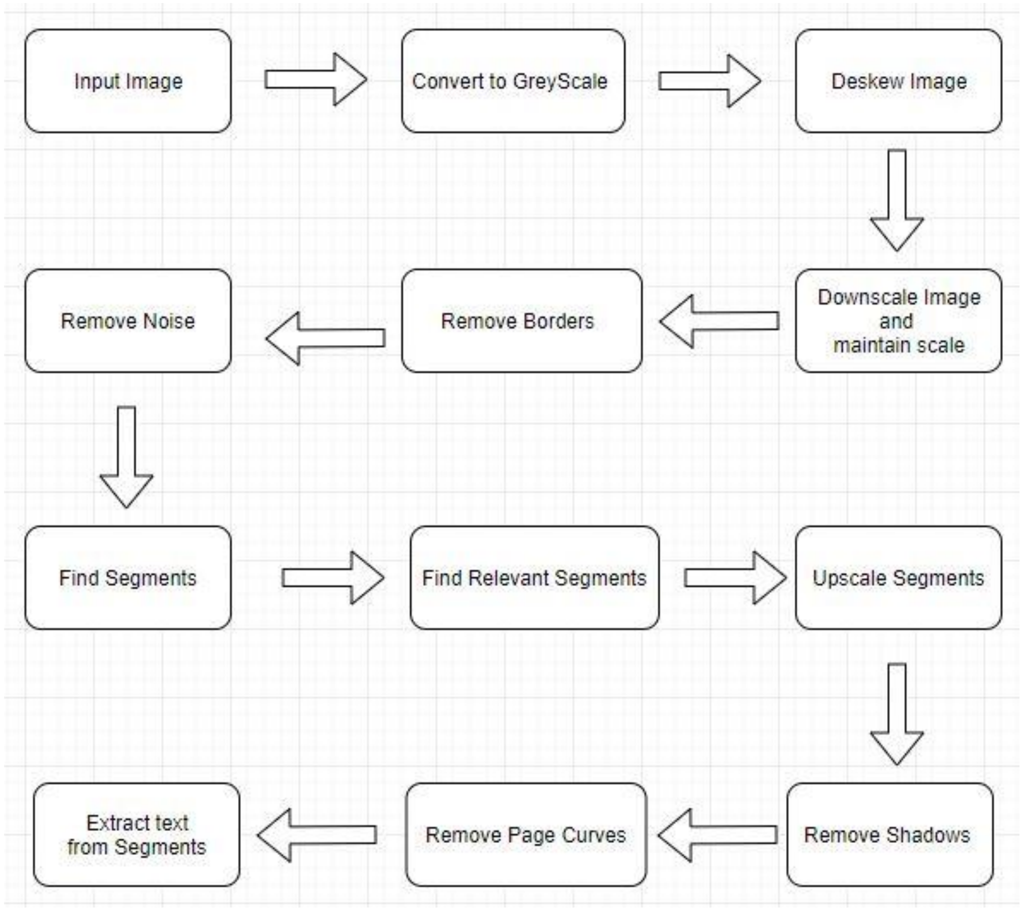
Resume Image Text Extractor

Problem Statement: To extract the text from Image uploaded by seeker and pass this extracted text to Parser to extract entities of the Resume.

Approach: For extracting Text from Image, We can directly use Tesseract - OCR developed by google and including a dependency of Python wrapper for this (pytesseract). But, the problem arises when the image uploaded by seeker is not bright enough or not perfectly aligned with horizontal axis. In this case, text extracted from Tesseract – OCR is not right. Hence, the first step should be preprocessing of the Image, Making it horizontally aligned and properly brightened up.
Further, to add to this, to parse this text in the form of resume and extract the entities from this text, we can find out the segments from the image based on dynamic spacing and then categories it to one of our required categories.
Doing this, also reduces time complexity while extracting text from image using pytesseract. If we extract the text from whole image, time taken by pytesseract to extract the text is higher (approximately 2-3 times more) than the time taken by pytesseract to extract text from segments of the text containing relevant information only. This is because not all the pixels of the image need to be evaluated in this by pytesseract, only relevant text containing the useful information.

Flow Chart:



Code Flow:

We just need to pass the image in the **process_image** function. The steps this function takes, is as follows:

1) Step-1: Convert the Image into Grey Scale (i.e., Black and White).
2) Step-2: Find the horizontally aligned Image by calling **deskew** function:
   For calling this function, bit of preprocessing is involved. We need to convert Image into numpy array and normalize them by dividing with maximum possible pixel, i.e., 255.
   a) This function internally calls another function **determine_skew**, which return the skew angle of the image on the basis of **Hough Line Peaks**. One can read about **Hough transforms** and skew removal at https://www.sciencedirect.com/science/article/pii/S1319157816300015
   b) Sometimes, if the image is not bright enough, it is unable to find the skew angle, hence it gives a message "Bad Quality Image".
   c) After finding the skew angle, we rotate the image by that angle and return rotated image.
3) Step-3: Downscale the image by a factor so that both height and width of the image are less than 2048 pixels by calling **downscale_image** function and maintain this scale. This is done to reduce computation complexity at later stages.
4) Step-4: Remove Border Components. This takes following steps:
   a) Find contours by calling **findContours** function.
   b) Find Borders by calling **find_border_components** function.
   c) If borders are present, remove borders by calling **remove_border** function.

   This is done so that while finding segments, the whole region should not come up as a segment. Which will happen if a boundary is present in the image outside the text.

5) Step-5: Remove Noise in the filter by using **rank_filter** function.
6) Step-6: Find Segments by calling **find_components** function. This function does the following steps:
   a) Checks if the information in the image is contained in full page or less. If the information in the image is in lesser area than a predefined threshold, then it sets max_components as 16, Otherwise it is 32.
   b) Then, it dilates the image by calling **dilate** function to join the segments until number of segments found are less than or equal to max_components.
7) Step-7: Find relevant segments by calling **find_optimal_subsets** function. It does the following steps:
   a) It checks the FScore of each segment by using Precision and Recall to check the information contained in the segment. If the FScore is more than a predefined threshold, then it only it is allowed further.
   b) Then, it checks if segment is inside another segment and if it is, it takes the bigger segment.
   c) Then, it sorts the segments on the basis of position of segment to extract the text in the order it is supposed to be.
8) Step-8: For each segment, do the following steps:
   a) Upscale the segment by the scale by which it was downscaled and extract that portion from the image.
   b) Remove Shadows from the segment, if any, by calling **remove_shadows** function.

    c) Remove the curves that might occur in image of the page while clicking Photo from camera by calling **deskew_partial** function. This uses hysteresis of the image as tool to remove curves and skewness of the segment.
    d) Extract the text from Image by calling image_to_string function of pytesseract.
9) Step-9: Return the extracted text segments to pass to Parser for further processing.

Future Work: One can find new way to find the orientation of the image and deskewing. One can make their own OCR instead of using Google's Tesseract – OCR for handwriting detection.

Pre-requisites: The following must be installed in the system to set up this module:

1) Tesseract – OCR
2) Python 3.6
3) Following Python Libraries:
    a) opencv-python
    b) scipy
    c) scikit-image
    d) pytesseract
    e) numpy
    f) tensorflow
    g) pillow
4) The Path, where Tesseract – OCR is installed, should be there in the config file.