tomoro.ai Technical Assignment Final Report

Submitted by Abhishek Das on February 16, 2025

Table of Contents

1. Introduction

Generative AI applications leveraging Multimodal, Reasoning and Vision Language Models enable novel approaches to the quantitative[1] analysis of financial reports. They enable natural language querying of financial documentation containing semi-structured data (tables), unstructured data (images and charts) and text.

This approach eschews more traditional ML-driven ETL pipelines which combine computer vision (OCR, ICR and image recognition) and Natural Language Processing (n-grams, bag-of-words and boosted trees) to extract meaningful insights from documents.

A clear advantage of the former approach is incorporating multi-modal models such as 4o-mini to extract information from semi-structured and unstructured content such as tables, images and video files. GPT-4 and its variants may subsequently be used as general purpose solvers for financial text analysis, with its performance on QA shown the be excellent compared to other models.[2] Alternatively reasoning models such as o1 which bake in chain-of-thought, may be considered for quantitative analysis. Neither of these could be performed with more traditional ML approaches.

This report details how a multi-modal RAG pipeline is deployed to use natural language to query financial filings, which includes data in tables, to answer numerical reasoning questions such as calculating valuation ratios and year-over-year percentage changes in financial reports. The approach, solution architecture and evaluation metrics are discussed along with proposed future work to potentially enhance the results.

2. Dataset

The ***convfinqatrain.json*** dataset is a semi-structured representation of financial (pdf) reports along with associated quantitative analysis of tabular data present in the original financial reports in a conversational style. It is designed to study chain of numerical reasoning in conversational question answering.

There are 3,037 conversations in the json dataset containing 13,000+ questions.[3] Questions are one of three types: (i) surface level content (ii) numerical calculations from the report or (iii) the above two questions asked sequentially to form the conversation to cumulatively query more information or switch to other aspects.

Each conversation is structured as a dictionary of key-value pairs with several key fields for the analysis are presented in Table 1.

No.	Field	Definition
1	*pre_text*	Preamble text from a company's financial report preceding a figure or table which provides

[1] In Finance the term "quantitative" often refers to a subset of analysis concerning building mathematical (derivatives) pricing models or aggregate asset selection based on computational-driven statistical models. In this report the term "quantitative" refers to numerical and statistical analysis familiar to secondary school (GCSE/A-Level) students.
[2] Refer to this paper: https://arxiv.org/abs/2305.05862
[3] Note these counts are slightly lower to those provided in the original paper: https://arxiv.org/pdf/2210.03849

		context or summarises the function of the table of data
2	*post_text*	Additional text from a company's filings immediately following a table of data or chart which might provide additional context or conclusions based on the data contained in the table
3	*id / filename*	A document identifier which can be used to chunk conversations.
4	*table_ori*	List of strings containing table column names and numerical values
5	*qa.question*	Annotated question(s) asked based on information in the table_ori field
6	*qa.answer*	Answer to the question above
7	*dialogue_break*	Additional questions that may be answered by the contents of the table. May be used by RAG Fusion to reframe how questions are asked to the dataset.
8	steps	Numerical reasoning steps required to calculate the answer to a question

Table 1: Key Fields in Document Corpus

For evaluating the proposed approach in this solution, 100 conversations are selected that have non-null qa data. A conversation is defined as an element of the convfinqatrain.json[4] that spans all fields from "pre_text" to "exe_ans".

3. Task

tomoro.ai asked to build an LLM application allowing users to query the convfinqa (JSON) corpus of financial reports which contain text and tabular information. Users interrogating the corpus may ask questions involving numeric reasoning e.g. instead of simply asking for the net sales in 2008, the user may ask for the percentage increase or decrease in net sales from 2007 to 2009. Write a report discussing the approach and metrics used to evaluate the efficacy of the LLM application.

4. Methodology

The core objective is to allows users to query a corpus of financial documents stored in a JSON file. These documents include free-for text[5] and structured tabular data. User queries are predominantly numerical in nature, targeting the data embedded within tables. To address this, a pipeline is build leveraging a multi-modal RAG approach that combines robust data ingestion, modality-specific processing, semantic embedding generation and retrieval-augmented response generation. While the source data contains free form text and string representations of tables, the approach described below can be extended to additional modalities of financial reporting data including charts and images.

4.1 Key Design Decisions (KDD)

[4] The source JSON file is structured as a list of dictionaries
[5] Free-form text and narrative text are used interchangeably in this document and refer to the aggregation of text in the *pre_text*, *post_text* and *dialogue_break* fields.

KDD 1: Multimodal Integration ensures free-form text and numerical tabular data are processed in a way that preserves their unique characteristics while making them comparable through a unified embedding space.

KDD 2: Preprocessing and embedding strategies are applied to financial tabular data to maintain numerical fidelity, which is essential for numerical queries that demand calculative precision.

KDD 3: Efficient indexing and retrieval mechanisms (using vector stores) are employed to handle a large corpus of financial documents ensuring scalability and performance.

KDD 4: For time and cost constraints, OpenAI's *text-embedding-3-small* and multimodal *GPT-4o-mini* are selected.

The proposed multi-modal RAG (mmRAG) solution is a hybrid retrieval-augmented generation system designed to handle both textual and tabular data. Multimodal LMs assess the inputs of different modes of data within a corpus which is used as context for generating responses to numerical queries on the corpus.

4.2 mmRAG Solution Architecture

The solution architecture consists of two main pipelines: document ingestion and retrieval-generation.

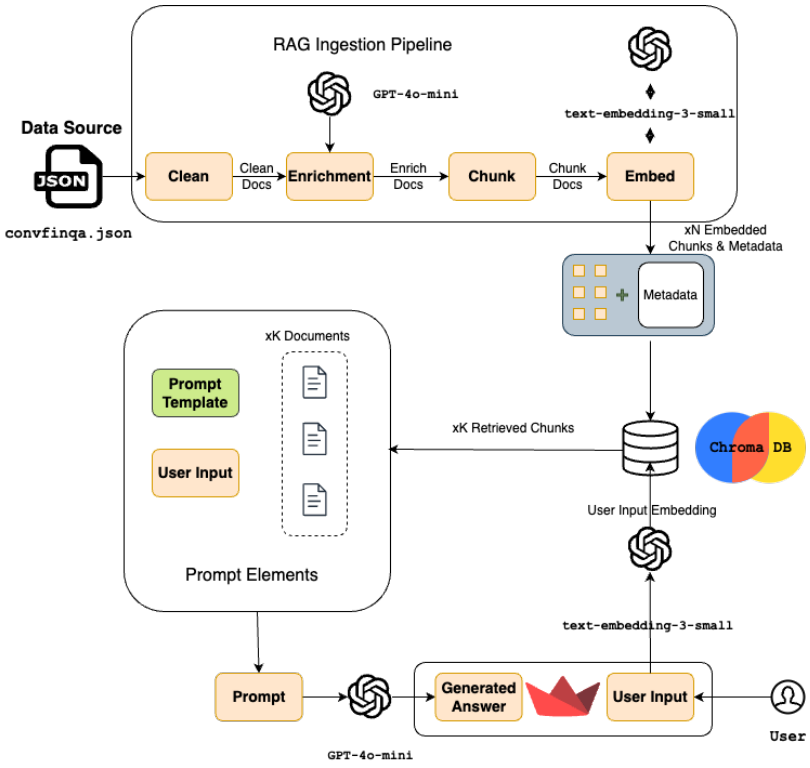

Figure 1: mmRAG Ingestion and Retrieval-Generation Pipelines

4.2.1 mmRAG Document Ingestion Pipeline

1. Source data in a JSON file which encapsulates a collection of financial reports is ingested and NLP techniques are used to handle

both narrative text and tabular data with configurable chunk sizes and overlap parameters to maintain context.
2. Each document is parsed via a dual-path processing system which separates tabular and textual content while maintaining their relationships through parent (uuid) based referencing. The separation is critical as each type requires a tailored processing strategy.
3. NLP techniques including tokenization, cleaning and normalization are applied to prepare the narrative textual content for embedding. Tabular data is processed to retain numerical values and relational structures by converting tables into a structural format that preserves row-column relationships and adding descriptive text format to each table.
4. Content summaries are generated using GPT-4 to create semantic representations of both narrative text chunks and tables, optimizing for retrieval effectiveness via a unified embedding strategy (text-embedding-3-small). This step ensures that semantically similar content across both modalities is represented in a comparable high-dimensional space.
5. The system maintains the original content alongside generated content in a multi-vector storage structure[6] enabling rapid similarity searching during query time. The multimodal index provisions a single query to retrieve information from both free-form text and tables.

4.2.2 mmRAG Retrieval-Generation Pipeline

6. User queries entered through a Streamlit front end[7].
7. Query processing leverages both semantic embeddings (using OpenAI's text-embedding-3-small) and the original content to ensure accurate numerical answers are retrieved.
8. The system employs a hybrid context formation strategy that combines relevant narrative text and table content into a unified aggregate prompt
9. A zero temperature, GPT-4o-mini generative model uses this context to generate a coherent and precise numerical analysis and answer.
10. The final answer to the numerical question integrating insights from both modalities is returned to the user via Streamlit front end along with an explanation (including calculations) of how the answer was reached.

The source code for the entire mmRAG solution including both pipelines can be viewed in the *multimodalragindexretrieval.ipynb* notebook in the repo of the submitted technical assignment. See here for location.

4.3 Methodological Limitations

No solution drafted under the constraints of an interview is perfect! Here are the key limitations of the current approach:

1. The mmRAG pipeline treats tables as flat text structures and may not preserve the inherent relationships and hierarchies present in tabular finance data despite using GPT-4 to articulate the purpose of the tables. This could impact the accuracy of complex numerical

6 The mmRAG solution uses Chroma as the vector store backend with an in-memory document store for rapid retrieval. Special attention is given to maintaining the relationships between summaries and original content through UUID-based referencing, ensuring that numerical precision in tables is preserved while leveraging the semantic understanding capabilities of the language model.
7 Apologies to Albert Phelps

queries that require understanding table structure and cell relationships.
2. The system does not cater to the interconnectivity between conversational questions asked of the financial data. While the corpus provides a sequence of questions that can be queried by the user to the dataset, the deployed solution does not keep these questions in any type of memory.
3. The efficacy of GPT-4 models to handle financial data is an avenue of research to be explored despite preliminary results showing promise. The system can be improved by embedding models such as FinBERT or using Language Models such as BloombergGPT **trained on financial documents** or explicitly handling financial-domain specific units of measurement, currency conversions or valuation accounting specific terminology.
4. The simple user query search may be supplanted by hybrid search techniques such as RAG Fusion which combine query expansion, vector searches and intelligent re-ranking.
5. While the current approach does not cater to image data such as charts or figures, the multimodal vector database and index can be easily extended to accommodate these types of data.

Additional areas for improvement are discussed in this section.

5. LLM Application Folder Structure

Installation instructions at https://github.com/factorwonk/tomoro.git repo.

```
FINANCIAL_QA_RAG/
├── chroma_db/                                  # Vector DB store
├── data/                                       # Source data
├── sample_notebooks/                           # Example implementations
│   ├── 01_multimodalragindexretrieval.ipynb    # End to End mmRAG
│   ├── 02_multimodalragas.ipynb                # RAGAS evaluation of mmRAG
│   ├── 03_multimodalragstreamlit.ipynb         # Streamlit app in a notebook
│   ├── mmRAGEvaluationScores.csv               # RAGAS Evaluation Scores for mmRAG
│   ├── notebook_streamlit.py
│   ├── rag_system.py
│   └── streamlit_app.py
├── src/
│   ├── app/                                    # Streamlit App Package
│   │   ├── ui/                                 # User interface components
│   │   │   ├── components.py
│   │   │   ├── layout.py
│   │   │   └── sidebar.py
│   │   └── main.py                             # Main application entry point
│   ├── rag/                                    # RAG App Package
│   │   ├── prompts/
│   │   │   └── templates.py                    # Prompt Templates
│   │   └── core.py                             # End-to-End mmRAG pipeline
│   └── config.py                               # RAG config settings
├── tests/                                      # Placeholder
├── .env.example
├── .gitignore
├── final_report.pdf                            # This report
├── README.md
├── requirements.txt
└── setup.py                                    # Package install script
```

Figure 2: LLM application structure

- chroma_db/ folder visible upon application run.
- init files not shown.

6. Results

The LLM Application uses the RAGAS framework for benchmarking and evaluation.

6.1 Evaluation Metrics and Results

The following metrics are considered for evaluating the mmRAG solution along with a justification for why these metrics were chosen.

No.	Metric	Definition
1	*Faithfulness*	Assess the factual consistency of the LLM generated answer versus the provided context i.e. whether the generated answer is factually supported by the retrieved context. It helps detect hallucinations.
2	*Answer Relevancy*	Evaluates the relevancy of a generated answer to the user's query. It ensures the response directly addresses the users prompt via similarity metrics. Incomplete answers and answers that contain redundant information leads to a lower score.
3	*Context Precision*	Measures how much of the retrieved context is relevant to answering the query based on the proportion of relevant chunks in the retrieved (top k) context.
4	*Context Recall*	Assess whether the retrieval system is retrieving the necessary information for answering the query. High values indicate the system is retrieving all relevant information.
5	*Semantic Similarity*	Determines whether the generated answers are close in meaning to the expected responses. The LM understands the general concept but might miss precise numerical details.
6	*Answer Correctness*	Accuracy of the generated answer compared to human annotated ground truth response.

Table 2: RAGAS Evaluation Metrics and Definitions

The first 100 examples of the ***convfinqa*** train.json dataset were used to calculate the above evaluation metrics. All scores are scaled between 0 and 1 with higher values indicating the system being more performant in a particular domain.

Metric	Score	Interpretation
Faithfulness	0.7528	Generated answers grounded in retrieved context 75% of the time. 25% of the time there are instances where the model generates information not explicitly present in the retrieved chunks (hallucinations).
Answer Relevancy	0.5566	Only 56% of the generated answers are highly relevant to the question so the model may be missing the intent of the question or generating partially aligned responses.
Context Precision	0.4591	Around 46% of retrieved context is relevant to answer the question. 54% of the time retrieval returns irrelevant information.
Context Recall	0.0280	The system only retrieves 2.8% of the actual relevant context necessary to answer the question. The retrieval pipeline is failing to fetch the most import supporting documents

		which adversely affects downstream answer quality.
Semantic Similarity	0.7688	Generated answers are close in meaning to expected response but not correlated with higher numerical precision.
Answer Correctness	0.3388	When compared to ground truth values the generated answers are accurate only 34% of the time, likely due to low context recall.

Table 3: mmRAG Evaluation Metrics

The source code for the RAGAS evaluation of the mmRAG solution and the evaluation scores for each of the 100 questions can be viewed in the *multimodalrags.ipynb* notebook and *mmRAGEvaluationScores.csv* files respectively, in the repo of the submitted technical assignment. See here for location.

6.2 Key Insights

The biggest issue with the mmRAG results is the very low context recall (0.03) which indicates the system is no retrieving enough relevant context. This can have an adverse effect on answer correctness and relevancy. As faithfulness is relatively high (0.75) but answer correctness is low (0.34), the mmRAG system adheres to the retrieved chunks but the top k retrieved chunks themselves are often insufficient to answer the numerical query. If context recall can be improved the high faithfulness score should translate into more correct answers. The model has relatively high semantic similarity (0.77) but low answer correctness (0.34) meaning that even though the output sounds right the numeric calculations are wrong. This suggests that GPT-4o-mini understands the questions semantically but lacks precise numerical reasoning. Context precision (0.46) is much higher than context recall (0.03) which suggests that when retrieval does find useful information it is precise, but it often misses a large amount of critical information which affects performance.

Several strategies may be employed to improve the low recall of the RAG pipeline. Domain-specific embedding models such as FinBERT may be used to embed the narrative text and tabular data. Alternate chunking strategies and chunk sizes can be explored. Hybrid retrieval via BM25 and dense embeddings can be explored. RAG-Fusion, combining query expansion (using an LLM to create variants of the original question) to capture more relevant context and using a reciprocal rank function (RRF) can be used to improve retrieval precision using re-ranking models. Finally, adjusting LLM parameters and additional prompt engineering may be used to try to improve model faithfulness.

The next section describes how to deploy the mmRAG solution developed for this technical assignment to a cloud platform environment.

7. mmRAG PoC Solution Architecture

This section provides High Level Solution Design using aws[8] services to deploy and scale a Proof-of-Concept mmRAG solution. The terminology **PoC** is used to differentiate this from a hardened, industrialised/production solution incorporating full LLM Ops pipelines and separated environments.

[8] AWS is selected due to the author's own experience, but the PoC solution can be replicated on Azure, Databricks or GCP-Vertex AI.

The solution design is comprised of the document ingestion pipeline and the document retrieval and generation pipeline. Each are presented below, and a logical sequencing of activities are provided. This design attempts to address the shortcomings of the mmRAG solution developed for the technical assignment discussed in the previous section.

7.1 PoC Ingestion Pipeline (AWS)

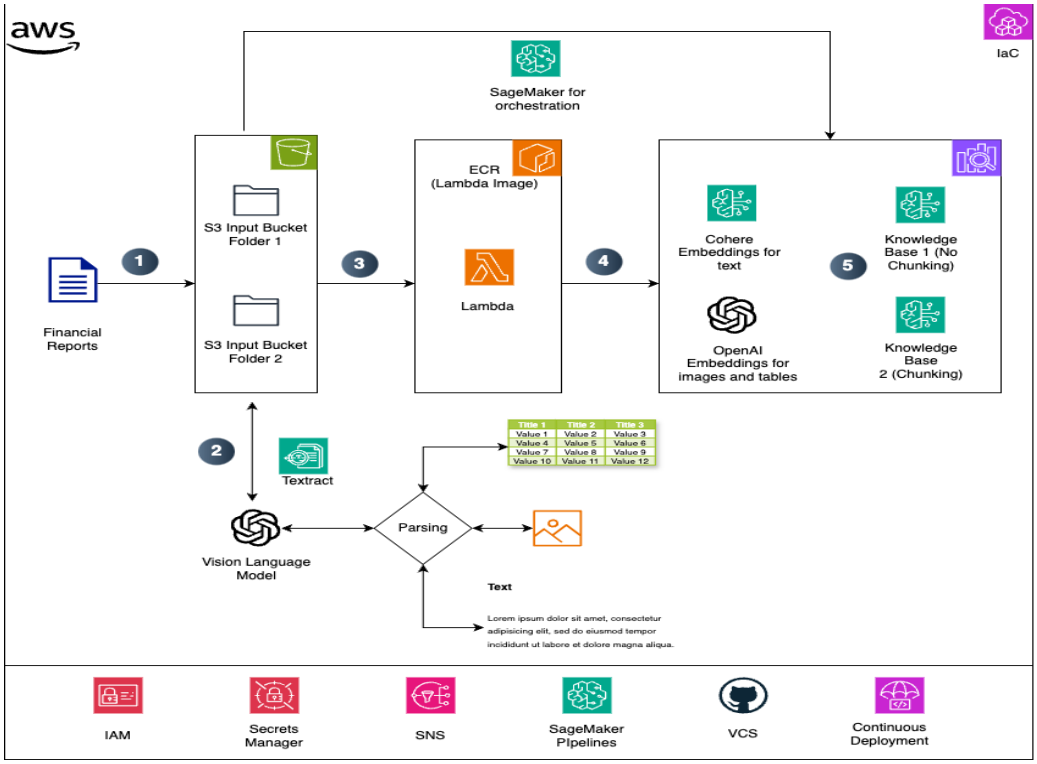

Figure 3: PoC Ingestion Pipeline

1. Financial documents are stored in an Amazon S3 bucket in two separate folders. Each folder will correspond with a specific Bedrock knowledge base.
2. Computer vision services such as Textract and Vision Language Model such as GPT-4 are run on each page of the financial documents. These can be run with SageMaker for the PoC and with Amazon Lambda in the industrialised (prod) solution. These models will parse text, image and tabular data separately.
3. Sagemaker notebooks are used in the PoC and Lambda functions or SageMaker pipelines can be used in the industrialised solution for pipeline orchestration
4. Multiple embedding models can be run on each document for resiliency (two are shown in the diagram) so that poor embedding from a single model can be mitigated by superior embedding from another model. For example, the low recall of the mmRAG solution may be mitigated by using FinBert or Cohere embedding models along with OpenAI models.
5. Two bedrock knowledge bases are created: one with no chunking and another with default chunking to be used for RAG Fusion and the Reciprocal Rank Function

Supporting aws services for identity management, secrets management version control, notification and CICD can be spun up with an Infrastructure as Code approach.

7.2 PoC Retrieval and Generation Pipeline (AWS)

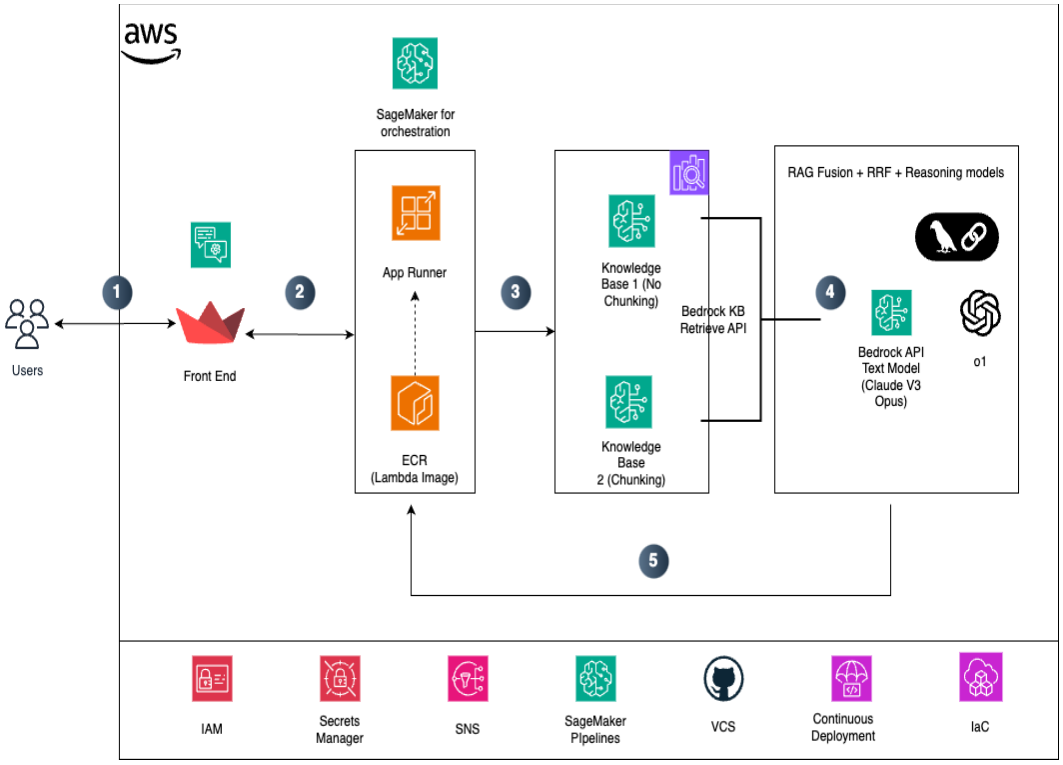

Figure 4: PoC Retrieval and Ingestion Pipelines

1. User queries the embedded financial documents using a Streamlit front end (or Amazon Lex for a basic chatbot experience)
2. SageMaker notebook performs workflow orchestration for PoC Solution. App Runner can be used for the industrialised solution.
3. Document chunks are extracted from both knowledge bases for RAG fusion.
4. RAG Fusion and Reciprocal Ranking Function and LangChain is used to query Claude Opus or Open AI text models. Reasoning models such as o1 or deepseek-r1 may be used for numerical calculations based on the user query and the Top 3 chunks are returned based on similarity metric.
5. Output returned to the user via the front end.

8. Further Enhancements

Further enhancements to the multi-modal RAG pipeline, which could not be explored due to time and resource constraints, are suggested below:

1. Using Language Models fine-tuned on Financial Documents such as Bloomberg-GPT.
2. Hybrid search methods such as RAG-Fusion which combine query expansion with intelligent reranking to provide a single unified ranking which yields better results than any individual system and better results than standard reranking. Variants of the user input question are generated using language models and used to query the financial corpus while the Reciprocal Rank Function is used to generate a consensus about the most relevant top K context documents to provide to the numerical output generating GPT-4o model.

3. Using a Knowledge Graph RAG architecture – the advantage of a knowledge graph-based RAG architecture is the ability to link related financial data across disparate conversations (multi-hop reasoning) in contrast to vanilla RAG or RAG Fusion. In Knowledge Graphs financial data is connected through nodes and relationships allowing follow-up queries to be performed. Knowledge graphs can handle context changing in questions: a question about book value can be followed by a question about labour costs and multi-hop connections can be made related to financial data nodes that enhances contextually relevant answers.
4. Creating Agentic Workflows – AI Agents are LLM orchestrated applications designed to autonomously perform tasks. Agents have additional capabilities beyond probabilistic reason driven text generation. They comprise of a planning component designed to analyse a problem and break it down into steps. The have access to tools such as calculators to perform numerical analysis and have memory to store and retrieve information such as previous questions asked allowing them to learn from past questions and make informed decisions.
5. Using alternative tools such as the **unstructured** python library to extract tabular and image data from documents or incorporating reasoning models such as GPT-o1 or deepseek-r1 when performing numerical calculations based on the data in the JSON file or using enterprise grade vector stores such as pinecone to store multimodal embeddings.
6. Incorporating conversational memory using LangChain which allows chatbots to respond to queries in a chat-like manner without treating every query independently as in the current solution. Conversational memory allows non-stateless agents to remember previous interactions including numerical questions asked by the user.

9. Conclusion

An LLM application which allows users to interrogate a corpus of financial documents (reports, filings) that include tables and charts was built using a multi-modal RAG architecture. Using Streamlit front-end users can ask numerical questions involving calculations based on information available in tabular data extracted from the reports. The GPT-4 family of language models are shown to be moderately performant in retrieving answers from financial text data that require numerical analysis. The RAGAS framework is used to evaluate the performance of the retrieval and generation pipelines. While the language model appears to be relatively faithful and semantically coherent, the RAG pipeline struggles primarily with retrieval recall which leads to incorrect and/or incomplete answers. This language model may lack the right supporting context and numerical precision. Scaling the PoC solution onto cloud infrastructure is discussed and suggestions for improving RAG low context retrieval are discussed.

10. References

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.T., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. arXiv preprint arXiv:2005.11401*. Available at: https://arxiv.org/abs/2005.11401

Chen, W., Hu, H., Chen, X., Verga, P., & Cohen, W.W. (2022). 'MuRAG: Multimodal Retrieval-Augmented Generator for Open Question Answering over Images and Text'. *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 5564–5580. Available at: https://aclanthology.org/2022.emnlp-main.375/

Şakar, T., & Emekci, H. (2024). *Maximizing RAG efficiency: A comparative analysis of RAG methods*. *Natural Language Processing*, [online] Available at: https://www.cambridge.org/core/journals/natural-language-processing/article/maximizing-rag-efficiency-a-comparative-analysis-of-rag-methods/D7B259BCD35586E04358DF06006E0A85

Rackauckas, Z. (2024). *RAG-Fusion: a New Take on Retrieval-Augmented Generation*. *arXiv preprint arXiv:2402.03367*. Available at: https://arxiv.org/abs/2402.03367

Li, X., Chan, S., Zhu, X., Pei, Y., Ma, Z., Liu, X., & Shah, S. (2023). 'Are ChatGPT and GPT-4 General-Purpose Solvers for Financial Text Analytics? A Study on Several Typical Tasks'. *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 399–410. Available at: https://aclanthology.org/2023.emnlp-industry.39/

Es, S., James, J., Espinosa-Anke, L., & Schockaert, S. (2025). 'Agentic Retrieval-Augmented Generation: A Survey on Agentic RAG'. *arXiv preprint arXiv:2501.09136*. Available at: https://arxiv.org/abs/2501.09136

Mombaerts, L., Ding, T., Banerjee, A., Felice, F., & Taws, J. (2024). 'Meta Knowledge for Retrieval Augmented Large Language Models'. *arXiv preprint arXiv:2408.09017*. Available at: https://arxiv.org/abs/2408.09017

Chen, Z., Li, S., Smiley, C., Ma, Z., Shah, S., & Wang, W.Y. (2022). 'ConvFinQA: Exploring the Chain of Numerical Reasoning in Conversational Finance Question Answering'. *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 6279–6292. Available at: https://aclanthology.org/2022.emnlp-main.421/

Es, S., James, J., Esponisa-Anke, L., & Schockaert, S. (2023). 'RAGAS: Automated Evaluation of Retrieval Augmented Generation'. *arXiv preprint arXiv: 2309.15217*. Available at: https://arxiv.org/abs/2309.15217

Safjan, K. (2023). 'RAG-Fusion', *Krystian Safjan's Blog*, 1 March. Available at: https://safjan.com/rag-fusion-enhancing-information-retrieval-in-large-language-models/

Maddula, S. (2024). 'Not RAG, but RAG Fusion? Understanding Next-Gen Info Retrieval', *Towards AI*, 15 October. Available at: https://towardsai.net/p/artificial-intelligence/not-rag-but-rag-fusion-understanding-next-gen-info-retrieval

'RAG-Fusion: a New Take on Retrieval-Augmented Generation' (2024). *Papers with Code*. Available at: https://paperswithcode.com/paper/rag-fusion-a-new-take-on-retrieval-augmented

'RAG-Fusion: Multi-Query Retrieval & Rank Fusion Techniques' (2024). *DocsBot*. Available at: https://docsbot.ai/article/advanced-rag-techniques-multiquery-and-rank-fusion