Amazon Machine Learning Challenge

Tech Hunters

September 16, 2024

Abstract

This document provides a comprehensive solution for extracting numerical values and their corresponding units from product images using Optical Character Recognition (OCR) technology. The approach focuses on preprocessing images, extracting text using PaddleOCR, and filtering irrelevant information. It also details the methodology for using regular expressions to accurately detect numbers associated with units (e.g., kg, cm, volt) and convert them to their full forms. By mapping unit abbreviations to their respective full names, this solution standardizes numerical information from a variety of formats, making it easier to integrate into datasets and reports. The final processed data is output to a CSV file for structured analysis. This method can be applied in various scenarios involving product specification extraction and standardization.

1 Introduction

The objective is to extract numerical information from images of product specifications and map the extracted units to their full forms. This is useful for standardized unit reporting in datasets containing mixed formats.

2 Problem Statement

Given a dataset of images with product specifications, the goal is to extract key information (e.g., weight, dimensions, voltage) using OCR and structure the output.

3 Approach & Methodology

3.1 Preprocessing

Images are loaded and PaddleOCR extracts text. Irrelevant information (e.g., '2in1') is filtered out. The OCR output is cleaned for meaningful text processing.

3.2 OCR and Filtering

The OCR is initialized with angle classification to handle rotated text. Results are filtered by removing unwanted words.

3.3 Number & Unit Extraction

Regex patterns are used to extract numbers followed by relevant units (e.g., gms, kg, cm) from the OCR results. Units are standardized.

3.4 Conversion and Tagging

Extracted values are converted to full form (e.g., 'kg' to 'kilogram'), and values are tagged based on context (e.g., height, weight, voltage). If multiple measurements are present, they are assigned as height, width, or depth.

3.5 Data Processing and Output

The extracted data is processed and output to a CSV file. The tagging system ensures that each value is mapped to the corresponding entity (e.g., height, wattage).

4 Code Explanation

4.1 Loading and Preprocessing the Data

The following code initializes PaddleOCR and loads images for text extraction.

```
from paddleocr import PaddleOCR

# Initialize the PaddleOCR model with angle classification enabled.
# This helps to correctly recognize text even if it is rotated in the image.
ocr = PaddleOCR(use_angle_cls=True, lang='en')

# Function to process the image and filter out irrelevant terms.
def ocr_filter(image_path):
    # Perform OCR on the input image and get the recognized text as a list.
    img = ocr.ocr(image_path)

    # List of terms to exclude (e.g., '2in1' is considered irrelevant here).
    exclusion_list = ['2in1']

    # Initialize an empty list to store recognized words.
    recognized_text = []

    # Iterate through the results from PaddleOCR (each 'line' is a block of text).
    for line in img:
        if line is None:
            continue
        # Extract words from each line and append them to the recognized_text list.
        for word_info in line:
            if word_info and len(word_info) > 1:
                recognized_text.append(word_info[1][0])

    # Filter the recognized text by removing unwanted words from exclusion_list.
    filtered_text = [word for word in recognized_text if word not in exclusion_list]

    # Return the filtered text for further processing.
    return filtered_text
```

4.2 Extracting Numbers and Units

This function uses regex to find numerical values followed by relevant units.

```
import re

# List of units that we want to recognize (e.g., grams, kilograms, centimeters, etc.).
units_list = ['gms', 'kg', 'cm', 'volt', ...]

# Function to extract numerical values and associated units from the filtered text.
def extract_numbers_and_units(text_list):
    # Regex pattern to capture numbers followed by any of the units from the units_list.
    pattern = r'(\d+(?:\.\d+)?\s*(?:' + '|'.join(units_list) + '))'

    # Initialize an empty list to store the extracted numbers and units.
    extracted = []
```

```
    # Iterate through the filtered text to find matches with the regex pattern.
    for text in text_list:
        # Convert text to lowercase for uniform matching.
        text_lower = text.lower()

        # Find all matches for the pattern in the current text.
        matches = re.findall(pattern, text_lower)

        # Append the matches to the extracted list.
        extracted.extend(matches)

    # Return the list of extracted numbers and units.
    return extracted
```

4.3 Conversion to Full Units and Tagging

This function converts abbreviated units to their full forms and tags them appropriately.

```
# Mapping of abbreviated units to their full forms.
unit_mappings = {
    'kg': 'kilogram',
    'cm': 'centimeter',
    'gms': 'grams',
    'volt': 'volts',
    ...
}

# Function to convert extracted units to full form and perform unit conversion.
def convert_to_full_units(extracted_list):
    # Initialize an empty list to store the converted values.
    converted_list = []

    # Regex pattern to capture number and unit separately from the extracted data.
    pattern = r'(\d+(?:\.\d+)?)\s*(\w+)'

    # Iterate over each extracted item.
    for item in extracted_list:
        # Match the pattern against the current item (number and unit).
        match = re.match(pattern, item)

        # If a match is found, proceed with conversion.
        if match:
            # Extract the number and unit from the matched groups.
            number = float(match.group(1))
            unit = match.group(2).lower()

            # Convert the abbreviated unit to its full form using the unit_mappings dict
            full_unit = unit_mappings.get(unit, unit)

            # Handle the case where a singular form of the unit is needed (e.g., 'kilogra
            if number == 1 and full_unit.endswith('s'):
                full_unit = full_unit[:-1]

            # Append the converted number and full unit to the converted_list.
            converted_list.append((number, full_unit))

    # Return the list of converted values.
    return converted_list
```

4.4 Tagging Dimensions

Recognized values are sorted by size and tagged as dimensions or other parameters.

```
# Function to categorize and tag the extracted values (e.g., dimensions, weight, voltage
def tag_dimensions(extracted_values):
    # Initialize empty lists for dimensions and a dictionary for other parameters (e.g.,
    dimensions = []
    other_params = {}

    # Iterate through the list of extracted values.
    for value, unit in extracted_values:
        # Get the full form of the unit (e.g., 'cm' -> 'centimeter').
        full_unit = unit_mappings.get(unit, unit)

        # Categorize the values based on the unit type.
        if full_unit in ['centimeter', 'inch', 'meter']:
            # Add to dimensions if the unit is a length measurement.
            dimensions.append((value, full_unit))
        elif full_unit in ['kilogram', 'pound']:
            # Assign item weight if the unit is related to mass.
            other_params['item_weight'] = f"{value} {full_unit}"
        elif full_unit == 'volt':
            # Assign voltage if the unit is volts.
            other_params['voltage'] = f"{value} {full_unit}"
        elif full_unit == 'watt':
            # Assign wattage if the unit is watts.
            other_params['wattage'] = f"{value} {full_unit}"

    # Sort dimensions by their magnitude in descending order.
    dimensions.sort(reverse=True, key=lambda x: x[0])

    # Return the tagged dimensions and other parameters.
    return other_params
```

5 Results

The code extracts numerical data from images and formats it into a CSV file. The solution is accurate for extracting weight, dimensions, voltage, and wattage.

6 Conclusion

This solution effectively extracts and standardizes numerical information from images using OCR. Regular expressions for unit extraction provide a reliable method for obtaining product specifications in a structured format.

7 References

- PaddleOCR Documentation: `https://github.com/PaddlePaddle/PaddleOCR`
- Python Regex Documentation: `https://docs.python.org/3/library/re.html`