Week 4 - Interim Report

Building an Amharic E-commerce Data Extractor

Introduction

This week's project focuses on establishing the foundational data pipeline for the Amharic E-commerce Data Extractor. The primary objective is to develop robust data ingestion and preprocessing mechanisms for Telegram channel data, critical steps towards fine-tuning Large Language Models (LLMs) for Named Entity Recognition (NER) in Amharic. Significant efforts were directed towards setting up a structured project environment, implementing a Telegram scraping utility, conducting initial exploratory data analysis (EDA), and designing a comprehensive text preprocessing component.

Methodology

1. GitHub Repository and Project Setup

The project repository, `EthioMart/`, has been meticulously structured to ensure modularity, reproducibility, and maintainability, aligning with best practices for machine learning projects.

- **Core Structure:** The directory layout includes `data/` (for raw and processed data), `src/` (for modular Python scripts), `notebooks/` (for exploratory analysis), `config/` (for centralized configurations), `tests/` (for unit tests), `models/` (for trained models), `outputs/` (for metrics and visualizations), and `photos/` (for scraped images).
- **Dependencies:** A `requirements.txt` file was created at the project root to manage all Python library dependencies, facilitating consistent environment setup.
- **Configuration Management:** A `config/config.py` module was introduced to centralize project-wide settings, including Telegram API credentials (loaded securely from a `.env` file) and defined target Telegram channel usernames.

2. Data Ingestion (Telegram Scraping)

The initial phase involved collecting raw Telegram messages, which serve as the primary dataset for the NER task.

- **Script (`src/telegram_scraper.py`):** A custom Python script was developed to connect to the Telegram API using `telethon`. This script is responsible for iterating through specified Telegram channels and extracting relevant message attributes.

- **Target Channels:** The scraper was configured to target specific e-commerce channels: `@ZemenExpress`, `@ethio_brand_collection`, `@Leyueqa`, `@Fashiontera`, and `@marakibrand`.
- **Data Fields Captured:** For each message, the scraper extracts and stores `channel_title`, `message_id`, `date`, `text`, `views`, `reactions_count`, and `image_path` (for attached media).
- **Image Storage:** Photos detected within messages are downloaded and saved into the `EthioMart/photos/` directory, with their relative paths recorded in the output CSV.
- **Output:** The collected raw data is saved to `EthioMart/data/raw/telegram_data.csv`.

3. Exploratory Data Analysis (EDA)

To gain a preliminary understanding of the ingested data's characteristics, an EDA notebook was developed.

- **Notebook (`notebooks/data_ingestion_eda.ipynb`):** This Jupyter notebook facilitates loading and inspecting the `telegram_data.csv`.
- **Key Analyses:** It performs checks for missing values (especially in the `text` column), visualizes the distribution of messages across different channels, analyzes message lengths (character count), and provides insights into engagement metrics such as `views` and `reactions_count`. This helps in identifying data quality issues and understanding the nature of the raw content.

4. Data Preprocessing/Cleaning

A dedicated module was developed to cleanse the raw Telegram text, transforming it into a more structured and standardized format suitable for downstream NLP tasks.

- **Script (`src/preprocessor.py`):** This script takes the raw `telegram_data.csv` as input and applies a series of sophisticated cleaning and normalization routines to the `text` column, saving the output to `EthioMart/data/processed/clean_telegram_data.csv`.
- **Cleaning Steps Implemented:**
    - **Amharic Character Normalization:** Basic standardization of commonly interchanged Amharic characters.
    - **Emoji and Symbol Removal:** Comprehensive removal of diverse emojis and decorative symbols prevalent in Telegram communications.
    - **Telegram-Specific Patterns:** Stripping out visual clutter like multiple dots, asterisks, tildes, and other repeated characters.
    - **URL, Mention, and Hashtag Removal:** Eliminating web links, `@username` mentions, and `#hashtag` elements to focus on core text content for NER.

    - **Currency Standardization:** Converting various currency expressions (e.g., "ብር", "Br") into a consistent "ETB" format and removing currency symbols.
    - **Phone Number Cleaning:** Standardizing phone number formats by replacing them with a `<PHONE_NUMBER>` placeholder to facilitate later entity recognition.
    - **Whitespace and Non-Amharic/Non-English Character Cleanup:** Normalizing spaces and removing any remaining extraneous characters outside of the Amharic script, English alphabet, digits, and basic punctuation.

Challenges & Solutions

- **Challenge: Async Iterator Mocking in Unit Tests:**
    - During the development of `tests/test_telegram_scraper.py`, correctly mocking `telethon.TelegramClient.iter_messages` (an asynchronous generator) proved challenging, leading to `async for` errors.
    - **Solution Attempted:** Multiple iterations of mocking techniques were explored, specifically focusing on how `AsyncMock`'s `side_effect` interacts with asynchronous generators to ensure the `async for` loop could correctly consume the yielded mock messages. While the test is still undergoing final debugging, the understanding of async mocking has significantly deepened.
- **Challenge: `preprocessor.py` Outputting Only Headers:**
    - Upon initial execution, `src/preprocessor.py` was observed to write only the CSV header to `clean_telegram_data.csv`, indicating an issue with data processing or loading.
    - **Solution in Progress:** Debugging statements were strategically added to `src/preprocessor.py` to trace the DataFrame's state (loaded row count, head of columns, etc.) at various stages, allowing for identification of the exact point where data loss occurs. This debugging is currently underway.

Future Plan

Building upon the established data ingestion and preprocessing foundation, the next steps will focus on preparing the data for Named Entity Recognition model training and subsequent evaluation.

1. **Resolve Preprocessor Issue:** Finalize the debugging of `src/preprocessor.py` to ensure it correctly processes and outputs the `clean_telegram_data.csv` with all expected rows.
2. **Data Labeling (`src/data_labeler.py`):** Develop a script to take the `clean_telegram_data.csv` and the provided `labeled_telegram_product_price_location.txt` to generate training data in a CoNLL-like format, suitable for NER models. This will involve using rule-based patterns and aligning them with SpaCy's `DocBin` format.

3. **NER Data Preparation (`notebooks/ner_model_training.ipynb`):** Utilize a notebook to load the labeled data, perform quality checks, and split it into appropriate training, validation, and testing datasets in SpaCy's binary (`.spacy`) format.
4. **Model Fine-Tuning (`src/model_finetuning/`):** Implement scripts to fine-tune a pre-trained Amharic or multilingual LLM (e.g., XLM-RoBERTa) using the prepared `.spacy` datasets. This will involve leveraging Hugging Face's `transformers` library and potentially SpaCy's training pipeline.
5. **Model Comparison and Interpretability:** After training, evaluate the model's performance using appropriate metrics (Precision, Recall, F1-score) and explore interpretability techniques like SHAP or LIME.
6. **Vendor Analysis and Lending Score Calculation:** (Future task) Develop scripts to derive insights about vendors and calculate a "Lending Score" based on the extracted entities and other metrics.

Conclusion

This week's efforts have laid a robust groundwork for the Amharic E-commerce Data Extractor project. Despite minor debugging challenges, significant progress has been made in establishing a well-structured repository, implementing reliable data ingestion, and developing a comprehensive preprocessing pipeline. This foundation is crucial for the upcoming Named Entity Recognition tasks and positions the project well for successful completion.