Stores in RAG:** Building on LLM Lingua, we compressed parsed context chunks for storage in the vector database. This *Compressed Document Store* was implemented as an extension of the Pathway Document Store, integrating compression as a *transformation* over parser outputs.
- **Fine-Tuning GPT-4 for Token Efficiency:** For larger prompts and complex queries, we fine-tuned **GPT-4** with tailored examples and a system prompt, reducing token consumption while maintaining response quality.

7.3 Structured Output for General LLMs

This solution was pivotal for generalizing across all LLMs. While LangChain supports Google's Gemini, its structured output implementation remains suboptimal [10].

To obtain structured output from LLMs without predefined methods in LangChain, we used the `PydanticOutputParser` for LLaMA and `Instruct` for Gemini. Hosted on Replicate, LLaMA lacks structured output, and we often encountered invalid JSON with the LangChain's Pydantic parser. To address this, we created a custom parser that integrates the `OutputFixingParser` with the Pydantic Parser, automatically correcting faulty JSON. To overcome Instruct's lack of Runnable compatibility with LangChain Expression Language we created a wrapper class based on LangChain's `BaseModel`. This solution allows easy integration of not only LLaMA or Gemini but also all other models supported by Replicate and Instruct with LangChain.

[1] Latency provided in Results table is subject to local environment

Method	**Correctness**	**Faithfulness**	**Response Relevancy**	**Context Precision**	**p50 Latency (s)**	**Avg. Token Usage**
Indexing Methods						
Semantic	0.58	0.82	0.98		11.16	3494.52
BM25	0.64	0.88	0.99		10.68	4194.75
Hybrid	**0.68**	**0.89**	**0.99**		**9.12**	**4174.94**
Varying k in top-k						
$k = 3$	0.65	0.84	0.99		8.18	3571.83
k = 5	**0.68**	**0.89**	**0.99**		**9.12**	**4174.94**
$k = 7$	0.69	0.89	0.99		11.50	4189.58
$k = 9$	0.66	0.92	0.99		14.22	4460.59
Ablation Study for Retriever						
Naive Rag	0.52	0.88	0.74	0.65	2.74	1033.31
+ Metadata filtering	0.52	0.83	0.93	0.65	27.11	2314.92
+ Reranker	0.51	0.92	0.79	0.65	6.49	2309.84
+ Metadata filtering with table-value map	0.71	0.96	0.91	0.75	29.73	4548.62
+ Doc Relevance Without Feedback	0.74	0.97	0.79	0.65	12.05	3176.73
+ Doc Relevance With Feedback	0.72	0.95	0.92	0.71	39.78	3162.36
+ with Hyde	0.69	0.96	0.92	0.73	40.68	3150.42
+ Answer Grader	**0.81**	**0.99**	**0.99**	**0.75**	**48.45**	**4221.19**
Addition of table-value map and Quantitative Qualitative Routing						
Without table value map	0.72	0.95	0.99	0.69	66.05	3120.88
With table value map	**0.81**	**0.99**	**0.99**	**0.75**	**48.45**	**4221.19**
Finance specific Embedding Model on Naive RAG all single hop						
text-embedding-ada-002	0.49	0.91	0.73	0.65	2.80	1032.72
voyage-finance-2	0.41	0.88	0.59	0.61	2.67	885.4
bge-m3	**0.57**	**0.92**	**0.76**	**0.65**	**5.27**	**1406.52**
Decomposer Experiments						
Naive	0.33	0.59	0.53		5.08	92,550
Only Parallel	0.37	0.39	0.89		38.40	17210.40
Basic Series Parallel	0.38	0.38	0.88		40.65	17026.44
Generator Critic - for series and parallel	0.42	0.35	0.93		44.66	20583.39
RRR	**0.72**	**0.80**	**0.93**		**83.50**	**42398.7**
RRR using Different Embedding Models						
text-embedding-ada-002	0.72	0.80	0.93		83.50	42398.70
bge-m3	0.70	0.86	0.86		108.67	55,118.88
Varying LLMs (using RRR on multi hop dataset)						
OpenAI (GPT-4o)	0.72	0.80	0.93		83.50	42398.7
Anthropic (Claude 3.5 Haiku)	0.72	0.80	0.88		195.93	41219.37
Mixtral (8x7b)	0.69	0.81	0.88		260.14	39157.64
Llama (70b)	0.70	0.79	0.81		187.88	36084.52
Financial Q&A - 10k Benchmark (A single hop dataset)						
Naive	0.37	0.78	0.49		2.94	1131.49
RRR	**0.76**	**0.88**	**0.88**		**47.81**	**3440.48**
With Compressed Chunks (On a smaller dataset with 6 documents)						
Uncompressed	0.79	0.86	0.95		21.47	4221.19
60% Compressed Text, 40% Compressed Table and Uncompressed KV	0.79	0.83	0.92		13.81	3187.17
40% Compressed Text, 20% Compressed Table and Uncompressed KV	0.77	0.87	0.95		16.18	3152.53
20% Compressed Text, 20% Compressed Table and 20% Compressed KV	0.68	0.86	0.87		16.44	3769.30

Table 1. All Results

7.4 Logging Tree Construction

The `app.get_state` feature provided by LangGraph facilitates the retrieval of states across events but does not inherently supply information about parent nodes. This limitation posed a significant challenge, as the parent node information is essential for constructing the logging tree, a core component of our interface. We addressed this challenge by introducing a dedicated `log_tree` variable within the system state, necessitating precise tracking and update mechanisms at each node to maintain logging tree integrity.

Hierarchical information sharing encountered additional complexity due to incompatibility between custom classes and LangGraph's current serialization infrastructure. To mitigate this limitation, we implemented an explicit serialization and deserialization protocol for custom class objects during state read and write operations. Custom reducer functions were developed to enable seamless information aggregation across parallel execution pathways.

7.5 Bounding Box Issue With OpenParse

Openparse uses bounding boxes to parse document content, classifying them as tables or text chunks. However, evaluation revealed hallucinated quantitative answers due to missing data and incorrect table value mapping, caused by inaccurately defined bounding boxes. To address this, we provided the entire page to the Vision LLM for detailed table parsing and separated **Table value map** chunks[5.1], achieving a **19%** improvement in accuracy. Despite this, accuracy still relies on the Vision LLM's ability to extract structured text from table images effectively.

7.6 Metadata Extraction Issues

Our metadata filtering handles companies with multiple names (e.g., ”Apple” vs. ”Apple Inc.” or linking ”YouTube” to its parent ”Alphabet.”) by combining LLM parametric knowledge with a **Postgres database** of company names extracted from user documents. During indexing, the first 10 pages are scanned for company names, which are stored in the database. The LLM then matches query company names with the most relevant database entry, improving retrieval accuracy.

7.7 Handling summarization Questions

Previously, our RAG decomposer agent struggled with handling summary-type questions, as highlighted in our midterm submission. To address this limitation for our final submission, we implemented two key approaches:

- **Persona RAG:** Persona RAG enables splitting summary-type questions into sub-questions, each addressed by different personas with unique perspectives. This approach is effective for complex queries, as it breaks them into smaller parts, allowing each persona to generate focused responses that improve the depth and relevance of the overall answer.
- **Query Expansion:** This helps the decomposer agent handle summary or open-ended questions. The agent automatically reformulates queries by expanding their scope or adding context, enabling it to address broader or more abstract queries.

8. Resilience to Error Handling

Several mechanisms were implemented to manage service failures, handle callback errors, and ensure continuity in workflow execution:

- **LLM Fallback Mechanism:** We created a unified LLM interface supporting Anthropic, Mistral, and Llama models, ensuring consistent structured output. The custom LLM class enables dynamic model switching, transitioning to open-source models if API access to closed-source models fails, maintaining workflow continuity. Structured output details are in Section 7.3.
- **Diverse Web Retrieval Strategy:** Web information retrieval follows a fallback system: Tavily is the primary method, with Google Search and scraping as the secondary option. If both fail, Bing Search API is used. Scraped content is processed by removing non-essential elements and converting text to markdown for LLM use.
- **Retry Management:** To prevent infinite loops during query refinement, the system sets configurable retry limits with different thresholds for fast and precise modes. If the limits are reached, alternative strategies like web search and tool-specific fallbacks are activated.
- **Workflow Checkpointing:**
  The error recovery mechanism uses checkpointing to handle node-level failures. The system retrieves the last known state via a checkpoint_id, allowing precise workflow resumption and reducing recovery time.
- **Parsing Reliability:** For complex parsing tasks, such as table extraction from PDFs using Pathway's Openparse, we employed a visual LLM-based text extraction approach. The `ExponentialBackoffRetryStrategy` was implemented to manage potential parsing errors, progressively adjusting retry intervals to enhance parsing reliability.

9. User Interface

9.1 UI features

- **Chat Interface**: The layout is intuitive and clean, featuring a chat space for conversation threads. Users have the option to upload files and select different run modes.
- **Conversations and Workspaces**: The interface supports conversational history and workspace management, allowing conversation persistence per user or per workspace.
- **Multimodal Input**: Our system supports input in various forms, including text, images, and speech, enabling a highly versatile and multimodal experience with autocomplete suggestions.
- **Citations**: Citations are shown for each answer. Users can hover over them to view the relevant chunk, and the document on the right displays the relevant page.
- **Tracing/Visualizations**: Logging of nodes allows users to trace and visualize the exact paths taken by the agents and the system. This enhances explainability and transparency. The visualization mode can be toggled on or off.

The UI features will be demonstrated more clearly in the the accompanying video.

9.2 Implementation Details

The web application leverages modern web technologies, integrating real-time communication, data visualization, and advanced state management to deliver a seamless user experience. The front end is built with **React 18** and **Vite**, featuring a modular structure with reusable components. It utilizes **Context API** for global state management and includes utility functions for efficient API integration. The design is fully responsive, powered by **Tailwind CSS**, while **Framer Motion** adds smooth animations. Accessible UI components ensure an enhanced user experience across various devices. Real-time functionality is achieved through **WebSocket**-driven live chat updates, dynamic graph visualizations, and drag-and-drop file management.

On the backend, **FastAPI** handles API requests and WebSocket communication, while **SQLAlchemy ORM** is used for database interactions with asynchronous operations. **Alembic** manages database migrations, ensuring smooth updates. **JWT authentication** secures the platform, complemented by input validation and secure file handling mechanisms. Performance is optimized using lazy loading, efficient database queries, and caching strategies, ensuring scalability. This architecture results in a highly scalable, maintainable, and secure web application, delivering a superior user experience.

10. Responsible AI Practices

Our RAG system is designed to align with the Responsible AI Standard [6], incorporating robust mechanisms to ensure ethical and responsible deployment. Below, we outline the key guardrails implemented to uphold these principles:

10.1 Hallucination Mitigation and Citation Integrity

We mitigate hallucinations at every generation step using the HHEM model to evaluate and discard reasoning artifacts, enhancing reliability. Final outputs include citations, with references rigorously checked for relevance. Relevant document chunks are stored and cited explicitly to ensure transparency and trust.

10.2 Safety Assurance Mechanisms

Safety is a top priority in our pipeline, with a dedicated safety agent ensuring compliance with the MLCommons AI Safety Benchmark [11][11]. Unsafe queries are flagged and exited, while permissible ones are rephrased to maintain user intent while following safety guidelines.

10.3 Reliable Web Searching Framework

To enhance source reliability, our system screens search results in alignment with user-defined preferences. Users can specify sites to exclude, and these preferences persist across sessions leveraging Pathway's UDFs for caching, ensuring alignment with individual trust requirements. This approach enhances transparency and allows users greater control over the sources utilized for their queries, fostering adherence to responsible AI practices. Further details are provided in Appendix A.5.

11. Lessons Learned

11.1 Insights Gained

Through the development of our Retrieval-Augmented Generation (RAG) framework, we gained several key insights with significant implications for the future of language models and AI architectures.

- **Effectiveness of Open-Source Models**: Our comparison of the BGE-M3 open-source model with OpenAI's Ada embedding models revealed that open-source models often outperform proprietary models. In addition, open-source models provide flexibility for customization and adaptation.
- **Shift Towards Domain-Specific RAG Architectures**: We observed that general-purpose LLMs are less effective for tasks requiring domain-specific knowledge. Our findings suggest that multi-agent RAG architectures are the next step in AI development. By distributing processing across specialized agents, these architectures can provide more accurate and relevant responses, offering a pathway towards more scalable and efficient AI systems.
- **Importance of Reasoning in AI Models**: Models that incorporate reasoning — performing intermediate steps before answering — demonstrated greater accuracy and reliability. Integrating reasoning processes in RAG frameworks allowed models to better understand context and generate more precise answers.

11.2 Potential Enhancements

Some areas for enhancement have been identified that could further improve the functionality, compliance, and usability of the system.

- **Improved Data Deletion for GDPR Compliance**: Currently, Pathway does not remove cached embeddings when a document is deleted; it only removes data from the table. This results in the embeddings being available again once the document is re-uploaded, even though the document is no longer queryable. To align with GDPR regulations and increase transparency, it is necessary to delete embeddings permanently when a document is deleted.
- **Domain-Specific Personas**: While our current implementation uses general personas for reasoning, there is an opportunity to finetune these models for specific domains, such as finance. This would enable the system to provide more targeted and contextually relevant responses, enhancing the overall user experience in specialized use cases.
- **Expanded Output Modalities**: To further improve the usability of the system, additional output modalities like speech can be integrated. Furthermore, incorporating capabilities such as scraping links and providing summaries would enhance the system's ability to deliver comprehensive, multimodal responses.

12. Conclusion

Developing multi-agent RAG systems for finance is an intricate task that demands addressing challenges like latency, explainability, and domain-specific complexity. Our approach integrates cutting-edge solutions, ensuring scalability, precision, and robustness.

To handle metadata extraction and filtering, we incorporate methods that distinguish between static metadata (e.g., company names and years) and dynamic intra document topic tags, optimizing relevance and retrieval accuracy. For numerical data, table map values are employed to streamline inefficiencies, ensuring precise data extraction and alignment with financial reporting standards.

To address latency challenges, we devised a triple-server architecture that maintains seamless query handling while running a more efficient indexing method in the background. This ensures rapid response times without compromising performance or scalability.

The system offers multiple tailored workflows, including Recursive Residual Resolution (RRR), a refined decomposer RAG agent that enhances performance by resolving residual inconsistencies; persona-based workflows for personalized, domain-specific insights; and KPI-driven financial analysis workflows that extract and analyze key metrics for actionable outcomes. Together, these components power FinSight, a comprehensive financial insights platform that combines cutting-edge retrieval-augmented generation with domain-specific intelligence. While primarily designed for finance, FinSight is domain-agnostic, adaptable to any field through its flexible persona workflow.

Robustness is further enhanced through fallback mechanisms integrated at multiple stages. These include alternative fallback LLMs, diverse web retrieval strategies, retry management protocols, and workflow checkpointing to prevent bottlenecks or failures. A key achievement is the consistency of our results: *regardless of the LLM* used, we achieve performance with less than 2% variance, ensuring reliability and scalability across different models.

A. Appendix

A.1 Key Performance Indicator Design

To support a comprehensive evaluation of financial data, we developed an extensive KPI (Key Performance Indicator) workflow organized across multiple analytical domains. This workflow facilitates granular and actionable insights into financial, operational, and strategic performance. The KPIs are categorized into thematic topics for targeted analysis:

1. **Balance Sheet Reviews**: Liquidity and capitalization assessment through asset-to-liability ratios and asset growth metrics.
2. **Income Statement Reviews**: Revenue trends and profit variability analyzed via return on investments and earnings margins.
3. **Cash Flow Analysis**: Cash cycle efficiency and projection accuracy using cash-to-debt ratios and liquidity metrics.
4. **Industry Analysis and Benchmarking**: Competitive positioning and sector comparison with profitability and market growth benchmarks.
5. **Trend Analysis**: Financial trajectory evaluation with nonlinear regression and revenue growth trends.
6. **Working Capital Analysis**: Operational capital efficiency via short-term asset-liability ratios and liquidity metrics.
7. **Liquidity Analysis**: Financial flexibility analysis through cash flow models and obligation coverage ratios.
8. **Profitability Analysis**: Profit generation efficiency via gross profit margins and return-on-assets metrics.
9. **Solvency Analysis**: Long-term financial health assessment using debt-to-equity ratios and debt sustainability metrics.
10. **Break-Even Analysis**: Cost-revenue balance determination with contribution margins and sales volume metrics.
11. **Capital Structure Analysis**: Financial leverage optimization using weighted cost of capital and equity ratios.
12. **SWOT Analysis**: Strategic position evaluation using revenue growth and customer satisfaction indices.
13. **Scenario Analysis**: Outcome modeling and risk mitigation using Monte Carlo simulations and cash flow projections.
14. **Customer Acquisition Analysis**: Cost of customer acquisition and conversion efficiency using CAC and CLV-to-CAC ratios.
15. **Customer Lifetime Value Analysis**: Revenue from customer retention and value generation via churn rate and cross-sell rates.
16. **Employee Productivity Analysis**: Workforce contributions and training impact with task completion rates and engagement scores.
17. **Valuation Analysis**: Financial worth and supply chain efficiency using P/E ratios and productivity metrics.
18. **Risk Analysis**: Exposure to operational, financial, and market risks.
19. **Profitability Driver Analysis**: Contributions to profit and benchmark performance with operating margins and industry standards.
20. **Company Debt Analysis**: Financial leverage, debt sustainability and compliance with interest coverage ratios and credit ratings.
21. **Evaluation Efficiency**: Operational efficiency relative to benchmarks using revenue-per-employee and variance metrics.

A.2 Exact Fallback Mechanism for Missing Documents

- **Company and Year Extraction**: Used spaCy's Named Entity Recognition (NER) to extract company names and years.
- **Fallback Company Identification**: In cases where spaCy failed, we matched company names against a preloaded list of 500 companies using case-insensitive comparisons.
- **Document Retrieval**: We utilized pdfkit to download the 10-K report of the identified company for the specified year from the internet.

A.3 System Operational Modes

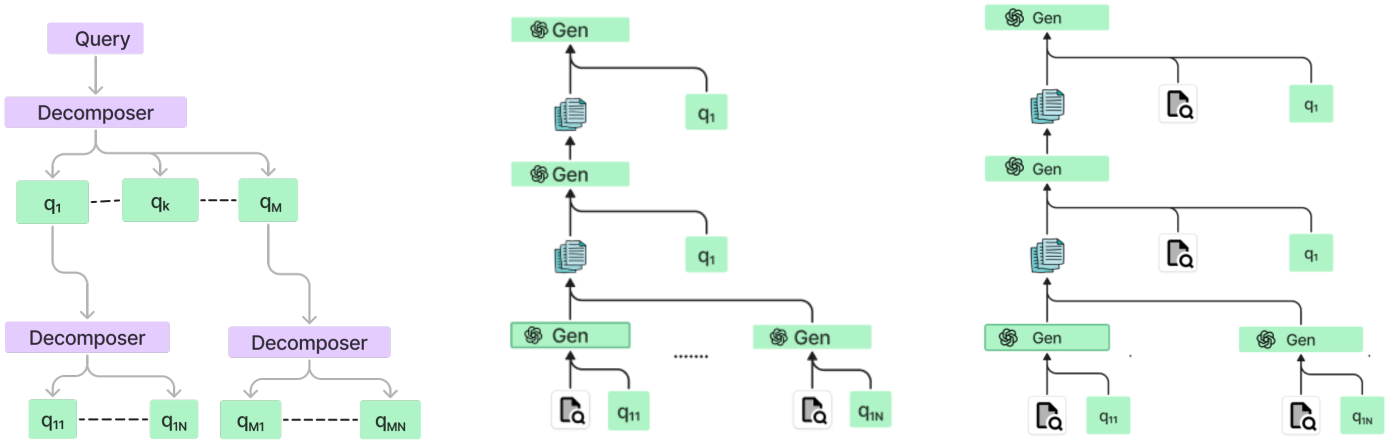
Our system operates in two distinct modes—**Fast Mode** and **Precise Mode**—to balance efficiency and depth of interaction. These modes dynamically adjust operational parameters to optimize performance based on user preferences or task requirements. The key differences are as follows:

- **Depth of Query Decomposition**: Fast Mode restricts hierarchical query decomposition to fewer(2) levels, ensuring rapid response generation, while Precise Mode allows deeper decomposition (3 levels) for a more detailed understanding of user intent.
- **Contextual Memory Retrieval**: Fast Mode retrieves up to 3 previous messages for conversational context, while Precise Mode expands this retrieval to include 5 previous messages.
- **Persona Generation**: Fast Mode generates a maximum of 3 personas, whereas Precise Mode supports up to 4 personas for a more comprehensive interpretation of the query. Personas in Precise Mode are permitted to ask more questions.
- **Clarifying Questions**: Fast Mode prioritizes efficiency by restricting the clarifying questions to only address significant ambiguities. In contrast, Precise Mode permits a wider variety of questions, enabling detailed exploration and reduced residual uncertainty.

This comparative framework ensures that users can select the mode best suited to their specific needs, whether prioritizing speed and efficiency or depth and detail.

A.4 Experimenting on Decomposition

Apart from the main workflow for decomposition (RRR), we implemented 2 other architectures. Firstly is contregen, where the primary objective is to construct a recursive tree-based architecture capable of dynamically breaking down questions into simpler sub-questions. This process continues until further decomposition is either impossible or the specified tree depth is reached. After the decomposition, we implemented 2 frameworks for answering the question. ConTReGenv1 (as shown in the paper) had document retrieval at each node of the tree, and the answers were combined while going up the tree, and ConTReGenv2 (our implementation) had the document only in



(a) Decomposition of questions **(b)** Answering segment of ConTReGenv2 **(c)** Answering segment of ConTReGenv1

Figure 9. Visual representation of implementation of the ConTReGen framework

the last layer. The implementation is explained through Figure 9. Further we had also implemented the Generator-Critic workflow, where the decomposer works on the feedback given by a critic and updates its decomposition. The workflow is shown through figure below.

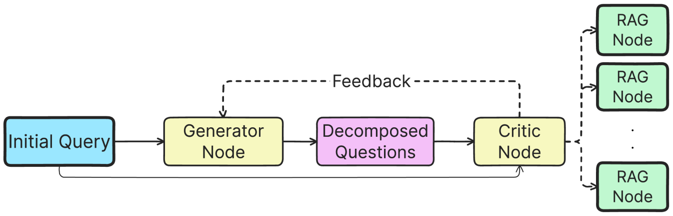


Figure 10. Generator-Critic decomposed workflow

A.5 Explorations in Responsible AI

Responsible AI is a critical aspect of building AI applications that is often overlooked. It encompasses key principles such as Accountability, Transparency, Fairness, Reliability and Safety, Privacy and Security, and Inclusiveness. These principles can be approached through techniques like prompt engineering, data safeguarding for training, and embedding guardrails in models.

Our team is committed to addressing each of these elements within our solution framework. Through research and initial implementation, we have begun incorporating responsible AI practices. Our focus has been on red-teaming approaches till midterm, searching for potential issues.

A.5.1 Common Challenges

- Malicious actors attempting to exploit AI systems.
- AI inadvertently generating harmful content due to biased training data.
- Hallucinations leading to lack of confidence in users

A.5.2 Literature Review Conclusion

There are various methods to ensure Responsible AI. MetaAI offers robust safeguards with open-source models. We plan to integrate their models alongside prompt engineering to promote responsible AI. Our work till midterm focused was on red-teaming efforts, building foundational solutions.

A.5.3 Experiments

Experiments focused on red-teaming various models based on the paper [13].

Designing Prompts Since we are not working with LLM fine-tuning, our red-teaming and solution efforts primarily involve prompt engineering.

Using GPT-4 The model could generate prompts, but they were generally safe and compliant.

Using Gemini Pro

- Initial tests with safety on resulted in refusal to generate certain prompts.
- When safety was turned off in AIStudio, it generated prompts capable of jailbreaking models.

Evaluation on Experiments Our evaluation of various models, including GPT-4o, LLaMA, and Mistral, demonstrated differing levels of resilience to adversarial prompts, with LlamaGuard adding an effective layer of safety across models.

- **GPT-4o**: The model primarily generated safe responses, maintaining high compliance with safety standards and proving resistant to most adversarial prompts.
- **LLaMA (all sizes)**: Even the smaller models in the LLaMA series generally produced safe outputs and successfully resisted jailbreak attempts.
- **Mistral**: This model was relatively easy to jailbreak, showing higher susceptibility to adversarial prompts compared to GPT-4o and LLaMA.

Overall, GPT-4o and Llama showed strong safety compliance, especially when augmented with LlamaGuard. Mistral's vulnerability to jailbreaking highlights areas for improvement, particularly in filtering adversarial prompts effectively. This shows that a layer of LlamaGuard on any model can make it safer.

In conclusion, there exists a trade-off with time; the more agents applied, the costlier the operation becomes. We optimized our architecture to address this effectively.

A.6 Reasons for not using other benchmarks

- **FinQA**[2]: provides financial data but documents are primarily from before 2023, limiting relevance for current financial analysis. Large document corpus requires extensive caching, affecting processing efficiency.
- **ConvFinQA**[1]: focuses on quantitative datasets, offering only numerical answers, which makes it unsuitable for qualitative or reasoning-based queries. Additionally, the steps in the conversation are not sufficiently well-defined for effective use.
- **FinBen**[15]: is generally usable but faced access limitations. We did not receive full access to the dataset, which hindered our ability to fully evaluate its potential.
- **FinanceBench**[4]: was a promising option, but we did not receive a response to our inquiry for access, limiting further exploration of this benchmark.

B. Workflow Demonstrations

B.1 Compressed Prompt Example

Original Prompt (431 tokens): You need to give 2 things for this chunk ensure no field is left empty: 1. Please give a short succinct context to situate this chunk within the overall document for the purposes of improving search retrieval of the chunk. 2. Please assign one broad Topic from the list of broad topics you have assigned already to some chunks of this document, that best describes the content of this chunk. If you cannot find any topic that fits this chunk, then return a new BROAD topic that best describes the content of this chunk.

Remember you have assigned num_topics topics till now for the chunks of this document. If the above number is ˜10, then keep care of the new topics you are assigning to the chunks of this document and make them very broad so that they can be used for the future chunks of this doc as well. If the above number is ˜1-2, then do not worry about the new topics you are assigning to the chunks of this document and make them very specific to the content of the chunk.

Compressed Prompt (290 tokens): You to give 2 things chunk ensure no is empty. Please a short succinct to the document of improving search retrieval the Please assign one broad Topic the of topics you have assigned to some chunks document, that the you cannot any this then return new BROAD that Remember have assigned _ topics the the above is ˜10 then keep the them broad the future chunks of this doc as well. If the above number is ˜1-2, then do not worry about the new topics you are assigning to the chunks of this document and make them very specific to the content of the chunk. **Resulting in a compression ratio of 1.5x**

B.2 Recursive Residual Resolution Example

This section provides a step-by-step demonstration of our RAG agent's capabilities using a representative example question related to the analysis of financial documents. The following example highlights the end-to-end workflow of the agent, illustrating its ability to retrieve, analyze, and generate accurate and insightful responses.

Question: "In the 2023 financial reports, what was the total amount of fixed lease payment obligations for Apple as of September 30th and the total amount of future lease payments under operating leases for Google as of December 31st?"

Decomposed Question-Answer pairs

- What was the total amount of fixed lease payment obligations for Apple as of September 30, 2023?
  - As of September 30, 2023, Apple had fixed lease payment obligations totaling $15.8 billion, with $2.0 billion payable within 12 months
- What was the total amount of future lease payments under operating leases for Google as of December 31, 2023?
  - As of December 31, 2023, the total amount of future lease payments under operating leases for Google was $17.7 billion. This total includes all individual yearly payments and reflects a weighted average remaining lease term of 8.1 years and a weighted average discount rate of 3.1% .

Then we implement an aggregator node which checks if the subquestion-answer pairs fully answer the original question.

Aggregator

- sufficient_answer: No
- justification: The subquestions provide the total amount of future lease payments under operating leases for Google, which includes all individual yearly payments. However, the breakdown of these payments is not explicitly provided, which is a significant part of the original question.

Since we do not have a sufficient answer yet, the aggregated information is used to generate a new question which just deals with the unknown information.

New Question : What are the individual yearly payments and the breakdown of the future lease payments under operating leases for Google as of December 31, 2023?

This question is then passed into the decomposer node again, but now it also takes into account the history of subquestions already asked, so as to not ask similar questions again which provides further subquestion-answers

Decomposed Question-Answer pairs:

- What are the individual yearly lease payments for Google for the next five years as of December 31, 2023?
  - As of December 31, 2023, the individual yearly lease payments for Google for the next five years are as follows: 2024: $3,179 million, 2025: $2,929 million, 2026: $2,450 million, 2027: $1,951 million, 2028: $1,488 million
- What is the total amount of variable lease payment obligations for Google as of December 31, 2023?
  - The query cannot be answered with the provided context,
- What are the specific terms and conditions of the operating leases for Google as of December 31, 2023?
  - As of December 31, 2023, Alphabet Inc. (Google) had the following specific terms and conditions regarding their operating leases:
    1. Weighted Average Remaining Lease Term: 8.1 years.
    2. Weighted Average Discount Rate: 3.1%.
    3. Total Future Lease Payments: $17.7 billion, which includes: 2024: $3,179 million , 2025: $2,929 million, 2026: $2,450 million , 2027: $1,951 million, 2028: $1,488 million , Thereafter: $5,685 million.
    4. Less Imputed Interest: $2,431 million, which is deducted from the total future lease payments.

Aggregator

- sufficient_answer: Yes
- justification: The subquestions provide the total fixed lease payment obligations for Apple and the total future lease payments under operating leases for Google, which fully addresses the first part of the original question. Additionally, the individual yearly lease payments for Google over the next five years are also provided, addressing the comparison aspect of the original question. Therefore, all parts of the original question are answered.

Now that the aggregator has given the answer to be sufficient, the combine answer node then take in the history of subquestion answer pairs so far with the original question to produce this final answer.

Final Answer (with citations):

"In the 2023 financial 10-K reports, IBM's Infrastructure revenue decreased from $15,288 million in 2022[[1/IBM.pdf/21]] to $12,593 million in 2023 [[2/IBM.pdf/21]], resulting in a year-to-year percentage change of -4.5% [[3/IBM.pdf/21]]. Additionally, Microsoft's total revenue for the fiscal year ending June 30, 2023, was $211.915 billion [[4/MSFT.pdf/93]]."

B.3 Persona Analysis

Question: "Analyse and compare the revenues of Apple and Google for the year 2022"

For analysis and reasoning based queries, the path decider routes them to Persona workflow, where different analyst personas are created to discuss the query and arrive on a satisfactory answer. For the above query, these personas are created:

- **Financial Analyst**: Specializes in financial performance metrics, focusing on revenue trends, including product and service streams for 2022.
- **Market Analyst**: Focuses on market trends and external factors shaping Apple and Google's 2022 revenues.
- **Technology Analyst**: Examines technological advancements like product launches and innovations influencing 2022 revenue.
- **Strategic Analyst**: Evaluates strategic decisions such as partnerships and acquisitions impacting 2022 financials.

For each of the expert analyst, following persona specific question are generated to answer:

- What were the total revenues of Apple and Google in 2022, and how do these figures compare in terms of growth rates year-over-year?
- What market trends and consumer behaviors impacted the revenues of Apple and Google in 2022, and how did their market shares change during the year?
- What key technological advancements or product launches did Apple and Google introduce in 2022, and how did these innovations contribute to their revenue generation?
- What strategic decisions did Apple and Google make in 2022 that influenced their revenues, including any mergers or partnerships, and how did these decisions affect their overall financial performance?

Final Answer

Total Revenues and Growth Rates: Apple reported total revenues of $394,328 million, an 8% increase from $365,817 million in 2021, driven by iPhone sales ($205,489 million) and services ($78,129 million). Google generated $282,836 million in revenues, with significant contributions from Google Search, YouTube ads, and Google Cloud. Growth was noted across Google's key segments.

Market Trends and Consumer Behaviors: Macroeconomic factors like inflation and rising interest rates dampened consumer spending in 2022. Competition from emerging tech companies and shifts toward subscription services and eco-friendly products influenced market shares and purchasing patterns for both Apple and Google.

Technological Advancements and Product Launches: Apple launched iOS 16, macOS Ventura, and new hardware, boosting revenue. Google updated Android and introduced products like Pixel phones and Nest devices, contributing to its revenue streams.

Strategic Decisions: Apple modified its App Store policies, potentially affecting app developer revenue. Google's strategic focus on advertising and Cloud services, alongside partnerships, supported its financial performance.

In summary, Apple led with $394,328 million in total revenues and an 8% growth rate, driven by strong iPhone sales and services. Google reported $282,836 million in revenues, reflecting solid performance across search, advertising, and Cloud segments. Both companies navigated challenges from economic conditions, increased competition, and shifting consumer preferences, leveraging innovations like software updates and new hardware alongside strategic moves to shape their financial outcomes in 2022."

C. Post Processing

We show the detailed workflow of the insight and chart generator post processing agent in 11.

User Query: "Compare trends in revenue, profit margin, and expense ratios for the Alphabet Inc. and microsoft over the last 5 years?"

Insights and charts based on the final response

Revenue Trends

- Alphabet Inc. (Google's Revenue Contribution): 2020: $182.527 billion 2021: $257.637 billion (55% increase) 2022: $282.836 billion (10% increase) 2023: Revenue figures for Alphabet Inc. are not explicitly provided, but Google's revenue has shown significant growth, indicating its critical role in driving Alphabet's overall financial performance.
- Microsoft: 2022: $198.27 billion 2023: $211.915 billion (6.88% increase) 2024: $245.122 billion (15.67% increase) Microsoft has demonstrated consistent revenue growth, particularly in digital services and cloud computing, reflecting strong performance in the technology sector.

Profit Margin Analysis

- Alphabet Inc.: While specific profit margin figures for Alphabet are not provided, it is understood that the company's reliance on advertising revenue can lead to variability in profitability. The fluctuations in advertising demand

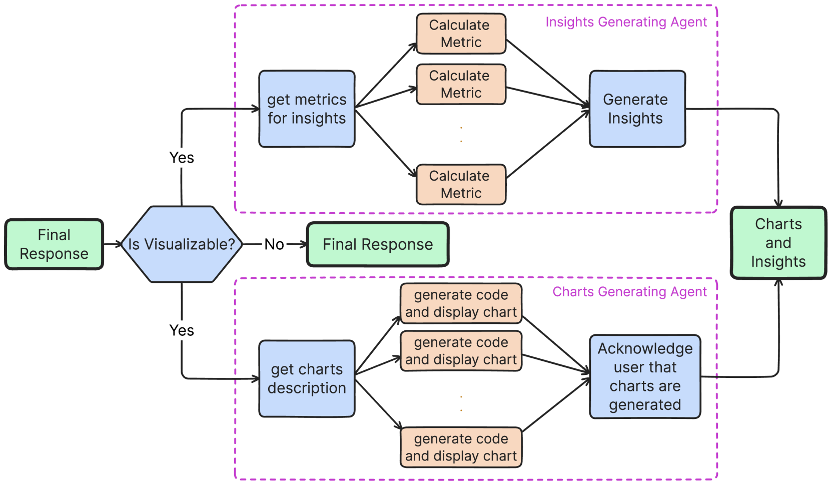


Figure 11. Post Processing Workflow

and regulatory challenges have influenced its profit margins.

- Microsoft: As of September 30, 2024, Microsoft boasts an operating profit margin of 35.61%. This higher margin reflects its business model focused on software and cloud services, which typically incur lower variable costs compared to Alphabet's advertising-driven model. The strategic focus on innovation and subscription-based services has bolstered Microsoft's profitability.

Expense Ratios

- Alphabet Inc.: Over the past five years, Alphabet's expense ratios have shown a consistent increase. For the twelve months ending September 30, 2024, operating expenses reached $234.744 billion, marking a 7.5% increase year-over-year. The rising expenses are attributed to significant investments in research and development, marketing, and high Traffic Acquisition Costs (TAC), which impact overall efficiency.
- Microsoft: While specific expense ratios for Microsoft are not detailed, it is known that the company manages expenses related to R&D, sales, and marketing. If Microsoft has maintained or improved its expense ratios while growing revenue, it would indicate strong operational efficiency.

Based on the answer, some of the charts generated by our agent are as follows:

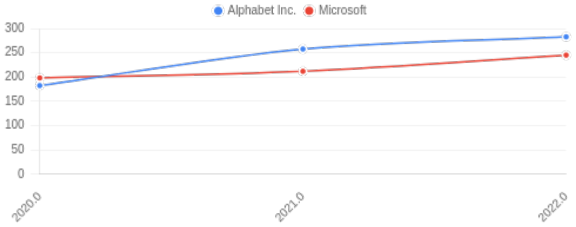


Figure 12. Chart generated for Revenue Growth Trend of Alphabet Inc. and Microsoft

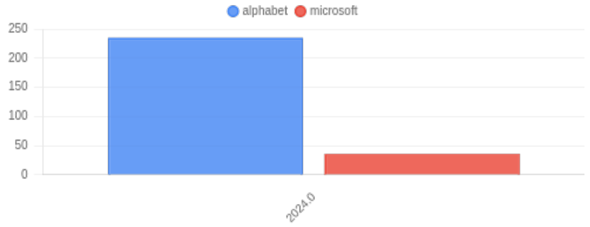


Figure 13. Financial Trends Comparison: Expense Ratios, and Profit Margin

References

[1] Zhiyu Chen et al. *ConvFinQA: Exploring the Chain of Numerical Reasoning in Conversational Finance Question Answering*. 2022. arXiv: 2210.03849 [cs.CL]. URL: https://arxiv.org/abs/2210.03849.

[2] Zhiyu Chen et al. *FinQA: A Dataset of Numerical Reasoning over Financial Data*. 2022. arXiv: 2109.00122 [cs.CL]. URL: https://arxiv.org/abs/2109.00122.

[3] Nicole Cho et al. "FISHNET: Financial Intelligence from Sub-querying, Harmonizing, Neural-Conditioning, Expert Swarms, and Task Planning". In: *arXiv preprint arXiv:2410.19727* (2024).

[4] Pranab Islam et al. *FinanceBench: A New Benchmark for Financial Question Answering*. 2023. arXiv: 2311.11944 [cs.CL]. URL: https://arxiv.org/abs/2311.11944.

[5] Huiqiang Jiang et al. "LLMLingua: Compressing Prompts for Accelerated Inference of Large Language Models". In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP 2023)*. Dec. 2023. URL: https://www.microsoft.com/en-us/research/publication/llmlingua-compressing-prompts-for-accelerated-inference-of-large-language-models/.

[6] Microsoft. *Microsoft Responsible AI Standard, v2*. URL: https://blogs.microsoft.com/wp-content/uploads/prod/sites/5/2022/06/Microsoft-Responsible-AI-Standard-v2-General-Requirements-3.pdf.

[7] Zhuoshi Pan et al. "LLMLingua-2: Data Distillation for Efficient and Faithful Task-Agnostic Prompt Compression". In: *The 62nd Annual Meeting of the Association for Computational Linguistics (Findings of ACL 2024)*. ACL. Mar. 2024. URL: https://www.microsoft.com/en-us/research/publication/llmlingua-2-data-distillation-for-efficient-and-faithful-task-agnostic-prompt-compression/.

[8] Kashob Kumar Roy et al. "ConTReGen: Context-driven Tree-structured Retrieval for Open-domain Long-form Text Generation". In: *arXiv preprint arXiv:2410.15511* (Oct. 2024). Accepted at EMNLP'24 Findings. URL: https://arxiv.org/abs/2410.15511.

[9] Vectara Team. *HHEM v2: A New and Improved Factual Consistency Scoring Model*. Accessed: 2024-12-06. 2024. URL: https://www.vectara.com/blog/hhem-v2-a-new-and-improved-factual-consistency-scoring-model.

[10] ToyHugs. *Structured Output doesn't work with advanced schema*. GitHub issue opened on July 13, 2024, with 21 comments. 2024. URL: https://github.com/langchain-ai/langchain/issues/24225.

[11] Bertie Vidgen et al. *Introducing v0.5 of the AI Safety Benchmark from MLCommons*. 2024. arXiv: 2404.12241 [cs.CL]. URL: https://arxiv.org/abs/2404.12241.

[12] Jim Allen Wallace. *Semantic caching for faster, smarter LLM apps*. Accessed: 2024-12-04. July 2024. URL: https://redis.io/blog/what-is-semantic-caching/.

[13] Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. *Jailbroken: How Does LLM Safety Training Fail?* 2023. arXiv: 2307.02483 [cs.LG]. URL: https://arxiv.org/abs/2307.02483.

[14] Shijie Wu et al. *BloombergGPT: A Large Language Model for Finance*. 2023. arXiv: 2303.17564 [cs.LG]. URL: https://arxiv.org/abs/2303.17564.

[15] Qianqian Xie et al. *FinBen: A Holistic Financial Benchmark for Large Language Models*. 2024. arXiv: 2402.12659 [cs.CL]. URL: https://arxiv.org/abs/2402.12659.

[16] Hongyang Yang, Xiao-Yang Liu, and Christina Dan Wang. *FinGPT: Open-Source Financial Large Language Models*. 2023. arXiv: 2306.06031 [q-fin.ST]. URL: https://arxiv.org/abs/2306.06031.

[17] Hongyang Yang et al. *FinRobot: An Open-Source AI Agent Platform for Financial Applications using Large Language Models*. 2024. arXiv: 2405.14767 [q-fin.ST]. URL: https://arxiv.org/abs/2405.14767.

[18] Ye Yuan et al. *A Hybrid RAG System with Comprehensive Enhancement on Complex Reasoning*. 2024. arXiv: 2408.05141 [cs.CL]. URL: https://arxiv.org/abs/2408.05141.