OCR for Handwritten Form

-Mihir Verma

Problem Statement:

Given a scanned query document, you have to predict (extract) the (a) Date (b) Bank a/c number. The document template is same across all images. The model will be evaluated on the basis of validation set as provided.

Implementation:

- The current system has been implemented via Python and it's robust dependencies. OpenCV and NumPy have been used to process images and work with them with effectively.
- Use of OS module and ArgParse has been done at places due to their requirements, like storage and retrieval of files.
- The entire work has been done on linux based system – Mac so to be considered while considering paths and modules.

Targets and achievements:

1.) To recognize data elements within a scanned document.

- **Converting and Loading:**
  - The given data (scanned documents) are available in pdf form, so before going ahead with processing the documents, they need to converted into image form.
  - This can be easily achieved with a module called 'pdf2image' for python available online on GitHub.

  (Due to shortage of storage on my Mac's system, dependencies for the above module could not be installed, so the pdfs were converted to png files manually)
  - Next, the images need to be loaded in the proposed system for further processing. This is done using 'OpenCV' for python.

  (All the other modules used for image processing through-out the project use OpenCV for achieving results.)

- **Preprocessing:**

  To recognize data elements within a fed scanned document, the document first needs to be pre-processed.

  To preprocess the data, the proposed system uses multiple filters.
  - Firstly, the system checks if the image is portrait or landcape. As in the data set, portrait images repeat the same information over three folds, it crops one third of the document image vertically in case of a portrait image.
  - Then, it has to crop the scanned black edges from the document, in order for better skew angle detection. For this the system uses 'AutoCroppy' tool developed by 'gerwin3' available on GitHub
    (reference below), with adjusted parameters.
  - Next, it has to correct the skewness of the document for better text recognition. For this, the system uses a model for Skew Correction proposed by 'Félix Abecassis' on his blogpost. (references below). Although it is not always successful, the model helps in correcting skewness for many of the documents.
  - Finally, the brightness and contrast of the image is adjusted for better text detection. For this, I developed a script that changes alpha value (for brightness and contrast respectively) using OpenCV. (This was with the help of an answered question on StackOverflow regarding the same topic, which is referenced below).

- **Finding Square Hulls:**

  The Date and Bank A/C Number are written in the documents inside square (or rectangle) boxes. So this feature was considered as an important part of useful text detection and subsequently these specific regions are looked for, by the system.
  - Using the MSER (Maximally Stable Extremal Region) Class provided by OpenCV, the MSE regions are extracted from the document.
  - Further, the convex hulls among those regions are isolated, for ease of square/rectangle detection.
  - Next, a pattern of convex hulls with four polygonal sides is looked for, in the set of convex hulls and these specific hulls are returned within the given image.

This concludes the recognition of useful data elements within the scanned document.

2.) To extract useful information, in given case: 1. Date 2. Bank A/C number.

- **Extracting Date:**
  - To extract the date from given document, the document is divided into a grid of 4x4, and square hulls from the top right grid are searched for.
  - These square hulls are combined using MSER regions merging algorithm proposed by 'GaneshTata' in a StackOverflow answer for identifying text lines in OCR (reference below).
  - This image is saved locally for further processing.

- **Extracting Bank A/C Number:**
  - A square/rectangle hull with the longest side in the given document represents the bank a/c number column on the form. This logic is used for extracting the column of Bank A/C Number from the document.
  - Thus for all the square hulls produced, an iterator keeps track of the hull with a longest side. This hull is returned as region of Bank A/C Number.
  - The image with this region is saved locally for further processing.

This concludes the extraction of Date and Bank A/C number regions from the given document.

3.) To recognize the handwritten digits from the isolated text.

- **Isolating Characters:**
  - Isolating the characters remains a bottle neck of the current system. This is mainly due to some persistent issues with the data set which have been further mentioned in the report.
  - I developed a module 'GetChars' that filters square hulls from extracted data images with area filter. Experimentally, the approximate area value of contours/hulls representing character values was determined. A filter around min Area and max Area around this empirical area is applied in the logic to isolate characters.
  - Furthermore, the characters can be checked for enclosing substantial amount of data pixels, this can further help in isolating blank characters from required characters. (Due to the lack of time, I was not able to implement this.)
  - These characters are locally stored too, for examination purposes.
  - The method of finding MSE regions and later filtering them with Square Hulls and area-wise filter still causes the system to generate a lot of duplicate regions.
  - This regions cause multiple single digit entries in the system and in turn causes ineffective prediction.

- **Recognizing the Characters:**
  - The current system uses a kNN algorithm to recognize handwritten digits.
  - A customized neural network can be designed for achieving good results. (Which, again, the lack of time could not permit me to)

    - The isolated characters are fed into the kNN model as well as saved locally for examining.

- **Presenting the Information:**
    - I simply reported the output in a text file which makes examining the output easy.
    - Currently, due to repetition and duplication of filtered regions from isolated text, and additionally, a weak predictive model for handwriting recognition causes a poor output.
    - Although, it can be worked upon and improved relatively easily, as the major tasks of preprocessing the data and isolating the text have been achieved so well.

Project Progress Milestones

- Pytesseract was used initially to recognize text from given documents. But due to its extremely poor performace, it had to be discarded.
- Text identification using Contours and Shape filtering contours failed. This forced me to try out new ways and ultimately MSER was elected as the best feasible choice.
- After being able to identify text all over the document, useful text identification posed a major issue.
- Initially, I attempted to identify 'Date' and 'Bank a/c number' written in the form using Pytesseract. I would later have to find the co-ordinates to those words and isolate the square region to the right of those words.
- Pytesseract failed here too, not being able to recognize simple printed text with dependable accuracy.
- Also, its module gives the output of identified words and characters in basic formats such as string and dictionaries, even for complete information tress. It is not feasible to manipulate these data types to generate fruitful results. So once again, Pytesseract was discarded.
- Later on, a simple logic was applied to extract date column. I decided to divide the document in grids and then look for text in the top right grid. Experimentally a 4x4 grid achieved good results.
- But the text boxes were still little boxes with no good overall text extraction, hence I looked more into being able to combine contour/MSER boxes and I succeeded.
- The Bank a/c number was extracted with a simple logic, its MSER had the longest side in the entire form if measured correctly.
- Although theoretically true, this logic failed for some images due to their uncorrectable skewness, which resulted into smaller MSER boxes instead of one large box.
- I tried different logics to extract the aforementioned information, but failed to do so. Thus, I moved on with this logic.
- Separating characters after isolating chunks of Date and A/c numbers from documents posed the most difficult challenge.
- Due to surrounding square boxes around handwritten digits, it became very difficult to single out characters.
- After many attempts with various logics, I settled on filtering square contour boxes with respect to their areas. I determined the approximate area experimentally and worked around its upper and lower limits.
- Finally, after all of this, no time to develop an efficient/robust prediction algorithm was available, so I settled on the classis kNN algorithm to predict the handwritten digits. As noticeable, it does not work very well with out data set.

Result:

- The dates and bank a/c columns are separated out from the rest of the documents very well.
- The characters are also isolated from extracted texts with quite ease and good enough results.
- With a more effective approach, like a neural network with a CNN model, the efficiency of recognition/prediction algorithm can be drastically improved even with limited efforts.

Issues:

1.) Scan borders around images

The black borders around some images prevent the system from effectively applying its skew correction algorithm.

2.) Skewed Images

Some images cannot be corrected with their skews and remain extremely skewed even after processing. To extract data boxes from such images poses a serious challenge for the system.

3.) Bad Data Entry

The data entered in form of digits in some of the documents has been very poor. The digits are spread outside the given boxes in some cases. This makes recognizing difficult for kNN algorithm, especially when combined with poor handwriting in some cases.

References

1. https://github.com/gerwin3/autocroppy for AutoCroppy.
2. https://www.pyimagesearch.com/2017/02/20/text-skew-correction-opencv-python/, http://felix.abecassis.me/2011/10/opencv-rotation-deskewing/
   for Skew Correction.
3. https://stackoverflow.com/questions/39308030/how-do-i-increase-the-contrast-of-an-image-in-python-opencv/50053219#50053219
   for Brightness and Contrast adjusment.
4. https://stackoverflow.com/questions/48615935/merging-regions-in-mser-for-identifying-text-lines-in-ocr
   for Merging regions in MSER for identifying text lines in ocr.
5. https://medium.com/@sudhirjain01dec/optical-character-recognizer-using-knn-and-opencv-part2-57637649079c
   for referencing kNN implementation using Python and OpenCV.