J Search Project: User Manual

Table of Contents

Project Overview

The J Search Project is a search engine specifically designed for desi women's clothing. It scrapes data from the J. website, processes both textual and visual data to create a searchable database, and provides a user-friendly interface to perform natural language searches.

Project Structure

The project is organized into two main modules, each with a specific purpose:

- **J_Scrapping:** Handles the scraping and processing of data.
- **J_Search:** Handles inference and provides the UI for user interaction.

Directory Structure

```
J_search_project/
│
├── J_Scrapping/
│   ├── requirements.txt
│   ├── config.json
│   ├── data_extracting.py
│   ├── Images/
│   ├── image_processing.py
│   ├── data_post_processing.py
│   ├── junaid_jamshed.csv
│   ├── database.py
│   └── main.py
│
└── J_Search/
    ├── .venv/
    ├── app.py
    ├── main.py
    └── requirements.txt
```

Setup Instructions

1. **Extract the Zipped Folder:**
   - Unzip the `J_search_project.zip` file to your desired location on your local machine.
2. **Install Required Libraries:**
   - Open a terminal and navigate to both `J_Scrapping` and `J_Search` directories to install the required libraries using the `requirements.txt` file:

     ```
     pip install -r requirements.txt
     ```

3. **Set Up Virtual Environment (Optional but recommended):**
   - Navigate to `J_Search` and set up a virtual environment to isolate project dependencies:

     ```
     python -m venv .venv
     source .venv/bin/activate
     .venv\Scripts\activate # For Windows use
     ```

4. **Database Configuration:**
    - Ensure MySQL is installed and running.
    - Configure the database settings in the `config.json` file located in the `J_Scrapping` directory.

Modules

1. J_Scrapping

This module is responsible for scraping data from the J. website, processing that data, and storing it in a MySQL database.

- **Files and Functionality:**
    - `requirements.txt`: Lists all the necessary libraries for scraping and processing. Use this to set up your environment.
    - `config.json`: Contains configuration settings like paths, URLs, database credentials, and hostnames. Make sure to update this file with your local setup details.
    - `data_extracting.py`: Scrapes women's clothing data from the J. website and saves it to a CSV file. Images are stored in the `Images` folder.
    - `Images/`: A folder that stores all the images scraped from the J. website.
    - `image_processing.py`: Utilizes the `dandelin/vilt-b32-finetuned-vqa` model to generate descriptions of the images.
    - `data_post_processing.py`: Processes the generated image descriptions, combines them with the textual data from the CSV, and shapes a final CSV file (`junaid_jamshed.csv`).
    - `junaid_jamshed.csv`: The final CSV file containing combined data from both textual and visual sources.
    - `database.py`: Handles the connection to the MySQL database, uploads the processed data, and provides querying functionality.
    - `main.py`: Integrates all the above scripts to perform end-to-end data scraping, processing, and database uploading.

2. J_Search

This module is used for searching the processed data and providing a user interface.

- **Files and Functionality:**
    - `.venv/`: The virtual environment containing the dependencies for the `J_Search` module.
    - `app.py`: A Streamlit-based user interface to facilitate natural language search functionality. Users can enter search queries, and the results are displayed based on relevance.
    - `main.py`: Handles backend operations, including fetching data from the MySQL database based on user queries.
    - `requirements.txt`: Lists all the necessary libraries for running the inference and UI.

Data Flow and Processing

1. **Scraping Data:**
   - Run `data_extracting.py` to scrape data from the J. website. The data is saved in a CSV file, and the images are stored in the `Images` folder.
2. **Generating Image Descriptions:**
   - Use `image_processing.py` to process images using the `dandelin/vilt-b32-finetuned-vqa` model. This script generates textual descriptions of the images.
3. **Combining Data:**
   - Run `data_post_processing.py` to combine the textual data from the CSV file with the image descriptions. The output is a refined CSV file named `junaid_jamshed.csv`.
4. **Uploading to Database:**
   - Execute `database.py` to connect to the MySQL database, create necessary tables, and upload the combined data for querying.
5. **Integration:**
   - Use `main.py` in the `J_Scrapping` module to automate the entire process from scraping to database upload.

Running the Project

1. **Data Processing:**
   - Navigate to the `J_Scrapping` directory:

     ```
     cd J_Scrapping
     ```

   - Run the main script to perform all tasks:

     ```
     python main.py
     ```

2. **Start the Search UI:**
   - Navigate to the `J_Search` directory:

     ```
     cd ../J_Search
     ```

   - Run the main script to start hosting the API on server:

     ```
     python main.py
     ```

   - Run the Streamlit app:

     ```
     streamlit run app.py
     ```

3. **Using the Search Interface:**
   - Open a web browser and navigate to the local URL provided by Streamlit.
   - Enter search queries in natural language to find specific clothing items from the J. collection.

Troubleshooting

- **Database Connection Issues:** Ensure MySQL is running, and the credentials in `config.json` are correct.
- **Missing Dependencies:** Run `pip install -r requirements.txt` in both `J_Scrapping` and `J_Search` directories.
- **Image Processing Errors:** Check that the `Images` folder exists and contains the scraped images. Ensure the `image_processing.py` script has access to these files.
- **Streamlit App Not Launching:** Verify that the virtual environment is activated and all dependencies are installed. Run `streamlit run app.py` from the correct directory.