SEC Filings QA Agent Document

Gautam Kumar
gautam.baranwal2003@gmail.com

1. Introduction

The document outlines the strategy implemented to build SEC Filings QA Agent.

- **Dataset** - Filing report of 15 Public Companies have been used as a Dataset. Downloaded all the files as(.html).
- **Vector store** - After parsing the .html files. I have created a vector store using chromaDB where the chunks created from html files are pushed using **Google embeddings**
- **QA Agent -** I have prepared a RAG(Retrieval Augmented Generation) model for this assignment as this is best out there as this has the most resources to study.

2. Getting Data in workable format

- Data Downloading -> For each of 15 tickers from different sectors I have downloaded last 5 10-K forms filed by the company and in between time duration other forms like 10-Q, 8-K, 3,4,5, DEF 14A. Together with a metadata.json file for each company having data about each file like filedAt, CiK, Name, Ticker etc.
- Data Abstraction ->
  - To clean the data and remove unnecessary tags data.
  - Used a regex based method to identify the sections that split the document into “items” (e.g., "Item 1A. Risk Factors"). This preserves the document's official structure.

    - This cleaned parsed data are ready to be created as chunks using a *RecursiveCharacterTextSpliter*. Now each chunk has it's own metadata from metadata.json and section.
    - Vector Store Creation: All processed chunks from all companies are embedded using GoogleGenerativeAIEmbeddings and stored in a single, persistent ChromaDB vector store (sec_filings_db).

3. QA Agent
    - Query Parser - This is the brain of the agent which will think for our queries and how we will search it in our vector store. The most crucial part of entire system.
    - Entity Extraction: An LLM (gemini-1.5-flash-latest) is used to parse the query and extract key entities (tickers, sectors, form types, years) into a structured QueryMetadata object.
    - Form Selection - To add relevant form types to search *FORM_TYPE_KEYWORDS* is used to match the query to correct form to look on. A query containing "insider trading" automatically adds Forms 3, 4, and 5 to the search filter. For broad analytical questions, it intelligently defaults to the 10-K.
    - Semantic Section Routing: The parser compares the core search query to a list of descriptions for key SEC sections. Using semantic similarity, it identifies the most relevant section (e.g., routing a question about "risks" to "Item 1A") and adds it to the filter.
    - Precision Retrieval: The generated metadata filter is passed to the ChromaDB retriever. The retriever uses search_type="mmr" (Maximal Marginal Relevance) to fetch a set of documents that is both relevant to the query and diverse, which is crucial for answering comparative questions.
    - **Synthesis:** The retrieved documents are "stuffed" into a custom, detailed prompt that instructs the final LLM to act as a senior financial analyst. The LLM's task is to synthesize a single answer based *only* on provided context, identify common themes, and state when the information is insufficient.

3. Challenges Addressed & Design Decisions

- The development of this agent was an iterative process of identifying and solving key challenges.
- Challenge: Data Loss from HTML Cleaning.
    - Problem: Initial attempts to clean the HTML by removing all <table> tags resulted in the complete loss of critical financial data.
    - Solution: We engineered a sophisticated extract_text_and_tables function that converts tables to a structured JSON format before cleaning, preserving all numerical data while still removing unwanted HTML noise. This was a critical breakthrough for enabling quantitative analysis.
- Challenge: The "Wrong Section" Problem.

  Initially the model was getting wrong section of documents for a query.
    - Solution: We implemented intelligent section splitting during data processing and semantic routing in the QueryParser.
- Challenge: The "Top-K" Problem in Comparative Questions.
    - Problem: When comparing two companies, the retriever would often find the top 8 most relevant documents, but they would all be from the same company, leading to a failed comparison.
    - Solution: We switched the retriever's search algorithm to search_type="mmr", which optimizes for diversity and ensures that the retrieved context is balanced across the different entities in the query.

4. Capabilities, Limitation and Trade offs

- Capabilities
    - The model worked extremely well Identifying ticker, time, sector, handling multi dimensional queries.
    - System is Robust to preserve information from messy .html files
    - Source attribution is reliable and precise.
    - No Hallucination. Through the testing part on some list of vague queries. The model is not answering falsely well. It just says it has insufficient information.
- Limitations
    - Limited resources - I have prepared the dataset for only last 5 year filling and restrict the no. of form 3,4,5 to 5 each.

    - Parser Reliance - The quality of answer is highly dependant on it as it identifies the ticker, section, form type etc. It can make mistakes on poorly phrased/ vague questions.
- Trade-offs:
    - Speed vs. Intelligence: The initial parsing step adds a small amount of latency to each query. This trade-off was explicitly made to prioritize the accuracy and reliability of the retrieval process over raw speed. A simple, fast agent that gives wrong answers is useless; a slightly slower agent that gives correct answers is invaluable.

5. Conclusion and Future Work

This project has successfully produced a high-quality, intelligent QA agent that meets all the requirements of the assignment. The final architecture is robust, scalable, and demonstrates a sophisticated understanding of both AI engineering and financial analysis.

Potential future improvements could include:

- **Implementing a Formal Evaluation Pipeline:** Using a framework like Ragas to create an automated evaluation dataset would allow for objective, repeatable measurement of the system's performance.
- **Hybrid Search:** Integrating a traditional keyword search (like BM25) alongside the semantic search could improve retrieval for queries containing specific financial acronyms or jargon.
- **User Interface:** Building a simple web interface (e.g., with FastAPI and Streamlit) would allow users to explicitly select filters, making the agent even more precise and user-friendly.