Josh Johnson

CSE 415 – Final Project

1) Optical Character Recognizer

2) Teammates: Solo-project

I was responsible for all parts of the project, including creating training data, creating test data, research, design, and creation of all the relevant python files, 4 in total.

3) The program is supposed to be able to receive an input of either a single character or multiple characters and when it does, figure out what character it is and output that to the user.

4) There were a couple AI techniques used. The first was a derivative of naïve Bayes that made an estimation about the probability of which letter the input letter was. The second technique was a set of parameters that made up the classifier. This was done by creating many training letters where the character was known and the relevant data was stored in a dictionary. This worked by having 8 parameters for every single character that was passed through. These included where the letter started on the horizontal, e.g. ‘w’ starts closer to the edge than ‘l’ does, the number of black pixels in the image, and the mean horizontal image, where this was a parameter that determined if a character was ‘heavy’ to one side, like an ‘L’ would be to the left, among other classifiers. On top of that, there was also a test of close the two characters were when they were converted to an array by using the cosine similarity. Finally, when the program outputs its guess, it asks the user to check if the character is what they expected. If the output is wrong, it asks the user to correct it and then anytime the program is asked about that specific file, it will be able to output a guaranteed right answer.

5)

An example of a test-run:

Preparing...

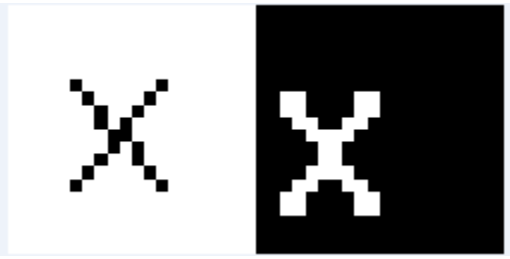
Ready for input.

Please enter file to analyze name or (q) to quit: Single Test/test.png

[outputs the image to show the user what it received versus what it thinks is the answer.]

Is this your letter: x (Y/N)? Y

Please enter file to analyze name or (q) to quit: q



Another example:

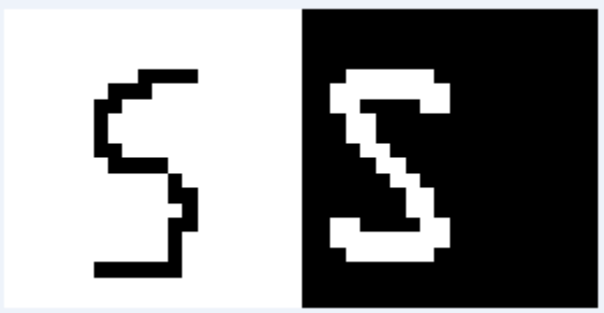
Preparing...

Ready for input.

Please enter file to analyze name or (q) to quit: Single Test/test9.png

Is this your letter: S (Y/N)? Y

Please enter file to analyze name or (q) to quit: q



However, it can also handle multiple characters at a time.

Preparing...

Ready for input.

Please enter file to analyze name or (q) to quit: Multi Test/200testerpic3.png

[This takes a bit longer than single chars, so I print out the progress.]

10 % complete

20 % complete

30 % complete

40 % complete

50 % complete

60 % complete

70 % complete

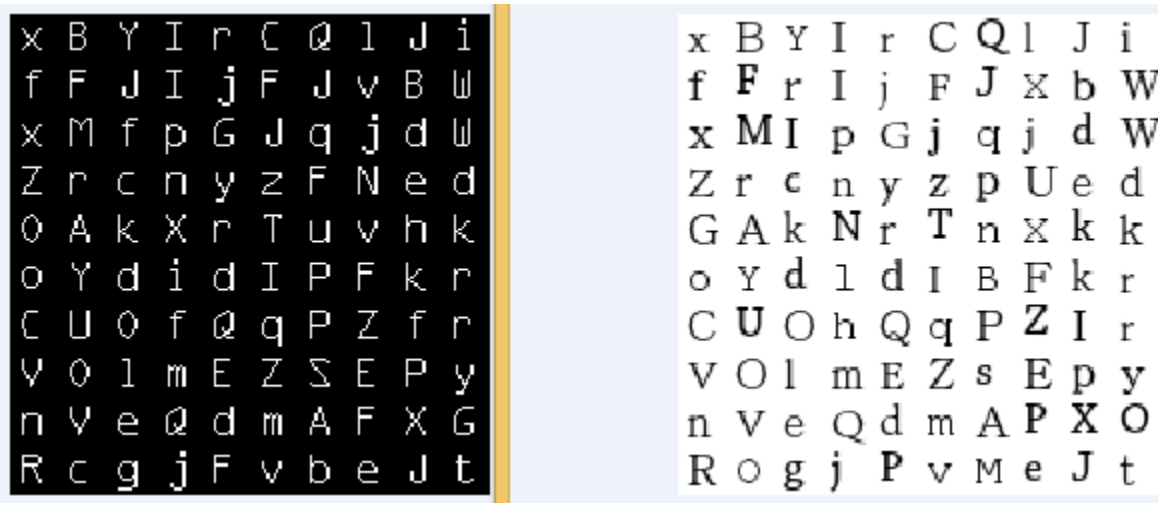
80 % complete

90 % complete

100 % complete

[output on the left this time]

Please enter file to analyze name or (q) to quit: q



6) Demo Instructions:

1. Run program and wait for it to prompt you.
2. Enter the file path to a .png file that contains either a 20x20 png image or a 200x200 png image.

3. Check out the program output and give feedback as necessary.
4. Run as many times as you'd like.

7)

```
def multi_image(image):
    newImg = Image.new("1", (200, 200), "black")
    draw = ImageDraw.Draw(newImg)
    font = ImageFont.truetype("C:\Windows\Fonts\OCRAEXT.ttf", 18)
    characterList = []
    for y in range(10):
        for x in range(10):
            crop_rectangle = (x * 20, y * 20, 20 + 20 * x, 20 + 20 * y)
            cropped_im = image.crop(crop_rectangle)
            pixelList = list(cropped_im.getdata())
            for x in range(len(pixelList)):
                pixelList[x] = int(not(pixelList[x] // 255))
            closestMatch, exact = get_closest(str(pixelList))
            characterList.append(closestMatch)
        print((1+y) * 10, "% complete")
    counter = 0
    for y in range(10):
        for x in range(10):
            character = characterList[counter]
            counter += 1
            draw.text((4 + 20 * x, 0 + 20 * y), character, 1, font=font)
    newImg.show()
```

This is the function for classifying the image with multiple characters. It starts by creating a new image with a black background. Then it iterates through the input picture and gets the closest match for each character and appends that character to a list. Finally, the function adds text to the image created at the beginning and shows that new image to the user.

8)

I learned a lot about how classifiers work and the importance of a large set of training data. My program became a lot more accurate when I increased the number of training characters to around 1000. I

became a lot more comfortable with the Pillow image processing module, as well as the unit-testing module, both modules I didn't have much experience with previously.

9)

There are a couple of features I would've liked to add if I had more time, here they are in order of assumed difficulty:

- Added the ability to accept an input of any size.
- Added a way to have direct input (not from an image), with the aid of a stylus.
- Ability to connect a camera and have it reading input continuously and output what it sees.

10) Citations

One of the papers I found relating to this was Peter W. Frey and David J. Slate's called Letter Recognition Using Holland-Style Adaptive Classifiers. It was hugely helpful and as of the completion of this project can be found at the following address: http://cns-classes.bu.edu/cn550/Readings/frey-slate-91.pdf