Interim Report: CFPB Consumer Complaint Vector Store for RAG

Author: Yodahe Tsegaye

Date: January 3, 2026

GitHub Repository: https://github.com/Yodahe2021/cfpb-rag-vectorstore

1. Executive Summary & Introduction

This interim report summarizes the development of a text processing and semantic retrieval pipeline for CFPB consumer complaints. The objective is to prepare a Retrieval-Augmented Generation (RAG) system by converting complaint narratives into a vectorized format suitable for semantic search. Key accomplishments include: Data Preparation & Sampling: Filtered and cleaned 10k–15k complaints stratified across major product categories. Text Chunking: Split long narratives into overlapping word-based segments to improve embedding quality. Embedding Generation: Created dense vector embeddings using the 'all-MiniLM-L6-v2' model. FAISS Vector Store: Built and persisted a FAISS vector index for efficient semantic similarity search. Metadata Alignment: Stored chunk-level metadata including Complaint ID and Product for retrieval context.

2. Business Understanding & Problem Statement

Consumer complaints provide insights into customer issues, fraud risk, and product weaknesses. A RAG system enables organizations to quickly retrieve relevant complaints, analyze trends, and support automated report generation and decision-making.

Key Challenges: Variable Text Lengths: Complaint narratives vary from a few words to several thousand words. Preserving Context: Splitting text without losing semantic meaning requires overlapping chunks. Efficient Search: Millions of chunks need to be embedded and searchable with low latency. Stratified Sampling: Ensure balanced representation of complaints across financial products.

3. Data Preparation & Exploration

Dataset Overview: - Source: CFPB consumer complaint dataset. - Filtered columns: Complaint ID, Product, clean_narrative. - Data size: Stratified sample of 10k–15k complaints. - Missing values: Removed any complaints without narratives.

Text Chunking: - Rationale: Embedding models perform optimally on shorter segments (~300 words). - Method: Overlapping chunks of 300 words with 50-word overlap. - Outcome: Each complaint split into multiple chunks while preserving context.

Example Chunk Metadata:

Complaint ID	Product	Chunk Snippet
12345	Credit card	I was charged multiple fees I did not authorize...
12345	Credit card	...contacted the bank but received no response for weeks...

4. Embedding & Vector Store Construction

- Model: 'all-MiniLM-L6-v2' (Sentence Transformers) - Each text chunk converted into a 384-dimensional dense vector. - FAISS Index: IndexFlatL2, stored at 'data/vector_store/complaints_faiss.index'. - Metadata: Stored at 'data/vector_store/complaints_metadata.csv'.

5. Sanity Test & Preliminary Retrieval

A preliminary semantic search was conducted: Query: 'credit card charged fees I did not authorize' Top Retrieved Chunks: 1. I was charged unexpected fees on my credit card and the bank did not respond... 2. After contacting customer support about fraudulent charges, I received no help... 3. The bank continued billing me despite multiple complaints about unauthorized fees...

6. Next Steps

1. Full Dataset Integration: Extend from 15k sampled complaints to the full dataset. 2. Vector Store Optimization: Explore GPU-based FAISS indices for large-scale retrieval. 3. Integration with LLMs: Connect FAISS store to RAG pipelines. 4. Evaluation Metrics: Recall@k and relevance scoring. 5. Automation: Convert pipeline into a reusable Python module.

7. Conclusion

The pipeline demonstrates effective text preprocessing, chunking, high-quality embeddings, a fully functional FAISS vector store with metadata, and promising preliminary retrieval results. The next phase will focus on full-scale deployment and integration with RAG models.