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: Christopher Search 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!): Mark Search

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

In [2]:

```
import zipfile


from PIL import Image

from PIL import ImageDraw

import pytesseract

import cv2 as cv

import numpy as np



# loading the face detection classifier

face_cascade = cv.CascadeClassifier('readonly/haarcascade_frontalface_default.xml')

######################

##### Functions ######

######################

def binarize(image_to_transform, threshold):

    output_image=image_to_transform.convert("L")

    for x in range(output_image.width):

        for y in range(output_image.height):

            # for the given pixel at w,h, lets check its value against the threshold

            if output_image.getpixel((x,y))< threshold: #note that the first parameter is actually a tuple object

                # lets set this to zero

                output_image.putpixel( (x,y), 0 )

            else:

                # otherwise lets set this to 255

                output_image.putpixel( (x,y), 255 )

    #now we just return the new image

    return output_image

# The newspaper texts should be extracted and saved for any future word search

# If a text file with the newspaper text is found omit the extraction

# But if there is no text file, proceed to extract text and save it to a txt file

def maketextdict(mizipo):

    Texts = {}

    for foto in mizipo.namelist():

        file = str(foto) +".txt"
```

```
        i = 0
        try:
            f = open(file, "r")
        except IOError:
            f = open(file, "a+")
            foto_text = Image.open(mizipo.extract(foto))
            BndW = binarize(foto_text, 80)
            texto = pytesseract.image_to_string(BndW)
            text = texto.replace("-\n", "")
            f.write(text)
            #f.close()
            i = 1
        finally:
            if i == 1:
                print(file + " has been saved")
            Texts[foto] = f.readlines()
            f.close()
    return Texts
#
def makeImagedict(what =[]):
    facesdict = {}
    for name in what:
        facesdict[name] = []
        faceslst = []
        n = 1
        cv_img=cv.imread(name)
        pil_img=Image.open(name)
        cv_img_bin=cv.cvtColor(cv_img,cv.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(cv_img_bin, 1.3, 5)
        drawing=ImageDraw.Draw(pil_img)
            # And plot all of the rectangles in faces
        for x,y,w,h in faces:
            n += 1
            picture = name[:len(name)-4] + str(n)
            faceslst.append(pil_img.crop((x, y, x+w, y+h)))
            facesdict[name] = faceslst
    return facesdict

def Contact_sheet(facesdict ={}):
    for name in facesdict.keys():
        if facesdict[name] == []:
            print("\nResults found in {}\nBut no faces were found in that file\n".format(name))
        else:
            print("Results found in {}".format(name))
            largest_x, largest_y = max(image.size for image in facesdict[name])
            count = len(facesdict[name])
            contact_sheet=Image.new('RGB', (largest_x*5, largest_y*(int(count/5) + (True))))
            x=0
            y=0
            # add images to contact sheet
            for img in facesdict[name]:
```

```
                x1, y1 = img.size
                if x1/largest_x < 1:
                    newsize = (largest_x, largest_y)
                    img = img.resize(newsize)
                contact_sheet.paste(img, (x, y) )
                # update position for next image:
                if x+largest_x == contact_sheet.width:
                    x=0
                    y=y+largest_y
                else:
                    x=x+largest_x

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



def search(values, searchFor):
    lista = []
    for k in values:
        for v in values[k]:
            if searchFor in v:
                lista.append(k)
            else:
                continue
    mylist = list(dict.fromkeys(lista))
    Contact_sheet(makeImagedict(mylist))
```

In [3]:

```
#
# Initialize the list of texts extracted from newspapers to search faces
#
mizip = zipfile.ZipFile("readonly/images.zip", mode='r')
Newspapers = maketextdict(mizip)
```

In [10]:

```
x
search(Newspapers, 'Mark')
```

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

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

```
Results found in a-10.png
But no faces were found in that file

Results found in a-13.png
```

```
Results found in a-2.png
```

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

```
Results found in a-8.png
But no faces were found in that file
```

In [4]:

```
search(Newspapers,"Christopher")
Results found in a-0.png
```

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

In [ ]: