ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY

COMPUTER VISION AND IMAGE PROCESSING

OCR Application Documentation

GROUP MEMBERS

1.ABDUSELAM JBRIL UGR/23323/13

2.DAGIM BEKELE UGR/23590/13

3.ABEL LEGESE UGR/22736/13

4.BELAYNEH SEWAREG UGR/23940/13

SUBMITTED DATE

DEC/30/2024

SUBMITTED TO Mr.

TAGEL ABONEH

Contents

1 Introduction

The OCR (Optical Character Recognition) Application is designed to transform images containing text into editable text formats. This application leverages web technologies to provide a user-friendly interface for capturing images via a device's camera or uploading images from local storage. The back-end utilizes Flask, OpenCV, and Tesseract OCR for image processing and text extraction.

2 Features

- **Capture Image**: Users can take photos directly from their device's camera.
- **Upload Image**: Users can select image files from their local storage.
- **Image Adjustment**: Brightness and contrast settings can be adjusted to enhance image quality before processing.
- **OCR Processing**: Extracts text from images using Tesseract OCR.
- **User-Friendly Interface**: Responsive design with dark mode support.
- **Help and About Sections**: Provides users with guidance on using the application and information about the contributors.

3 Technologies Used

- **Frontend**:
    - HTML5
    - CSS3
    - JavaScript
- **Backend**:
    - Flask (Python Framework)
- **Image Processing**:
    - OpenCV (for image manipulation)
- **OCR**:
    - Tesseract OCR (for text recognition)
- **Others**:
    - NumPy (for numerical operations)

4 Installation

To set up the OCR application locally, follow these steps:

Prerequisites

- Python 3.x installed on your machine.
- Tesseract OCR installed. Follow Tesseract Installation Guide for your operating system.

Steps

1. **Clone the Repository**:

```bash
git clone https://github.com/yourusername/ocr-application.git
cd ocr-application
```

2. **Set Up a Virtual Environment** (optional but recommended):

```bash
python -m venv venv
source venv/bin/activate  # On Windows use `venv\Scripts\activate`
```

3. **Install Dependencies**:
   Create a requirements.txt file and include:

```plaintext
Flask==2.0.3
opencv-python==4.5.3.20210926
pytesseract==0.3.8
numpy==1.21.2
```

Then run:

```bash
pip install -r requirements.txt
```

4. **Configure Tesseract Path**:
   Modify the path in app.py to point to your Tesseract installation:

```python
pytesseract.pytesseract.tesseract_cmd = r"C:\Program Files\Tesseract-OCR\tessera
```

5. **Run the Application**:

```bash
python app.py
```

5 Configuration

Tesseract Configuration

Ensure that Tesseract OCR is properly installed and configured. You can test the installation by running:

```bash
tesseract --version
```

Flask Configuration

In app.py, you can modify the Flask application settings, such as enabling debug mode or changing the host and port:

```python
app.run(debug=True, host='0.0.0.0', port=5000)
```

6 Usage Guide

User Interface

1. **Home Screen**: Users are greeted with options to capture an image or upload one.
2. **Capture Image**: Click the "Capture Image" button to take a photo.
3. **Upload Image**: Use the "Upload Image" button to select a file.
4. **Adjust Image**: Use sliders to adjust brightness and contrast.
5. **Process OCR**: Click "Process OCR" to extract text.
6. **View Output**: The extracted text appears below the buttons.

Dark Mode

Users can toggle dark mode for better visibility in low-light environments by clicking the "Toggle Dark Mode" button.

Help and About Sections

Click on the "Help" or "About" buttons to view more information about the application and the team behind it.

7 Architecture Overview

Architecture Diagram

Include a diagram illustrating the architecture of the application, showing the interaction between the frontend, backend, and OCR processing.

Components

- **Frontend**: Built using HTML, CSS, and JavaScript for user interactions.
- **Backend**: Flask handles requests and serves the web pages.
- **OCR Processing**: Images are processed using OpenCV and Tesseract, where text extraction occurs.

Data Flow

1. User captures or uploads an image.
2. The image is sent to the Flask backend.
3. The backend processes the image using OpenCV and Tesseract.
4. The extracted text is sent back to the frontend for display.

8 Error Handling

Client-Side Errors

- Input validation for file uploads (e.g., checking file types).
- Display error messages if the camera cannot be accessed.

Server-Side Errors

- Handle cases where no image data is provided.
- Return appropriate error messages if OCR processing fails.

Logging

Implement logging for both frontend and backend to help track errors and application flow. Use Python's built-in logging module in the backend.

9 Contributing

Contributions are welcome! If you would like to contribute to the OCR application, please follow these steps:

1. Fork the repository.
2. Create a new branch for your feature or bug fix.
3. Make your changes and commit them with descriptive messages.

4. Push the branch to your forked repository.
5. Submit a pull request for review.

Code of Conduct

Please adhere to the project's code of conduct during your contributions.

10 License and Contact Information

License

This project is licensed under the MIT License. See the LICENSE file for details.

Contact Information

For questions or feedback, please reach out to:

- **Your Name**: bekeledagim3@gmail.com
- **GitHub**: https://github.com/dagimbe/Optical-Character-Recognition.git