Code Documentation: Entity Information Extraction from Images using PaddleOCR

#### **Overview:**

This Python code processes images to extract specific entity values such as "item weight", "item volume", "voltage", etc., using Optical Character Recognition (OCR) and regex matching. The images are fetched from URLs provided in a CSV file, preprocessed to improve OCR accuracy, and the extracted information is saved back to a CSV file.

### **Modules Used:**

1. **Pandas (pd)**: For handling CSV file operations and dataframes.
2. **Requests**: To download images from the web using URLs.
3. **Pillow (PIL)**: To manipulate and preprocess images.
4. **NumPy (np)**: For numerical operations on images.
5. **OpenCV (cv2)**: Used for image processing tasks like thresholding and blurring.
6. **PaddleOCR**: OCR tool used to extract text from images.
7. **Regex (re)**: To define patterns for extracting specific entity information.
8. **Logging**: To provide detailed debug information throughout the process.

### **Key Components:**

#### **1. Unit Normalization:**

A unit\_normalization\_map standardises unit representations in the text extracted from images. For instance, "kg" is normalised to "kilogram", "lb" to "pound", etc. This ensures consistent units for later processing as only certain units are allowed in the final test\_out csv file.

CODE:

unit\_normalization\_map = {

'g': 'gram', 'kg': 'kilogram', 'lbs': 'pound', 'ml': 'milliliter',

'v': 'volt', 'w': 'watt', 'cm': 'centimeter', 'm': 'meter'

}

#### **2. Reading the CSV:**

The read\_csv function from the Pandas reads the input CSV file containing URLs of images and the corresponding entity names. It reads all the image\_links and thus downloads and processes the images and thus gets them ready for the PaddleOCR.

CODE:

def read\_csv(file\_path):

return pd.read\_csv(file\_path)

#### **3. Image Preprocessing:**

To improve OCR accuracy, images are preprocessed with several steps:

* **Resize**: Enlarge the image to make the text clearer.
* **Contrast Enhancement**: Improves visibility of text.
* **Binarisation**: Converts the image to black-and-white for cleaner text extraction.
* **Denoising**: Reduces noise in the image using Gaussian blur.
* **Adaptive Thresholding**: Applies a more dynamic threshold to improve text visibility.

CODE:

def preprocess\_image(image):

image = resize\_image(image)

image = enhance\_contrast(image)

image = binarize\_image(image)

image = denoise\_image(image)

image = adaptive\_threshold(image)

return image

#### **4. OCR with PaddleOCR:**

The ocr\_paddleocr function uses PaddleOCR to extract text from preprocessed images. This function processes each image, extracts the text in blocks, and returns the extracted text in a structured format. It is used for improved accuracy over EasyOCR and Pytesseract.

CODE:

def ocr\_paddleocr(image, ocr\_instance):

result = ocr\_instance.ocr(np.array(image), cls=True)

...

extracted\_text\_str = "\n".join(extracted\_text)

return extracted\_text\_str if extracted\_text else ""

#### **5. Regex for Entity Extraction:**

Each entity (e.g., weight, volume, voltage) has a corresponding regular expression pattern for extracting numerical values and units. These regular expressions are mainly used to search for the relevant information asked in the “entity\_name” and contain the relevant units. For example, the pattern for extracting "item\_weight" is defined as follows:

CODE:

regex\_patterns = {

'item\_weight': r'(\d+(?:\.\d+)?)\s\*(g|kg|lb?|ounce|gram|pound?)',

'voltage': r'(\d+(?:\.\d+)?)\s\*(v|volt|mv|millivolt)'

}

The extract\_info function runs the OCR, applies the regex pattern based on the entity name, and extracts the matched value and unit.

#### **6. Predictions and Saving Results:**

The predict\_and\_save function iterates over the rows in the CSV, fetching images, running OCR, and applying regex to extract relevant information. The extracted data is appended to a list, which is then saved to an output CSV file.

CODE:

def predict\_and\_save(csv\_data, output\_file\_path, ocr\_instance):

predictions = []

cache = {} # Caches OCR results to avoid redundant processing

for index, row in csv\_data.iterrows():

image\_url = row['image\_link']

entity\_name = row['entity\_name']

predicted\_value = extract\_info(image\_url, entity\_name, ocr\_instance, cache)

predictions.append({'index': index, 'prediction': predicted\_value})

predictions\_df = pd.DataFrame(predictions)

predictions\_df.to\_csv(output\_file\_path, index=False)

### 

### **Detailed Walkthrough of Functions:**

1. **normalize\_unit(unit)**:
   * Uses a dictionary to convert shorthand unit representations (like kg, g, etc.) into their full forms (kilogram, gram, etc.).
2. **format\_value(value)**:
   * Keeps the extracted numeric value in the correct format without modifying it.
3. **Image Preprocessing Functions**:
   * resize\_image(): Enlarges the image for better OCR results.
   * enhance\_contrast(): Boosts image contrast.
   * binarize\_image(): Converts the image to black and white.
   * denoise\_image(): Removes noise using Gaussian blur.
   * adaptive\_threshold(): Applies dynamic thresholding to improve text clarity.
4. **OCR Functions**:
   * ocr\_paddleocr(): Calls PaddleOCR to extract text from an image.
   * run\_ocr(): Preprocesses the image, performs OCR, and extracts text.
5. **Main Prediction Function**:
   * extract\_info(): Fetches the image from a URL, preprocesses it, performs OCR, and uses regex to extract specific entity information (like weight or voltage).

### **Logging:**

Logging is used extensively to provide detailed information on each step of the process. This helps in debugging and tracking the flow of the program, such as loading images, normalizing units, and performing OCR.

CODE:

logging.info("Preprocessing image for OCR")

logging.debug(f"Normalized unit '{unit}' to '{normalized}'")

### **Main Execution Flow:**

1. **Read the CSV file**.
2. **Initialize PaddleOCR**.
3. **Iterate over each image URL**, preprocess the image, run OCR, and extract information using regex.
4. **Save the predictions** to a new CSV file.

CODE:

if \_\_name\_\_ == "\_\_main\_\_":

main()

### **Summary:**

This Python code reads the CSV file extracts link to the images and then processes images to extract entity values using OCR. It leverages PaddleOCR for text extraction, preprocesses images for better accuracy, and uses regular expressions to identify specific patterns. The results are then stored in a CSV file, making it useful for automating the extraction of structured data from images.