## **Overview**

OCR technology is useful for processing scanned documents and images, but it is extremely challenging to process engineering documents. A dedicated area needs to be designated to extract the required text and counts such as regularization need to be used after OCR to adjust the accuracy and format of the data. Extracting accurate text and numbers from tables, info boxes and technical information is important to automate processes and get a highly accurate result. Unfortunately, there is no high-quality open-source solution that implements the entire process.

## **Methodology**

图示, 工程绘图

描述已自动生成

The area of the information part in all PDF documents is fixed, which means that OCR can be performed by specifying the area to avoid interference from information from other parts.

### **1.3 OCR Library Test and Evaluation**

In our OCR library evaluation, we compared three popular OCR tools: PytesseractOCR, EasyOCR, KerasOCR and PaddleOCR, highlighting each tool's unique approach to extracting text from images.

PytesseractOCR: PytesseractOCR is a widely used OCR tool that excels at recognizing text in images. When extracting text from a rectangular object using PytesseractOCR, we first identify the rectangular box surrounding the text area in the image. During the subsequent text extraction process, we take appropriate segmentation measures to maintain the structure of the text lines. Text extraction is performed by using the pytesseract.image\_to\_string(roi) function, where Roi represents the region of interest, which is the detected text area in the rectangle. We can leverage its ability to recognize text to infer the structure of the table. By analyzing the layout and position of the text, we can infer the rows and columns of the table and further predict the position and boundaries of the cells. PytesseractOCR can effectively identify the boundaries and positions of text lines and extract the text content.

EasyOCR: EasyOCR is another powerful OCR tool known for its simplicity and efficiency in extracting text from images. EasyOCR uses an end-to-end deep learning model and can process text in a variety of languages and fonts. By simply calling the function easyocr.Reader(), we can load the EasyOCR model and extract text from the image. EasyOCR's flexibility and accuracy make it ideal for handling a variety of OCR tasks.

KerasOCR: KerasOCR is an OCR tool built on the Keras framework and has simple and powerful features. It uses advanced deep learning technology to accurately recognize text in images. By using KerasOCR’s pre-trained model, we can easily extract text from images. KerasOCR's model flexibility and customizability make it ideal for specific OCR needs.

PaddleOCR: PaddleOCR is an open-source OCR tool developed by Baidu and is based on the PaddlePaddle deep learning platform. The tool is specifically optimized to recognize Chinese and many other languages, making it perform well in multilingual environments. By calling PaddleOCR’s API, users can easily load models and extract text from images. PaddleOCR's efficiency and accuracy are due to the continuous optimization and updating of its back-end deep learning model, which can handle many text recognition tasks quickly and accurately. In addition, PaddleOCR supports customized training and fine-tuning so that it can adapt to specific business needs and application scenarios.

For the task of table extraction, PytesseractOCR utilizes the following models:

1. **Text Area Detection (Tesseract Engine):** The Tesseract engine is adept at detecting text areas within an image, accurately outlining the boundaries and positions of these areas. This is essential for identifying individual lines and blocks of text which form the basis of tables.
2. **Text Line Recognition (Tesseract OCR):** Once text areas are detected, the Tesseract OCR algorithm is used to recognize and transcribe the text lines within these areas. This step ensures the precise extraction of textual content from the image.
3. **Table Structure Recognition (Custom Algorithms):** PytesseractOCR can be integrated with custom algorithms designed to recognize the layout and structure of tables. By analyzing the arrangement and alignment of text, it predicts the rows, columns, and cell boundaries within the table, facilitating effective structured data extraction.

These components allow PytesseractOCR to excel in table detection and extraction, offering a robust solution for extracting structured information from various images.

### 1.4 Result Analysis

Using Keras OCR and PaddleOCR has proven difficult. Keras OCR's environment setup is complex because Keras has been integrated into TensorFlow. However, TensorFlow's compatibility issues pose a problem. For example, after TensorFlow 2.10, GPU acceleration on Windows is only possible through WSL2. PaddleOCR relies on the PaddlePaddle framework, which has complicated dependencies and is relatively niche. The official documentation often lags behind the latest versions, and many submitted issues remain unresolved for extended periods. Given these difficulties, we will exclude Keras OCR and PaddleOCR from consideration for extracting content from documents.

There are no complex formats in the Table area and Notes area, so there is no difference in extraction between several OCR tools. The Rev Table section has shown a simple table format, and the differences between OCR tools have been reflected:

表格

描述已自动生成

The above figure shows the results between Easy OCR and Pytesseract OCR. It can be found that Pytesseract OCR has more accurate extraction, and the formatting of the extracted content is consistent with the table.

Then check the extraction results of the complex format Status table:

手机屏幕截图

描述已自动生成

Upon analyzing the results from both PytesseractOCR and EasyOCR, it's evident that PytesseractOCR excels in maintaining the integrity of the original layout and structure of complex documents, thanks to its sophisticated OCR engine. This superiority is rooted in the specific components and algorithms it employs.

PytesseractOCR leverages the Tesseract engine, which combines traditional computer vision methods and machine learning to enhance text recognition. A key feature is its Adaptive Recognition technology, which dynamically adjusts to the content it processes, improving accuracy across diverse layouts and text densities. Additionally, it uses the Connected Component Analysis (CCA) algorithm to identify text components based on their connectivity, crucial for maintaining the document's original structure. This algorithm excels at mapping spatial relationships and alignments, essential for reproducing complex layouts accurately. PytesseractOCR also incorporates an advanced layout analysis system that segments text into lines, blocks, and paragraphs. This system employs a heuristic approach based on the geometric attributes of text components, aiding in preserving the document’s original structure. These capabilities make PytesseractOCR particularly effective for documents where the format and structure are integral, ensuring high fidelity in layout preservation.

In contrast, EasyOCR, while proficient in text detection and recognition across various languages and fonts, may not focus as extensively on preserving the exact spatial and formatting nuances of the source material. This can result in less fidelity in the representation of the original document layout, especially in complex structured documents.

### 1.5 Challenge

Optical Character Recognition (OCR) enables us to identify text within images or tables. However, the challenge lies in organizing this text into meaningful context. OCR can read individual characters, but it does not inherently understand how these characters form words, sentences, or the structure of a document. This is why the more specialized field is called "Natural Language Processing" rather than "Natural Language Understanding." Incorrect organization of extracted text can make the automation of information extraction difficult, especially when the text extraction has flaws, such as confusing I with 1, or O with 0. This leads to regularization rules tailored to one PDF failing on others.

A more general approach is to train a specialized model using many engineering drawings. Such a model would be more accurate and applicable for extracting information from engineering drawings compared to generic OCR tools and would not struggle with special symbols.

However, we do not have enough engineering drawings, nor do we have sufficient computational resources to train this model. An open-source tool worth investigating and using is eDOCr. This tool is based on Keras-OCR and is designed for recognizing mechanical engineering drawings. However, as previously mentioned, Keras-OCR has its issues, which are further compounded in eDOCr that is based on Keras-OCR.