ECE 528: Cloud Computing

Assignment #5

Midterm Tutorial Execution: Deploying an API with AWS Lambda and AWS API Gateway

Michael Bowyer

March 31, 2021

1 Statement of the Problem

2 Description of Solution

```
usage: Assignment2.py [-h] [--student-list STUDENT_LIST] [--output-image-dir
    OUTPUT_IMAGE_DIR] [-fd]

Read in student list csv file, then plot all head shots with names. Optionally
    run facial detection.

optional arguments:
  -h, --help          show this help message and exit
  --student-list STUDENT_LIST
                      Path to student list
  --output-image-dir OUTPUT_IMAGE_DIR
                      Path to directory where to save output student matrix image
  -fd, --run-face-detection
                      Set when you want to run facial detection for each student
                          image
```

The main function of this program is shown in the below code snippet. Most lines are function calls which are defined in other created files under the lib folder, namely student_info and img_utils. The student_info library contains classes for storing and creating new students and the img_utils library is responsible for downloading, plotting, and running facial detection on images. Each line of the code snippet contains an important functionality of the program and is accompanied by a descriptive comment. Each step of the program is further described in the itemized list below the code snippet.

```
from lib import student_info
from lib import imgs_utils

#1. Grab argument of location of image, and master information csv
   args = parser.parse_args()
   studentListFilePath = args.student_list
   outputImgDir = args.output_image_dir
   if not os.path.exists(outputImgDir):
      os.makedirs(outputImgDir)

#2. Read in csv file
   studentInfos = student_info.Students(studentListFilePath)

#3. Download Images and update students image file path
```

```
    validStudents = imgs_utils.download_images_update_students(studentInfos,
        outputImgDir)

#4. Sort students based on last name
    validStudents.sort(key=lambda x: x.lastname)

#5. Display images
    imgs_utils.plot_all_students(validStudents)

#6. Run facial recognition detection
    if(args.run_face_detection):
        imgs_utils.runFacialDetection(validStudents)
```

Here is a further description of each step.

1. Arugment parsing reads the arguments passed to the program: source student list, output image directory, flag to run facial detection.

2. Student file reading reads in all student information from the input source student list and stores them in the variable studentInfos.

3. All student information from studentInfos is then passed to a created function called download_images_update_students(), which downloads all images from the studentInfo, then appends the local path to where the image was stored locally. All students which an image was sucessfully downloaded for are then returned and saved in validStudents.

4. The students are sorted in place by last name.

5. All student images are then combined into a single matrix, then this matrix is saved locally.

6. Finally, if the user wants to run facial detection, then the program will grab the image for the first student in the list, find and plot bounding boxes around any found faces. This image will be displayed in a popup. This popup is shown until the user of the program presses another key which indicates they want to look at the next image. When the user presses a key, the existing popup is replaced with the output of the

facial detection on the next students image. This repeats until all images are exhausted or the program is terminated.

3 Encountered Problems

There were a few problems which were encountered while developing this solution. These are briefly described in the following list along with the solution used:

1. Students had incorrect URLs - The solution used was to create a list of valid students, which only contained students who an image could be sucessfully downloaded for.

2. Student images had same file name - the solution for this problem was to check if there was a image already saved locally with the same name which the download image would be saved to. If there was an existing file name with the same name, then the newly downloaded image would be renamed to include the students name. This can be seen in some of the output section debug statements.

3. Facial detection did not always detect faces - The facial detection algorithm used is imperfect and is not able to detect faces in all images. The best solution found was to play with the detection parameters to try to make it work well for the dataset of student images.

4 Testing and Output

Below is the output log of the program when the user of the program cycled through three student images. There would be further debug statements with "faces found", but the program was terminated early for the sake of brevity. The output matrix of student images can be seen in figure **??** and a few example output images from the facial detection algorithm can be be seen in figure **??**.

```
python.exe Assignment2.py --student-list ./StudentInformation.csv
    --output-image-dir ./StudentImages --run-face-detection
INFO: Parsing input arguments
INFO: Input student information file: ./StudentInformation.csv
INFO: Saving downloaded images to: ./StudentImages
INFO: Reading in student information
INFO: Downloading student images locally
DEBUG: Image for Ahmed Zaki sucessfully Downloaded: meface.jpg
DEBUG: Image for Michael Bowyer sucessfully Downloaded:
    BowyerHeadshot_150x150.PNG
DEBUG: Image for Hamka Maya sucessfully Downloaded: me.png
DEBUG: Image for Ponkshe Aishwarya sucessfully Downloaded: AP.png
DEBUG: Image for Brown Patrick sucessfully Downloaded: SMALL.PORTRAIT2.png
DEBUG: Image for Delisi Justin sucessfully Downloaded: headshot.png
DEBUG: Image for Daniel Zajac sucessfully Downloaded: dz_150x150.jpg
DEBUG: Image for Steiner Tristan sucessfully Downloaded:
    Steiner_photo_150x150.jpg
DEBUG: Image for VK Reshma sucessfully Downloaded: reshmavk-150x150.png
DEBUG: Image for Topolovec Kenny sucessfully Downloaded:
    KennyTopolovec_150_150.jpg
DEBUG: Image for Benczarski Joe sucessfully Downloaded: joe_150x150.png
DEBUG: Image for Paul Graff sucessfully Downloaded:
    PaulGraffPictureForECE528_2.jpg
DEBUG: Image for Bhopatrao Prachi sucessfully Downloaded: prachi_img_new.png
DEBUG: Image for Ujjwal Pratul sucessfully Downloaded: image.jpg
DEBUG: Image for Bhor Sonal sucessfully Downloaded: SonalBhor.png
DEBUG: Image for Chinnireddy Balaji sucessfully Downloaded: image.png
DEBUG: Image for Fozey Alqirsh sucessfully Downloaded: fozey.png
DEBUG: Image for Bokhari Aidan sucessfully Downloaded: HowDoYouDoFellowKids.png
DEBUG: Image name for Sivagiri Manjunadh already exists, renaming
DEBUG: Image for Sivagiri Manjunadh sucessfully Downloaded:
    Sivagiri_Manjunadh_Image.jpg
DEBUG: Image for Bhatti Tejbir sucessfully Downloaded: IMG.png
DEBUG: Image for Patel Kush sucessfully Downloaded: kush.png
DEBUG: Image for Gilkey Kelsey sucessfully Downloaded: KG428.png
```

```
DEBUG: Image for Blum Bruce sucessfully Downloaded: myImage.png
DEBUG: Image for Khambam Sanjaykumar sucessfully Downloaded: Sanjay.png
DEBUG: Image for Hourani Hadi sucessfully Downloaded: Hourani.png
DEBUG: Image for Virigineni Srija sucessfully Downloaded: myimage.jpg
DEBUG: Image for Patel Amar sucessfully Downloaded: small.png
INFO: Sorting students by last name
INFO: Generating matrix of student images
INFO: Running facial detection on each student image
DEBUG: Found 1 faces!
DEBUG: Found 1 faces!
DEBUG: Found 1 faces!
```

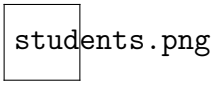

Figure 1: Output student image matrix

Figure 2: Three example output images from facial detection algorithm

5 Code

All code generated within scope of this assignment can be found in the Assignment 2 folder in the GitHub repository: `https://github.com/mikebowyer/ECE528_Assignments`.