🧠 Intelligent Complaint Analysis for Financial Services

RAG-Powered Chatbot for CrediTrust Financial

Date: July 8, 2025
Author: Ebtisam Muzemil

📌 1. Introduction

CrediTrust Financial is a rapidly growing digital finance provider across East Africa, offering products like credit cards, personal loans, BNPL, savings accounts, and money transfers. With over 500,000 users, the company receives thousands of customer complaints per month.

This project aims to convert unstructured complaint data into actionable insights using a **Retrieval-Augmented Generation (RAG)** chatbot. The tool allows product and compliance teams to ask natural-language questions and receive concise, context-aware answers based on real customer narratives.

🔍 2. Exploratory Data Analysis (EDA)

We began by loading the CFPB complaint dataset, which includes millions of records. After filtering for the five target product categories, we found:

- The top complaint volumes were associated with **BNPL** and **Credit Cards**.
- Complaint narratives varied widely in length: from 10 to over 1000 words.
- Around **300,000** entries included usable narrative text.
- Many complaints lacked narrative fields and were excluded from modeling.

A key insight was that complaints about BNPL often involved late fees and unclear refund processes—exactly the kind of pattern our chatbot should surface.

🧹 3. Data Preprocessing

To prepare the data for embedding:

- We filtered for only five product types:
    - Credit Card
    - Personal Loan
    - Buy Now, Pay Later (BNPL)
    - Savings Account

    - Money Transfers
- Removed entries with empty narratives.
- Cleaned the text by:
    - Lowercasing all characters
    - Removing special characters, URLs, and boilerplate phrases
    - Trimming extra spaces

The cleaned and filtered dataset, saved as `filtered_complaints.csv`, contained approximately **287,345** high-quality complaint narratives.

🔗 4. Text Chunking, Embedding & Vector Store Indexing

Long complaint narratives were chunked using **LangChain's RecursiveCharacterTextSplitter**, with:

- `chunk_size = 300`
- `chunk_overlap = 30`

We chose **`all-MiniLM-L6-v2`** from Sentence Transformers for embedding due to its speed and robust semantic performance.

Embeddings were stored using **FAISS**, with associated metadata (product type, complaint ID, original text). This setup enables fast semantic retrieval and source traceability.

---

5. Retrieval-Augmented Generation (RAG) Pipeline

We implemented a RAG pipeline with these steps:

◆ Query Processing:

- A user inputs a plain-English question.
- It's embedded using the same model as the corpus.
- Top-5 semantically similar chunks are retrieved via FAISS.

◆ Prompt Engineering:

Prompt Template:

```
makefile
CopyEdit
You are a financial analyst assistant for CrediTrust. Your task is to answer
questions about customer complaints. Use the following retrieved complaint
```

```
excerpts to formulate your answer. If the context doesn't contain the answer,
state that you don't have enough information.

Context: {context}
Question: {question}
Answer:
```

◆ LLM Generation:

The context-augmented prompt is sent to a language model (e.g., Mistral 7B via HuggingFace or LangChain). The model returns a grounded, summarized answer.

📊 6. System Evaluation

We evaluated the chatbot using real user questions. Below is a sample of the evaluation table:

Question	Score (1-5)	Retrieved Sample(s)	Comments
Why are people unhappy with BNPL?	5	"Refund not issued after cancellation"	Captured key themes (refunds, late fees)
What are top complaints for savings?	4	"Account blocked without explanation"	Accurate but some irrelevant chunks
Do users report credit card fraud?	5	"Unauthorized charges keep appearing"	Very strong retrieval
Are money transfers delayed often?	4	"Transfer marked complete, funds missing"	Good context, minor prompt tweaking needed
Any common issues with personal loans?	3	"Loan terms changed after signing"	Partially relevant but vague in summary

Key Observations:

- Retrieval quality is strong, especially for specific terms like “BNPL” or “fraud”.
- Prompt robustness is essential to avoid hallucinations.
- Retrieval + prompt tuning improves response accuracy.

💬 7. Streamlit App Interface

We developed a **Streamlit-based** UI allowing non-technical teams to interact with the chatbot.

Features:

- Text input for plain-English questions
- "Ask" button to generate responses
- Generated answer displayed clearly
- **Source excerpts shown below** the answer to enhance transparency
- Clear button to reset session
- Optional: Token-by-token streaming for better UX

Screenshot Preview:
(Insert screenshot of the app here)

---

8. Conclusion & Learnings

This project successfully demonstrated how RAG systems can turn noisy complaint data into strategic insights. Key learnings:

✅ Successes:

- Enabled internal teams to understand customer issues in real-time
- Achieved response times in seconds vs. hours of manual reading
- Built a reusable vector search + LLM framework

⚠️ Challenges:

- Variable text quality in narratives
- Tuning chunking vs. retrieval granularity
- Preventing hallucinations in generation

🔮 Future Improvements:

- Add support for multi-language complaints
- Expand to real-time streaming complaints
- Fine-tune LLM for financial compliance domain

📁 Appendix

Code Repo Structure:

├── data/

│ └── filtered_complaints.csv

├── notebooks/

│ └── eda_and_preprocessing.ipynb

├── src/

│ ├── chunk_embed_index.py

│ └── rag_pipeline.py

├── vector_store/

│ └── faiss_index.bin

├── app.py ← Streamlit UI

└── report/

└── final_report.mdCode Repo Structure: