OCR Processing Using Tesseract in Python

OCR (Optical Character Recognition) is a technology that converts different types of documents—such as scanned paper documents, PDF files, or images taken by a digital camera—into editable and searchable data.

**Introduction to OCR**

**What is OCR?**

OCR is a technology that enables the extraction of text from images or scanned documents. It uses machine learning and pattern recognition to interpret the visual representation of text and convert it into machine-readable characters.

**Example Uses of OCR**:

* Digitizing printed documents.
* Extracting text from scanned books or articles.
* Automating data entry tasks.
* Creating searchable PDF files.

**Use Cases for OCR**

1. **Document Digitization**: Converting physical documents into digital formats.
2. **Data Entry Automation**: Automating the extraction of data from forms and invoices.
3. **Text Searchability**: Making scanned documents searchable by text.
4. **Language Translation**: Translating printed text into different languages by first converting it to digital text.

**Using Tesseract for OCR**

**Installing Tesseract and pytesseract**

**Tesseract** is an open-source OCR engine developed by Google. **pytesseract** is a Python wrapper for Tesseract that makes it easier to use Tesseract in Python code.

* **Installation**:
  1. **Install Tesseract**:
     + **Windows**: Download the installer from the [Tesseract GitHub page](https://github.com/UB-Mannheim/tesseract/wiki) and follow the installation instructions. Add the installation path (e.g., C:\Program Files\Tesseract-OCR) to your system's PATH environment variable.
     + **macOS**:

brew install tesseract

* + - **Linux**:

sudo apt-get install tesseract-ocr

* 1. **Install pytesseract**:

pip install pytesseract

**Extracting Text from Scanned PDFs**

To extract text from scanned PDFs, you'll first need to convert the PDF pages to images and then apply OCR.

**Example**:

import pytesseract

from PIL import Image

import pdf2image

# Convert PDF pages to images

pages = pdf2image.convert\_from\_path('scanned\_document.pdf')

# Perform OCR on each page

for page\_number, page in enumerate(pages):

text = pytesseract.image\_to\_string(page)

print(f"Page {page\_number + 1} text:\n{text}")

**Note**: Install pdf2image using pip install pdf2image and ensure you have poppler installed for PDF conversion.

**Preprocessing Images for Better OCR Results**

Preprocessing can significantly improve OCR accuracy, especially with noisy or low-quality images.

**Common Techniques**:

* **Grayscale Conversion**:

from PIL import Image, ImageOps

img = Image.open('example.png')

gray\_img = ImageOps.grayscale(img)

gray\_img.save('gray\_example.png')

* **Thresholding**:

import cv2

img = cv2.imread('example.png', cv2.IMREAD\_GRAYSCALE)

\_, thresh\_img = cv2.threshold(img, 128, 255, cv2.THRESH\_BINARY)

cv2.imwrite('thresh\_example.png', thresh\_img)

* **Noise Reduction**:

import cv2

import numpy as np

img = cv2.imread('example.png', cv2.IMREAD\_GRAYSCALE)

denoised\_img = cv2.fastNlMeansDenoising(img, None, 30, 7, 21)

cv2.imwrite('denoised\_example.png', denoised\_img)

**Advanced OCR Techniques**

**Handling Noisy and Low-Quality Scans**

For noisy and low-quality scans, advanced preprocessing and OCR techniques can help improve results.

* **Noise Reduction**: Use filters to remove noise before OCR.
* **Image Enhancement**: Apply contrast adjustment and sharpening to make text more distinguishable.

**Example**:

import cv2

from PIL import Image

import pytesseract

# Load and preprocess image

img = cv2.imread('noisy\_scan.png', cv2.IMREAD\_GRAYSCALE)

denoised\_img = cv2.fastNlMeansDenoising(img, None, 30, 7, 21)

\_, thresh\_img = cv2.threshold(denoised\_img, 128, 255, cv2.THRESH\_BINARY)

# Save preprocessed image

cv2.imwrite('preprocessed\_scan.png', thresh\_img)

# Perform OCR

preprocessed\_img = Image.open('preprocessed\_scan.png')

text = pytesseract.image\_to\_string(preprocessed\_img)

print(text)

**Extracting Structured Data from Scanned Documents**

For documents with structured data (e.g., tables), additional processing might be required to extract data in a structured format.

* **Table Extraction**: Use libraries like pytesseract combined with image processing libraries or specialized tools for table recognition.

**Example**:

import pytesseract

from pytesseract import Output

import cv2

# Load image

img = cv2.imread('table\_scan.png')

# Perform OCR with layout analysis

detection\_boxes = pytesseract.image\_to\_boxes(img)

data = pytesseract.image\_to\_data(img, output\_type=Output.DICT)

# Process detected text

for i, word in enumerate(data['text']):

if word.strip() != "":

print(f"Word: {word}, Position: ({data['left'][i]}, {data['top'][i]})")

**Summary**:

* **OCR Basics**: OCR converts images of text into machine-readable text. It is used for digitizing documents and automating data extraction.
* **Tesseract and pytesseract**: Install Tesseract and its Python wrapper to extract text from images and PDFs.
* **Advanced Techniques**: Improve OCR accuracy by preprocessing images and handling structured data.