Retrieval-Augmented Generation (RAG) Chatbot Report

Amlgo Labs
GitHub Repository: github.com/bilalahmad0210/rag-chatbot

July 12, 2025

1. Document Structure and Chunking Logic

The source document used by the chatbot was a legal agreement from eBay, which includes formal sections such as user obligations, dispute resolution, and disclaimers. To handle such a complex and structured document, it was necessary to divide it into smaller, manageable pieces while maintaining contextual integrity.

We used the `RecursiveCharacterTextSplitter` with a chunk size of 500 characters and an overlap of 50 characters. This ensures that each chunk retains enough context from the previous segment to help the model understand content in flow.

Listing 1: Chunking Logic

```
from langchain.text_splitter import RecursiveCharacterTextSplitter

text_splitter = RecursiveCharacterTextSplitter(
    chunk_size=500,
    chunk_overlap=50
)
chunks = text_splitter.split_text(raw_text)
```

2. Embedding Model and Vector Database

Embedding Model

For semantic search, we used the `all-MiniLM-L6-v2` model from the Sentence Transformers library. It generates 384-dimensional embeddings and is optimized for fast inference on CPU-only systems. It was chosen for its balance between performance and efficiency.

Listing 2: Embedding and Indexing

```
from sentence_transformers import SentenceTransformer
import faiss, pickle

embedding_model = SentenceTransformer("all-MiniLM-L6-v2")
index = faiss.read_index("vectordb/index.faiss")

with open("vectordb/chunks.pkl", "rb") as f:
    chunks = pickle.load(f)["chunks"]
```

Vector Store: FAISS

FAISS (Facebook AI Similarity Search) is used as the vector index. It allows fast cosine similarity searches for top-k matching vectors. This makes it ideal for real-time query processing even in offline and resource-limited setups.

3. Prompt Format and Generation Logic

To generate answers, the top-k retrieved chunks are fed into a prompt template. The prompt guides the model to respond strictly using the provided context and avoid speculation. This minimizes hallucinations and ensures reliability.

Listing 3: Prompt Formatting

```
def build_prompt(context_chunks, query):
    context = "\n".join(context_chunks)
    prompt = (
        "### Instruction:\n"
        "You are a helpful assistant. Use the following context to
           answer the question.\n"
        "If the answer is not present, say: 'The answer is not
           available in the provided document.'\n"
        "\n### Context:\n" + context +
        "\n### Question:\n" + query +
        "\n### Answer:"
    )
    return prompt
```

Language Model: `Mistral-7B-Instruct` (Quantized GGUF Q4_K_S) was chosen for its performance and local deployment capability. It runs entirely offline using `ctransformers`:

Listing 4: Model Initialization

```
from ctransformers import AutoModelForCausalLM

model = AutoModelForCausalLM.from_pretrained(
    "TheBloke/Mistral-7B-Instruct",
    model_file="model/mistral-7b-instruct.Q4_K_S.gguf",
    model_type="mistral",
    max_new_tokens=300
)
```

4. Example Queries and Evaluation

The chatbot was tested with both direct fact-based and inference-based queries. Below are some example results.

Successful Queries

- **Q:** What services does eBay offer to users?
  A: eBay offers tools for listing, pricing, sourcing, and AI-driven features.
- **Q:** Is eBay guidance mandatory?
  A: No, it's completely optional.

- **Q:** What is the role of the arbitrator?
  A: To resolve disputes regarding arbitration terms.

Failure Cases

- **Q:** Can users use AI suggestions from eBay?
  A: Incorrectly answered "not available" despite clues being present.
- **Q:** Does eBay help ship vehicles?
  A: Model failed to infer that eBay is not a broker or shipping agent.

5. System Architecture Diagram

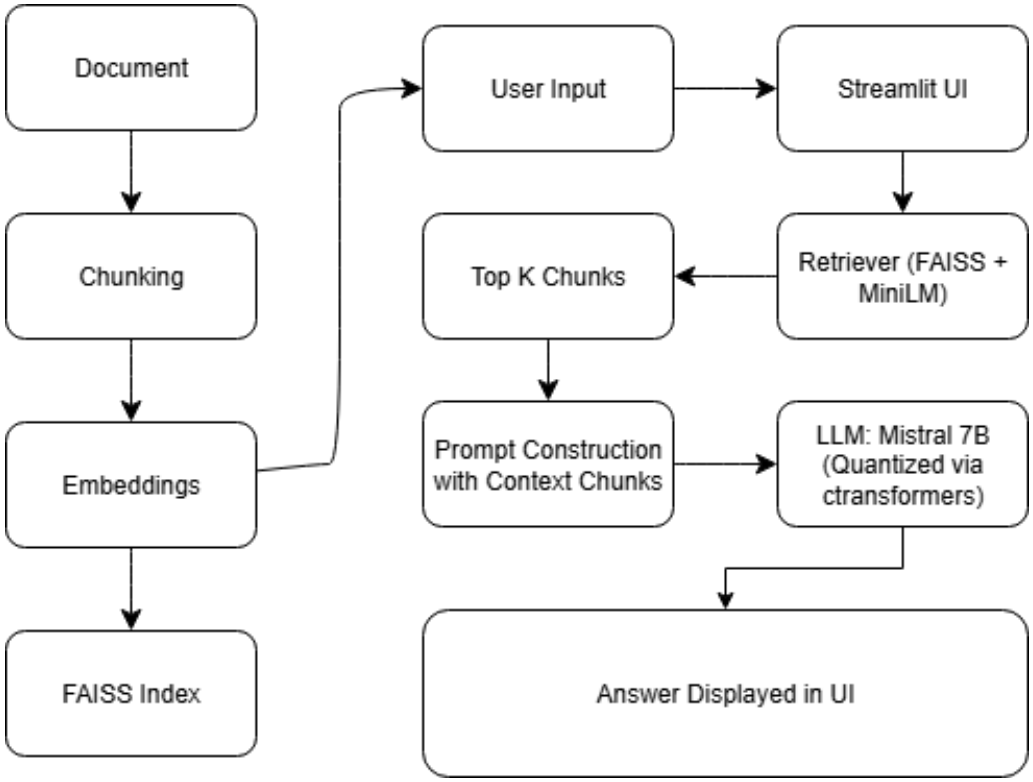

Figure 1: System Architecture of the RAG Chatbot

6. Limitations and Observations

- **Hallucinations:** Mostly avoided due to strict prompting, but possible when queries are vague or poorly phrased.
- **Token Truncation:** Inputs are trimmed to stay under the 512-token limit imposed by the model.
- **Latency:** 6–8 seconds per response on CPU-only setups (Ryzen 5, 16GB RAM).
- **Failure Modes:** Phrasing mismatches and subtle semantic gaps still cause occasional retrieval errors.

7. Conclusion

This RAG chatbot is capable of accurately answering grounded questions using document-specific context. Its offline-first architecture using FAISS, MiniLM, and quantized Mistral 7B makes it suitable for restricted environments. Future improvements could include chunk reranking, answer verification, and question rewriting.

GitHub Repository: https://github.com/bilalahmad0210/rag-chatbot.git