Project Report

Ritik Verma

September, 2024

—

OCR Image Text Extraction and Keyword Search Application

—

Internship at PARIMAL IIT Roorkee

INTRODUCTION

This project involves developing a web application using Streamlit that allows users to upload images and extract text using Optical Character Recognition (OCR). The application also enables users to search for specific keywords in the extracted text and highlights these keywords for better visibility.

FEATURES AND TECHNOLOGY STACK

FEATURES

1. **Image Upload**: Users can upload images in JPEG or PNG formats.
2. **Text Extraction**: The application uses Tesseract OCR to extract text from the uploaded images.
3. **Keyword Search**: Users can input keywords, and the application will highlight these keywords in the extracted text.
4. **User-Friendly Interface**: Built with Streamlit for a seamless user experience.

TECHNOLOGY STACK

- **Python**: Programming language used for backend development.
- **Streamlit**: A Python library to create web applications quickly and easily.
- **Pytesseract**: Python wrapper for Google's Tesseract-OCR Engine.
- **OpenCV**: A library for computer vision tasks.
- **Pillow (PIL)**: Python Imaging Library to handle image processing.
- **NumPy**: Library for numerical operations on arrays.

Code Overview

The main components of the code are as follows:

1. Importing Libraries:

```python
OCR_Project.py
import streamlit as st
import pytesseract
from PIL import Image
import cv2
import numpy as np
```

2. Text Extraction Function: This function converts the uploaded image to grayscale and applies Tesseract OCR to extract text:

```python
def extract_text(image):
    # Convert the image to grayscale
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    # Use pytesseract to do OCR on the images
    text = pytesseract.image_to_string(gray)
    return text
```

3. Keyword Highlighting Function: This function highlights specified keywords in the extracted text using HTML <mark> tags:

```python
# Function to highlight keywords in text
def highlight_keywords(text, keywords):
    highlighted_text = text
    for keyword in keywords:
        if keyword:
            highlighted_text = highlighted_text.replace(keyword, f"<mark>{keyword}</mark>")
    return highlighted_text
```

4. Streamlit App Structure: The main interface allows users to upload images, displays the extracted text, and takes input for keywords:

```python
# Streamlit app
st.title("OCR Image Text Extraction and Keyword Search")

# Upload image file
uploaded_file = st.file_uploader("Choose an image...", type=["jpg", "jpeg", "png"])
```

5. Other Complete Code:

OCR_Project.py

OCR_Project.py > extract_text

```python

# Streamlit app
st.title("OCR Image Text Extraction and Keyword Search")

# Upload image file
uploaded_file = st.file_uploader("Choose an image...", type=["jpg", "jpeg", "png"])

if uploaded_file is not None:
    # Open and display the image
    image = Image.open(uploaded_file)
    st.image(image, caption='Uploaded Image.', use_column_width=True)

    # Convert image to a format suitable for OCR
    image_cv = np.array(image)

    # Extract text from the image
    extracted_text = extract_text(image_cv)

    # Display extracted text
    st.subheader("Extracted Text:")
    st.text_area("Text", extracted_text, height=300)

    # Input for keywords to search
    keywords_input = st.text_input("Enter keywords to search (comma-separated):")
    keywords = [kw.strip() for kw in keywords_input.split(",") if kw.strip()]

    # Highlight keywords in the extracted text
    if keywords:
        highlighted_text = highlight_keywords(extracted_text, keywords)
        st.subheader("Search Results:")
        st.markdown(highlighted_text, unsafe_allow_html=True)
```

Installation Requirements

Streamlit: To run the application, the following requirements must be installed-

requirements.txt

OCR_Project / requirements.txt

ritikverma123 Update requirements.txt

Code Blame 5 lines (5 loc) · 96 Bytes Code 55% faster with GitHub Copilot

```
streamlit==1.38.0
pytesseract==0.3.13
pillow==10.4.0
opencv-python==4.10.0.84
numpy==2.1.1
```

packages.txt

OCR_Project / packages.txt

ritikverma123 Add files via upload

Code Blame 6 lines (6 loc) · 69 Bytes Code 55% faster with GitHub Copilot

```
libgl1
libglib2.0-0
libsm6
libxext6
libxrender-dev
tesseract-ocr
```

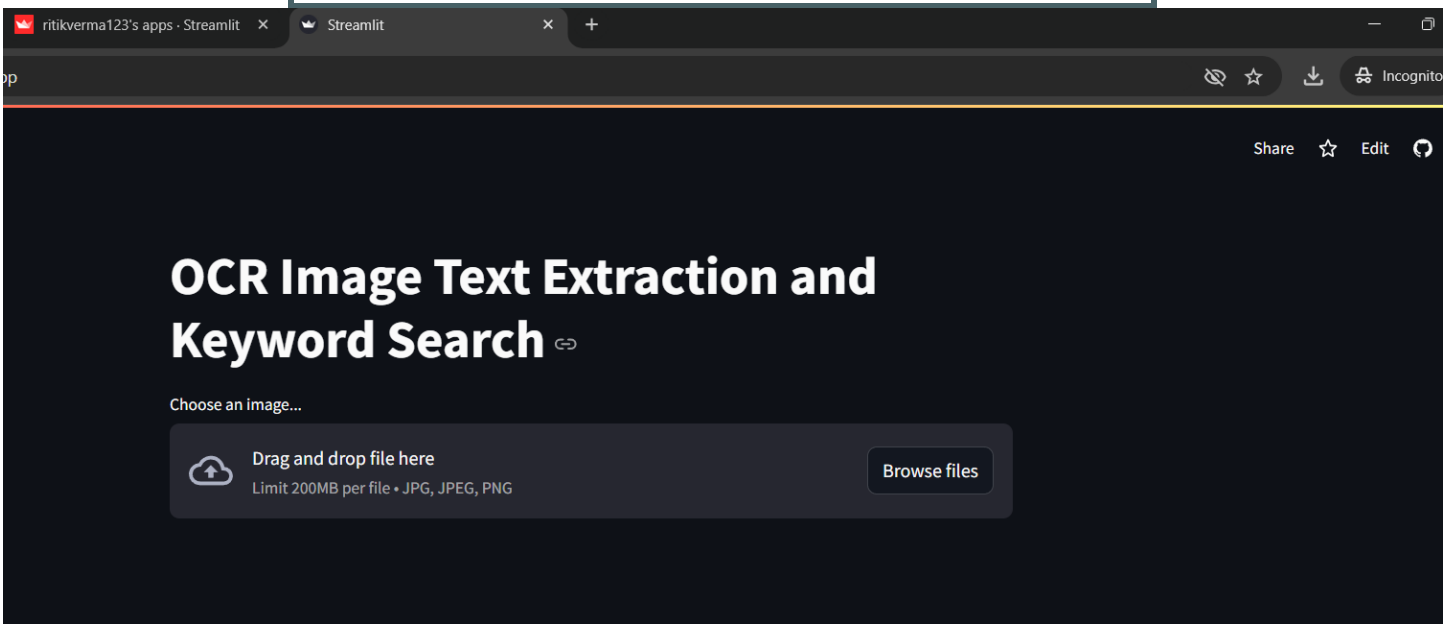
USER INSTRUCTION

- **Upload an Image:** Click on the "Choose an image..." button to upload an image file (JPEG or PNG).
- **View Extracted Text:** After uploading, the extracted text will be displayed in a text area.
- **Search for Keywords:** Enter keywords in the input box (comma-separated) and click enter. The application will highlight the keywords in the displayed text.

CONCLUSION

This project demonstrates the effective use of Python and various libraries to create an interactive web application for OCR text extraction and keyword search. The user-friendly interface and robust functionality make it a useful tool for anyone needing to process and analyze text from images.

Output Screenshots



Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG

Browse files

Imagetxt.png 190.2KB

Task 3: Deployment

1. **Deploy the Web Application**:
   - Deploy the web application on platforms like Hugging Faces, Streamlit Sharing, or any other suitable platform.
   - Ensure the application is accessible via a public URL.

Deliverables

1. **Code Submission**:
   - Python scripts for the web application, including the OCR processing and search functionality.
   - A README file explaining how to set up the environment, run the web application locally, and details about the deployment process.
2. **Live Web Application**:
   - The live URL of the deployed web application where the OCR and search functionalities can be tested.
3. **Extracted Text and Search Output**:
   - JSON or plain text output of the extracted text from the uploaded image.
   - Demonstration of the search functionality with example keywords.

Evaluation Criteria

- **Accuracy**: How well the OCR model extracts text from both Hindi and English sections of the image.
- **Functionality**: The web application should correctly handle image uploads, extract text, and allow keyword searches.

Extracted Text:

Text

Task 3: Deployment

1. Deploy the Web Appli :
o Deploy the web application on platforms like Hugging Faces, Streamlit Sharing,
or any other suitable platform.
© Ensure the application is accessible via a public URL.

Deliverables

1. Code Submission:
o Python scripts for the web application, including the OCR processing and search

Enter keywords to search (comma-separated):

Submission

Enter keywords to search (comma-separated):

Submission

Search Results:

Task 3: Deployment

1. Deploy the Web Appli : o Deploy the web application on platforms like Hugging Faces, Streamlit Sharing, or any other suitable platform. © Ensure the application is accessible via a public URL.

Deliverables

1. Code Submission : o Python scripts for the web application, including the OCR processing and search functionality. o AREADME file explaining how to set up the environment, run the web application locally, and details about the deployment process.
2. Live Web Application: o The live URL of the deployed web application where the OCR and search functionalities can be tested.
3. Extracted Text and Search Output: o JSON or plain text output of the extracted text from the uploaded image. o Demonstration of the search functionality with example keywords.

Evaluation Criteria

e Accuracy: How well the OCR model extracts text from both Hindi and English sections of the image.

e Functionality: The web application should correctly handle image uploads, extract text, and allow keyword searches.

e User Interface: The web interface should be simple, intuitive, and functional.