Document Identification and Data Extraction

Priyam Garg
Dept. of Metallurgy and Materials Science
Indian Institute of Technology, Bombay
Mumbai, India
Roll No.: 210111001
priyamgarg2002@gmail.com

Rohit Vijaykumar Kangoni
Dept. of Metallurgy and Materials Science
Indian Institute of Technology, Bombay
Mumbai, India
Roll No.: 210110094
rohitkan247@gmail.com

I. Introduction

Financial institutions and keep a massive volume of consumer documents. These documents are crucial to the company's operations, and there are statutory obligations on how they must be processed and stored. To identify the documents and extract data from them, a significant amount of human processing is required. The motivation behind this project was to cut short this human processing through automation.

This is the report for our solution "FileClassifierEngine" that performs generic identification, classification and splits different documents from uploaded files into different document type classes.

II. Methodology

Our solution can be broken down to three parts-

1) To handle different input file types.
2) To extract the different documents present in an input file as images.
3) To extract text from each extracted document image and identify document class.

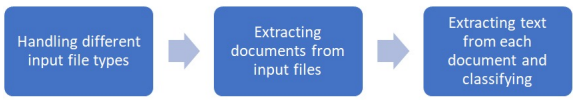

Fig. 1. Pipeline for Classifier Engine

III. Libraries Used

img2pdf to convert a .png and .jpg file to a .pdf file, so that it can be processed more easiy. **PyMuPDF** to process PDFs and extract images of documents from them. **pytesseract** and **tesseract-ocr** to extract the text in a image and use that information to classify the document. Also made use of **Pandas** to build out the final dataframe containing all the extracted information.

IV. Solution

A. Handling different Input File Types

The first task in this problem statement was to take a file as input from the user. We achieved this by using Google Colab's "upload.files()" functionality, that directly allows a user to upload a file from his local disc. It is then saved on the Google Colab notebook's virtual disc and allows the notebook to work with it. Once the file has been saved, it's name is copied and stored so that it can be stored in the dataframe. The name of the input file is stored in the dataframe as it will work as a reference for the origin of a particular document.
Since this input document is user defined, one problem we encounter is that the file type would not be standardised, i.e. different users would upload different file types. In order to ease processing, we converted *.png* and *.jpg* files, using the **img2pdf** library to *.pdf* files.

B. Splitting File for different Documents

By the end of the first part of our solution, we are left with a single PDF file that contains different user documents that are of use to the bank. We now need to separate out each of these documents and store them as different images before we can classify them and extract data from them.To establish this functionality, we made use of the **PyMuPDF** library.
On iterating over all the pages in the PDF, and using the *get_images()* function, a list of all images present in the PDF file is generated. The images are then labelled with their page number and serial number on said page and stored on the local disc of the Google Colab notebook.

C. Identifying Documents and Extracting Data

We now iterate over all images present in a particular PDF file and make use of the **tesseract-ocr** library to extract text from the images.
The OCR in **tesseract-ocr** stands for Optical Character Recognition. On iterating over every pixel of the image, the module extracts the text. We then make use of said extracted text and search for key identifier words for every document class and tag it by the document name.
Then all this information, the input file name, document image name, document type and data extracted from the image, are stored in a dataframe *df*. Even this information is entered for

every image at the end of every iteration using the *pandas.loc* function.

V. Road Ahead

What we have in mind for our FileCLassifierEngine is to convert it into a full stack project. Create a robust environment with all the libraries and packages used in the engine, the backend or the core working of the engine is what we have ready with us, followed by building a suitable user interface for our application and completing the front end.

References

https://www.geeksforgeeks.org/python-convert-image-to-pdf-using-img2pdf-module/
https://datatofish.com/images-to-pdf-python/
https://stackoverflow.com/questions/2693820/extract-images-from-pdf-without-resampling-in-python
https://towardsdatascience.com/5-quick-tips-to-manage-your-strings-in-python-4e7e48f72560
https://stackoverflow.com/questions/30053329/elegant-way-to-create-empty-pandas-dataframe-with-nan-of-type-float
https://www.geeksforgeeks.org/how-to-extract-images-from-pdf-in-python/
https://www.geeksforgeeks.org/how-to-extract-text-from-images-with-python/
https://stackoverflow.com/questions/2693820/extract-images-from-pdf-without-resampling-in-python
IEEE Conference Template (1)