**Python Interview Code Report: Product Recommendation System with OCR Integration**

* Author: Mohammed Audu
* Date: 05th May 2024

**Introduction**

This report details the Python code developed for a product recommendation system with integrated Optical Character Recognition (OCR) functionality. The code demonstrates skills in data cleaning, vectorization, leveraging a vector database (Pinecone), and interacting with external tools like Tesseract for OCR.

* **Objective:** Develop the capability to extract text from images using OCR technology.
* **Key Actions:** Integrate and configure an OCR tool (e.g., Tesseract).

**web\_scraping.py**

This script focuses on image scraping for potential training data:

* Defines functions for:
  + Core OCR processing using pytesseract and Pillow's Image class.
  + Downloading images from a provided base URL across multiple pages, storing them in a designated directory.
* Employs requests library to download webpage content.
* Utilizes BeautifulSoup for parsing the HTML structure and extracting image URLs.
* Downloads images that have absolute URLs (http or https).
* Creates a directory structure if it doesn't exist to store the downloaded images.

**Key Decisions:**

* **Tesseract Integration:** Tesseract was chosen as a widely used and open-source OCR tool. Depending on project requirements, alternative libraries like EasyOCR or cloud-based OCR services could be explored.

**Challenges and Solutions:**

* Potential challenges include:
  + **OCR Accuracy:** Tesseract's accuracy can vary depending on image quality and text complexity. Strategies like image pre-processing or exploring more advanced OCR libraries could be considered for improvement.
  + **Web Scraping Ethics:** Respecting robots.txt guidelines and avoiding overwhelming website servers are crucial considerations during web scraping.

**Module 3: OCR-Based Query Processing**

**Task 5: OCR-Based Query Processing**

* **Objective:** Process handwritten queries using the same logic as the text-based recommendation service (Endpoint 1).
* **Key Actions:** Extract text from image files and integrate it with the existing recommendation logic.

**ocr.py and ocr.html**

These files implement an OCR web application:

* **ocr.py** defines a Flask app for processing image uploads and text extraction:
  + Provides a route for the homepage (/).
  + Handles file uploads through a POST request to the /upload route.
  + Reads the uploaded image file using Flask's request object.
  + Converts the image to an OpenCV image format for compatibility with Tesseract.
  + Extracts text from the image using pytesseract.
  + Returns the extracted text as the response.
* **ocr.html** provides a simple web interface for users to upload handwritten query images:
  + Includes an HTML form with a file upload field for selecting an image.
  + Submits the form to the /upload route for processing.

**Conclusion**

This code demonstrates the ability to build a product recommendation system that can handle both text-based and handwritten queries through OCR integration. The use of external tools like Tesseract and web scraping techniques (if applicable) showcases adaptability and an understanding of practical considerations.

**Future Enhancements:**

* Explore more advanced OCR libraries or techniques to improve accuracy for handwritten queries.
* Implement pre-processing steps for image noise reduction or skew correction before OCR processing.