Financial Assistant Chatbot Documentation

1. Overview

The Financial Assistant Chatbot is a Gradio-based web application designed to answer financial queries by combining multiple data sources and processing methods. It integrates:

- **Gradio UI** for user interaction.
- **Gemini LLM (Google Generative AI)** to generate responses.
- **Alpha Vantage API** for real-time stock data.
- **NewsAPI** to fetch the latest company news.
- **FAISS with SentenceTransformer** for context retrieval from pre-defined financial documents.
- Branching logic to handle various query types (image analysis, sentiment analysis, news lookup, direct stock quotes, and general questions).

---

2. Setup

2.1 Prerequisites

- **Python 3.8+**
- Required libraries:
  - `gradio`
  - `google.generativeai`
  - `faiss` (or `faiss-cpu` depending on your environment)
  - `numpy`
  - `sentence-transformers`
  - `requests`
  - `Pillow`
  - `base64`
  - `re`

2.2 Installation

1. **Clone the repository** or place the project code in your working directory.

Install dependencies using pip. For example:

pip install gradio google-generativeai faiss-cpu numpy sentence-transformers requests pillow

2.
3. **Set up API Keys:**
   Replace the placeholder API keys in the code (or set them as environment variables) for:

   - Gemini LLM: `gemini_api_key`
   - Alpha Vantage: `av_api_key`
   - NewsAPI: `news_api_key`
4. For example, you can export keys in your shell or configure them in a `.env` file and load them via Python.

Run the application:

python your_script.py

5. This will launch the Gradio interface locally where you can interact with the chatbot.

---

3. API Usage

3.1 Gemini LLM API

- **Purpose:**
  Handles various tasks such as generating natural language responses, converting company names to ticker symbols, and performing prompt-based analyses (sentiment, image interpretation).

- **Usage in Code:**

  - The function `call_gemini_llm(prompt, gemini_api_key, model="gemini-2.0-flash")` configures the Gemini API using the provided key, constructs a prompt, and returns the generated text.
  - It's used across multiple modules (e.g., ticker conversion, context-based general query responses, and image analysis).
- **Notes on Fine-Tuning:**
  While direct fine-tuning of Gemini LLM may not be exposed, prompt engineering is used to adapt responses. For specialized behavior, consider adjusting prompt templates or using additional context in the prompt.

3.2 Alpha Vantage API

- **Purpose:**
  Fetches real-time stock data such as current price, change, and change percent for a given ticker symbol.

- **Usage in Code:**

    - The function `get_stock_quote_alpha_vantage(symbol, av_api_key)` builds the API request URL, sends the query, and parses the JSON response.
    - The function extracts key fields from the “Global Quote” returned by Alpha Vantage.
- **Important Considerations:**
  Ensure that you monitor API usage quotas and handle error responses gracefully. The code includes basic error handling to capture exceptions or missing data.

3.3 NewsAPI

- **Purpose:**
  Retrieves recent news headlines about a specified company, which can be used for sentiment analysis or just to display current news.

- **Usage in Code:**

    - The function `fetch_company_news(company, news_api_key, count=5)` constructs the query parameters (including language, sorting, and count), sends the request, and extracts headlines from the returned JSON.
    - These headlines are then used by `perform_sentiment_analysis` to generate a sentiment summary via Gemini LLM.

4.1 Fine-Tuning vs. Prompt Engineering

- **Fine-Tuning:**
    - Direct fine-tuning of large language models like Gemini is typically handled on the backend by the provider.
    - **CSV Dataset Fine-Tuning:** In this project, we also implement fine-tuning using CSV datasets containing domain-specific financial data, terminology, and example queries with their corresponding responses.
    - This creates a more specialized model that better understands financial contexts and terminology specific to our application.
- **Prompt Engineering Techniques:**
    - **Role Specification:** Prompts often begin with a statement like "You are a financial analyst…" to set the context.
    - **Clear Instructions:** Each branch of the logic (image analysis, sentiment analysis, ticker conversion) constructs specific prompts to direct the LLM.

- **Context Inclusion:** For general queries, relevant financial documents are fetched from a vector database and included in the prompt.

4.2 Customizing Prompts

- **Ticker Conversion Example:**
  The prompt instructs the LLM to output only the stock ticker symbol, ensuring concise conversion.
- **Image Analysis:**
  The image file is converted to a base64 string and inserted into a detailed prompt that explains the type of analysis required.
- **Sentiment Analysis:**
  A prompt that lists recent news headlines is built to have the LLM provide a sentiment summary with reasoning.

---

5. Context Retrieval via Vector Database

5.1 Embedding Model Setup

- **Tool:**
  `SentenceTransformer` model “all-MiniLM-L6-v2” is used to compute text embeddings.
- **Purpose:**
  Converts financial documents into dense vector representations for similarity search.

5.2 FAISS Integration

- **Index Creation:**
  - The embeddings of all context documents are computed and stored in a NumPy array.
  - An IVF (Inverted File) index is created using FAISS for efficient approximate nearest neighbor search.
  - The index is trained on these embeddings and then used to add vectors for search.
- **Context Retrieval Function:**
  - `retrieve_context(query, top_k=2)` computes an embedding for the query, searches the FAISS index, and retrieves the top-k most similar documents.
  - This retrieved context is then appended to the prompt for the LLM, helping it generate a more informed answer.

---

6. Fallback Mechanisms

6.1 Branching Logic

The application's main function `process_chat(user_query, image_file)` uses a series of conditional checks to determine which branch of logic to follow:

1. **Image/Graph Analysis Branch:**
    - Triggered if an image is provided or if the query includes keywords like "upload image" or "analyze chart."
    - If no image file is provided despite the query hint, a message prompts the user to upload an image.
2. **Sentiment Analysis Branch:**
    - If the query mentions "sentiment analysis," the function attempts to extract a company name.
    - If extraction fails, a fallback message indicates the issue.
3. **News Headlines Branch:**
    - Checks if the query contains "news."
    - If no company name can be determined, the user is prompted to specify a company.
4. **Direct Stock Quote Branch:**
    - Looks for phrases such as "stock price" or "quote for."
    - Uses regex to extract a potential ticker or company name, then converts the name via Gemini LLM if necessary.
5. **General Query Branch (Fallback):**
    - When none of the above conditions are met, the application defaults to the general query process.
    - This branch retrieves additional context from the vector database and constructs a combined prompt for the Gemini LLM.

6.2 Error Handling and Default Responses

- **API Error Handling:**
    - Each API call (Gemini LLM, Alpha Vantage, NewsAPI) is wrapped in try-except blocks to catch and return descriptive error messages.
- **Graceful Fallback:**
    - If specific branches (like ticker extraction or sentiment analysis) fail to extract necessary information, the system responds with a clear fallback message instructing the user on how to modify their query.

---

7. Summary

This project integrates multiple data sources and processing techniques to deliver a responsive financial assistant chatbot. By:

- Setting up a robust environment with Gradio, FAISS, and various API integrations,
- Leveraging prompt engineering for controlling the Gemini LLM output,
- Employing a vector search mechanism for context enrichment, and
- Implementing multiple fallback strategies to handle errors or ambiguous queries,

the system is designed to provide detailed and context-rich answers to a wide range of financial questions.