Title

Intelligent Complaint Analysis for Financial Services: Interim Report

---

1. Introduction

This report documents the progress made in building an intelligent RAG (Retrieval-Augmented Generation) system that helps product managers, compliance officers, and customer support teams query customer complaints in natural language and receive concise, evidence-backed answers.

This interim report focuses on:

- Exploratory Data Analysis (EDA) and data preprocessing
- Embedding and vector database indexing of complaint narratives

---

2. Exploratory Data Analysis (EDA) and Preprocessing

Objective

Understand and prepare the CFPB complaints dataset for embedding and retrieval. The primary focus was filtering for relevant products, cleaning textual data, and summarizing key trends.

Filtering Criteria

We filtered the data to only include the following financial products:

- Credit card
- Personal loan
- Buy Now, Pay Later (BNPL)
- Savings account
- Money transfers

Additionally:

- Rows with missing `Consumer complaint narrative` fields were removed.
- The final cleaned dataset is saved at: `data/filtered_complaints.csv`

Text Cleaning

To improve semantic search quality, we performed:

- Lowercasing
- Removal of special characters and template/boilerplate text (e.g., "I am writing to...")
- Trimmed whitespace and non-informative phrases

EDA Highlights

- **Complaints per Product**: BNPL and Credit Cards had the highest complaint counts.
- **Narrative Length Distribution**: Most narratives ranged from 40 to 250 words.
- **Empty Narratives**: Over 25% of the raw dataset had empty narratives — these were excluded.

Conclusion: The dataset is now filtered, cleaned, and ready for embedding.

3.Text Chunking, Embedding, and Vector Indexing

Goal

Transform cleaned narratives into dense vector representations for semantic search using a vector database.

Text Chunking

Long complaint texts were broken into smaller overlapping chunks using:

- Tool: `LangChain`'s `RecursiveCharacterTextSplitter`

Embedding Model

- Model: `sentence-transformers/all-MiniLM-L6-v2`
- Reason: Lightweight, fast, and well-suited for semantic search on short to medium-length texts.

Vector Database

- Library: `FAISS` (Facebook AI Similarity Search)
- Metadata stored: `complaint ID`, `product`, `original text chunk`
- Purpose: Enables traceability of retrieved vectors

Artifacts

- Persisted vector store: `vector_store/`
- Chunking and embedding logic: in `src/embedding.py`

Conclusion: We now have an efficient vector search setup that supports fast and relevant retrieval for downstream question-answering.

---

4. Next Steps

In the final phase of the project, we will:

1. **Build the RAG Pipeline**
   Implement retrieval + generation + prompt engineering for intelligent answers.
2. **Evaluate Results**
   Use qualitative analysis with test questions and scoring.
3. **Launch an Interactive App**
   Use Streamlit or Gradio to make the tool accessible to non-technical users.

Appendices

- Cleaned dataset: `data/filtered_complaints.csv`
- Code for embedding/indexing: `src/embedding.py`
- GitHub Actions pipeline: `.github/workflows/cicd.yml`