Method for Text Extraction and Conversion from Images

1 Introduction

This document outlines the method developed for extracting and processing text from images to determine various physical and electrical properties. The properties in question include weight, dimensions (height, width, depth), voltage, wattage, and volume. The approach utilizes Optical Character Recognition (OCR) to extract text from images and regular expressions to identify and convert measurement values.

2 Libraries and Tools Used

2.1 Python Libraries

- **NumPy**: Provides support for numerical operations and handling arrays.
- **Pandas**: Used for data manipulation and analysis.
- **Matplotlib**: Used for plotting (though not directly utilized in the provided code).
- **Requests**: Handles HTTP requests to fetch image data.
- **OpenCV**: An open-source computer vision library (though not directly used in this script).
- **Pillow (PIL)**: Facilitates image processing and enhancement.
- **EasyOCR**: A library for Optical Character Recognition (OCR), used for text extraction from images.
- **io**: Provides the ability to handle byte streams.
- **re**: Facilitates pattern matching through regular expressions.

3 Methodology

3.1 Data Loading

The dataset containing image URLs and entity names is loaded using Pandas:

```
test_data = pd.read_csv("test_file_4.csv")
```

3.2 OCR Initialization

An EasyOCR reader is initialized to process images in the English language:

```
reader = Reader(['en'])
```

3.3 Unit Mapping Definition

A dictionary ('unit$_m$*apping*') *is created to standardize units by mapping common abbreviations and variatio*

```
unit_mapping = {
    'g': 'gram', 'grams': 'gram', 'gram': 'gram',
    'kg': 'kilogram', 'k9': 'kilogram', 'kilogram': 'kilogram', 'kilograms': 'kilogra
    'lb': 'pound', 'lbs': 'pound', 'pound': 'pound', 'pounds': 'pound',
    'oz': 'ounce', 'OZ': 'ounce', '0z': 'ounce', 'ounce': 'ounce', 'ounces': 'ounce',
    'cm': 'centimetre', 'centimetre': 'centimetre', 'cms': 'centimetre',
    'inch': 'inch', 'inches': 'inch', 'ii': 'inches', 'in': 'inch', '"': 'inch',
    'foot': 'foot', 'feet': 'foot', 'm': 'metre',
    'millimetre': 'millimetre', 'mm': 'millimetre', 'millimetres': 'millimetre',
    'yard': 'yard', 'yards': 'yard', 'volt': 'volt', 'volts': 'volt', 'u': 'volt',
    'kilovolt': 'kilovolt', 'millivolt': 'millivolt', 'v': 'volt', 'kV': 'kilovolt',
    'W': 'watt', 'w': 'watt', 'wt': 'watt', 'watt': 'watt', 'watts': 'watt',
    'kilowatt': 'kilowatt', 'kw': 'kilowatt', 'kW': 'kilowatt',
    'litre': 'litre', 'litres': 'litre', 'liter': 'litre', 'liters': 'litre',
    'millilitre': 'millilitre', 'ml': 'millilitre', 'millilitres': 'millilitre',
    'gallon': 'gallon', 'gallons': 'gallon', 'pint': 'pint', 'pints': 'pint',
    'quart': 'quart', 'quarts': 'quart'
}
```

3.4 Text Extraction from Images

A function, `extract_text_from_image`, is defined to:

- Fetch the image from the provided URL.
- Open and process the image using PIL.
- Extract text using EasyOCR.

```
def extract_text_from_image(image_url):
    response = requests.get(image_url)
    img = Image.open(BytesIO(response.content))
    results = reader.readtext(img)
    extracted_text = ' '.join([result[1] for result in results])
    return extracted_text
```

3.5 Parsing and Converting Values

The `extract_value_based_on_entity` function uses regular expressions to parse text based on the entity type. The function identifies relevant values and units, normalizes the units using `unit_mapping`, and selects appropriate values based on the entity type.

```
def extract_value_based_on_entity(extracted_text, entity_name):
    patterns = {
      'item_weight': r'(\d+\.?\d*)\s?(g|grams|gram|kg|k9|kilogram|kilograms|lb|lbs|po
      'width': r'(\d+\.?\d*)\s?(centimetre|centimeters|cm|cms|foot|feet|inch|inches|"
      'depth': r'(\d+\.?\d*)\s?(centimetre|centimeters|cm|cms|foot|feet|inch|inches|"
      'height': r'(\d+\.?\d*)\s?(centimetre|centimeters|cm|cms|foot|feet|inch|inches|
      'voltage': r'(\d+)\s?(volt|volts|kilovolt|kilovolts|millivolt|millivolts|v|u|V|
      'wattage': r'(\d+)\s?(wt|w|W|watt|watts|kilowatt|kw|kW|kilowatts)?',
      'item_volume': r'(\d+\.?\d*)\s?(litre|litres|liter|liters|millilitre|millilitre
    }

    pattern = patterns.get(entity_name)
    if not pattern:
        return None

    matches = re.findall(pattern, extracted_text, re.IGNORECASE)
    print(f"Matches found: {matches}")

    values = []
    for value, unit in matches:
        normalized_unit = unit_mapping.get(unit.lower(), unit)
        if normalized_unit in unit_mapping.values():
            values.append((float(value), normalized_unit))

    if values:
        if entity_name in ['height', 'width', 'depth']:
            if entity_name == 'height':
                value, unit = max(values, key=lambda x: x[0])
            elif entity_name == 'width':
                sorted_values = sorted(values, key=lambda x: x[0], reverse=True)
                if len(sorted_values) >= 2:
                    value, unit = sorted_values[1]
                else:
                    value, unit = sorted_values[0]
            elif entity_name == 'depth':
                value, unit = min(values, key=lambda x: x[0])
        else:
            value, unit = max(values, key=lambda x: x[0])

        return f"{value} {unit_mapping.get(unit.lower(), unit)}".strip() if unit else

    return None
```

3.6 Generating Predictions

The method processes each row of the dataset, extracting text from images and determining values based on entity names. Results are collected into a DataFrame and saved as a CSV file.

```
predictions = pd.DataFrame(columns=['index', 'prediction'])
predictions['prediction'] = None

for i, row in test_data.iterrows():
    print(f"Processing row {i+1}/{len(test_data)}")

    image_url = row['image_link']
    entity_name = row['entity_name']

    extracted_text = extract_text_from_image(image_url)
    print(f"Extracted text: {extracted_text}")

    predicted_value = extract_value_based_on_entity(extracted_text, entity_name)
    print(f"Entity: {entity_name}")
    print(f"Extracted value: {predicted_value}")

    new_row = pd.DataFrame({'index': [row['index']], 'prediction': [predicted_value]}
    predictions = pd.concat([predictions, new_row], ignore_index=True)

predictions.to_csv('predictions_new_4.csv', index=False)
```

4 Conclusion

This method provides a systematic approach to extract and process measurements from images using OCR and regular expressions. It handles diverse unit representations and ensures consistent output for various physical and electrical properties. The results are stored in a CSV file for further use or analysis.