

## Capstone 2: Text Search

### **Motivation:**

When text documents are stored as chars, it is easy to apply dfa searching algorithms to find where substrings are in the given text. When documents are given though only as images, searching becomes a manual task, often time-consuming.

### **Problem Statement:**

This project will attempt to convert images into a char array such that it is possible to search for given substrings in an automated way.

### **Data-Set:**

The data is taken from [this](#) dataset on Kaggle. The data was manually extracted from [NIST](#), lines of handwritten text and separated into 28 X 28 images of individual upper case chars.

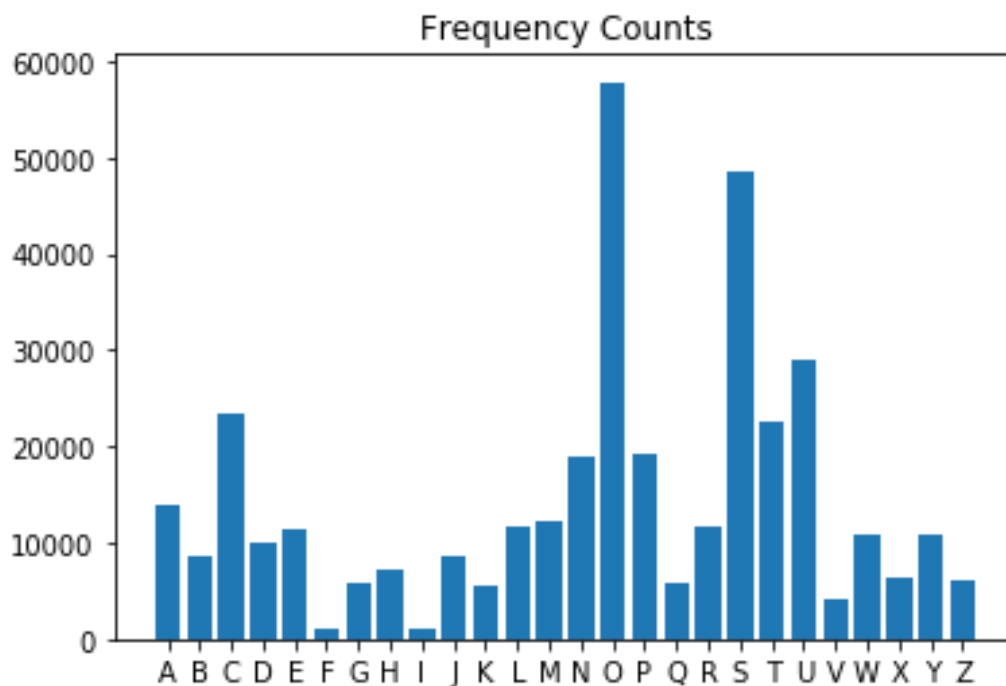
I imported the data as an array with numpy. The data was just a 372451, 785 array of floats. I then reshaped the array into 2 different arrays; one with shape 28 X 28 X 1, representing the actual 28 by 28 image, and the other a 372451 X1 array with a domain of 0 to 25 with the actual char value (ex - A, B, C...)

### **Exploration:**

After seeing the actual shape of the data arrays, I wanted to see what one of the images actually looked like. To do this, I used the built-in Keras `array_to_img` function. (I could have used `PIL.Image` too but I was going to use Keras later anyways). I observed that the images were all black backgrounds with a white handwritten char in the middle.

I then wanted to confirm that the images of all the letters were together. Having this property isn't strictly needed, but makes the dataset a bit easier to use. I looped over all the pictures and observed that the number of times that the letter changed was only 25 times, confirming that they were all in order.

Finally, I was interesting in the distribution of the letters. I found that the letter that appeared the most number of times was 'O' and the least 'I'. I then plotted the distribution.



The number of pictures per letter isn't even close to being evenly distributed. The range between 'O' and 'I' was 57825 pictures with 'O' to 1120 with 'I', a range of 56,705 pictures.

The deviation was high at 13094 and the avg was 14325.

Because there are so many pictures with even 'I' having over 1000, this shouldn't affect the model too much.

## Statistical Analysis

I conducted a statistical analysis to see if the distribution of letters in the data set is comparable to the distribution in actual texts. The actual true frequency most likely differs depending on the context the text is taken from. I chose to use the count given here.

<http://practicalcryptography.com/cryptanalysis/letter-frequencies-various-languages/english-letter-frequencies/>.

That dataset analyzed 4.5 billion chars of text in order to get their percentages. I immediately noticed that the data set I was using has a much different distribution.

I wanted to validate that this difference is significant. Intuitively, I thought it was significant because that dataset contained 4.5 billion chars and the percentages in my data set were quite different.

I set up a hypothesis with a single letter, 'I'. The null hypothesis was that there was no significant difference. The alternate was that there was. I also represented this problem as a binomial

distribution. Success means that the letter 'l' appears. A failure means that it doesn't. I used  $\sqrt{pq/n}$  to estimate sigma as that is the standard error for using p as the point estimate.

Because n was really large at 4.5 billion chars, sigma was very small, almost close to 0. This happens because the binomial point estimator is consistent.

[https://en.wikipedia.org/wiki/Consistent\\_estimator](https://en.wikipedia.org/wiki/Consistent_estimator)

Now the actual difference between the percentage of 'l' and the real percentage was large at 3 percent. Because of this, it is unlikely that if the data was taken from a data set that follows this distribution.

This shouldn't affect the model but it is something that is important to notice about the data.

Preprocessing:

First I inverted the colors of half of the images (black background to white background). This would help the model be able to predict whether the char is black or white. I also then fitted the images to an ImageDataGenerator, a Keras class which allows for augmentation. I chose just to have some of the images be shifted vertically and horizontally up to 20 percent. I didn't want to rotate the images because some of the chars were already slightly rotated as they were written by hand and letters can blend together if rotated too much.

