Untitled

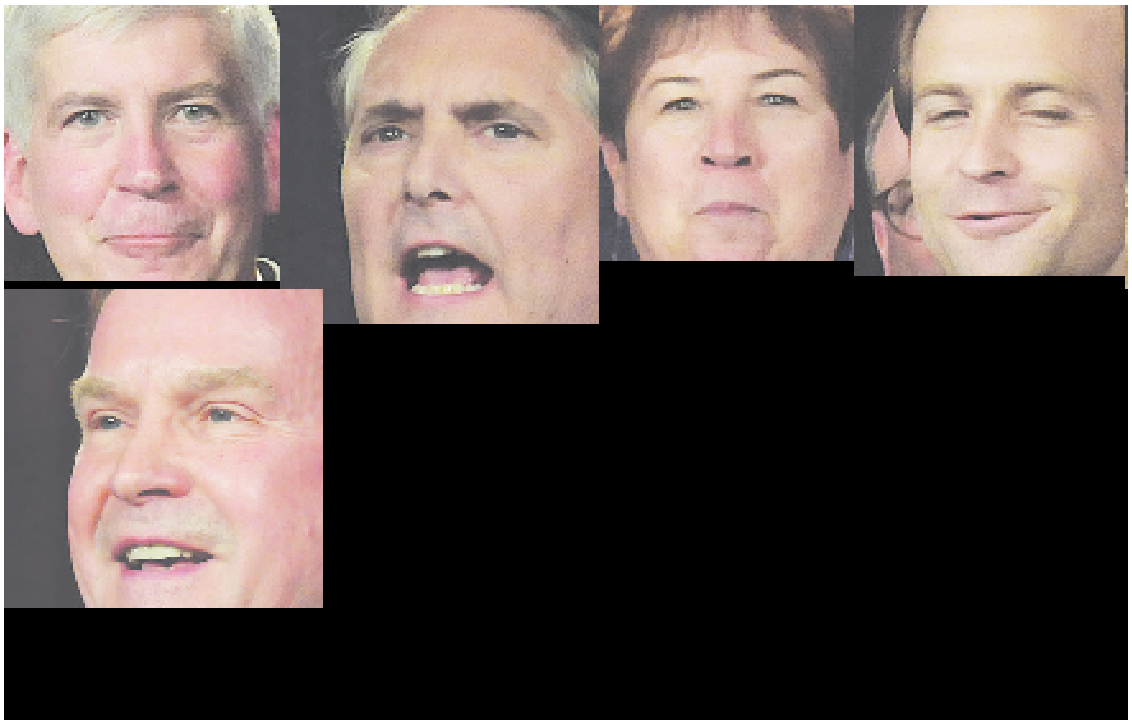
January 22, 2020

```
In [13]: import zipfile

         from PIL import Image
         import pytesseract
         import cv2 as cv
         import numpy as np

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

         # the rest is up to you!

In [14]: #Function that creates the zipfile object
         def create_zp():
             filer = input("Enter zipfile name: (ex: images.zip)")
             #The zipfile object and a list of tuples with a name and a zipinfo object for eve
             zip_file = zipfile.ZipFile("readonly/{}".format(filer))
             return zip_file

         #Function that creates a list with a dictionary for every image in the zipfile.
         #Each dictionary contains the file name, the PIL.image object, the text and the faces
         def create_diclst(zip_file):
             files_lst = zip(zip_file.namelist(), zip_file.infolist())
             #Loop that creates a list of dictionaries with every PIL.image object binarized w
             dic_lst = []
             i = 0
             for file_name, info_obj in files_lst:
                 dic_lst.append({})
                 pil_image = Image.open(zip_file.open(info_obj)).convert("1")
                 dic_lst[i]["name"] = file_name
                 dic_lst[i]["pil_image"] = pil_image
                 i += 1
             #loop that extracts the text from the images and add it to each dic in dic_lst
             for item in range(len(dic_lst)):
                 text = pytesseract.image_to_string(dic_lst[item]["pil_image"])
                 dic_lst[item]["text"] = text
             #Loops that adds the faces to each dic in dic_lst
             for image in range(len(zip_file.infolist())):
                 p_i = Image.open(zip_file.open(zip_file.infolist()[image])).convert("L")
                 p_i.save("file{}.png".format(image))
                 cv_img = cv.imread("file{}.png".format(image))
                 faces = face_cascade.detectMultiScale(cv_img, 1.3, 5)
                 dic_lst[image]["faces"] = []
                 pil_img=Image.open(zip_file.open(zip_file.infolist()[image]))
                 # Cut the faces
                 for x,y,w,h in faces:
                     cro = pil_img.crop((x,y,x+w,y+h))
                     dic_lst[image]["faces"].append(cro)

             return dic_lst
```

In [15]:
```
        #functions that gets the height of the contact sheet
        def get_cs_height(i, dic_lst):
            height = 0
            #Loop to get the highest height of the images in dic_lst[i]
            for img in dic_lst[i]["faces"]:
                if img.height > height:
                    height = img.height
                else:
                    continue
            #loop to guess how many times we have to add the img height to itself to get the
            width = 0
            for img in dic_lst[i]["faces"]:
                width += img.width
                if width + img.width >= 900:
                    width = 0
                    height += height
                else:
                    continue

            return height

        #Function that creates the contact sheet with the faces
        def create_cs(i, dic_lst):
            #for i in range(len(dic_lst)):
            height = get_cs_height(i, dic_lst)
            contact_sheet=Image.new(dic_lst[i]["faces"][0].mode, (880, height))
            x=0
            y=0
            for img in dic_lst[i]["faces"]:
                max_size = (250, 250)
                img.thumbnail(max_size)
                # Lets paste the current image into the contact sheet
                contact_sheet.paste(img, (x, y) )
                # Now we update our X position. If it is going to be the width of the image,
                # and update Y as well to point to the next "line" of the contact sheet.
                if x+img.width >= contact_sheet.width:
                    x=0
                    y=y+img.height
                else:
                    x=x+img.width

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

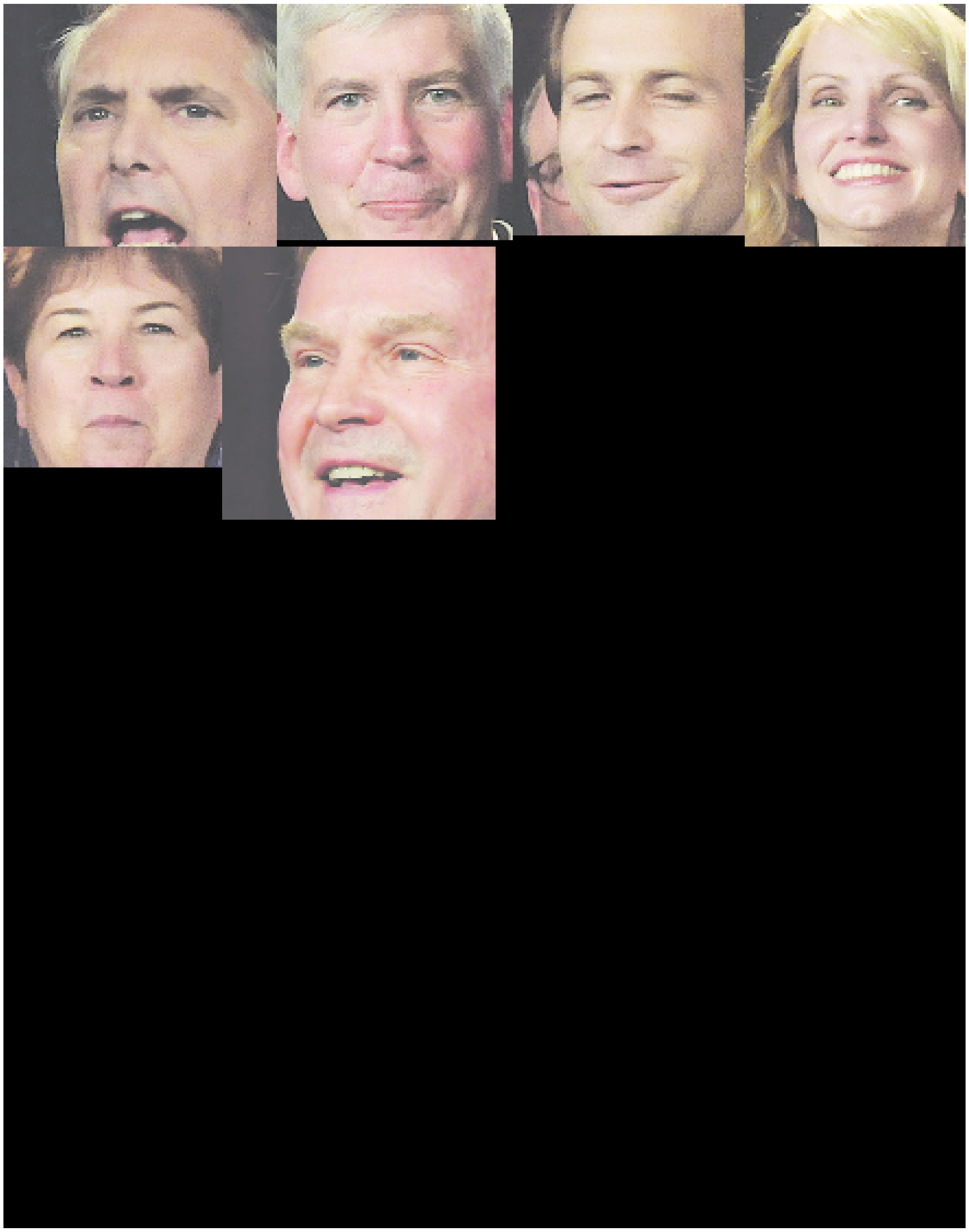
```
In [16]: def search(key):
            ''' Looks up the key(string) in the text of the images in the zip file,
            if the key is there it displays the faces that appears in that
            image in a contact sheet'''
            zip_file = create_zp()
            dic_lst = create_diclst(zip_file)
            for i in range(len(dic_lst)):
                if key in dic_lst[i]["text"]:
                    if len(dic_lst[i]["faces"]) > 0:
                        print("Results found in {}".format(dic_lst[i]["name"]))
                        create_cs(i, dic_lst)
                    else:
                        print("Results found in {}".format(dic_lst[i]["name"]))
                        print("But there were no faces.")
                else:
                    continue

        search("Chris")
        search("Mark")
```

```
Enter zipfile name: (ex: images.zip)small_img.zip
Results found in a-0.png
```



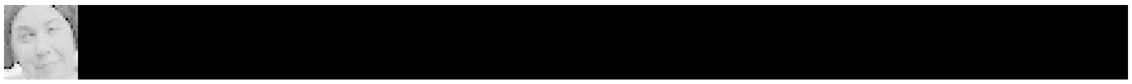
```
Results found in a-3.png
```

```
Enter zipfile name: (ex: images.zip)images.zip
Results found in a-0.png
```



Results found in a-1.png

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

```
In [ ]:
```