Internship Report

Arya G

PES1UG22EE012

Mentor – Prabindh Sundareson

Duration – 14th August 2024 – 30th October 2024

Goal of this Project

The goal of this project is to develop a robust and scalable **OCR (Optical Character Recognition) system for Indian languages**, with a particular focus on **Kannada, Tamil, Telugu**, and other Indic scripts. The system must provide high accuracy for recognizing text in various **fonts, sizes, and layouts** while supporting the extraction and reconstruction of **tables with multiple rows and columns**. By leveraging modern computing resources and cloud infrastructure, the tool aims to automate the digitization of printed books, converting them into accessible digital formats to serve visually impaired individuals and enhance accessibility for libraries and institutions working in this space.

Repository - https://github.com/prabindh/ctsoc-stdu/tree/main/TableExtraction-main/TableExtraction-main

https://colab.research.google.com/#fileId=https%3A//huggingface.co/microsoft/Florence-2-large/blob/main/sample_inference.ipynb

https://drive.google.com/open?id=15uO5LejwIb50gCCLrHz3_ldq0edZGTH3

Overview

The TableExtraction-main repository is designed for detecting and extracting tables from scanned documents or images. It employs deep learning models, such as YOLO or Detectron2, for accurately identifying table boundaries, coupled with OCR engines like Tesseract to extract text from table cells. The tool processes input images to identify table structures, reconstructs rows, columns, and cells, and exports the content in machine-readable formats like CSV or JSON. It is particularly useful for automating data extraction from documents such as invoices, research papers, and reports. The repository uses Python libraries like OpenCV and Pandas for image processing and data management. While it excels in detecting and parsing simple tables, it may face challenges with complex layouts, such as nested tables or merged cells. Additionally, it lacks native support for Indic languages like Kannada, Tamil, or Telugu. To enhance its utility, future improvements could include Indic script OCR integration, support for complex table structures, and scalability for large datasets through cloud deployment. This tool has significant potential for document digitization and accessibility applications.

Learnings from repository

1. **Table Detection**:
    - The repository uses a model for detecting table structures in scanned images or PDFs, likely based on deep learning frameworks such as **Detectron2** or **YOLO**.
    - Bounding box detection for tables is effective for identifying table regions within complex document layouts.
2. **Table Structure Recognition**:
    - Once a table is detected, the repository extracts rows, columns, and cells, ensuring semantic relationships (e.g., headers, data).
    - The system uses post-processing techniques to refine table structures and align content properly.
3. **Image Preprocessing**:
    - Preprocessing steps, such as resizing, denoising, and thresholding, are likely used to improve the quality of input images for table detection.
    - Handling noise and skew corrections ensures better accuracy in real-world scenarios with imperfect scans.
4. **Integration with OCR**:
    - The repository integrates table detection with OCR engines, extracting text from table cells and associating it with the detected table structure.
    - Tools like **Tesseract** may be used for text recognition within table cells.
5. **Output Formats**:
    - The repository converts extracted tables into structured formats such as **CSV, JSON**, or other tabular representations, facilitating further use in analysis or document reconstruction.

Potential Steps for the future

Enhancing Table Detection and Recognition

- **Expand Dataset**:
    - Train the table detection model on datasets that include tables in Indian languages (Kannada, Tamil, Telugu, etc.) with diverse layouts.
    - Collect samples from real-world sources like textbooks, research papers, and government documents to improve robustness.
- **Advanced Models**:
    - Upgrade the table detection model using Transformer-based architectures like **DETR** (Deformable DETR) or **TableNet** for higher accuracy in complex layouts.

    - Explore **Graph Neural Networks (GNNs)** to model the relationships between cells and improve table structure understanding.

2. Indic Language Support

- **Text Extraction**:
    - Integrate or fine-tune OCR models (e.g., Tesseract, EasyOCR, or Google Vision) for Indic scripts, ensuring accurate recognition of multilingual table content.
    - Address challenges such as recognizing ligatures, diacritics, and varying fonts common in Indian languages.
- **Language Context**:
    - Use **NLP models** (e.g., IndicBERT or BERT-based models) to correct OCR errors based on the language context, improving text accuracy.

3. Table Semantics and Nested Structures

- **Semantic Relationships**:
    - Enhance the system to identify table headers, sub-headers, and hierarchies for better semantic understanding of table content.
    - Support complex table structures with merged cells, spanning multiple rows/columns.
- **Nested Tables**:
    - Add support for nested tables (tables within tables), which are common in financial and legal documents.

4. Scalability and Performance

- **Cloud Deployment**:
    - Deploy the system on cloud platforms like AWS, Azure, or Google Cloud to handle large-scale table extraction tasks.
    - Use serverless architectures or distributed processing frameworks to improve scalability.
- **Batch Processing**:
    - Enable batch processing of scanned books or documents, allowing simultaneous extraction of tables from multiple files.

5. User Interface and Ease of Use

- **Interactive UI**:
    - Develop a web-based interface where users can upload scanned documents, visualize detected tables, and edit results manually if needed.
    - Provide real-time feedback on detected table structures and extracted content.
- **Customizable Outputs**:

    - Allow users to choose output formats, such as ePub (for accessibility), LaTeX (for publications), or structured spreadsheets.

6. Expanding Beyond Tables

- **Document Layout Analysis**:
    - Integrate with broader layout analysis tools to detect other components like figures, captions, and side notes in addition to tables.
    - Enable full document reconstruction with tables as a part of the overall layout.

7. Incorporating AI for Error Correction

- **Post-OCR Correction**:
    - Implement AI-based error correction for tables where OCR inaccuracies or alignment issues occur.
    - Use contextual learning models to predict missing or distorted characters in table cells.

8. Accessibility for Visually Impaired Users

- **Accessible Formats**:
    - Extend output formats to ePub with embedded semantic markup, enabling screen readers to correctly interpret table structures.
    - Ensure tables are tagged with appropriate accessibility metadata.
- **Voice Feedback**:
    - Integrate voice-based table summaries for blind users, describing table content interactively.

REFERENCES

1. https://huggingface.co/microsoft/Florence-2-large

2. https://github.com/Sghosh1999/Computer-Vision-OCR-Florence2.

3. Bin Xiao, Haiping Wu, Weijian Xu, et al. "Florence-2: Advancing a

Unified Representation for a Variety of Vision Tasks." *arXiv preprint*

arXiv:2311.06242 (2023).https://doi.org/10.48550/arXiv.2311.06242

4. https://github.com/JaidedAI/EasyOCR

5. Bhatt, B. (n.d.). **EasyOCR Demo**. GitHub.

https://github.com/bhattbhavesh91/easyocr-demo?tab=Apache-2.0-1-o

v-file

6. PaddleOCR Documentation

7. https://github.com/prabindh/ctsoc-stdu/tree/main/TableExtraction-main/TableExtraction-main