Amazon ML Hackathon

Problem Statement Overview

Objective: Develop a machine learning model to extract specific entity values from product images. This is crucial for digital marketplaces to automatically gather key product details (e.g., weight, dimensions) from images that lack textual descriptions.

Data Description

Dataset Files:

1. **Training File (`dataset/train.csv`):** Contains labelled data with entity values.
2. **Test File (`dataset/test.csv`):** Contains images for which entity values need to be predicted.
3. **Sample Files:**
    - `dataset/sample_test.csv`: Example test input file.
    - `dataset/sample_test_out.csv`: Example output file showing the correct format.

Data Columns:

- **index:** Unique identifier for each data sample.
- **image_link:** URL to download the product image.
- **group_id:** Category code of the product.
- **entity_name:** Name of the product entity (e.g., "item_weight").
- **entity_value:** Value of the product entity (e.g., "34 gram").

Output Format

Required CSV Output File:

- **index:** Unique identifier from the test file.
- **prediction:** Predicted entity value in the format "x unit" (e.g., "2 gram", "12.5 centimetre"). If no value is detected, return an empty string.

Machine Learning Approach

Algorithm Overview:

1. **Data Preparation:**
    - **Image Download:** Use `download_images` function from `src/utils.py` to fetch images from URLs.
    - **Data Preprocessing:** Normalize and resize images for model input.
2. **Feature Extraction:**
    - Use a Convolutional Neural Network (CNN) to extract features from images. Popular architectures include ResNet, Inception, and EfficientNet.
    - Optionally, use pre-trained models like those from TensorFlow or PyTorch and fine-tune them on the provided dataset.
3. **Entity Extraction:**
    - Implement a Multi-Label Classification model to predict the entity values based on extracted features.
    - Alternatively, use object detection models (e.g., YOLO, Faster R-CNN) to identify and classify entities within images.
4. **Post-Processing:**
    - Format predictions to match the required output format. Ensure predictions are in standard formatting and valid units as listed in `src/constants.py`.


```
import pytesseract
from PIL import Image

def extract_text_from_image(image_path):
    # Open the image file
    with Image.open(image_path) as img:
        # Use pytesseract to do OCR on the image
        text = pytesseract.image_to_string(img)
    return text

if __name__ == "__main__":
    # The code below was changed to use the global variable image_path instead of expecting a command line argument.
    image_path = "/content/Amazon ML.jpg"
    text = extract_text_from_image(image_path)
    print("Extracted Text:")
    print(text)
```

```
Extracted Text:
Power in a Snap

Magnets align phone for faster
and easier charging
```

Evaluation Metrics

F1 Score Calculation:

- **True Positives (TP):** Predictions that match ground truth.
- **False Positives (FP):** Predictions that do not match ground truth.
- **False Negatives (FN):** Ground truth values that were not predicted.

$$\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}} \quad \text{Recall} = \frac{\text{TP}}{\text{TP}+\text{FN}} \quad \text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision}+\text{Recall}}$$

Dataset Utilities:

- **Sanity Checker and Output Format:** `src/sanity.py` ensures the output file meets formatting requirements.
- **Image Downloading:** `src/utils.py` helps in fetching images.

Conclusion: Successfully addressing this problem will enable the automation of data extraction from images, enhancing the efficiency and accuracy of product information retrieval in digital marketplaces. By implementing a robust solution, we can bridge the gap where textual descriptions are missing, thereby streamlining product data management and improving the overall user experience in e-commerce platforms.

Improved Data Accuracy: Automated extraction reduces human error and ensures consistent data quality for product listings.

Efficiency Gains: Speed up the process of data entry and product catalog management by automating the extraction of key details from images.

Enhanced User Experience: Provide users with accurate product information without relying on textual descriptions, improving their decision-making process.

Scalability: Facilitate the handling of large volumes of product images, making it easier for digital marketplaces to expand their catalogs.