📑 Interim Documentation: Contributions and Project Implementation

Project Overview

This project aims to build a **Data Warehouse** that stores data from Ethiopian medical businesses scraped from various **Telegram channels**. The data is processed and transformed for analysis, which helps identify insights related to the Ethiopian medical sector. Additionally, the project integrates **object detection** using the **YOLO model** and exposes the data via a **FastAPI** application for seamless access.

Contributions and Tasks

1. **Data Scraping from Telegram (Task 1)**
    - **Objective**: Extract and collect raw data from specified Telegram channels related to Ethiopian medical businesses.
    - **Technologies**: **Telethon (Python)**, **Telegram API**.
    - **Implementation**:
        - Developed a Python script (`telegram_scraper.py`) that uses the **Telethon** library to scrape public Telegram channels. The following Telegram channels were targeted for scraping:
            - `https://t.me/DoctorsET`
            - `Chemed Telegram Channel`
            - `https://t.me/lobelia4cosmetics`
            - `https://t.me/yetenaweg`
            - `https://t.me/EAHCI`
            - Additional channels from `https://et.tgstat.com/medicine`.
        - Extracted key details from messages, including **text content, dates, sender info**, etc.
        - Stored the raw data as JSON objects in `data/raw_data.json`.
    - **Challenges**:
        - Handling large amounts of data: Managed Telegram message limits and ensured robust logging to track scraping progress and potential issues.

2. **Data Cleaning & Transformation (Task 2)**

    - **Objective**: Clean the raw scraped data, standardize it, and transform it into a usable format.

    - **Technologies**: **Pandas**, **DBT (Data Build Tool)**, **SQLite**.

    - **Implementation**:

        - Cleaned the data using the **Pandas** library. The following tasks were carried out:
            - Removed duplicate messages based on unique message IDs.
            - Removed any empty or missing values from the data.
            - Standardized the date format to **YYYY-MM-DD HH:MM:SS**.
            - Saved the cleaned data as `data/cleaned_data.csv`.
        - Implemented an **SQLite database** to store the cleaned data (`data/medical_business.db`), making it easily accessible for future queries.
        - Integrated **DBT** to handle SQL transformations. Created SQL models for transforming the data and generating the final database schema.
        - Successfully ran the **DBT models** to apply data transformations, improving its usability for analysis.
    - **Challenges**:

        - Inconsistent data format: Addressed missing and malformed data through rigorous cleaning steps.
        - Efficient storage: Chose **SQLite** as the database for its simplicity and effective storage.
3. **Object Detection Using YOLO (Task 3)**

    - **Objective**: Enhance data analysis by performing object detection on images collected from Telegram channels.

    - **Technologies**: **YOLO (You Only Look Once)**, **OpenCV**, **TensorFlow/PyTorch**.

    - **Implementation**:

        - Collected relevant images from the scraped Telegram channels (e.g., product images, medical supplies).
        - Integrated the **YOLO object detection** model to analyze and detect objects in the images, extracting bounding boxes, confidence scores, and class labels.
        - Stored the detection results in a structured database for easy querying.

   - **Challenges**:
      - Image data processing: Handling variations in image quality and format, and ensuring YOLO works optimally across different image types.
      - Storage of detection results: Properly storing object detection results alongside other medical business data for integrated analysis.
4. **Exposing Data with FastAPI (Task 4)**
   - **Objective**: Build a FastAPI service to expose the collected and processed data through API endpoints.
   - **Technologies**: **FastAPI**, **SQLAlchemy** (for ORM), **Pydantic** (for data validation).
   - **Implementation**:
      - Developed a FastAPI application to expose the cleaned and processed data as RESTful API endpoints. The project structure includes:
         - **main.py**: FastAPI app definition with routes.
         - **database.py**: Configured the database connection using **SQLAlchemy**.
         - **models.py**: Defined SQLAlchemy models for storing medical business data.
         - **schemas.py**: Defined **Pydantic** models for data validation and serialization.
         - **crud.py**: Implemented CRUD operations for interacting with the database.
      - Exposed data endpoints for querying, inserting, and updating medical business data.
   - **Challenges**:
      - Data access control: Ensured the API had robust error handling and validation mechanisms.
      - Database connection: Managed efficient database connections and ensured data integrity.

Key Achievements

1. **End-to-End Data Pipeline**: Successfully implemented a pipeline for scraping, cleaning, transforming, and exposing data for analysis.
2. **Integration of Object Detection**: Integrated **YOLO** for image analysis, contributing valuable insights into product and medical supply recognition.

3. **Data Exposure via FastAPI**: Created an API for external systems to query medical business data in real-time.
4. **Database Design**: Ensured clean, standardized data storage using **SQLite** and made it accessible via FastAPI.
5. **DBT Transformations**: Automated and simplified the data transformation process using **DBT**, improving scalability.

Challenges Faced & Solutions

1. **Data Quality Issues**: Raw data often contained errors like missing values, inconsistent formatting, and duplicate entries.
   Solution: Utilized **Pandas** for data cleaning, ensuring consistency and accuracy before loading the data into the database.

2. **Telegram API Limitations**: Scraping Telegram channels sometimes hit rate limits, requiring a well-managed and logged process.
   Solution: Implemented logging in the scraper to track progress and identify issues early.

3. **Database Storage**: Efficiently storing and querying large amounts of data while maintaining quick access.
   Solution: Chose **SQLite** for simplicity and integrated it with **FastAPI** for real-time querying.

4. **YOLO Integration**: Ensuring YOLO worked effectively on varying image qualities.
   Solution: Pre-processed images for standardization and optimized YOLO parameters for better accuracy.

Next Steps

- **Task 3**: Improve YOLO model accuracy and include more data for object detection.
- **Task 4**: Add authentication and rate limiting to the FastAPI service for better security and access control.
- **Final Integration**: Complete the data warehouse, and integrate all parts to ensure seamless querying and reporting.

Conclusion

This interim documentation outlines the significant steps taken in building the data warehouse for Ethiopian medical businesses. The contributions span **data collection**, **cleaning**, **object detection**, and **exposing data via FastAPI**, ensuring that the system is robust, scalable, and ready for analysis. Further enhancements will be made in the coming tasks to increase data processing power and expand functionality.
