# PDF OCR Extraction Tool

This Python script extracts text from a PDF file using Optical Character Recognition (OCR) and saves the results as both a Word document and an Excel file. It processes a specified number of pages, converts them to images, applies OCR, cleans the extracted text, and organizes the output into structured formats.

## Features

* Converts PDF pages to high-resolution images (300 DPI) using PyMuPDF.
* Performs OCR on images using EasyOCR, supporting English and Bengali languages.
* Cleans extracted text by merging short lines, removing Arabic text, and eliminating empty lines.
* Splits text into columns (Title and Description) based on delimiters (: or -).
* Saves extracted text to a Word document (Output.docx) with page headings.
* Saves structured data to an Excel file (Output.xlsx) with page numbers and categorized columns.

## Prerequisites

To run this script, ensure you have Python installed along with the following dependencies: - PyMuPDF (for PDF processing) - easyocr (for OCR) - python-docx (for Word document creation) - pandas (for Excel output) - tqdm (for progress bars) - regex (for text cleaning)

Install the dependencies using pip:

pip install PyMuPDF easyocr python-docx pandas tqdm

Additionally, ensure you have a PDF file named book.pdf in the same directory as the script or update the INPUT\_PDF path in the configuration.

## Configuration

The script uses the following configuration variables, defined at the top of the script: - INPUT\_PDF: Path to the input PDF file (default: book.pdf). - OUTPUT\_DIR: Directory to store output files (default: output). - PAGES\_TO\_CONVERT: Number of pages to process (default: 20).

Modify these variables as needed before running the script.

## Usage

1. Place the input PDF file (book.pdf) in the script’s directory or update the INPUT\_PDF path.
2. Run the script:

* python main.py

1. The script will:
   * Create an output directory with subfolders images, word, and excel.
   * Convert the specified number of PDF pages to PNG images.
   * Perform OCR on the images to extract text.
   * Clean and process the text, removing Arabic lines and merging short lines.
   * Save the extracted text to output/word/Output.docx.
   * Save structured data to output/excel/Output.xlsx.

## Output

* **Images**: PNG files for each processed page are saved in output/images/ (e.g., page\_01.png, page\_02.png, etc.).
* **Word Document**: A file named Output.docx in output/word/ contains the extracted text, organized by page with bold headings.
* **Excel File**: A file named Output.xlsx in output/excel/ contains a table with columns:
  + PageNumber: The page number from the PDF.
  + Title: The text before a delimiter (: or -) or the full paragraph if no delimiter is present.
  + Description: The text after a delimiter, if present.
  + FullText: The complete paragraph text.

## Text Cleaning

The script includes a clean\_text function that: - Removes empty lines. - Skips lines containing Arabic text (based on Unicode ranges). - Merges short lines (less than 50 characters) to form coherent paragraphs. - Strips extra whitespace.

The split\_columns function organizes text into Title and Description columns if a delimiter (: or -) is present, facilitating structured output in the Excel file.

## Notes

* The script processes only the first PAGES\_TO\_CONVERT pages of the PDF or the total number of pages if fewer are available.
* EasyOCR is configured for English and Bengali (['bn', 'en']). Modify the easyocr.Reader languages if needed.
* Arabic text is filtered out using a regular expression to avoid irrelevant content. Adjust the arabic\_re pattern for other languages if required.
* Ensure sufficient disk space for image files, as high-resolution PNGs can be large.
* The script assumes the input PDF is readable and not encrypted.

## Example

For a PDF named book.pdf with 20 pages: 1. The script converts pages 1–20 to PNG images in output/images/. 2. OCR extracts text, which is cleaned and organized into paragraphs. 3. A Word document (Output.docx) is created with page-wise text. 4. An Excel file (Output.xlsx) is created with structured data, e.g.:

| PageNumber | Title | Description | FullText |
| --- | --- | --- | --- |
| 1 | Chapter 1 | Introduction to… | Chapter 1: Introduction to… |
| 1 | Section 1.1 | Overview of… | Section 1.1 - Overview of… |

## Troubleshooting

* **PDF not found**: Ensure the INPUT\_PDF path is correct and the file exists.
* **OCR errors**: Verify that EasyOCR is properly installed and the language models (bn, en) are downloaded.
* **Memory issues**: Reduce PAGES\_TO\_CONVERT or lower the DPI in page.get\_pixmap(dpi=300) for large PDFs.
* **Output issues**: Check write permissions for the OUTPUT\_DIR and its subfolders.

For further assistance, refer to the documentation of the used libraries or contact the script maintainer.

## Supporting Resources

- PyMuPDF Documentation: https://pymupdf.readthedocs.io/

- EasyOCR GitHub: https://github.com/JaidedAI/EasyOCR

- python-docx Documentation: https://python-docx.readthedocs.io/

- pandas Documentation: https://pandas.pydata.org/docs/

- tqdm GitHub: https://github.com/tqdm/tqdm

Note: This script is not currently hosted in a live-hosted link. For local use, ensure all dependencies are installed and the input PDF is correctly configured.