Title - Text detection using Python

By - Zaid Ahmed Khan

Introduction

This report provides an overview of the complete code implementation present in the Jupyter Notebook. The main tasks include converting PDF pages to images, extracting text from these images, processing the extracted text, and saving the final structured data. The code is well-structured to handle the intricacies of Optical Character Recognition (OCR) and data cleaning, making it suitable for large-scale document processing tasks..

Converting PDF Pages to Images

The first step in the process involves converting PDF pages into images using the `pdf2image` library. This is a crucial step as it prepares the document for Optical Character Recognition (OCR).

- **Process**: The PDF is split into individual pages, and each page is converted into a high-resolution image. This step is critical as it transforms the document into a format suitable for Optical Character Recognition (OCR).
- **Outcome**: The output is a list of image files, one for each page of the PDF.

Text Extraction from Images

The next step involves extracting text from the images using OCR, specifically with the Tesseract engine.

Process: The images generated in the previous step are processed with Tesseract to extract textual information. The extracted text is stored in a list.

Outcome: The text extracted from each image is now available for further processing, enabling analysis and manipulation.

Processing Extracted Text

After extracting text, the next task is to parse and clean the data. This involves identifying relevant information, such as voter details, and organizing it into a structured format.

Process: The raw text is parsed to extract specific details like voter names, relative names, gender, age, and EPIC numbers. This involves splitting the text into blocks, cleaning the data, and structuring it into a DataFrame.

Outcome: A structured dataset is obtained, which can be saved or further manipulated as needed.

Epic No. Calculation

Cropping Epic No. part in each block and applying the text detection algorithm on that crop image to get better output of Epic No.

Watermark Detection Algorithm

Finally, the watermark detection algorithm is implemented as previously discussed. This step involves estimating the watermark from a set of images using median filtering and Poisson reconstruction.

Methodology -

A watermarked image `J` is obtained by imposing a watermark `W` over an unwatermarked image `I` with a blend factor α. Specifically, we have the following equation:

$$J(p) = \alpha(p)W(p) + (1 - \alpha(p))I(p)$$

Where `p = (x, y)` is the pixel location. For a set of `K` images, we have:

$$J_k = \alpha W + (1 - \alpha)I_k, \quad k = 1, 2....K$$

Although we have a lot of unknown quantities (J_k, W, α), we can make use of the structural properties of the image to determine its location and estimate its structure. Using these basic ideology we can detect the watermark in a image.

Saving the Final Output

Finally, the processed data is saved to an Excel file, making it accessible for reporting or further analysis.

Process: The structured data is saved in an Excel format.

Outcome: The final dataset is stored in an easily accessible format, ready for use.
