Arvind Vinod
Stubbs
COE 379L

Project 4 Report

The goal of this project was to develop a natural language question-answering system for my personal financial data using Retrieval-Augmented Generation (RAG). Recognizing the severe limitations of existing personal finance applications, particularly around bank integration and manual entry support, this project aimed to create an intelligent layer on top of my finance tracking spreadsheet while leaving room for actually building an app that logs transactions as well in the future. The core problem addressed was the tedious and time-consuming nature of extracting meaningful insights from the spreadsheets through manual filtering, formulas, and pivot tables. The proposed solution leverages natural language processing and information retrieval techniques to allow me to ask conversational questions about spending habits, trends, and financial status, thereby transforming a simple and static data repository into an interactive financial assistant. This endeavor focused on addressing key challenges such as multi-currency support, semantic understanding of financial queries, temporal reasoning, and contextual retrieval of relevant transactions. The ultimate aim was not to replace my existing tracking methods but to significantly enhance their analytical capabilities and improve their use cases through an intuitive natural language interface.

The primary data source for this project was a personal finance spreadsheet I maintained for over a year. Each entry in the spreadsheet included fields for the transaction date, the place of expenditure (merchant name or context), the amount spent in USD, the equivalent amount in INR (calculated using Google Finance), the expense category (e.g., groceries, rent, travel), and the source of the funds (bank accounts, credit cards, or cash). The spreadsheet also tracked transactions in NTD and THB for periods of study-abroad and travel in 2024, with dynamic currency conversion facilitated by Google Finance. A crucial aspect of why I chose to maintain the data was the inclusion of cash transactions and local credit unions, which are often poorly supported by automated finance applications but are essential for a comprehensive understanding of personal finances, especially when traveling. To facilitate semantic search and LLM-powered reasoning, each individual transaction was converted into a natural language string. For example, a transaction on May 30th, involving a TWD 907 purchase at Macho Tacos using cash and categorized as 'Eating/Drinking Out', was represented as: "On May 30th, I spent TWD 907 ($28.28) at Macho Tacos using cash. It was categorized as 'Eating/Drinking Out'." These textual representations formed the basis for embedding and subsequent retrieval.

The project employed a data processing pipeline to transform the raw spreadsheet data into a format suitable for the RAG system. The initial step involved data loading, which was designed to handle potential variations in file encodings (UTF-8, Latin-1, ISO-8859-1, CP1252) and automatically detect common date formats (MM/DD/YYYY vs. DD/MM/YYYY). I found ISO-8859 worked the best for my data. The system also addressed inconsistencies in currency formatting, such as the use of comma separators, which was useful when making the sheet but not when utilizing the data elsewhere. Following data loading, each transaction was converted into a natural language description, incorporating the date, merchant, amount in the original currency and its USD equivalent, the expense category, and the payment source. For multi-currency transactions, both the native currency amount and the converted USD figure were included to maintain currency awareness. These strings were then processed by the embedding model, SentenceTransformers' all-MiniLM-L6-v2, to generate 384-dimensional vector embeddings. This process aims to capture the semantic meaning of each transaction, allowing for effective similarity-based retrieval. This is what allows me to type in a query that is close enough and it still gets the relevant data. The generated embeddings were indexed and stored in the FAISS (Facebook AI Similarity Search) library, which provides a data structure for performing similarity searches.

A key component of the project was the smart retrieval system, which aimed to enhance the relevance of retrieved transactions beyond basic vector similarity search. This system incorporates dynamic K selection which automatically adjusts the number of transactions to retrieve based on the complexity of the query itself. Pattern matching was used to detect references to dates, categories, currencies, and aggregations within the query, leading to an increase in the number of retrieved transactions (k) for more complex or temporally broad queries. However I noticed I could not exactly nail down the accuracy all the time with the dynamic selection so there is also a manual slider for the amount of transactions it receives.

Pre-filtering techniques were also implemented to extract explicit and implicit date ranges (e.g., "last month," "Q1 2024") and identify mentioned currencies and categories, applying Boolean filters before semantic search even takes place to improve precision and reduce hallucination rates with the LLM. Decision logic was implemented to determine the optimal search strategy for each query. To handle potentially large result sets, the system incorporated techniques for processing results in chunks to avoid exceeding the language model's context window. This involved analyzing representative samples from the beginning, middle, and end of the result set and combining findings across these chunks to provide comprehensive answers. In practice, I found this technique to be somewhat redundant as most of the relevant transactions were found during the first chunk anyways.

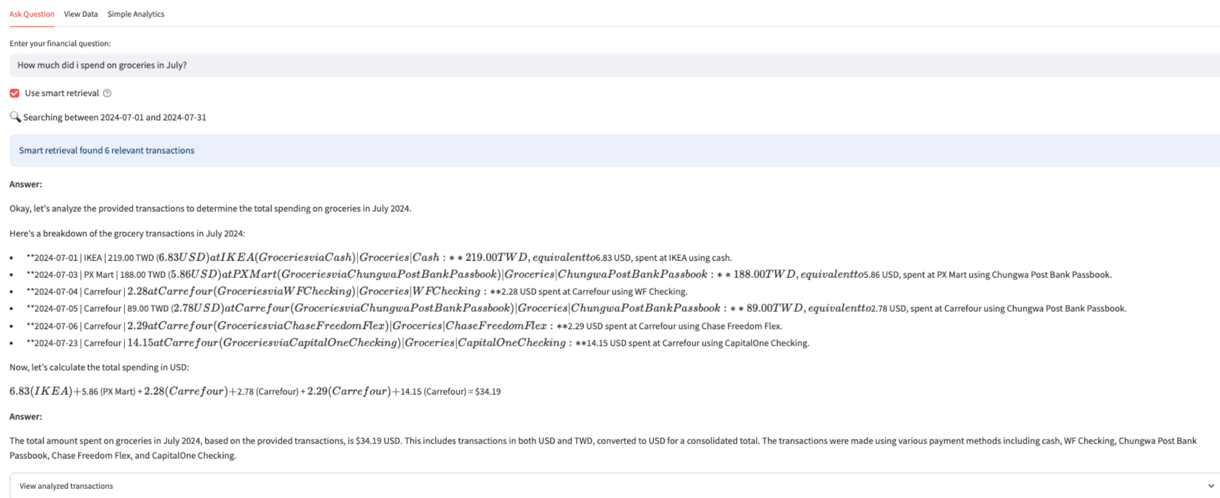
The query processing and response generation phase involved sophisticated techniques for understanding user queries and formulating informative answers. The system was designed to perform date range extraction, recognizing absolute references (e.g., "January 2024") with the ability to infer the year from the context when ambiguous. It also included currency and category detection, identifying currency codes (USD, INR, THB) and names (dollars (assumes the USD), rupees), as well as recognizing common expense categories (dining out is the same as Eating Out) from the query context, while maintaining awareness of the primary versus secondary currencies within the dataset. Prompt engineering played a critical role in guiding the language model. Prompts were structured with the retrieved transactions presented in a consistent format, along with dataset-level metadata such as the overall date range and available currencies. Explicit guidelines were provided within the prompts to instruct the LLM on how to handle currency conversions and comparisons accurately without hallucinations or pulling unnecessary outside information. The generated responses were formatted to always display the original currency and its USD equivalent, highlight significant patterns or outliers identified in the data, and preserve source information when relevant to the user's query.

To illustrate the system's capabilities, several example queries and their responses are recorded below:

Example 1: Basic Spending Analysis

Query: "How much did I spend on groceries in July?"

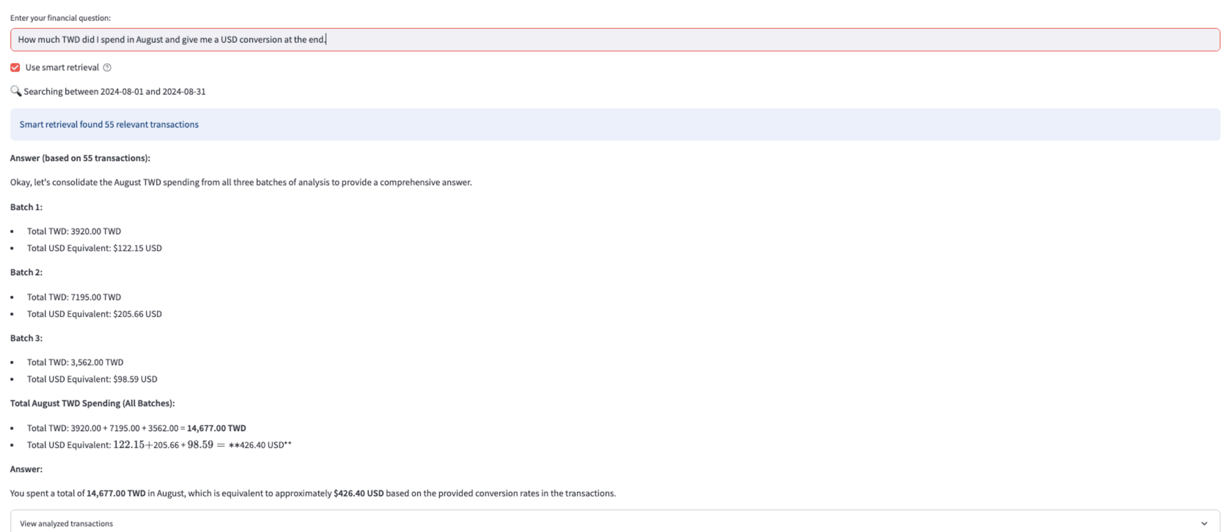
Response:



Example 2: Multi-Currency Comparison

Query: "How much TWD did I spend in August and give me a USD conversion at the end."

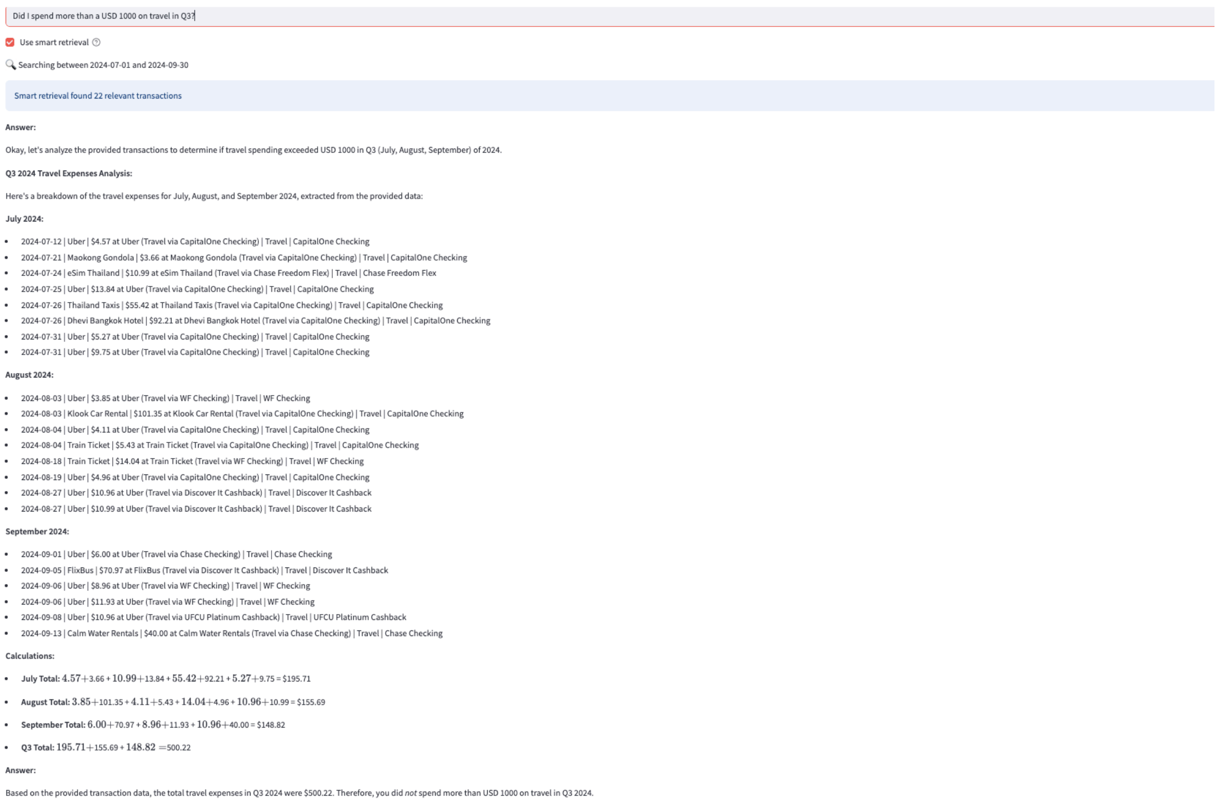
Response:



Example 3: Comparative Temporal Analysis

Query: "Did I spend more than a USD 1000 on travel in Q3?"

Response:



Did I spend more than a USD 1000 on travel in Q3?

Use smart retrieval

Searching between 2024-07-01 and 2024-09-30

Smart retrieval found 22 relevant transactions

Answer:

Okay, let's analyze the provided transactions to determine if travel spending exceeded USD 1000 in Q3 (July, August, September) of 2024.

Q3 2024 Travel Expenses Analysis:

Here's a breakdown of the travel expenses for July, August, and September 2024, extracted from the provided data:

July 2024:

- 2024-07-12 | Uber | $4.57 at Uber (Travel via CapitalOne Checking) | Travel | CapitalOne Checking
- 2024-07-21 | Maokong Gondola | $3.66 at Maokong Gondola (Travel via CapitalOne Checking) | Travel | CapitalOne Checking
- 2024-07-24 | eSim Thailand | $10.99 at eSim Thailand (Travel via Chase Freedom Flex) | Travel | Chase Freedom Flex
- 2024-07-25 | Uber | $13.84 at Uber (Travel via CapitalOne Checking) | Travel | CapitalOne Checking
- 2024-07-26 | Thailand Taxis | $55.42 at Thailand Taxis (Travel via CapitalOne Checking) | Travel | CapitalOne Checking
- 2024-07-26 | Dhevi Bangkok Hotel | $92.21 at Dhevi Bangkok Hotel (Travel via CapitalOne Checking) | Travel | CapitalOne Checking
- 2024-07-31 | Uber | $5.27 at Uber (Travel via CapitalOne Checking) | Travel | CapitalOne Checking
- 2024-07-31 | Uber | $9.75 at Uber (Travel via CapitalOne Checking) | Travel | CapitalOne Checking

August 2024:

- 2024-08-03 | Uber | $3.85 at Uber (Travel via WF Checking) | Travel | WF Checking
- 2024-08-03 | Klook Car Rental | $101.35 at Klook Car Rental (Travel via CapitalOne Checking) | Travel | CapitalOne Checking
- 2024-08-04 | Uber | $4.11 at Uber (Travel via CapitalOne Checking) | Travel | CapitalOne Checking
- 2024-08-04 | Train Ticket | $5.43 at Train Ticket (Travel via CapitalOne Checking) | Travel | CapitalOne Checking
- 2024-08-18 | Train Ticket | $14.04 at Train Ticket (Travel via WF Checking) | Travel | WF Checking
- 2024-08-19 | Uber | $4.96 at Uber (Travel via CapitalOne Checking) | Travel | CapitalOne Checking
- 2024-08-27 | Uber | $10.96 at Uber (Travel via Discover It Cashback) | Travel | Discover It Cashback
- 2024-08-27 | Uber | $10.99 at Uber (Travel via Discover It Cashback) | Travel | Discover It Cashback

September 2024:

- 2024-09-01 | Uber | $6.00 at Uber (Travel via Chase Checking) | Travel | Chase Checking
- 2024-09-05 | FlixBus | $70.97 at FlixBus (Travel via Discover It Cashback) | Travel | Discover It Cashback
- 2024-09-06 | Uber | $8.96 at Uber (Travel via WF Checking) | Travel | WF Checking
- 2024-09-06 | Uber | $11.93 at Uber (Travel via WF Checking) | Travel | WF Checking
- 2024-09-08 | Uber | $10.96 at Uber (Travel via UFCU Platinum Cashback) | Travel | UFCU Platinum Cashback
- 2024-09-13 | Calm Water Rentals | $40.00 at Calm Water Rentals (Travel via Chase Checking) | Travel | Chase Checking

Calculations:

- **July Total:** 4.57+3.66 + 10.99+13.84 + 55.42+92.21 + 5.27+9.75 = $195.71
- **August Total:** 3.85+101.35 + 4.11+5.43 + 14.04+4.96 + 10.96+10.99 = $155.69
- **September Total:** 6.00+70.97 + 8.96+11.93 + 10.96+40.00 = $148.82
- **Q3 Total:** 195.71+155.69 + 148.82 =500.22

Answer:

Based on the provided transaction data, the total travel expenses in Q3 2024 were $500.22. Therefore, you did *not* spend more than USD 1000 on travel in Q3 2024.

This project successfully demonstrated the potential of Retrieval-Augmented Generation to change the way I see personal financial data analysis. By creating a natural language interface, the system transforms the process of gaining insights from spreadsheets from a technical exercise into a simple conversation. The implementation effectively addresses the complexities of multi-currency transactions, temporal reasoning, and the semantic understanding of financial queries. Key contributions of this project include a robust multi-currency RAG system, dynamic retrieval strategies that adapt to the complexity of the query, advanced natural language date parsing, and chunked processing techniques for efficiently handling large transaction datasets. Future work could explore the integration of anomaly detection in spending patterns, comparison features between temporal contexts, automatic identification of recurring transactions, predictive analytics for future expenses. The big roadmap puts the end goal at an app that uses Plaid to get all the transaction info from the big banks but also allows manual entry for cash and credit union purchases. All of this data would then be compiled into a csv that is fed into the workings of this project to empower the analysis features. The RAG approach demonstrated here holds promise for application in various other personal data domains beyond finance.

References:

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., … Amodei, D. (2020, July 22). Language models are few-shot learners. arXiv.org. https://arxiv.org/abs/2005.14165

Charming Data. (n.d.). *Integrate AI + RAG into an App on NYC Buildings (Part 3)*. YouTube. https://www.youtube.com/watch?app=desktop&v=tciN6xKCEnk&ab_channel=CharmingData

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., & Wang, H. (2024, March 27). Retrieval-augmented generation for large language models: A survey. arXiv.org. https://arxiv.org/abs/2312.10997

Google. (n.d.). *Gemini API | google AI for developers*. Google. https://ai.google.dev/gemini-api/docs

Streamlit docs. Streamlit documentation. (n.d.). https://docs.streamlit.io/