System Ticket Data Summary

This project is a Streamlit-based application designed to process customer ticket data, generate summaries using LLMs (Large Language Models), and display the results in an interactive timeline format. The application supports caching LLM responses in a SQLite database to optimize performance and reduce redundant API calls.

Features

- **Data Preprocessing**: Processes uploaded ticket data files (CSV or TXT) to remove the unused categories.
- **Category Mapping**: Maps each category to its product, to work based on products, then groups them by customer and product.
- **Summary Generation**: Uses LLMs [Ollama (locally) or OpenRouter] to generate structured summaries for each customer-product group.
- **Interactive Timeline**: Displays summaries in a timeline format for easy visualization.
- **Caching**: Stores LLM responses in a SQLite database to avoid redundant API calls.
- **Retry Mechanism**: Retries LLM API calls up to 3 times in case of generation or parsing errors.

Implementation Steps

1. Data Preprocessing

- The uploaded file is processed as mentioned earlier, using the `process_data` function.
- Data is grouped by `Customer Number` and `Product` to create subsets for analysis.
- A content hash is generated for each group to uniquely identify its data.

2. Category Mapping

- Work with the general products rather than their categories.

3. Summary Generation

- Summaries are generated using LLMs from (Ollama or OpenRouter).
- A structured prompt is created for each group to ensure consistent output.
- The application retries API calls up to 3 times in case of errors.

4. Caching System

- Responses are cached in a SQLite database to avoid redundant API calls.

5. Results Filtering

- Filters are provided in the sidebar to allow users to filter data by:
  - Customer Number
  - Product
- Only the selected groups are shown.

6. Interactive Timeline

- Summaries are displayed in a timeline format using the `streamlit-timeline` library.
- Each phase of the summary (e.g., Initial Issue, Follow-ups) is shown as an event on the timeline.

Code Documentation

1. Main Application (`main.py`)

- Handles the Streamlit UI and user interactions.
- Processes uploaded files and applies filters.
- Retrieves cached summaries or generates new ones using LLMs.
- Displays summaries in an interactive timeline.

2. Summary Generator (`summary_generator.py`)

- Contains logic for generating summaries using LLMs.
- Includes retry mechanisms for API calls.
- Formats summaries for display.

3. Timeline Helper (`timeline_helper.py`)

- Converts LLM responses into a timeline-compatible format.
- Handles date extraction and formatting.

4. Database Helper (`db_helper.py`)

- Manages SQLite database operations.
- Stores and retrieves cached LLM responses.

User Guide

1. Setup

- Install the required dependencies and run the Streamlit app:

  ```
  pip install -r requirements.txt
  streamlit run main.py
  ```

  - Make sure to have openrouter or running Ollama server.
- Alternatively, use Docker to build and run the application in a container:

  ```
  docker compose up
  ```

- Access the application in your browser at `http://localhost:8501`.

2. Using the Application

- **Upload File**: Upload a CSV or TXT file containing ticket data.
- **Select LLM Provider**: Choose between Ollama and OpenRouter in the sidebar.
- **Apply Filters**: Use the filters in the sidebar to select specific customers or products.
- **View Summaries**: Summaries are displayed in an interactive timeline format.

3. Caching

- The application automatically caches LLM responses in a SQLite database.
- Cached responses are retrieved when the same data group is processed again.

4. Error Handling

- The application retries API calls up to 3 times in case of errors.
- Errors are displayed in the UI if all retries fail.

Dependencies

- `streamlit`
- `pandas`
- `plotly`
- `openai`
- `python-dotenv`
- `requests`
- `streamlit-timeline`
- `db-sqlite3`
