Web Scraper Design Documentation

Introduction

This document provides a detailed overview of a web scraper designed to collect business-related data from the platform *AmbitionBox*. The scraper extracts information such as company name, ratings, and domain location, processes the data, and saves it in a clean and usable format. This tool aligns with the task of developing a scalable and efficient scraper for open-source platforms.

Features

1. Data Collection

- **Platform**: AmbitionBox (https://www.ambitionbox.com/list-of-companies)
- **Scraped Fields**:
    - Company Name
    - Ratings
    - Domain Location

2. Data Processing and Cleaning

- Handles missing data by excluding incomplete records.
- Removes duplicate entries to ensure unique company information.
- Processes the data into a tabular format for easy integration with AI/ML models.

3. Scalability and Efficiency

- Implements retry mechanisms for failed web requests.
- Uses robust data extraction techniques with BeautifulSoup to handle large datasets.
- Respects platform policies by adhering to ethical scraping practices.

Workflow and Architecture

Step 1: Initialization

The scraper is initialized with the following configurations:

- **Base URL**: The platform's URL is to be scraped.
- **Headers**: Custom user-agent to simulate browser behavior.
- **Retries**: Exponential backoff for robust request handling.

Step 2: Data Collection

1. Make HTTP requests to the target URL.
2. Parse the HTML content using BeautifulSoup.
3. Extract relevant fields from specific HTML elements.

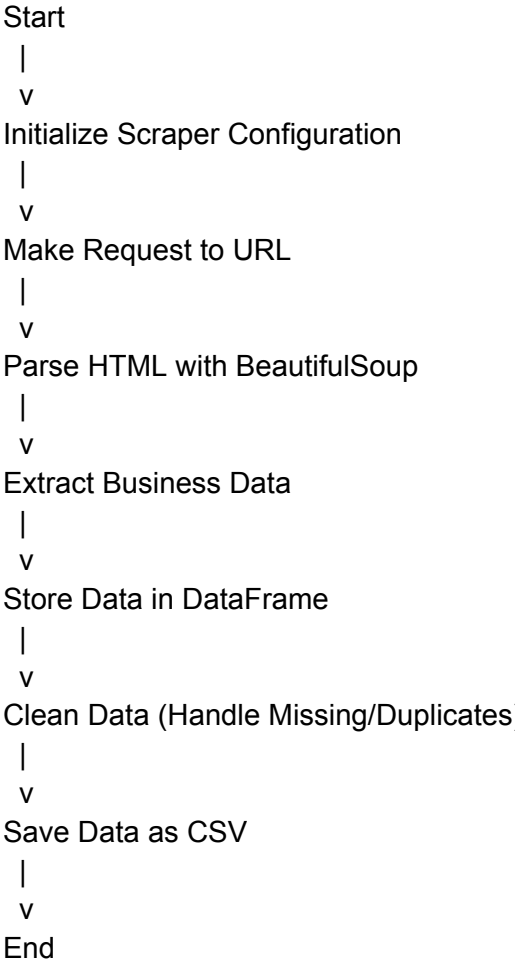
Step 3: Data Processing

1. Convert raw data into a structured pandas DataFrame.
2. Clean the data by handling missing values and duplicates.
3. Save the processed data as a CSV file.

Step 4: Data Storage

The cleaned data is stored locally in a CSV format, ensuring compatibility with downstream AI/ML processes.

Flowchart



Code Overview

Key Functions

1. `make_request(url: str)`
    - Sends a GET request with headers and retries.
2. `extract_business_data(soup: BeautifulSoup)`
    - Extracts structured data from HTML content.
3. `scrape_data()`
    - Combines request and extraction functions to return cleaned data.
4. `save_data(df: pd.DataFrame, filename: str)`
    - Saves the processed DataFrame as a CSV file.

Key Technologies

- **Python Libraries**: Requests, BeautifulSoup (bs4), pandas, logging.
- **File Format**: CSV for data storage.

Ethical Considerations

- The scraper does not violate *AmbitionBox*'s terms of service.
- Data collected is publicly available and is anonymized when stored.

Conclusion

This scraper effectively collects and processes business data for further analysis. Its scalable and modular design ensures adaptability for other platforms and future enhancements.