TEAM BIG BANG

Explanation:

This code performs Optical Character Recognition (OCR) using PaddleOCR, processes images, and extracts values based on specified entities from a CSV file. Here's an explanation of the code step by step:

1. Importing Libraries

The code starts by importing several libraries, including:

- **os**: For interacting with the operating system (e.g., handling file paths).
- **pandas**: For loading and processing CSV files.
- **re**: Regular expressions for pattern matching.
- **PIL (Pillow)**: For handling images.
- **PaddleOCR**: A tool for Optical Character Recognition.
- **matplotlib**: For displaying images.

These libraries are used throughout the code to manipulate images, perform OCR, and handle data.

2. OCR with PaddleOCR

```python
ocr = PaddleOCR(lang='en', use_angle_cls=True, use_gpu=True)
```

Here, PaddleOCR is initialized with support for the English language and the ability to detect text at different angles. The use_gpu=True enables GPU acceleration, making the OCR process faster.

3. OCR Function

```python
def ocr_with_paddle(image_path):
    finaltext = ''
    try:
        img = PILImage.open(image_path).convert("RGB")
        img_np = np.array(img)
        result = ocr.ocr(img_np)
        ...
    except Exception as e:
        ...
    return finaltext
```

This function takes an image path as input, opens the image, and converts it to a format suitable for OCR. It then applies PaddleOCR to extract text from the image. The extracted text is stored in finaltext, which is returned at the end.

4. Loading and Processing CSV File

```python
csv_file = '/content/train_sample_subset.csv'
df = pd.read_csv(csv_file)
```

The CSV file is loaded into a Pandas DataFrame (df). This CSV contains information such as image_link, which refers to the images that will be processed.

5. Looping Through Images and Performing OCR

```python
for index, row in df.iterrows():
    img_filename = row['image_link'].split('/')[-1]
    img_path = os.path.join('/content/train_images', img_filename)
    ...
```

This loop iterates through each row of the DataFrame. For each row, the code extracts the image_link and constructs the image path. If the image exists, it performs OCR on the image and appends the extracted text to a list (image_texts).

6. Extracting Values from Text Based on Entities

The most complex part of the code is where it extracts specific values and units (e.g., weight, height, voltage) from the text based on the entity type.

Regular Expression Patterns

```python
patterns = {
    'item_weight': r'(\d+\.?\d*)\s*[-.,]?\s*(gram|kilogram|microgram|milligram|ounce|pound
    'width': r'(\d+\.?\d*)\s*(centimetre|foot|inch|metre|millimetre|yard|cm|ft|in|m|mm|yd|
    ...
}
```

These are regular expressions that capture numeric values and corresponding units. For example, for item_weight, it captures values like "10 kg" or "5 pounds". Each pattern is associated with a specific entity (e.g., weight, width, height).

Value Extraction Function

```python
def extract_value_unit(entity_name, image_text):
    ...
    matches = re.findall(pattern, image_text)
    ...
    for match in matches:
        value, unit = match
        value = float(value)
        ...
        values.append((value, unit))
    ...
    return f"{max_value[0]} {max_value[1]}"
```

The function extract_value_unit uses regular expressions to find numeric values and units in the image_text. It then selects the most appropriate value based on the entity type:

- **For weight and volume**: It selects the maximum value found.
- **For dimensions (width, height, depth)**: It applies different rules, such as returning the minimum width or maximum height.

7. Applying the Extraction to the DataFrame

```python
df1['extracted_value'] = df1.apply(
    lambda row: extract_value_unit(str(row['entity_name']), str(row['output_text'])),
    axis=1
)
```

This line applies the extract_value_unit function to each row in the DataFrame, using the entity_name and output_text fields to extract the desired value (e.g., weight or height) from the OCR-processed text.

8. **Saving the Updated DataFrame**

```python
output_csv_file = '/content/train_sample_with_comparison.csv'
df1.to_csv(output_csv_file, index=False)
```

Finally, the updated DataFrame (with the extracted values) is saved to a new CSV file.

Summary of Key Steps:

- **Perform OCR** on images using PaddleOCR.
- **Extract values** (such as weight, height, voltage) from the recognized text based on specific entity names using regular expressions.
- **Store and save** the extracted values in a CSV file for further analysis or use.

This code combines image processing, text extraction, and data handling to automatically extract structured information from images (e.g., product labels).