project

December 13, 2019

In [10]:
```
#
# Course Project - Python Project: pillow, tesseract, and opencv
# Bob Kozdemba, 12/12/2019
#
# Use pillow, tesseract, and opencv to perform optical character
# and facial recognition from a collection of scanned newspaper pages.
#

import zipfile
import pytesseract
import PIL
from PIL import Image
from PIL import ImageFile
import pytesseract
import cv2 as cv
import numpy as np
import math

# Load the face detection classifier
face_cascade = cv.CascadeClassifier('readonly/haarcascade_frontalface_default.xml')

#
# Global data structure for storing results.
#
# Example:
# globalData = [{'filename': string, pilImage': object, 'text' : string, 'faces': [] }
#
globalData = []

def getSmallestFace(imageDict):
    """
    Find the dimensions of the face (a tuple) with the smallest width.

    :param imageDict: An image dictionary from global storage.
    :returns: The tuple with the smallest face width.
    """
    sortedList = sorted(imageDict['faces'], key = lambda x: x[2])
```

```
    smallestDim = (sortedList[0][2], sortedList[0][3])
    return (sortedList[0][2], sortedList[0][3])

def getFacesFromImage(pil_image):
    """
    Find the faces in an image.

    :param imageDict: The PIL image to process.
    :returns: A list of faces found.
    """
    open_cv_image = np.array(pil_image)
    # Convert RGB to BGR
    open_cv_image = open_cv_image[:, :, ::-1].copy()
    gray = cv.cvtColor(open_cv_image, cv.COLOR_BGR2GRAY)
    # This function returns a list of objects as rectangles.
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.35, minNeighbors=4)
    return faces

def getImageFromZip(zfile, filename):
        """
        Extract a PIL compatible image from a zip archive.

        :param zfile: The path to a zip archive (string).
        :param filename: Image filename within the zip archive (string).
        :returns: A `~PIL.Image.Image` object.
        """
        myzip = zipfile.ZipFile(zfile, 'r')
        fp = myzip.read(filename)
        p = ImageFile.Parser()
        p.feed(fp)
        img = p.close()
        return img

def getTextFromImage(image):
    """
    Return a string of text from an image.

    :param image: PIL compatible image
    :return text(string)
    """

    image = image.convert('L')
    image = image.convert('1')
    text = pytesseract.image_to_string(image)
    return text

def displayFaces(i):
        """
```

```
        Display the faces in this image.

        :param imageDict: An image dictionary from global storage.
        :returns: Nothing
        """
        cellDims = getSmallestFace(i)
        cellDims = (192, 192)
        totalCells = len(i['faces'])
        cellsPerRow = 5
        sheetDims = (cellsPerRow * cellDims[0], math.ceil(totalCells / cellsPerRow) *

        contact_sheet=PIL.Image.new(i['pilImage'].mode, sheetDims)

        x = 0
        y = 0

        for rec in i['faces']:
            c = (rec[0], rec[1], rec[0] + rec[2], rec[1] + rec[3])
            faceImage = i['pilImage'].crop(c)
            r = faceImage.resize(cellDims)
            contact_sheet.paste(r, (x, y) )

            if x + cellDims[0] == contact_sheet.width:
                x = 0
                y = y + cellDims[1]
            else:
                x = x + cellDims[0]

        # Resize and display the contact sheet
        contact_sheet = contact_sheet.resize((int(contact_sheet.width/2),int(contact_s
        display(contact_sheet)

def processImageSet(zipArchive, globalDataStructure):
    """
    This function searches each image in the zipfile
    for text and faces. The results are stored as dictionaries
    in the globalDataStructure.

    :param zipArchive: The path to a zip archive (string).
    :param globalDataStructire: The globalDataStructure object.
    """
    #for file in ['a-0.png']:  # For testing a single image.
    for file in zipfile.ZipFile(zipArchive, 'r').namelist():
        d = {}
        d['filename'] = file
        d['pilImage'] = getImageFromZip(zipArchive, file)
        print('Running OCR on file:', file)
        d['text'] = getTextFromImage(d['pilImage'])
```

```
            print('Running FCC on file:', file)
            d['faces'] = getFacesFromImage(d['pilImage'])
            globalDataStructure.append(d)

    def searchAndDisplay(searchTerms, globalDataStructure):
        """
        This function searches for text and displays faces from global storage.

        :param searchTerms: A list of strings to search for.
        :param globalDataStructire: The globalDataStructure object.
        """

        for i in globalDataStructure:
            for search in searchTerms:
                if search in i['text']:
                    print('\nResults found in file {}'.format(i['filename']))

                    if len(i['faces']) > 0:
                        displayFaces(i)
                    else:
                        print('But sorry, there were no faces in that file!')


    #
    # Begin main
    #
    print('This will take several minutes ...')
    zipArchive = 'readonly/small_img.zip'
    searchTerms = ['Christopher']
    print('Started processing {} image archive.'.format(zipArchive))
    processImageSet(zipArchive, globalData)
    print('Finished processing {} image archive.'.format(zipArchive))
    print('\nSearching for {} in {}'.format(searchTerms, zipArchive))
    searchAndDisplay(searchTerms, globalData)

    zipArchive = 'readonly/images.zip'
    searchTerms = ['Mark']
    del globalData
    globalData = []
    print('Started processing {} image archive.'.format(zipArchive))
    processImageSet(zipArchive, globalData)
    print('Finished processing {} image archive.'.format(zipArchive))
    print('\nSearching for {} in {}'.format(searchTerms, zipArchive))
    searchAndDisplay(searchTerms, globalData)
```

```
This will take several minutes ...
Started processing readonly/small_img.zip image archive.
Running OCR on file: a-0.png
```

```
Running FCC on file: a-0.png
Running OCR on file: a-1.png
Running FCC on file: a-1.png
Running OCR on file: a-2.png
Running FCC on file: a-2.png
Running OCR on file: a-3.png
Running FCC on file: a-3.png
Finished processing readonly/small_img.zip image archive.

Searching for ['Christopher'] in readonly/small_img.zip

Results found in file a-0.png
```

```
Results found in file a-3.png
```

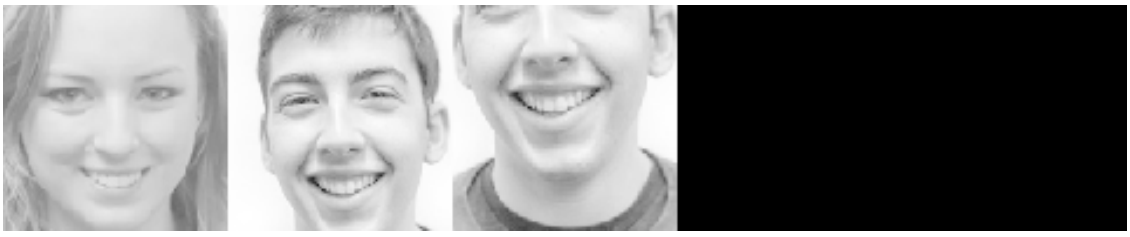
```
Started processing readonly/images.zip image archive.
Running OCR on file: a-0.png
Running FCC on file: a-0.png
Running OCR on file: a-1.png
```

```
Running FCC on file: a-1.png
Running OCR on file: a-10.png
Running FCC on file: a-10.png
Running OCR on file: a-11.png
Running FCC on file: a-11.png
Running OCR on file: a-12.png
Running FCC on file: a-12.png
Running OCR on file: a-13.png
Running FCC on file: a-13.png
Running OCR on file: a-2.png
Running FCC on file: a-2.png
Running OCR on file: a-3.png
Running FCC on file: a-3.png
Running OCR on file: a-4.png
Running FCC on file: a-4.png
Running OCR on file: a-5.png
Running FCC on file: a-5.png
Running OCR on file: a-6.png
Running FCC on file: a-6.png
Running OCR on file: a-7.png
Running FCC on file: a-7.png
Running OCR on file: a-8.png
Running FCC on file: a-8.png
Running OCR on file: a-9.png
Running FCC on file: a-9.png
Finished processing readonly/images.zip image archive.

Searching for ['Mark'] in readonly/images.zip

Results found in file a-0.png
```

```
Results found in file a-1.png
```

```
Results found in file a-10.png
But sorry, there were 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 sorry, there were no faces in that file!
```

In [ ]: