**A Case Study: Optical Character Recognition (OCR) Engine for extracting structured table information from scanned Images or PDFs**

**Abstract:**

Portable Document Format (PDF) is a current way to exchange data and information. As PDF file type is universal for different systems and comes with trusted security, it is used extensively in corporate world. In one of the hypothesis, we deal extensively with scanned images or PDFs which are source of unstructured data. Tables are ubiquitous document component and is valuable information source. Thus, there is an effective need to extract the data stored in tables without disturbing its structure. The purpose of this case study is to explain the functionality of OCR engine developed by us to extract structured table information from PDFs and scanned images.

This Case study will clearly define the method used to build the software, its features, and shortcomings.

**Introduction:**

Tables are among the most common means of presenting and structuring data with high information density in any document including PDFs. However, it is not an easy task because tables can be of varying formats. There are some free source tools available on internet which works fine with text based PDFs but they are error prone with scanned PDFs and don’t work well scanned PDFs. Our solution incorporates use of Tesseract software an optical character recognition engine, which takes TIFF image file format as input and reads text inside the file. We are also leveraging OpenCV, a library of programming functions mainly aimed at real-time computer vision. We are using several image processing techniques with the help of OpenCV like Canny’s Edge Detection algorithm, Hough Probabilistic Line Transformation algorithm, Morphological Structuring elements etc.

This OCR engine extract structured table data in sequential manner from scanned images or PDFs. The algorithm is explained in methods section.

**Approach:**

As stated above in introduction, we are utilizing several image processing techniques with the help of OpenCV and utilizing Tesseract software to OCR image to read text inside the images. The idea here is to detect the table and identify its location within the image and run loops over each horizontal and vertical lines and read cell data in sequential format to maintain the table structure. We have used Python for coding purpose. Below is the elaboration of Algorithm:

1. Converting the scanned PDF in TIFF image file formats:

We are utilizing the free available software online for converting PDF into TIFF images keeping the order of pages.

1. Skew correction:

Skewed images can cause problems and result in false outcomes. Therefore, it is essential to identify the skew and correct it before moving ahead. There are multiple methods available to do this task.

1. Detect horizontal and vertical lines and knowing the line coordinates:

Now by this step we will have pages of our PDF as image TIFF files preprocessed. The general feature of any table is it consists of many vertical and horizontal lines. Hence, this step is about detecting the lines and identifying the location of table, in turn each cell. For doing so below are the steps:

* Transform the source image to gray scale
* Apply Adaptive Thresholding at bitwise\_not of gray image to segment it
* Create structure elements for extracting horizontal lines through morphology operations
* Create structure elements for extracting vertical lines through morphology operations
* Mask the identified horizontal and vertical lines to have a complete table. The masked image only contains horizontal and vertical line on bitwise\_ not of gray image
* Apply probabilistic hough line transformation algorithm to identify horizontal and vertical lines with coordinates in our image
* As hough line transformation can identify multiple lines for single edge, remove duplicate lines (lines in nearby pixels) to keep single hough line with coordinates for an edge. Using coordinate information to identify the duplicate lines for nearby pixels (in range of 20 pixel distance)
* As we require to keep the structure of table, we require to sort the lines to loop over those lines in sequential manner
* Now we have 2 coordinates (X1, Y1, X2, Y2) for each horizontal and vertical sorted line and hence we can leverage this information to crop each cell in sequential manner. Run a loop on sorted vertical lines inside the loop of sorted horizontal lines and utilize coordinate information to identify region of interest (roi) or cells
* Use Tesseract to read text inside the region of interest or cells and store it in a list.
* Store the list of rows in another list and in similar fashion we can store the multiple table information.
* We can write information in the desired format in a new output file

**Extraction Speed:**

The software is taking approximately 500 seconds (8-9 minutes) for extracting information from a 20-page scanned PDF considering every page has only tables across it (Example of heavy processing) using a quad core processor. As there is sequential processing and every step is essential to serve our purpose and maintain the table structure, the above processing time is justified. The major time-consuming step is extracting text from roi using Tesseract. Processing time also depends on quality of image, if the quality is too good or too bad, the software might take even longer to process it.

The input file type:

Scanned images, PDFs, Databases, Email, MS Office Documents, Faxes and Standard archived data.

Processing time:

Depends on quality of image, will be more if quality if too high or too low. On average takes 30 – 40 seconds for 5-page PDF (claimed by Deloitte consultant).

Where it fails:

When the document templates have variations.

When its best:

When similar document template.

**Other market tools available:**

Tabula:

It is an open-source software, that helps you liberate data tables trapped inside PDF files. Tabula allows to extract data provided in PDFs in CSV format using a simple web interface. It works well with text-based PDFs with fast processing speed but fails with scanned PDFs. However, there is an option to auto detect tables in their web interface which accurately detects Tables even in scanned images. Tabula works line to line (horizontally), which makes it difficult for our purpose as there might be cases where a single cell in table can have multiple lines which Tabula will represent as different rows.