Savitribai Phule Pune University
Modern Education Society's College of Engineering, Pune
19, Bund Garden, V.K. Joag Path, Pune – 411001.

ACCREDITED BY NAAC WITH "A" GRADE (CGPA – 3.13)

DEPARTMENT OF COMPUTER ENGINEERING

A REPORT

ON

Using Face Detection and OCR for help in Crime Investigation

B.E. (COMPUTER)

SUBMITTED BY

Mr. Yogen Ghodke (71818291L)
Mr. Sudesh Pawar (71818502B)
Miss. Shravani Kanade (71818357G)

UNDER THE GUIDANCE OF

Dr. Jayshree R. Pansare

(Academic Year: 2020-2021)

Savitribai Phule Pune University
Modern Education Society's College of Engineering, Pune
19, Bund Garden, V.K. Joag Path, Pune – 411001.

ACCREDITED BY NAAC WITH "A" GRADE (CGPA – 3.13)

DEPARTMENT OF COMPUTER ENGINEERING

Certificate

This is to certify that project entitled

Using Face Detection & OCR for help in Crime Investigation

has been completed by

Mr. Yogen Ghodke (PRN. 71818291L)
Mr. Sudesh Pawar (PRN. 71818502B)
Miss. Shravani Kanade (PRN. 71818357G)

of BE COMP I in the Semester - II of academic year 2020-2021 in partial fulfillment of the Fourth Year of Bachelor degree in "Computer Engineering" as prescribed by the Savitribai Phule Pune University.

Dr. J. R. Pansare
Project Guide

(Dr.(Mrs.) N. F. Shaikh)
H.O.D

Place: MESCOE, Pune.
Date: /05/2021

ACKNOWLEDGEMENT

It gives us great pleasure and satisfaction in presenting this mini project on "Using Face Detection and OCR for help in Crime Investigation".

We would like to express my deep sense of gratitude towards all faculty for their support and advice during the development.

We have furthermore to thank Computer Department HOD **Dr.(Mrs.) N. F. Shaikh** *and our Project Guide* **Dr. Jayshree Pansare** *to encourage us to go ahead and for continuous guidance. Her incisive and objective guidance and timely advice encouraged us with a constant flow of energy to continue the work. Finally, we must say that no height is ever achieved without some sacrifices made at some end and it us here we owe our special debt to our parents and our friends for showing their generous love and care throughout the entire period.*

We would like to thank all those, who have directly or indirectly helped us for the completion of the work during this mini project.

Yogen Ghodke (71818291L)
Sudesh Pawar (71818502B)
Shravani Kanade (71818357G)
B.E. Computer

Contents

List of Figures

Abstract

Many a times, a formal investigation of an ongoing criminal case involve going through archives of old newspapers looking for information on relevant news articles on the case.

Doing this work manually can be quite a hassle and a very time consuming task as sometimes the exact date, the article was printed is sometimes not known and manually looking for relevant info in newspapers over a span of a couple of months can be almost impossible.

This project aims to use the Optical Character Recognition of the Python Tesseract library and Facial Recognition OpenCV modules to search for certain keywords (such as names of people, location, etc.) in the scanned digital archives of the newspaper and return the pages and the faces detected in the photographs printed on that page for faster skimming of relevant news.

Keywords-*OCR, Face Detection, OpenCV, Newspaper, Tesseract, Haar Cascade*

Chapter 1

INTRODUCTION

1.1 Introduction

Many a times, a formal investigation of an ongoing criminal case involve going through archives of old newspapers looking for information on relevant news articles on the case.Doing this work manually can be quite a hassle and a very time consuming task and manually looking for relevant info in newspapers over a span of a couple of months can be almost impossible.

This project aims to use the Optical Character Recognition of the Python libraries and modules to search for certain keywords (such as names of people, location, etc.) in the scanned digital archives of the newspaper and return the pages and the faces detected in the photographs printed on that page for faster skimming of relevant news.

1.2 Motivation

Many a times, a formal investigation of an ongoing criminal case involve going through archives of old newspapers looking for information on relevant news articles on the case.
Doing this work manually can be quite a hassle and a very time consuming task as sometimes the exact date, the article was printed is sometimes not known and manually looking for relevant info in newspapers over a span of a couple of months can be almost impossible.
Therefore, we designed a tool to search for certain keywords (such as names of people, location, etc.) in the scanned digital archives of the newspaper and return the pages and the faces detected in the photographs printed on that page for faster skimming of relevant news.

Chapter 2

PROBLEM STATEMENT

2.1 Problem Statement

This project uses Python Tesseract and OpenCV modules to search for certain keywords in a newspaper and return the Faces detected in the Photographs on that particular page.

2.2 Objectives

1. To aid Investigative Agencies in Investigation.
2. Save Time by replacing manual search methods.
3. Help digitally archive news clippings.
4. Reduce Manual Errors in identification.
5. Use Python Tools available to develop a robust system.
6. Emphasis on avoiding biased treatment towards any newspaper sources.

Chapter 3

PROJECT REQUIREMENTS

3.1 Packages Used

1. Python Imaging Library (PIL)
2. OpenCV
3. PyTesseract
4. ZipFile
5. iPython
6. Miniconda

3.2 Dataset

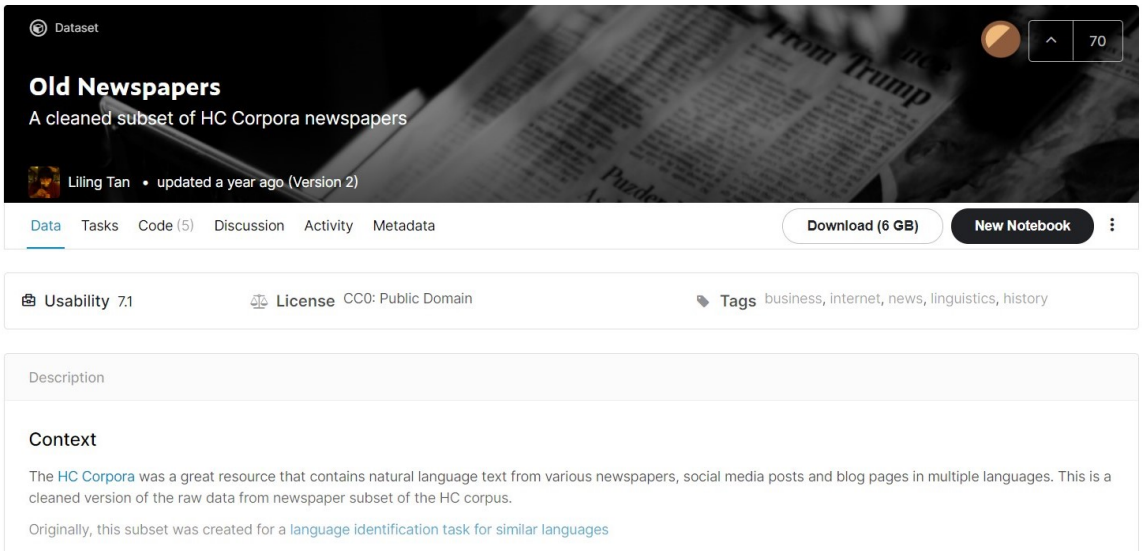

Figure 3.1: Kaggle Page

A dataset containing old newspaper clippings is downloaded and curated using Python tools. The Dataset contains varied news clippings of various media houses like The Sun, New York Times, Washington Post, USA Today etc.

These Newspaper clippings are not necessarily very well defined and therefore we use Python Pre-processing for further enhancement on these articles to create our final dataset for usage.

3.2.1 Dataset Details

1. No. of Newspapers : 60
2. Size : 800 MB
3. No. of Images in Newspaper: Approx 1160
4. Site of Origin: Kaggle
5. Concentrated on: North American Region

3.2.2 Technical Workflow

1. OCR returns output in String
2. String is converted into JSON

3.3 Tech Stack

3.3.1 Pillow

Figure 3.2: Pillow

The Python Imaging Library adds image processing capabilities to your Python interpreter. This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities.

The core image library is designed for fast access to data stored in a few basic pixel formats. It should provide a solid foundation for a general image processing tool.

3.3.2 OpenCV

Figure 3.3: OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.

The library has more than 2500 optimized algorithms. These algorithms can be used to detect & recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models, stitch images together to produce images, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc.

3.3.3 PyTesseract

Figure 3.4: PyTesseract

Python-tesseract is an optical character recognition (OCR) tool for python. That is, it will recognize and "read" the text embedded in images.

Python-tesseract is a wrapper for Google's Tesseract-OCR Engine. It is also useful as a stand-alone invocation script to tesseract, as it can read all image types supported by the Pillow and Leptonica imaging libraries, including jpeg, png, gif, bmp, tiff, and others. Additionally, if used as a script, Python-tesseract will print the recognized text instead of writing it to a file.

3.3.4 Zipfile

Figure 3.5: Zipfile

The ZIP file format is a common archive and compression standard. This module provides tools to create, read, write, append, and list a ZIP file. Any advanced use of this module will require an understanding of the format.

This module does not currently handle multi-disk ZIP files. It can handle ZIP files that use the ZIP64 extensions. It supports decryption of encrypted files in ZIP archives, but it currently cannot create an encrypted file. Decryption is extremely slow as it is implemented in native Python rather than C.

Chapter 4

IMPLEMENTATION OF PROJECT

4.1 Modules description

4.1.1 OCR Module

The first module takes the newspaper as input and then performs OCR on the newspaper using PyTesseract to find the text in the article , further processing the article to find keywords which will be passed as input to the second module for detecting the faces in it.

4.1.2 Face Detection Module

This is the second module that takes the input string which is the name of the suspect. Then this name is searched in the list generated by the OCR Module and then uses OpenCV to perform Face Detection on the newspaper images and returns all the matches on this string.

4.2 Development Approach

1. Open the zip file (through iteration). There's a lot of ways to do this but you should explore and look through documentation (such as https://docs.python.org/3/library/zipfile.html).
2. Open the image to see what you're working with. This really helps clarify what our code should do and we had to use what we learned from past weeks to do this.
3. We used Pytesseract to convert the text into strings so we can sort through each png that have X name. There are some lectures on how to do this.
4. After converting to NumPy, we used the face haar cascade thing up above to find faces and return coordinates. With these coordinates, we cropped out new images that we could now use (this was the hardest thing for us and it required a lot of experimentation).
5. Using a contact sheet similar to the one in the W1 project, I pasted in all the images found after converting them to PIL images.

4.3 Screen shots of Project

```
images = []
with zipfile.ZipFile("small_img.zip","r") as zip_ref:
    for file in zip_ref.infolist():
        unzip = zip_ref.open(file)
        newimage = {}
        newimage['filename'] = file.filename
        newimage['image'] = Image.open(unzip)
        newimage['image'].save(file.filename)
        newimage['facesimages'] = []
        images.append(newimage)
print(len(images),"images were scanned.")
```

Figure 4.1: Scanning Module

```
for image in images:
    cvimage = cv.imread(image['filename'])
    gray = cv.cvtColor(cvimage, cv.COLOR_BGR2GRAY)
    image['faces'] = face_cascade.detectMultiScale(gray,1.5)

for image in images:
    for face in image['faces']:
        foundface = image['image'].crop((face[0],face[1],face[0]+face[2],face[1]+face[3]))
        foundface = foundface.resize((250,250))
        image['facesimages'].append(foundface)
```

Figure 4.2: Face Processing Module

```
def find_faces(faces):
    import math
    from PIL import ImageDraw
    contact_sheet= Image.new('RGB', (1250,math.ceil(len(faces)/5.0)*250))
    x=0
    y=0
    d = ImageDraw.Draw(contact_sheet)
    for face in faces:
        contact_sheet.paste(face, (x, y) )
        if x+face.width >= contact_sheet.width:
            x=0
            y=y+face.height
        else:
            x=x+face.width
    contact_sheet = contact_sheet.resize((int(contact_sheet.width/2),int(contact_sheet.height/2) ))
    return contact_sheet
```

Figure 4.3: Face Detection Module

```
def find_Person(name):
    for image in images:
        if (not (name in image['text'].split())):
            continue
        print('Results found in file {}'.format(image['filename']))
        if(image['facesimages'] == None or len(image['facesimages']) == 0):
            print('Results found in file {}'.format(image['filename']) + 'But there were no faces in that file!')
        else:
            display(find_faces(image['facesimages']))

find_Person('Mark')
```

Figure 4.4: Find Person Module

Chapter 5

CONCLUSION

Thus, the proposed system is able to detect the unknown faces by training the OCR model with the help of detection algorithm. It performs various preprocessing steps like Grayscale conversion, rescaling, feature extraction via usage of Haar Cascade, min max scaling and normalization and thus predicts the probable output with reasonable confidence.

The accuracy obtained by the system is 91%.

.