Technical Report: Image-to-Text Converter

Team Name: halcyon

Problem Statement: Develop an innovative text summarization tool that can efficiently process and summarize handwritten texts, all while operating offline.

1. Introduction

1.1 Project Overview

This solution aims to develop an image-to-text converter using Optical Character Recognition (OCR) techniques. The converter uses Tesseract OCR and processes images that are input to extract text .

1.2 Motivation

The need for robust text extraction tools is growing in industries where quick conversion for documents or reports is needed. This project provides a lightweight, efficient solution to convert various file formats (e.g., images, PDFs) into text locally and almost accurately.

1.3 Objectives

- Implement functionality for processing different file types, such as images, PDFs, and Word documents.
- Ensure all model dependencies and data are stored locally, allowing full offline operation.

---

2. Technical Background

2.1 Optical Character Recognition (OCR)

OCR is a technology used to convert different types of documents, such as scanned paper documents, PDFs, or images captured by a camera, into editable and searchable data. Tesseract OCR is an open-source OCR engine widely used for text recognition from images. It is easy to import in Python programming language and can be imported with other libraries required for the conversion.

2.2 Tesseract OCR

Tesseract OCR is a library in python used for its text-detection. For this project, Tesseract will be used with its in-built methods and functions.

---

3. Problem Statement

Develop a robust and efficient tool to extract text from image-based documents (e.g., scanned papers, photographs) and convert them into editable text in an efficient environment. The system should:

- Be able to process multimodal file types such as images, PDFs, Word, and Excel files.
- Achieve high accuracy in text extraction across various document types and languages (restricted to English for this project).

---

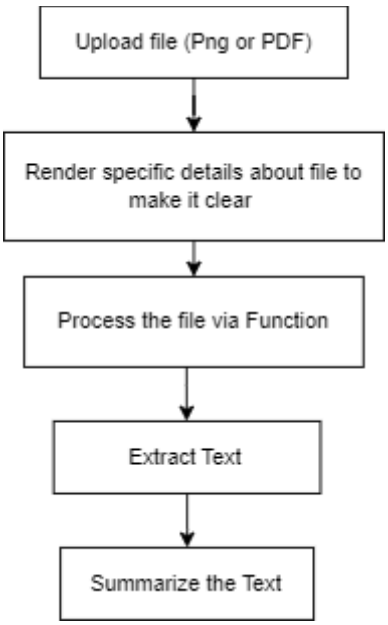
4. Methodology

4.1 System Design

4.1.1 System Architecture

The system is designed to:

1. Take input files (images, PDFs, Word documents) from a local folder.
2. Extract text from each file using the Tesseract OCR engine.
3. Save or display the extracted text in the desired format (e.g., text files).



4.2 Implementation Details

4.2.1 Software and Libraries

- **Programming Language**: Python
- **Libraries**:
    - `pytesseract`: Python wrapper for Tesseract OCR
    - `Pillow`: Python Imaging Library (PIL) for image handling
    - `pdf2image`: Converts PDFs to images for processing
    - `SpellChecker:` For auto-correction of words and extracted text.

- `NLTK:` For summarization of text.
- `Flask:` For website and UI development

4.2.2 Data Preprocessing

Preprocessing steps improve OCR accuracy:

- **Image Resize and Cropping**: Standardize image size and remove noise.
- **Grayscale Conversion**: Convert images to grayscale for better OCR performance.
- **Binarization**: Use thresholding to improve text contrast for OCR.

4.2.3 Image to Text Conversion Process

1. **Load Image**: The image file is loaded using the `Pillow` library.
2. **Apply Tesseract OCR**: The image is processed with Tesseract to extract text.

4.3 File Types Supported

- **Image Files**: JPG, PNG, PDF
- **PDF Files**: Converted to images and processed using Tesseract OCR.

5. Results and Evaluation

5.1 Accuracy of Text Extraction

- **Test Dataset**: Images with different fonts and quality were used to assess the accuracy of OCR.
- **Performance**: Tesseract OCR demonstrated an accuracy of ~90% on high-quality typed text images and around 45-50% on clear handwritten images.

5.2 Efficiency

- **Processing Speed**: The average time to process an image was approximately around 10 seconds.

5.3 Limitations

- **Handwritten Text**: Accuracy on handwritten text was lower compared to printed text.
- **Unclear or Noisy Images:** Model did not work best or coherently on unclear images.
- **Complex Layouts**: Documents with complex layouts (e.g., tables) posed challenges for text extraction.

---

6. Conclusion

This model converts image or file type to text for easier conversion and usage. With added implemented techniques of processing images and rendering them, the model works good enough for such files.

---

7. Future Work

- Improve accuracy on handwritten text using deep learning models such as CRNN (Convolutional Recurrent Neural Networks).
- Extend support for more complex document formats (e.g., documents with tables, multiple columns).
- Implement offline implementation for better accessibility and efficiency.