utf-8”project

May 4, 2019

1 The Project

1. This is a project with minimal scaffolding. Expect to use the the discussion forums to gain insights! It’s not cheating to ask others for opinions or perspectives!
2. Be inquisitive, try out new things.
3. Use the previous modules for insights into how to complete the functions! You’ll have to combine Pillow, OpenCV, and Pytesseract
4. There are 4 functions you need to complete to have a working project. These functions are described using the RE formula, which stands for Requires and Effects. Each function will have its own RE located directly above the function definition. The Requires section describes what is needed in the function argument (inbetween the function definition parenthesis). The Effects portion outlines what the function is supposed to do.
5. There are hints provided in Coursera, feel free to explore the hints if needed. Each hint provide progressively more details on how to solve the issue. This project is intended to be comprehensive and difficult if you do it without the hints.

1.0.1 The Assignment

Take a ZIP file of images and process them, using a library built into python that you need to learn how to use. A ZIP file takes several different files and compresses them, thus saving space, into one single file. The files in the ZIP file we provide are newspaper images (like you saw in week 3). Your task is to write python code which allows one to search through the images looking for the occurrences of keywords and faces. E.g. if you search for "pizza" it will return a contact sheet of all of the faces which were located on the newspaper page which mentions "pizza". This will test your ability to learn a new (library), your ability to use OpenCV to detect faces, your ability to use tesseract to do optical character recognition, and your ability to use PIL to composite images together into contact sheets.

Each page of the newspapers is saved as a single PNG image in a file called images.zip. These newspapers are in english, and contain a variety of stories, advertisements and images. Note: This file is fairly large (~200 MB) and may take some time to work with, I would encourage you to use small_img.zip for testing.

Here’s an example of the output expected. Using the small_img.zip file, if I search for the string "Christopher" I should see the following image: If I were to use the images.zip file and search for "Mark" I should see the following image (note that there are times when there are no faces on a page, but a word is found!):

Note: That big file can take some time to process - for me it took nearly ten minutes! Use the small one for testing.

[1]:
```
import zipfile
import PIL
from PIL import Image
from PIL import ImageFont
from PIL import ImageDraw
import pytesseract
import cv2 as cv
import numpy as np

# loading the face detection classifier
portraitpx = 100
textpx = 20
border = int(textpx / 2)
width = 500 + border * 2

def prepare_dict(zip,name):
    pages = []
    with zipfile.ZipFile(zip) as archive:
        for pic in archive.namelist():
            files.append(pic)
        for file in files:
            with archive.open(file) as myfile:
                image = Image.open(myfile)
                scans[file] = [image.convert('RGB')]
                scans[file].append(pytesseract.image_to_string(scans[file][0]))

        for file in files:
            if name in scans[file][1]:
                pages.append(file)
        return pages


def get_faces(file):
    face_cascade = cv.CascadeClassifier('readonly/
 ↪haarcascade_frontalface_default.xml')
    open_cv_image = np.array(scans[file][0])
    img = open_cv_image[:, :, ::-1].copy()
    gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
    for x, y, w, h in faces:
        box = (x, y, x+w, y+h)
        tmpimg = scans[file][0].copy()
        tmpimg = tmpimg.crop(box=box)
        scans[file].append(tmpimg)


def calc_rows(file):
```

```
    rows = (len(scans[file]) - 2) // 5
    if (len(scans[file]) - 2) % 5 != 0:
        rows += 1
    return rows

def calc_no_faces(file):
    if len(scans[file]) == 2:
        return textpx
    else:
        return 0
```

[2]:
```
#Create ContactHigh
def create_contact_sheet():
    rows = 0
    totalrows = 0
    no_faces = 0
    total_no_faces = 0
    x = border
    y = 0
    for file in pages:
        rows = calc_rows(file)
        totalrows += rows
        no_faces = calc_no_faces(file)
        total_no_faces += no_faces
    fnt = ImageFont.truetype("readonly/fanwood-webfont.ttf", size = textpx)
    background = PIL.Image.new('RGB', (portraitpx * 5, portraitpx), color =␣
 ↪"black")
    contact_sheet = PIL.Image.new('RGB', (width, (total_no_faces + portraitpx *␣
 ↪totalrows + (textpx + 2 * border) * len(pages) + 2 * border)), color =␣
 ↪"white")

    for file in pages:
        y += border
        d = ImageDraw.Draw(contact_sheet)
        d.text((x, y), "Results found in file {}".format(file), font=fnt,␣
 ↪fill=(0, 0, 0, 128))
        if len(scans[file]) == 2:
            y += textpx + 5
            d.text((x, y), "But there where no faces in that file!", font=fnt,␣
 ↪fill=(0, 0, 0, 128))
        y += (textpx + border)
        if len(scans[file]) > 2:
            for face in scans[file][2:]:
                face.thumbnail((portraitpx, portraitpx))
                if x + portraitpx > contact_sheet.width:
                    x = border
                    y += portraitpx
```

```
                if x == border:
                    contact_sheet.paste(background, (x, y))
                contact_sheet.paste(face, (x, y))
                x += portraitpx
            x = border
            y += portraitpx

    display(contact_sheet)
```

[3]:
```
# Search "Christopher"
scans = {}
files = []
pages = prepare_dict("readonly/small_img.zip","Christopher")
for page in pages:
    get_faces(page)
create_contact_sheet()

#Search "Mark"
scans = {}
files = []
pages = prepare_dict("readonly/images.zip","Mark")
for page in pages:
    get_faces(page)
create_contact_sheet()
```

Results found in file a-0.png

Results found in file a-3.png

Results found in file a-0.png

Results found in file a-1.png

Results found in file a-10.png
But there where no faces in that file!

Results found in file a-13.png

Results found in file a-2.png

Results found in file a-3.png

Results found in file a-8.png
But there where no faces in that file!

[ ]: