Interim Submission Report: Intelligent Complaint Analysis for Financial Services

Introduction

This interim report details the progress made on the "Intelligent Complaint Analysis for Financial Services" project, focusing on the initial stages of Exploratory Data Analysis (EDA) and Data Preprocessing (Task 1), and the foundational steps for Text Chunking, Embedding, and Vector Store Indexing (Task 2). The project aims to develop a Retrieval-Augmented Generation (RAG) powered chatbot for CrediTrust Financial to transform unstructured customer complaint data into actionable insights, thereby improving efficiency for product managers, support, and compliance teams.

1. Task 1: Exploratory Data Analysis and Data Preprocessing

The initial phase involved understanding the raw Consumer Financial Protection Bureau (CFPB) complaint dataset and preparing it for the RAG pipeline.

1.1 Key Findings from EDA

The raw dataset is extensive, comprising **9,609,797 rows and 18 columns**. Key observations from the EDA include:

- **Missing Data**: A significant challenge was the large number of missing values, especially in the Consumer complaint narrative column, with **6,629,041 (69%)** entries being empty. This highlighted the necessity of removing these records as they are central to the RAG system's functionality. Other columns like Tags and Consumer disputed? also exhibited extremely high rates of missing values (over 90%), rendering them largely unusable for direct analysis or retrieval.
- **Duplicate Narratives**: Approximately **2,108,758 duplicate complaint narratives** were identified. While some may be genuine repeated issues, this suggests the need for careful text cleaning to prevent redundant embeddings and ensure unique insights.
- **Product Categories**: The Product column contained 21 unique categories. A crucial finding was the **absence of "Buy Now, Pay Later (BNPL)" as a direct product category** within the dataset's Product column, requiring careful consideration for filtering this specific product line.

1.2 Data Filtering and Cleaning

To align with project requirements and manage data volume, the following preprocessing steps were executed:

- **Product Filtering**: The dataset was filtered to include only complaints related to CrediTrust's five key product categories: Credit Card, Personal Loan, Buy Now, Pay Later

(BNPL) (addressed through broader categories like 'Payday loan, title loan, personal loan, or advance loan' due to direct absence), Savings Account, and Money Transfer. This reduced the dataset from its original size to **1,045,146 rows and 18 columns**.

- **Narrative Removal**: Rows with empty Consumer complaint narrative fields were subsequently removed. This critical step further refined the dataset to **480,580 rows and 19 columns** (including the narrative_length column added during EDA), ensuring all remaining entries provide valuable textual feedback for the RAG system.
- **Basic Text Cleaning**: The Consumer complaint narrative text was converted to lowercase. An example of a cleaned narrative: 'a xxxx xxxx card was opened under my name by a fraudster. i received a notice from xxxx that an account was just opened under my name...' This initial cleaning helps standardize the text for embedding.
- **Dataset Persistence**: The cleaned and filtered dataset has been successfully saved to data/processed/filtered_complaints.csv.

1.3 Distribution Analysis and Narrative Length

As part of EDA, the distribution of complaints across the filtered product categories was analyzed and visualized using a bar chart. This provides a clear overview of which product areas generate the most complaints for CrediTrust.

Furthermore, the **length (word count)** of the Consumer complaint narrative was calculated and visualized using a histogram. This analysis revealed the spread of narrative lengths, identifying the presence of both very short and very long complaints. This insight is directly used to inform the text chunking strategy in Task 2.

2. Task 2: Text Chunking, Embedding, and Vector Store Indexing (Partial Progress)

This task focuses on preparing the cleaned text narratives for efficient semantic search within the RAG system.

2.1 Text Chunking Strategy Implementation

The preprocessed Consumer complaint narrative texts were chunked using LangChain's RecursiveCharacterTextSplitter. This method intelligently breaks down longer narratives into smaller, overlapping segments suitable for embedding.

- **Parameters**: A chunk_size of **1000 characters** and a chunk_overlap of **200 characters** were selected. This choice aims to balance retaining sufficient context within each chunk with ensuring chunks are not excessively large for effective embedding, informed by the narrative length distribution observed in Task 1.
- **Results**: From the 480,580 processed narratives, a total of **888,532 chunks** were created. Each chunk retains critical metadata from the original complaint, including complaint_id,

product, issue, sub_product, and company, enabling rich contextual retrieval.

2.2 Embedding Model Selection and Initialization

For generating vector embeddings, the sentence-transformers/all-MiniLM-L6-v2 model was chosen. This model is recognized for its efficient balance of performance and computational resource usage, making it a suitable choice for this large-scale semantic search application. The model has been successfully downloaded, cached locally, and initialized using HuggingFaceEmbeddings, ready for the embedding process. A sample embedding test confirmed its 384-dimension output.

2.3 Embedding and Indexing into Vector Store

This step, involving the generation of embeddings for all 888,532 chunks and their indexing into a ChromaDB vector store, has been initiated. This is a computationally intensive process given the volume of chunks.

- **Vector Store Choice**: **ChromaDB** was selected as the vector store due to its features for persistence and ease of handling metadata alongside embeddings.
- **Current Status**: The process of creating and persisting the vector store is currently underway. Due to the significant number of chunks and the computational demands, this step is expected to take a considerable amount of time (estimated 1-3.5 hours on Colab GPU, or several hours on a CPU-only machine). Upon completion, the vector store will be saved to the vector_store/ directory.

Next Steps

Upon successful completion and persistence of the vector store (Task 2, Part 3), the project will proceed to **Task 3: Building the RAG Core Logic and Evaluation**. This will involve implementing the retrieval and generation pipeline, designing prompt templates, and performing qualitative evaluation of the chatbot's performance.