**PyTesseract library**

**Objective:**

To read a pdf file along with the tables and create a CSV file for the same using the Pytesseract library

**Introduction**

Python-tesseract is an optical character recognition (OCR) tool for python. That is, it will recognize and “read” the text embedded in images.

Python-tesseract is a wrapper for [Google’s Tesseract-OCR Engine](https://github.com/tesseract-ocr/tesseract). It is also useful as a stand-alone invocation script to tesseract, as it can read all image types supported by the Pillow and Leptonica imaging libraries, including jpeg, png, gif, bmp, tiff, and others.

**Procedure**:

#### 

#### Acquiring the Image database (PDF files needs to be converted to image file)

#### Preprocessing/cleaning the image file

#### Reading the preprocessed image file using pytesseract

#### Applying Text labeling on the extracted text to filter out required entities

#### Storing the filtered important text to a dictionary,csv,json file

#### 

### **Preprocessing Image for Tesseract using OpenCV2**

-Pre Processing is a Must do to improve accuracy. But If certain pre-processing methods result in loss of accuracy you can choose to drop that method.

- To avoid all the ways your tesseract output accuracy can drop, you need to make sure the image is appropriately [pre-processed](https://github.com/tesseract-ocr/tesseract/wiki/ImproveQuality#image-processing).This includes rescaling, binarization, noise removal, deskewing,orientation detection etc.

- Resizing the image can also be performed if the image resolution is low.

-Dilation and erosion is important to increase/decrease font size to improve clarity and needs to adjusted manually according to image

IMPORTANT: Using the canny method for preprocessing images is resulting in significant accuracy drop(not recommended).

**Results**

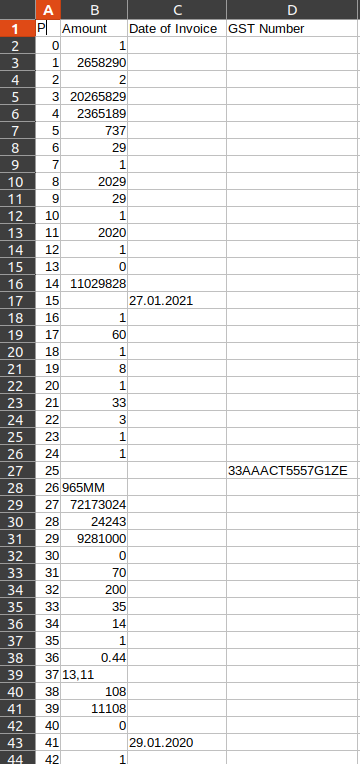
1- Semi-Structured/Unstructured Images are generating poor accuracy OCR results and require significant manual tweaking according to the image provided to get the exact text.

2-Csv,json, Dictionary creation has been easily done for results.

Git-hub: <https://github.com/aayushsrivastava0/Python_codes.git>

| Type of file | Accuracy(approx) | File name |
| --- | --- | --- |
| Semi-Structured | 50-60% | invoice2.jpg |
| Structured | 50-60% | invoice3.pdf |
| Unstructured | 50-60% | invoice.jpg |

**Analyzing Result**

****

-Target was to extract 4 entities as you can see from results only 3 entities got extracted, and the Amount entity wasn’t reliable due to OCR detecting gibberish and all the numbers in image making results(amount) unreliable.

**Advantages**

**1-** Black listing characters that you are confident that they will not appear in your text

**2-** TEXT TEMPLATE MATCHING supported - Take the example of trying to find where a date/ PAN number or “xyz” is in an image.

**3-** Tesseract supports customized pre-processing layers based on the user’s needs.

**4-** Tesseract performance is directly linked towards the quality of the image.

**5-** It can Detect Multiple languages but you need to provide an input on which languages you want to detect.

**6-** Pre-processing an image using OpenCV significantly improves the OCR results.

**Limitation**

**1-** It is not capable of recognizing handwritten text.

**2-** It requires a clear image as input. A poor quality scan may produce poor results in OCR.

**3-** It doesn’t give accurate results of the images affected by artifacts including partial occlusion, distorted perspective, and complex background.

**4-** It is not good at analyzing the normal reading order of documents. For example, you might fail to recognize that a document contains two columns, and might try to join the text across those columns.

**5-** It does not expose the font family’s text information.

**6-** Semicolons and different punctuations are getting detected as letters/numbers on poor quality images

**7-**Text template matching(bounding box creation) needs to be tweaked manually before hand to get accurate entity extraction results

**8-** The OCR is not as accurate as some commercial solutions available to us.

**9-** It may find gibberish and report this as OCR output.(frequently happening)

**10-** Number entity extraction example(amount) is detecting all the number present in image and giving it as output

**Conclusion**

Pytesseract has poor accuracy and requires users to heavily tweak the code to slightly better results.It is able to process both pdf and image file of all types.