**Project Documentation: Feature Extraction from Images**

**1. Introduction**

The goal of this project is to create a machine learning model that extracts entity values from product images. These extracted values are essential for various industries, such as healthcare, e-commerce, and content moderation. In digital marketplaces, product information like weight, volume, dimensions, and other attributes can often be missing, making it necessary to extract this data directly from images.

This project involves processing a dataset with product images and using Optical Character Recognition (OCR) to extract text from the images. The extracted text is further processed to identify and return the desired entity values such as dimensions, weight, or other attributes.

**2. Data Description**

The dataset contains the following columns:

* **index**: A unique identifier for each sample.
* **image\_link**: A URL pointing to the image of the product, which contains information like weight, size, or other entity values.
* **group\_id**: Category code of the product, representing the product's category (for example, electronics, home goods, etc.).
* **entity\_name**: The name of the product entity that needs to be extracted, e.g., "item\_weight," "dimensions," etc.
* **entity\_value**: The value of the product entity (target variable in training data) that needs to be predicted in the test dataset.

**3. Methodology**

**3.1 Data Loading and Preprocessing**

The dataset is split into two files: train.csv and test.csv. While the training dataset contains labeled entity\_value information, the test dataset lacks this column, and the task is to predict it based on the image content.

1. **Download Images**: Images are downloaded from the provided image\_link using a helper function from utils.py.
2. **Text Extraction from Images**: The core part of the project involves using Tesseract OCR to extract textual information from the images. This text will be further processed to extract the specific entity value required (e.g., weight or dimensions).
3. **Data Cleaning**: The extracted text often contains noise. Therefore, it's necessary to clean this data by removing irrelevant characters and ensuring it follows a specific format. For example, removing special characters and unwanted spaces in entity values is crucial to match the expected output format.

**3.2 Feature Extraction**

To extract the relevant entity values from the text, the following steps are taken:

* **Entity Matching**: Based on the entity\_name (for example, "item\_weight"), text patterns relevant to that entity are identified and extracted.
* **Standardization**: Ensure that the entity value follows a standard format (e.g., "x unit," such as "50 gram") and matches the expected units provided in constants.py.

**3.3 Model Prediction Pipeline**

1. **Loading the Test Dataset**: The test dataset (with columns index, image\_link, group\_id, and entity\_name) is loaded. The model processes each row, applies the predictor function, and generates a prediction for each image.
2. **Prediction Logic**: A custom predictor() function processes the extracted text and identifies the relevant entity value. The prediction is formatted according to the allowed units mentioned in the problem statement.
3. **Saving Predictions**: The predicted values are saved in a CSV file (test\_out.csv) in the required format for evaluation. This file contains two columns:
   * **index**: The unique identifier for the test sample.
   * **prediction**: The extracted and formatted entity value (e.g., "50 gram").

**3.4 Sanity Check**

To ensure that the generated output file follows the required formatting, a sanity.py script is provided. It validates that the output file:

* Has the correct number of rows.
* Has the expected column names (index and prediction).
* Matches the format of the provided sample output file.

**4. File Descriptions**

* **train.csv**: The training dataset that contains image\_link, entity\_name, and entity\_value columns.
* **test.csv**: The test dataset that contains the image\_link and entity\_name but not the entity\_value (which is to be predicted).
* **sample\_test.csv**: A sample test dataset provided to guide the structure of the test data.
* **sample\_test\_out.csv**: A sample output file provided to guide the formatting of predictions.

**5. Code Explanation**

**5.1 Code Structure**

1. **Main Code**
   * **Data Loading**: Loads the CSV files using pandas.
   * **Image Processing and Text Extraction**: Utilizes Tesseract OCR to extract text from images.
   * **Prediction Function**: Applies the prediction logic to extract relevant entity values from the OCR-extracted text.
   * **Output Generation**: Saves the predictions to a CSV file in the required format.
2. **Helper Scripts**
   * **sanity.py**: A script that checks whether the output CSV file is formatted correctly.
   * **utils.py**: Contains helper functions for tasks like downloading images from the web.

**5.2 Example Code Snippets**

**Loading and Renaming Data Columns:**

# Load the dataset

test\_df = pd.read\_csv('path\_to\_test.csv')

# Rename columns for ease of use

test\_df.rename(columns={'Image File': 'image\_file', 'Extracted Text': 'extracted\_text'}, inplace=True)

**Predictor Function:**

def predictor(extracted\_text):

# Logic to extract entity values from extracted text

if isinstance(extracted\_text, str):

return extracted\_text.split()[0] # Dummy logic for entity extraction

return ""

**Apply Predictor and Save Output:**

# Apply the predictor function to the dataset

test\_df['prediction'] = test\_df['extracted\_text'].apply(predictor)

# Save the output

output\_filename = os.path.join('output/', 'test\_out.csv')

test\_df[['index', 'prediction']].to\_csv(output\_filename, index=False)

**6. Challenges Faced**

1. **Text Noise**: The text extracted from images often contains noise (irrelevant characters or OCR errors), which makes it difficult to extract the correct entity values. Cleaning the text was crucial.
2. **Formatting Requirements**: Ensuring that the extracted entity values followed the required format was critical to passing the sanity checks.
3. **OCR Limitations**: The quality of the OCR output depended heavily on the quality and clarity of the images. In some cases, OCR failed to recognize key entity values, and manual rules had to be implemented to handle such cases.

**7. Future Improvements**

* **Model Enhancement**: Use advanced models like deep learning-based image recognition or more sophisticated text extraction models for better accuracy.
* **Error Handling**: Implement more robust error handling for missing or corrupt images, as well as improving the noise reduction in OCR text.
* **Entity Extraction**: Incorporate advanced Natural Language Processing (NLP) techniques to improve the extraction of entity values from text.

**8. Conclusion**

This project demonstrates the ability to extract valuable information from product images using OCR techniques. The solution can be applied to various industries where product details need to be extracted automatically from images. Although the project provides a baseline solution, there is significant potential for improving accuracy through more advanced techniques in both image processing and text extraction.

Top of Form

Bottom of Form