TD Bank AI Executive Advisor

Brianna Ta (bht2118), Dai Dai (yd2765), Gregor Hanuschak (gzh2101),

Sam Gabor (sg662), Xiqian Yuan (xy2655)

Introduction

Background

Executives at TD Bank face the challenge of reviewing large amounts of reports, proposals, and financial documents every day. With so much information, finding the most important details and making timely, informed decisions can be difficult. To help solve this problem, this project aims to build a Generative AI Executive Advisor that provides clear, reliable, and evidence-based insights to support better decision-making.

This project is a partnership between TD Bank and Columbia University's Master of Data Science program. The goal is to create an end-to-end application using Retrieval-Augmented Generation (RAG) techniques with GPT-4o to help executives quickly find the strategic information they need. The system is designed to read and process different types of public financial documents, such as TD Bank's annual reports, earnings call transcripts, and regulatory filings, and use them to answer questions from executives.

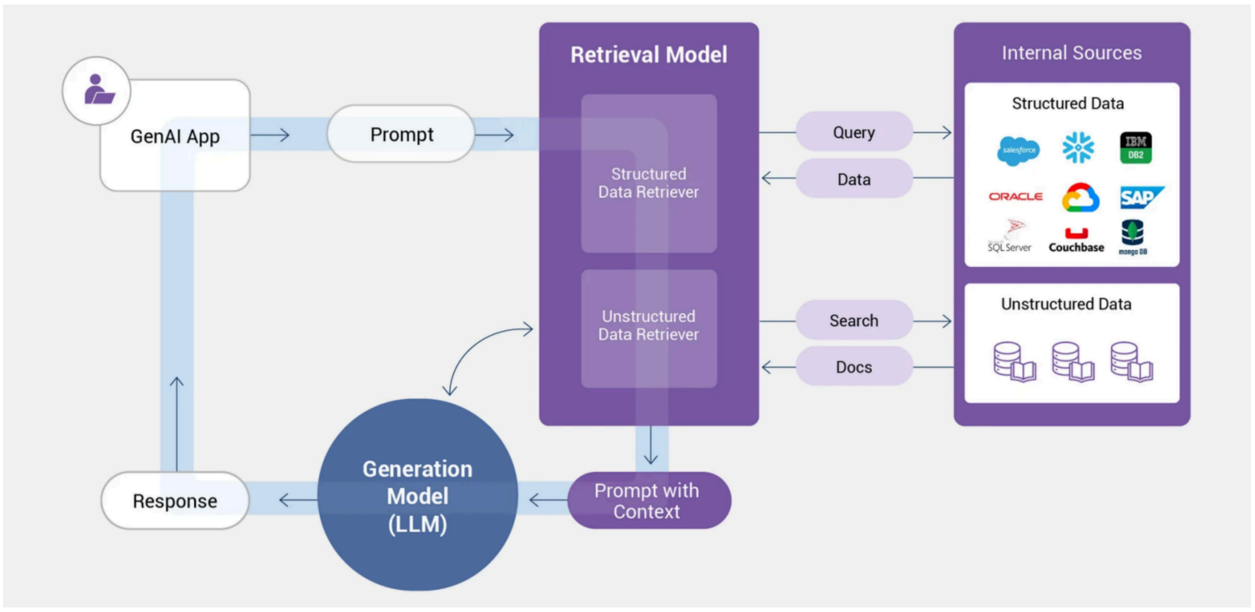
The main goal of this project is to design a chatbot that helps TD Bank executives quickly access necessary, evidence-based strategic information to support their decision-making. This solution aims to reduce the time and effort needed to find key insights within large amounts of financial documents, ensuring that the information provided is reliable, accurate, and relevant to their needs. By acting as a virtual executive assistant, the chatbot will help executives quickly retrieve trustworthy insights from financial reports, focus on high-quality, well-supported strategic information, and make more confident, informed decisions backed by clear evidence.

Led by Daniel Randles and Mohammad Nejad from TD Bank, the team is working to create a tool that turns large amounts of financial data into clear, helpful answers to support strategic planning and action.

Dataset

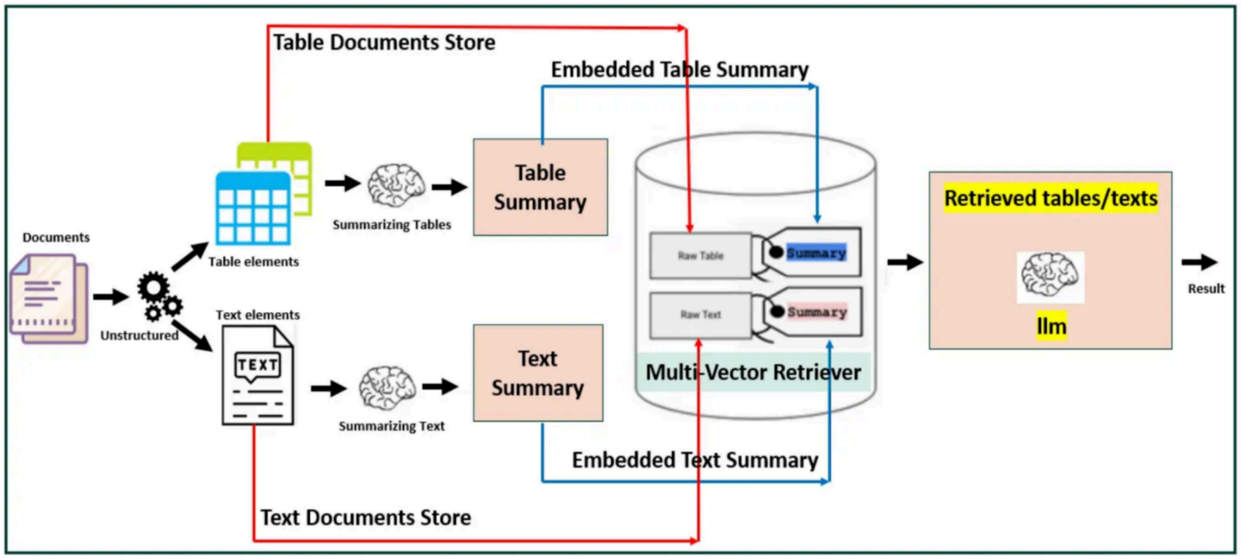
The dataset includes publicly available data that details TD Bank's financial information. We aggregated 27 documents, including quarterly financial reports, annual reports, and call transcripts from annual meetings from 2021 to 2025. We downloaded these documents from TD Bank's website as PDFs.

Previous Work



Original Approach:

- **Contextual Compression Chunking Pipeline** – The pipeline is parsed through the various types of documents and summarized in the content to extract the most salient information.
- **Multi-Vector Summary Indexing** – Provided a separate set of summary embeddings that point to longer form context to provide a faster retrieval across many documents with high-level information.
- **Cloud Vector Database** – Stored the context embeddings, summary embeddings, and associated metadata (e.g., page numbers, document titles). This cloud vector database was able to accept new data continuously, which does not require re-training. It worked in the scenario of increasing the volume of documents.
- **Context Retrieval** – Used the query questions to search the Cloud Vector Database summaries and retrieve the contents for the output. The response included the summary vector and metadata to allow the user to easily reference the original text and ensure evidence was provided in the response.
- **Guardrail & Quality Management** – For the output portion of the pipeline, the team also implemented guardrails to mitigate hallucination and introduced metrics to capture quality (e.g., relevance, groundedness, etc.)
- **User Interface and Experience** – Created a chatbot for users to interact with the system in a Q&A format. This chatbot allows in-memory questions. The generated responses have references with document names as evidence.

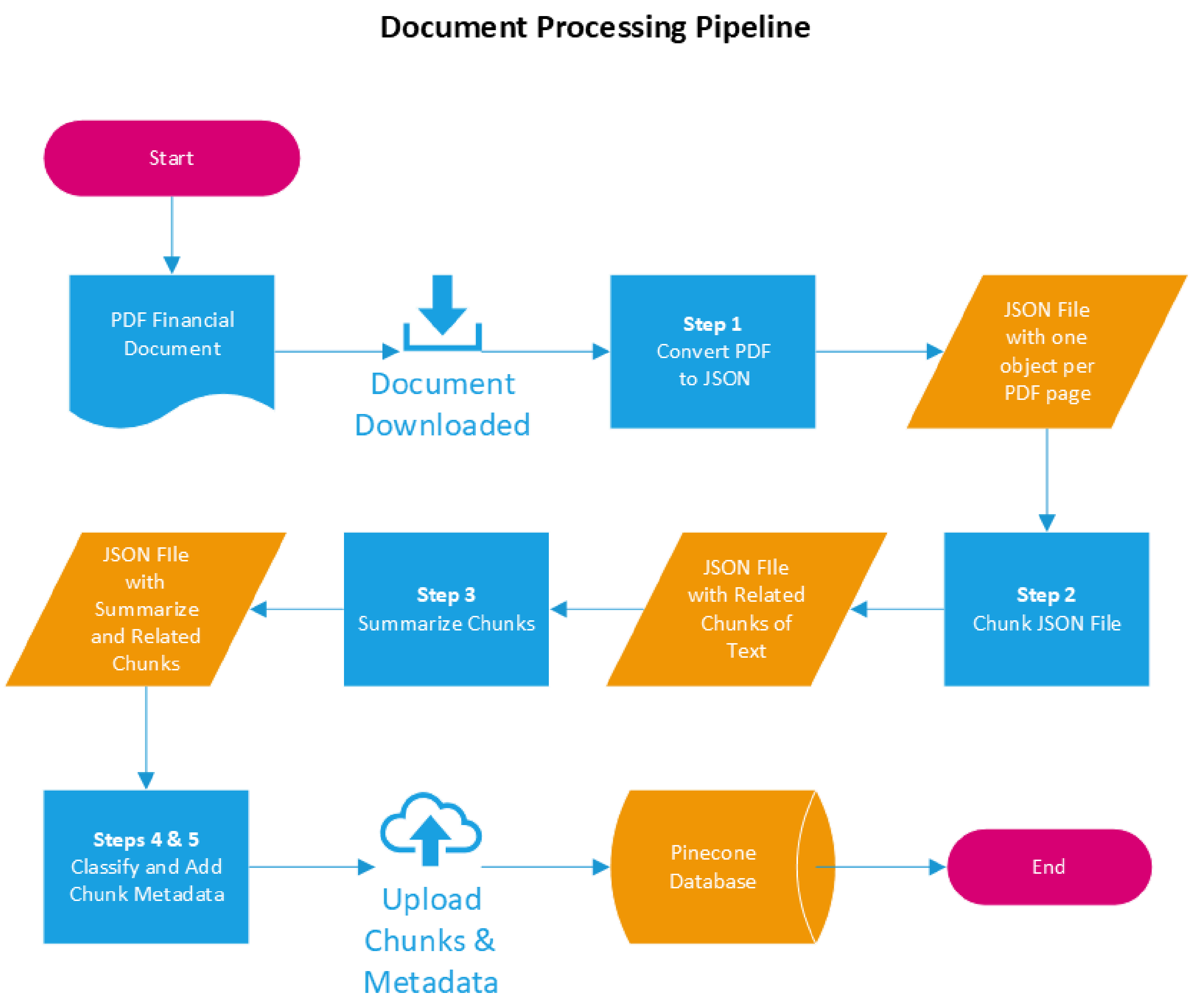


Adjustments and Improvements

The TD Bank Chatbot backend processes financial documents through a structured pipeline designed to enhance flexibility and efficiency. The system consists of five key pre-processing steps: converting PDFs to text, classifying content, chunking related pages, summarizing text, and uploading processed data into a Pinecone vector database for retrieval-augmented generation (RAG) queries. Unlike the previous implementation, which combined these steps into a single process, the new modular design allows for independent optimization. The backend now supports Ollama, Azure, and OpenAI, improving adaptability and performance.

To enhance document retrieval, we implemented hybrid search, which combines semantic search (finding conceptually similar content) with keyword search (matching exact terms). This method improves search quality, especially when queries use different wording than the indexed documents. Additionally, reranking refines retrieved results, improving relevance. We integrated the Cohere Rerank and Two Stage Rerank, which assigns scores to candidate documents based on contextual similarity, enhancing retrieval precision. We developed an abstention mechanism using G-Eval to reduce hallucination and ensure response reliability.

Data Processing


Document Processing Pipeline
Start
PDF Financial Document
Document Downloaded
Step 1 Convert PDF to JSON
JSON File with one object per PDF page
Step 2 Chunk JSON File
JSON FIle with Related Chunks of Text
Step 3 Summarize Chunks
JSON FIle with Summarize and Related Chunks
Steps 4 & 5 Classify and Add Chunk Metadata
Upload Chunks & Metadata
Pinecone Database
End

Once a PDF document is chosen for inclusion in the chatbot's knowledge base, it undergoes five discrete steps in the document processing pipeline:

1) Conversion to text
2) Related text chunking
3) Chunk summarization
4) Document classification
5) Indexing with Metadata

The PDF conversion to text produces a JSON file containing an object for each PDF page that includes the text of the page as well as the page number in the original document. Page numbers are retained to ensure that the chatbot can provide page references for any provided answers.

Once the JSON file with text and page information is produced, the next step is to chunk (i.e. group) related pages together in text chunks that may span multiple pages. This allows topics that span multiple pages to be grouped together in discrete, referenceable units. Chunk membership is determined by cosine similarity between successive pages. This step produces a JSON output file with one object per chunk and a page range that specifies the source pages in the original document where the chunk came from.

The chunks in the JSON file are then read and summarized using an LLM call. The model used (e.g. gpt-4o, mistral, etc.) for summarization as well as the LLM infrastructure (OpenAI, Ollama, Azure) are configurable. The output of this summarization step is another JSON file with each object containing the chunked text, source page range and chunk summary.

Finally, the JSON summarized chunk file is read and, for each chunk, metadata on the file and chunk which includes the original document date range, document type and vector encoding are added then uploaded to either a new (for complete document reloads) or an existing Pinecone database (for incremental document additions).

Output from each step of the document pipeline is preserved so it can be reviewed for problem determination and potential improvements

Reranking

Reranking is a method used to reorder initial search results based on their relevance to a query, improving the accuracy and contextual appropriateness of the retrieved documents. The previous system only implemented a basic cosine similarity ranking approach, which can miss nuanced relationships and subtle contextual details in the documents. To further advance retrieval precision and overcome limitations in capturing nuanced contextual relevance, this semester, we explored two advanced methods: Cohere reranking and two-stage reranking.

Cohere Rerank

Cohere rerank is an approach that leverages advanced natural language processing techniques to enhance the ranking of retrieved documents. It benefits retrieval systems by accurately capturing nuanced semantic relationships between queries and documents. After retrieving initial candidate documents through hybrid search (combining semantic and keyword searches), Cohere rerank assigns relevance scores to each candidate, prioritizing documents based on their contextual similarity and overall relevance to the user's query.

We integrated Cohere rerank into our retrieval pipeline by introducing an additional step after initial candidate retrieval. After obtaining candidate documents from Pinecone through semantic search, these documents are reranked using Cohere's rerank-english-v3.0 model. This reranking step refines the order of the top 10 documents, enhancing the precision and relevance of the final response provided to users.

Two-Stage Rerank

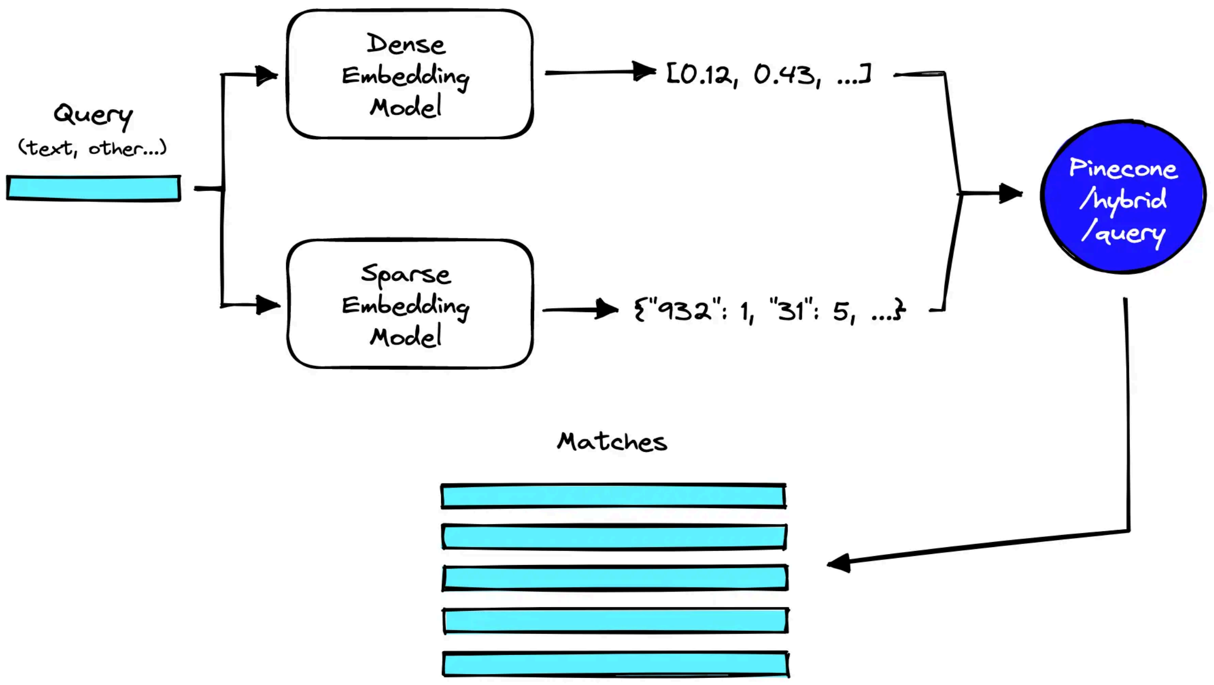
Two-stage reranking uses two phases: a fast retrieval stage with embedding models followed by a computationally intensive reranking stage to refine the retrieved documents and improve relevance.

Phase 1, the fast retrieval stage, is used to narrow down the search space so that fewer documents are passed on to the computationally expensive second phase. The fast retrieval embedding model quickly searches for potentially relevant documents based on semantic similarities to the user's query. The search retrieves as many potentially relevant documents as possible even if some of the documents are not the most accurate.

In phase 2 the re-ranking occurs to the smaller set of documents. This step is crucial for maximizing LLM performance. A cross-encoder re-ranking model takes the retrieved documents and the user's query as input and reorders them based on relevance to the query, prioritizing the most contextually accurate and relevant documents.

Hybrid Search

Hybrid search is a popular architecture pattern for vector embeddings that combines both semantic search and keyword search (also called token-based search). By integrating these approaches, hybrid search enhances search quality and performance by leveraging the strengths of both methods. Bhagdev et al. introduced the hybrid search concept in 2008, demonstrating that it significantly improved precision and recall compared to pure semantic or keyword search in a study involving 18,000 documents. One of the primary motivations for this approach was the limitation of semantic search, which relies solely on document content and struggles to infer similar meanings between words. Hybrid search addresses this shortcoming by incorporating keyword search, ensuring a more comprehensive retrieval of relevant information.



The previous system relies solely on semantic search using dense embeddings to find the documents with relevant content that is similar to the query's embedding. While this ensures high precision, semantic search is not able to handle finding similar meanings among words and always find exact matches. By integrating keyword search, we can increase the search quality for queries that might not use the same exact wording as our documents, as keyword search supports finding conceptually similar information even if there are no keyword matches in the index.

To implement the hybrid search, we will add a sparse embedding index to our vector database in Pinecone. We also need to adapt our current search algorithm to use a combination of semantic and keyword searches to pull relevant documents before our reranking step. We are in the process of implementing this and experimenting with how much our search will rely on semantic versus keyword search.

Abstention

Abstention is a critical mechanism in the Generative AI Executive Advisor, which is designed to mitigate hallucinations by preventing the model from generating responses when there is insufficient confidence in the retrieved evidence. In the context of this project, hallucinations would be referred to as instances where the model produces inaccurate or misleading answers, which are not grounded in the provided financial documents, to executives.

To achieve this, the system uses the abstention mechanism that enables the model to withhold an answer when the relevant information does not contain sufficient information, causing the confidence of the response to fall below a certain threshold. The process is implemented using G-Eval metrics, a structured LLM-Eval framework that employs LLM-as-a-judge principles to access model outputs. G-Eval leverages Chain of Thought (CoT) prompting, which breaks down the evaluation process into logical steps before determining the final score. Given an input query and a generated response, G-Eval systematically evaluates the response based on accuracy, completeness, coherence, and factual alignment with the retrieved documents. It first generates structured evaluation steps from an LLMTestCase, which includes the input and actual output. The model assigns a probability-weighted score between 1 and 5 and normalizes the response's confidence based on LLM token probabilities.

Performance

Document processing performance in the updated system was improved by a factor of 10. A complete corpus load of 27 PDF documents consisting of thousands of pages spanning 4 years of financial data takes approximately 2.5 hours versus 24+ hours in the original system. The performance improvement will allow new documents to be added quickly and with ease to expand the RAG potential of the chatbot.

We have completed the ConfidentCheck function and successfully tested it through unit tests, confirming its effectiveness in evaluating the responses. The next step is integrating the function into our chatbot, allowing it to determine whether to provide an answer based on confidence scores. As development progresses, we may also explore additional abstention methods to enhance the accuracy of the responses further.

Goal

Our metrics for success will be to look at three main metrics: context relevance, answer relevance, and groundedness. Context relevance measures the relevance of the retrieved sources with respect to the question asked, answer relevance measures the relevance of the answer with respect to the question asked, and groundedness measures the relevance of the answer with respect to the retrieved sources. We will compare our version of the AI Executive Advisor against the version we received from the previous semester. Our goal is to improve how the model scores on these metrics as well as 1.) increase the speed and efficiency of our Retrieval-Augmented Generation system and 2.) add support for different backend processing environments, such as Azure and Ollama.

Next Steps

At this point, all we have left to do is: 1.) the abstention function implementation so that our chatbot indicates if an answer can't be found in the documents and 2.) the hybrid search implementation to improve the efficiency and relevance of the documents. However, while we're working on those tasks, there will be some challenges and unknowns we will be up against.

The first of those challenges is speed. Despite our efficiency improvements to the data processing pipeline, we hope to further reduce the time it takes the chatbot to generate a response. With advanced AI available to the general public, such as ChatGPT, people are used to almost instantaneous answers and we want our chatbot to be as near instantaneous as possible.

Our second challenge is scalability. Currently, we only process 27 documents. Our Pinecone database that contains the embeddings of these documents can become expensive if we expand it. Additionally, the current retrieval and reranking process would take longer if we added more documents to include more historical data.

Finally, our third challenge is cost. For the sake of performance, we are using OpenAI's GPT 3.5 and 4 models for embedding and summarizing documents. This is state of the art technology, but depending on how frequently the chatbot is used and queried, this project can quickly become expensive and may not provide enough value to compensate for that cost. We've already run into issues with running out of credits early in our testing and development process and had to ask for more.

References

About hybrid search. (2015). Google Cloud. https://cloud.google.com/vertex-ai/docs/vector-search/about-hybrid-search

Bhagdev, Ravish & Chapman, Sam & Ciravegna, Fabio & Lanfranchi, Vitaveska & Petrelli, Daniela. (2008). Hybrid Search: Effectively Combining Keywords and Semantic Searches. 554-568. 10.1007/978-3-540-68234-9_41.

Briggs, J. (n.d.). Getting Started with Hybrid Search | Pinecone. https://www.pinecone.io/learn/hybrid-search-intro/

Lee, Robert. (2023). Hybrid search - Azure AI Search. https://learn.microsoft.com/en-us/azure/search/hybrid-search-overview

Goh, H. W. (2025, January 13). Benchmarking hallucination detection methods in RAG. Towards Data Science. https://towardsdatascience.com/benchmarking-hallucination-detection-methods-in-rag-6a03c555f063/

Mulani, A. (2024, February 24). Improve Retrieval Augmented Generation (RAG) with Re-ranking. *Medium*. https://medium.com/@ashpaklmulani/improve-retrieval-augmented-generation-rag-with-re-ranking-31799c670f8e