Expert Chatbots Using Retrieval-Augmented Generation (RAG) on Azure

Author: Niclas Svanström
Date: 2025-02-22

---

1. Introduction

This document describes the expert chatbot solution created using Retrieval-Augmented Generation (RAG) within a Django framework, deployed on Azure. By allowing users to upload documents—subsequently indexed for search—the system provides precise, domain-specific answers via an orchestrated retrieval and generative pipeline.

Key highlights:

- **Azure AI Search for retrieving relevant chunks.**
- **Reranker to reorder results for higher accuracy.**
- **Multiple model options (4o and 4o mini) for different performance needs.**
- **Continuous deployment via GitHub integration.**

---

2. Key Features

1. **RAG with Azure AI Search**
    - **Leverages user-uploaded data to ground answers in reality, reducing hallucinations.**
    - **Delivers more reliable responses than purely generative models.**
2. **Django-Based Backend**
    - **Handles file uploads, user authentication, and communication with Azure services.**
    - **Straightforward deployment on Azure and easy to maintain.**
3. **Azure Blob Storage**
    - **Secure storage for all uploaded documents.**
    - **Direct integration with Azure AI Search for indexing and retrieval.**
4. **Prompt Flow Integration**
    - **Oversees retrieval steps and calls either the 4o or 4o mini model for response generation.**
    - **Maintains logs of queries and answers for performance tracking.**
5. **Reranker Mechanism**

- **Reranks or shortlists initial search results to highlight the most contextually relevant content.**
- **Significantly boosts final answer quality and user satisfaction.**

6. **Multiple Model Options: 4o and 4o mini**
   - **4o: Provides more in-depth, elaborate answers.**
   - **4o mini: Optimized for faster response times, especially for smaller datasets.**
7. **Filtering for Distributed Content**
   - **Filters out extraneous chunks if important details are scattered across multiple documents.**
   - **Ensures the model only receives the best material for final answer generation.**

---

3. Workflow

1. **User Document Upload**
   - **Users upload files through Django (either a web form or an API endpoint).**
2. **Storage & Indexing**
   - **The uploaded files are stored in Azure Blob Storage and indexed by Azure AI Search.**
3. **Retrieval & Reranking**
   - **When a user queries the system, Azure AI Search returns relevant chunks.**
   - **A reranker reorders these chunks, pushing the most pertinent information to the top.**
4. **Prompt Flow & Model Invocation**
   - **The refined chunks are sent to the Prompt Flow pipeline, which calls either the 4o or 4o mini model.**
   - **Users receive accurate answers grounded in real data.**

---

4. Implementation Details

4.1 Django Backend

- **File Upload & Processing**
  - **Accepts and processes user-uploaded documents.**
  - **Automatically triggers indexing with Azure AI Search when uploads succeed.**

- **Metadata Management**
    - **Stores details (e.g., tags, ownership) to aid in filtering and search.**
    - **Useful for controlling access or narrowing result sets.**

4.2 Azure AI Search Indexing

- **Chunking**
    - **Splits larger files into smaller text chunks, improving retrieval performance.**
- **Schema Design**
    - **Each chunk is stored with fields for content, document ID, and any metadata.**
    - **Allows advanced filters and full-text queries.**

4.3 Reranking

- **Purpose**
    - **Adjusts the order of the retrieved chunks based on their contextual match to the user's query.**
- **Benefit**
    - **Surfaces the best candidates first, improving the overall quality of generated answers.**

4.4 Prompt Flow & Models

- **Pipeline Orchestration**
    - **Coordinates all steps (retrieval → reranking → model call).**
- **4o vs. 4o mini**
    - **4o handles comprehensive or complex questions.**
    - **4o mini delivers quicker responses on smaller scales.**

4.5 Filtering

- **Document-Level Filtering**
    - **Skips documents irrelevant to the user's query, using metadata or tags.**
- **Chunk-Level Filtering**
    - **Excludes sections lacking context, resulting in more concise and accurate prompts.**

---

5. Deployment on Azure (GitHub-Integrated)

1. **GitHub Repository**

    - **The Django project is stored in GitHub, making version control and collaboration straightforward.**

2. **Automatic Deployment**
    - **Azure automatically pulls new code from the designated GitHub branch whenever there's a push.**
    - **Eliminates manual deployment steps and simplifies continuous delivery.**
3. **Rollback & Version Management**
    - **If an issue arises, reverting to a previous commit on GitHub rolls back the deployment.**
    - **Ensures minimal downtime and consistent performance.**

---

6. Conclusion

By combining Azure AI Search with a reranker and prompt flow for models (4o or 4o mini), this chatbot platform provides reliable, domain-specific answers. The use of Django for the backend and Azure for storage/indexing creates a scalable, secure, and straightforward system. Continuous deployment through GitHub ensures updates and improvements can be delivered seamlessly.

Contact Information

- **Name: Niclas Svanström**
- **Email: Niclas.svanstrom@hotmail.com**