**Retrieval-Augmented Generation (RAG) Based Question Answering (QA) Bot**

Project Overview:

This project implements a Retrieval-Augmented Generation (RAG) model for a Question Answering (QA) bot for business use cases. The bot utilizes Pinecone DB for storing and retrieving document embeddings and Cohere's API for generating coherent, context-based answers. This system allows users to query documents efficiently and receive relevant, generated responses based on retrieved document chunks.

Key Components:

* **Vector Database (Pinecone):** Stores document embeddings for efficient retrieval.
* **Generative Model (Cohere API):** Generates coherent responses based on retrieved documents.
* **Embedding Model (Sentence-BERT):** Converts both documents and queries into vector representations for similarity search.

Project Structure:

The project follows a modular and scalable structure:

1. **Data Preprocessing (data\_preprocessing.py)**
2. **Document Retrieval (retrieval.py)**
3. **Answer Generation (generation.py)**
4. **Main Query Handler (app.py)**

**Setup Instructions**

**Prerequisites:**

* Python 3.7+
* Colab or Local Environment (ensure pip is installed)
* Pinecone API Key
* Cohere API Key

**Installation**

1. **Clone the repository:**

git clone https://github.com/your\_username/rag-qa-bot.git

cd rag-qa-bot

1. **Install dependencies:**

pip install -r requirements.txt

1. **Set up environment variables: Create a .env file in the root directory and add your API keys:**

PINECONE\_API\_KEY=your\_pinecone\_api\_key

COHERE\_API\_KEY=your\_cohere\_api\_key

**Running the Bot**

1. Running the Colab Notebook

* Open rag\_qa\_bot.ipynb in Google Colab.
* Follow the instructions in the notebook to upload your dataset, generate embeddings, and run queries.
* Example:

query = "How to create a new account?"

answer = answer\_query(query)

print(answer)

1. Running Locally

You can also run the bot locally (if you prefer not to use Colab):

**python app.py**

**Usage Instructions**

**Example Queries:**

Once the bot is running, you can ask questions like:

* "How to reset my password?"
* "What are the account recovery options?"

The system will retrieve the relevant sections of the document and generate a coherent answer.

**Example Output:**

**Query:**"How to create a new account?"

**Response:**"To create a new account, follow these steps: Go to the signup page, fill in your details, and submit the form. You will receive a confirmation email to verify your account."

**Challenges Faced**

**1. Document Splitting:**

We faced challenges splitting larger documents into manageable chunks for embedding. Initially, longer chunks caused issues with retrieval accuracy. To solve this, we tested different chunk sizes to ensure each one was contextually relevant yet small enough for efficient embedding.

**2. API Rate Limits:**

Cohere API has limits on token usage per request, which needed to be considered when creating the context for generative models. This required balancing between retrieving enough relevant information while staying within token limits.

**3. Efficient Retrieval:**

Pinecone offers scalable vector storage, but querying large datasets can introduce latency. We optimized retrieval by experimenting with the number of top k results to retrieve the most relevant information efficiently.

**Scalability Considerations**

* Larger Datasets: The system is designed to handle large document sets. As the number of documents grows, increasing Pinecone index replicas and adjusting chunk sizes will help ensure fast retrieval.
* Switching APIs: The system can easily integrate alternative generative APIs (such as OpenAI’s GPT models) by modifying the generate\_answer\_with\_context function in generation.py.
* Deployment: The backend could be deployed as a microservice, scalable via platforms like AWS Lambda (serverless architecture), and the frontend can be hosted separately, allowing for a fully decoupled architecture.

**Additional Documentation**

* **API References:**
  + Pinecone Documentation
  + Cohere Documentation
* **Deployment Instructions:**

Instructions for deploying the system on cloud platforms like AWS, Heroku, or using Docker containers are provided in DEPLOYMENT.md.