tomoro.ai Technical Assignment Final Report

Submitted by Abhishek Das on February 17, 2025

Table of Contents

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.

Dataset

The ***convfinqa*** 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] 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

1	*pre_text*	Preamble text from a company's financial report preceding a figure or table which provides 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.

Task

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.

Methodology

Retrieval-Augmented Generation (RAG) is a hybrid approach combining retrieval-based and generation-based methods for NLP tasks. The main idea is to retrieve relevant documents or passages from a corpus of text and use them as context for generating responses to questions posed by the user. RAG leverages the strengths of both retrieval and generation, providing more accurate and contextually relevant answers. RAG overcomes two primary limitations of LLMs. It provides access to external or latest data and prevents hallucinations enhancing the accuracy and reliability of AI models.

Multimodal RAG enhances vanilla RAG by using Multimodal LLMs such as GPT-4 and its variants, specifically GPT-4o-mini used in this project, to assess the inputs of different modes (types) within a corpus including but not limited to text, tables and images. The assessment of these modes of data is used as context for generating responses to numerical queries on the corpus.

A RAG system is comprised of three main independent modules:

1. **Ingestion pipeline:** A batch or streaming pipeline to populate the vector DB
2. **Retrieval pipeline:** A module that queries the vector DB and retrieves relevant entries to the user's input question
3. **Generation pipeline:** The layer which uses the retrieved data from (2) to augment the prompt and an LLM to generate answers

Proof-of-Concept (PoC) Solution Design

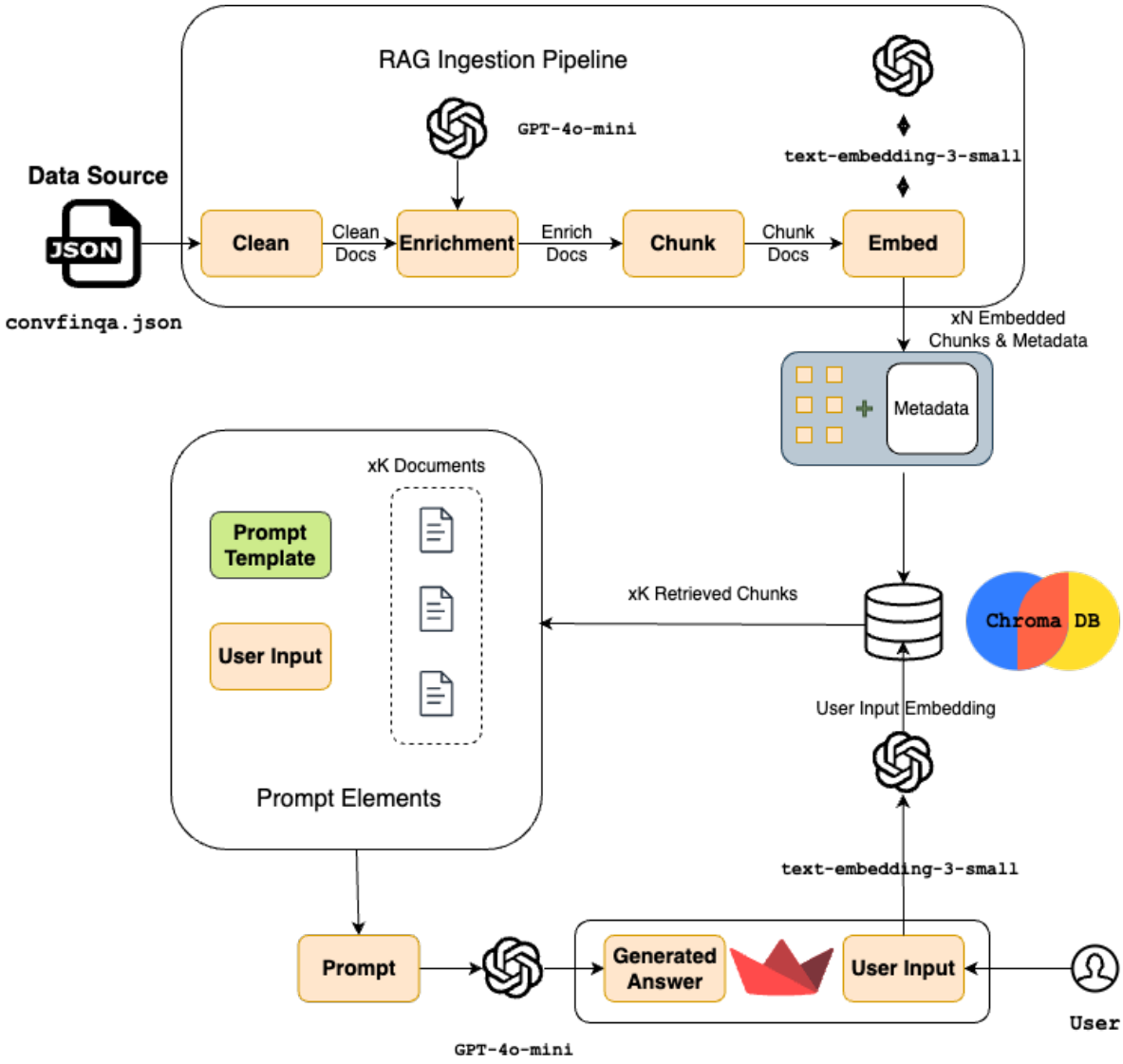

Figure 1: PoC Ingestion, Retrieval and Generation Pipelines

The PoC Multimodal RAG pipeline underlying the LLM application is executed sequentially with the following steps:

PoC Ingestion Pipeline

1. The ingestion pipeline is manually run. It loads the the JSON file and cleans, chunks, enriches and embeds the documents into the ChromaDB vector store.
2. It chunks conversations based on their unique ids.
3. GPT-4o-mini model is prompted to add contextualized text to the table_ori field – it describes what the table presents and what types of questions it can answer.
4. Pre-text, Post-text, QA and contextualized table information are merged.
5. The OpenAI text-embedding-3-small model is used to take chunks and project it into a dense vector packed with semantic value on ChromaDB
6. Embedded chunks along with metadata is used as an index to query similar chunks while metadata is used to access the information added to augment the prompt.

PoC Retrieval Pipeline

7. Take the user's input numerical query through the Streamlit front-end which is embedded into a vector space using the OpenAI text-embedding-3-small model.
8. The retrieval step projects the user's query into the same vector space as the embeddings used as an index in the Chroma vector DB. This is done using the same embedding model as is used in the RAG pipeline.
9. Find the top K most similar entries in the Chroma DB by comparing embeddings from the vector storage with the user's input vector. These entries serve as content to augment the prompt that is passed to the LLM to generate the answer.
10. The cosine distance metric (1 minus the cosine of the angle between the two vectors) is used to measure the similarity between the two vectors. The value is -1 if the two vectors are in opposite direction, 0 if they are orthogonal and +1 if they are in the same direction.
11. Avoid training-serving skew by processing (cleaning, chunking and embedding) the user input the same way as the raw documents in the ingestion pipeline are processed.

PoC Generation Pipeline

12. Take the user input and the top K retrieved chunks from the Chroma DB and pass them into a prompt template.
13. The prompt template contains both the system and user prompt and all prompt engineering is done within here.
14. The prompt template, user input, and top-k retrieved document chunks are passed as a prompt to the generation LLM (GPT-4o-mini).
15. The output is returned to the Streamlit front end.

Evaluation Metrics and Results

The codebase submitted for this assignment uses the RAGAS framework for benchmarking and evaluation. The following metrics are considered for evaluating the retrieval and generation pipelines along with a justification for why these metrics were chosen.

No.	Metric	Definition
1	*Context Precision*	Context precision evaluates whether all the ground-truth relevant chunks are both present

		in the context and ranked higher or not. When the relevant chunks have a higher ranking, the context precision value is higher.
2	*Faithfulness*	Factual consistency of the LLM generated answer versus the provided context. It is calculated from the answer and the retrieved context and scaled between 0 and 1 with higher score representing more faithfulness.
3	*Answer Relevancy*	Relevance of an answer to the given prompt. Incomplete answers and answers that contain redundant information leads to a lower score.
4	*Context Recall*	How does the retrieved content align with the annotated answer? Calculated using the user's input question, the ground truth and the retrieved context and scaled between 0 and 1 with higher score indicating better performance.
5	*Context Utilisation*	Evaluates whether all answer relevant items present in contexts are ranked higher or not. Ideally all relevant chunks are ranked at the top.
6	*Answer Correctness*	Accuracy of the generated answer compared to ground truth.

Table 2: RAGAS Evaluation Metrics and Definitions

For a larger retrieval size (K), the model has access to more text chunks resulting in better answer relevancy and answer correctness. The model may, retrieve more irrelevant chunks lowering context recall and utilisation. A smaller retrieval size improves context recall and utilisation as the model has fewer irrelevant chunks to sort through at the opportunity cost of higher answer relevancy and correctness as fewer context chunks are available to support the answer.

The first 100 examples of the ***convfinqa*** train.json dataset were used to calculate the multi-modal RAG evaluation metrics.

Metric	Score
Context Precision	
Faithfulness	
Answer Relevancy	
Context Recall	
Context Utilisation	
Answer Correctness	

Table 3: Naïve RAG Evaluation Metrics

Pilot HLSD on Cloud Infrastructure

This section provides High Level Solution Design using aws[4] services to deploy and scale multi-modal RAG pipelnes in a cloud environment. The terminology "Pilot" is used to differentiate this solution from a hardened, industrialised solution incorporating full LLM Ops pipelines with separated Development, UAT and Production environments.

[4] AWS is selected due to the author's own experience, but the Pilot 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.

Pilot Ingestion Pipeline (AWS)

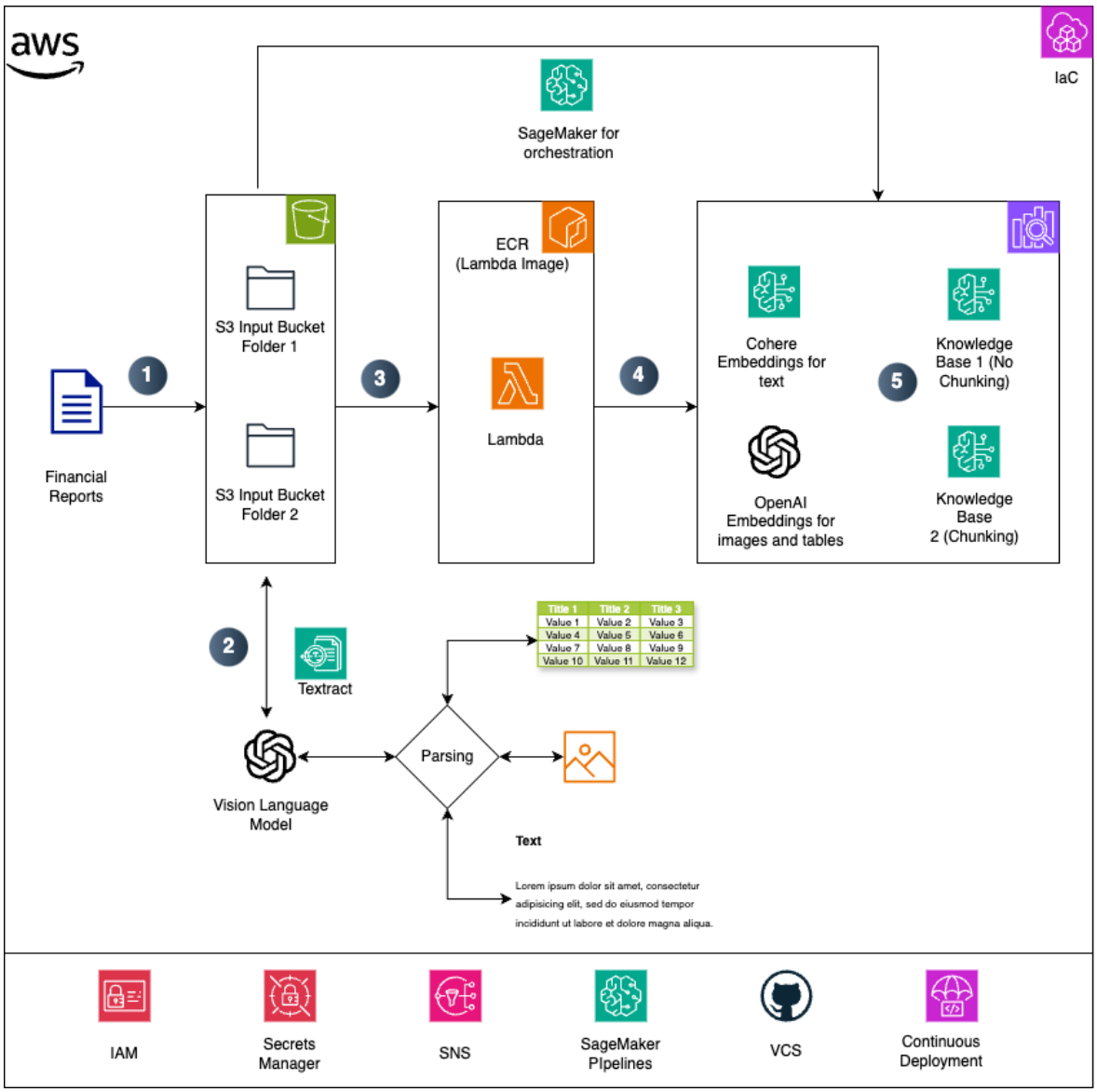

Figure 2: Pilot Ingestion Pipeline

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 Pilot 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 Pilot 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.
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.

Pilot Retrieval and Generation Pipeline (AWS)

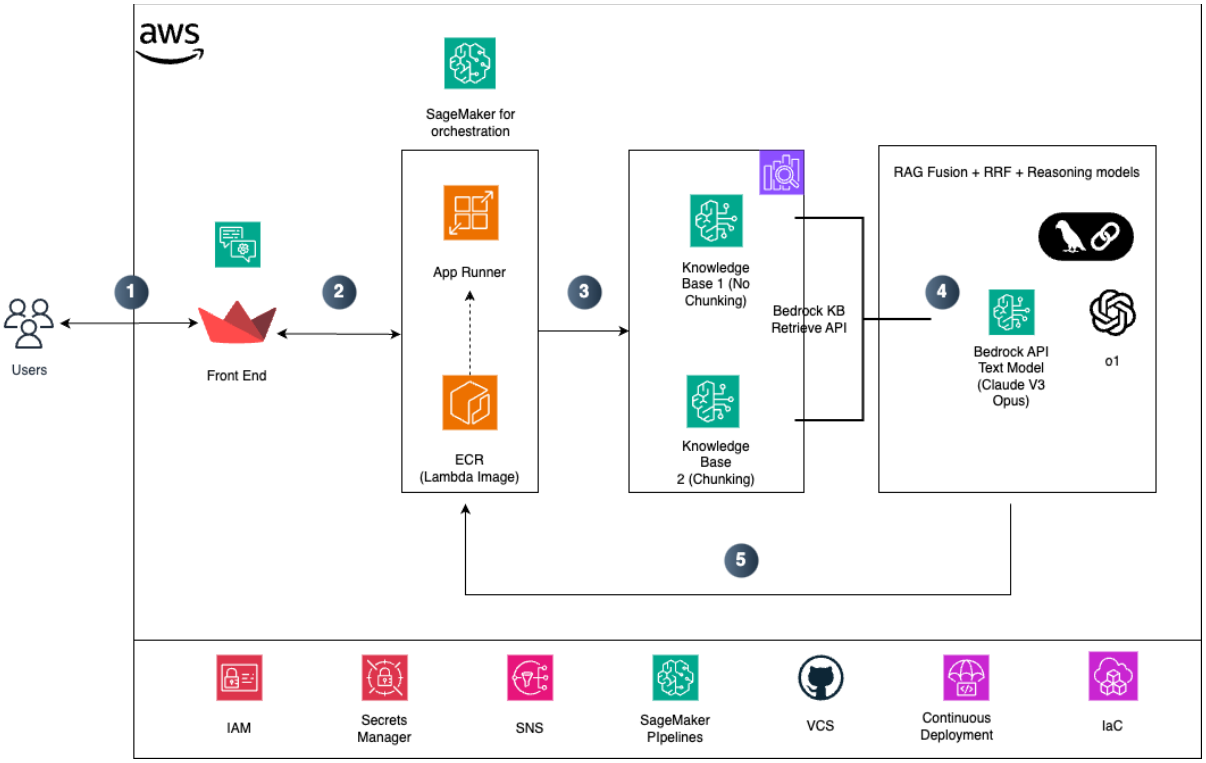

Figure 3: Pilot Retrieval and Ingestion Pipelines

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 Pilot. App Runner can be used for industrialised production.
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.

Further Enhancements to the PoC LLM Application

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

1. 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.
2. 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.
3. 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.
4. 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.

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 with suitable discussion of metrics. Scaling the PoC solution onto cloud infrastructure is discussed and suggestions for improving RAG evaluation metrics are provided.

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/

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

Farcas, M. (2025). 'Build a custom knowledge RAG chatbot using n8n', *n8n Blog*, 21 January. Available at: https://blog.n8n.io/rag-chatbot/

'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