CS 660 - Data Science at Scale

Lab 1: The Digital Universe
Size Estimation & Text Processing with Python

Group: Eric Jiang, Joonsoo Park, Franco Pettigrosso, McWelling Todman

1 Excercise 1 - Data Types

1. Text is only one type of data. Other types include music, pictures, videos, sensor data, reference data, personal data. Choose one of the following five slides and answer the questions to estimate how many bytes are needed. Present a similar story as the one I did with text.

Pictures

(a) Resolution and color

i. The resolution of a picture or image is determined by its width times its height. For example, a picture with resolution of 200 x 100 means there are 200 columns and 100 rows of pixels, that is 20000 pixels in total.
The size of a picture is determined by multiplying the number of pixels by the number of bytes per pixel. Each pixel is represented by the RGB color channels that it would need 8-bit data for Red channel, 8-bit for Green channel, and 8-bit for Blue channel. The 8-bit binary number can represent a scale from 0 to 255. Therefore, the three 8-bit channels imply 3 bytes per pixel, and up to 16.7 million possible color combinations (256 x 256 x 256).
In other words, a color picture with 200 x 100 resolution would have a size of 200 x 100 x 3 = 60000 bytes = 60 kilobytes, where the last " 3" is for 3 bytes of RGB color information per pixel.

(b) jpeg size

i. jpeg is a compression method used to make image files smaller by sacrificing colors. The original jpeg was never used. something much simpler taken its place, but we still just the ext JPEG or JPG. The scheme takes advantage of the fact we cannot see small changes in color intensity. This i done by downsizing the color samples and then compressing the file... like a zip. We can also change the compression to let more color or less. The less, small files and the latter for more color.

(c) Number of pictures? Number of pictures on a hard drive? On a phone?

i. **Number of pictures:** the number of pictures that can be held on any device depends upon the size of the pictures. A picture that is 1024x768 pixels (printed size of 3.41x2.56) is 3 Mb. So, if we take 3Mb as the size of a picture, in 1Gb, we can have about 333 pictures. In 2017 approximately 1.2T digital photographs were taken, assuming an average size of 3Mb per photo, that is 3.6 Trillion Mb or 3600 petabytes of data.

   ii. **Number of pictures on a hard drive:** A common hard drive that comes on a modern computer is 1Tb, which is 1,000Gb. So, on 1Tb (assuming we have nothing else on there), we can have about 333,333 pictures.

   iii. **Number of pictures on a phone:** Modern phones come with a range of memory sizes. The largest iPhone is 512Gb. On a phone with this much memory, we can have about 170,496 pictures. There are many phones built on the Android operating system that have expandable memory. The largest memory that Ive been able to identify is the LG G8 ThinQ, which has a base of 128Gb of internal storage that can be expanded up to 2Tb with a microSD card. This phone can hold around 708,624 pictures.

(d) Number of Facebook picture uploads?

   i. While several media outlets reference a white paper which states facebook's user population uploads approximately 350 million photographs per day, I was unable to locate the white paper in question to verify this claim. I did find a 2010 whitepaper from facebook's reasearch group explaining that their custom object store, Haystack ingests around 1b photos per week (avg 143m per day), a daily image data volume of 60 terabytes. At the time, Facebook would generate 4 images of differing size for each photo uploaded, meaning the 65b user photos that had been uploaded to date resulted in 260b images in storage or 20 petabytes of data.

References:

(a) Resolution and color

   i. https://www.scantips.com/basics1d.html

(b) jpeg size

   i. https://file.org/extension/jpeg

(c) Picture storage

   i. https://mylio.com/life-in-focus/tech-today/heres-how-many-digital-photos-will-be-taken-in-2017-repost-oct

   ii. http://www.rideau-info.com/photos/filetypes.html

(d) Facebook picture uploads

   i. https://research.fb.com/wp-content/uploads/2016/11/finding-a-needle-in-haystack-facebook-s-photo-storage.pdf?

2 Exercise 2 - Letter Frequency Analysis

1. Run the frequency count script on another book taken from www.gutenberg.org.
   (a) Do letter frequencies vary from author to author?
      i. No, letter frequencies only vary by a small degree between authors.
   (b) How would you compare two histograms of letter frequencies?
      i. You could visually compare by subtracting the values of one histogram from the other – positive values mean the initial book had a higher percentage of a certain letter, whereas if the value for a letter is negative, it means the second book had a higher percentage of a certain letter. It is important to use real terms (percentages) when doing this rather than nominal terms (letter counts) because both text might be of different lengths.
   (c) Look at the python package matplotlib and install it if you dont have it. How do you use it to make a histogram?
      i. See below (code on next page)

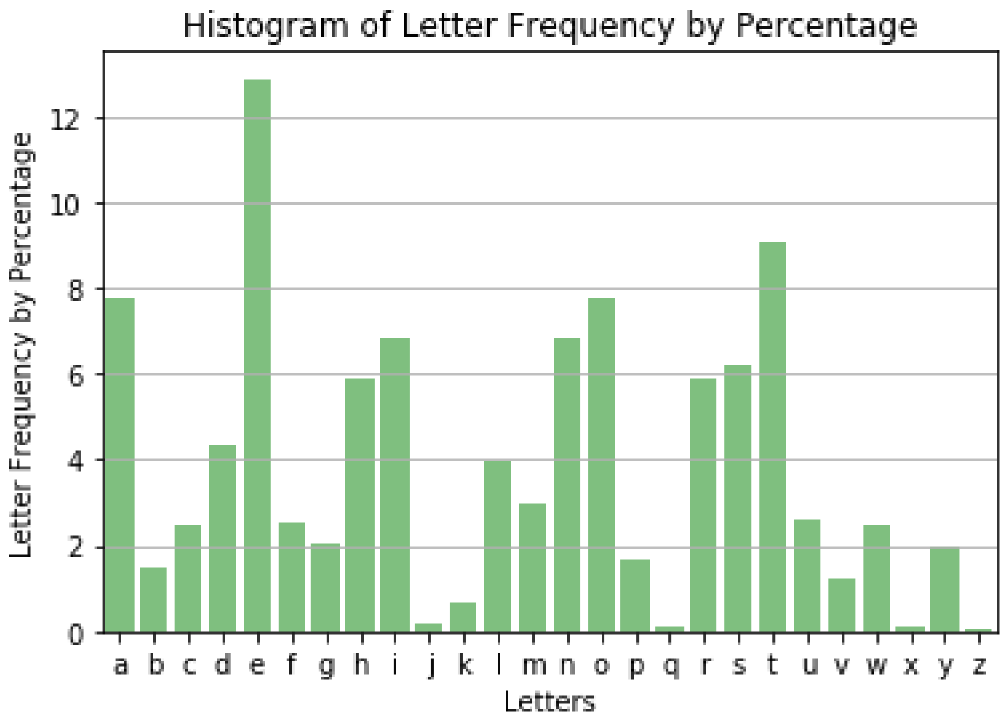

Figure 1: Histogram generated using python package Matplotlib (script below)

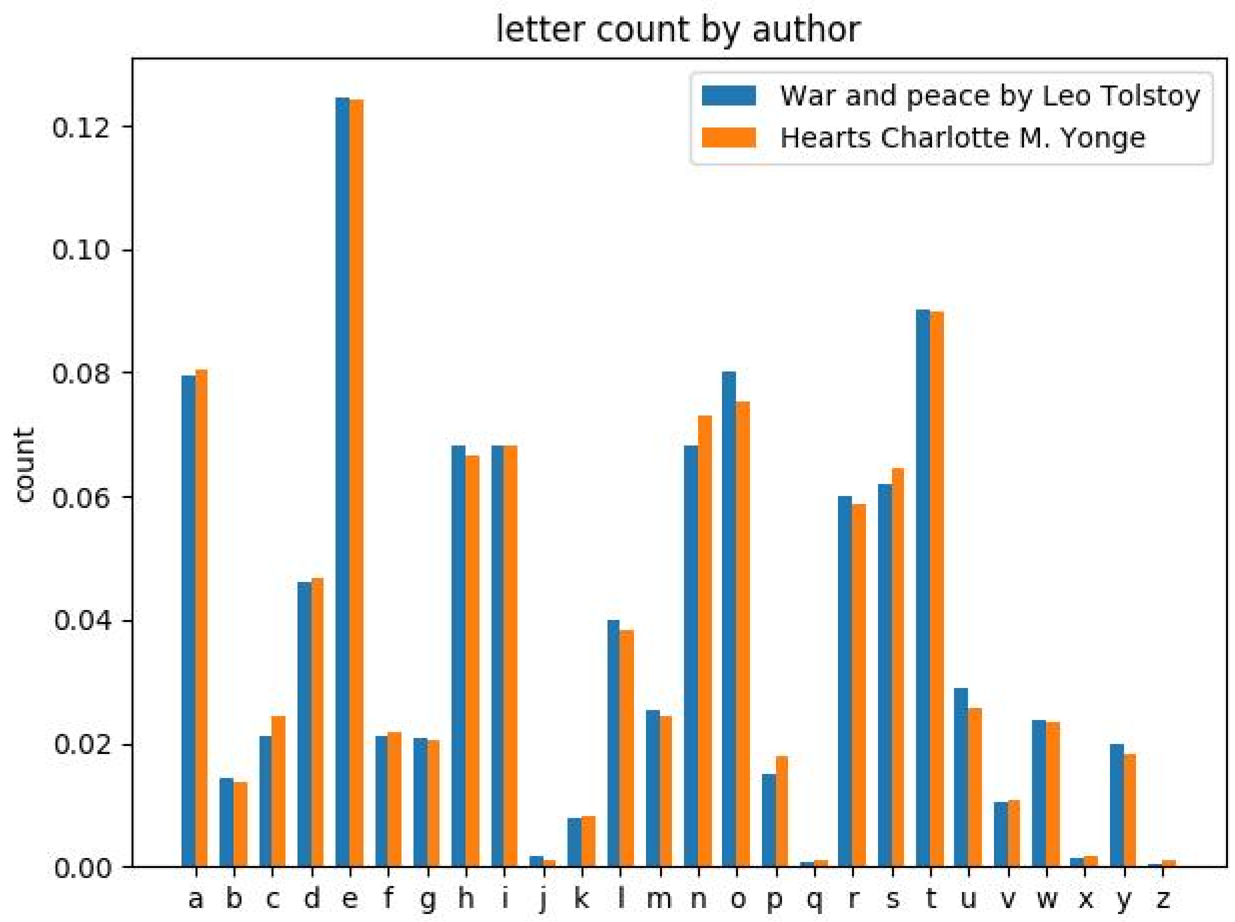

Figure 2: Comparative Histogram generated using python package Matplotlib (script below)

Exercise 2 - Python Script for Figure 1

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Jun 30 12:02:43 2019

@author: group (Eric, Franco, Joonsoo, Mac)
"""

import string
import wget
import matplotlib.pyplot as plt

url = 'http://www.gutenberg.org/files/23/23.txt'
corpus = wget.download(url)
infile = open(corpus,'r')

freq = {}
count = 0
for c in string.ascii_lowercase:
    freq[c] = 0

for line in infile:
    for c in line.lower():
        if c >= 'a' and c <= 'z':
            freq[c] = freq[c] + 1
            count = count + 1

keys = sorted(freq)

# the histogram of the data
fig = plt.figure()
ax = fig.add_subplot(111)
x = range(26)
ax.bar(x, [item/count * 100 for item in freq.values()], width=0.8,
       color='g', alpha=0.5, align='center')
ax.set_xticks(x)
ax.set_xticklabels(keys)
ax.tick_params(axis='x', direction='out')
ax.set_xlim(-0.5, 25.5)
ax.yaxis.grid(True)
ax.set_ylabel('Letter Frequency by Percentage')
plt.xlabel('Letters')
plt.title('Histogram of Letter Frequency by Percentage')
plt.show()
```

Exercise 2 - Python Script for Figure 2

```
'''
Lab1 Exercise 2

@author: group (Eric, Franco, Joonsoo, Mac)
'''
import string
import sys
import matplotlib.pyplot as plt
import numpy as np

ALPHABET = string.ascii_lowercase

def get_letter_count( file_path ):
    '''
    exactly the same code you gave us for the frequency script
    except it does it in a function
    '''
    print(f'Starting {file_path}')
    infile = open(file_path ,'r')
    freq = {}
    count = 0
    for c in ALPHABET:
        freq[c] = 0

    for line in infile:
        for c in line.lower():
            if c.isalpha():
                try:
                    freq[c] = freq[c] + 1
                    count += 1
                    #just so happens I pick a book with funny letters
                except KeyError:
                    continue

    for i in ALPHABET:
        freq[i] = freq[i] / count
    return freq

def get_values_out_of_Dictionary ( dictionary):
    '''
    to use the data we need to get the values out of the diction that
    comes out from get_letter_count. we do that here
    '''
    temp_list = []
    for i in ALPHABET:
        temp_list.append(dictionary[i])
    return temp_list


def main():
    '''
    the main script
    -gets the files
    -gets the letter count
    -creates the graph
    -and puts the data in the graph
    -compiles and creates the graph
    '''
    filename1 = '/Users/francopettigrosso/ws/CS660Summer/samples/Heartease.t
    filename2 = '/Users/francopettigrosso/ws/CS660Summer/samples/WarANDPeace
    dataset1 = get_letter_count(filename1)
    dataset2 = get_letter_count(filename2)

    #setup
    ind = np.arange(len(string.ascii_lowercase)) #used to show how big our g
    width = .35 #how much space we want between each letter

    fig, ax = plt.subplots()
    wap = get_values_out_of_Dictionary(dataset1)
    heart = get_values_out_of_Dictionary(dataset2)
    #here we are just creating the categories and giving them a label
    rects1 = ax.bar(ind - width/2, wap, width ,label = "War and peace by Leo
    rects2 = ax.bar(ind + width/2, heart, width, label='Hearts Charlotte M. 
    # Add some text for labels, title and custom x-axis tick labels, etc.
    ax.set_ylabel('count')
    ax.set_title('letter count by author')
    ax.set_xticks(ind)
    ax.set_xticklabels(ALPHABET)
    ax.legend()
    fig.tight_layout()
    plt.show()

# the script starts here
main()
```

3 Exercise 3 - Size Estimates, Word Count & Average Word length

1. Previously we estimated the amount of space required to store the library of congress. In order to do this, we needed to estimate the average number of bytes in a book. How might you calculate this?

   (a) First, we would need to know the total number of books in the library of congress. We can call this N.

   (b) Second, we would need to estimate the average size of a book. This could be accomplished by writing a script to download text files of all of the books available through project gutenberg, adding the size of each book (expressed in bytes) to a list, L.

   (c) Third, we could then take the average of the items stored in L and multiple this by N.

$$N * AVERAGE(L) = \text{ESTIMATE}$$

2. Write python program to read a book (as we did to compute letter frequencies) and calculate the number of words and the average number of letters in a word.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Jun 30 18:06:58 2019

@author: group (Eric, Franco, Joonsoo, Mac)
"""

import wget
import numpy as np

url = 'http://www.gutenberg.org/files/23/23.txt'
corpus = wget.download(url)
infile = open(corpus, 'r')
doc = infile.read()
word_vec = doc.split()

std_word_dict = {word.lower(): len(word.lower()) for word in word_vec}
word_count = len(std_word_dict.keys())
avg_word_length = np.array(list(std_word_dict.values())).mean()

print("Total Word Count: {n} | Average Word Length:\\
    {s}".format(n=word_count, s=avg_word_length))
```

4 Exercise 4 - Extra

1. Write a python program that reads a bunch of books and calculates the average number of words. You can get books from www.gutenberg.org.

```
'''
Exercise 4
'''
from pathlib import Path #used to get the contents of the file
from statistics import mean #used to get the average word size
import urllib.request

def Main():
    #same thing as Exercise 3 but now we are getting more
    #info so we need to keep track of all the books
    #we download
    word_count = []
    mean_count = []
    success = 0
    j = 0
    while success < 1000: #I want to get 1000 successful downloads
        url = f'http://www.gutenberg.org/files/{j}/{j}-0.txt'
        j += 1
        try:
            response = urllib.request.urlopen(url)
            data = response.read()      # a `bytes` object
            text = data.decode('utf-8-sig')
        except urllib.error.HTTPError:
            continue
        except Exception as e:
            print(e)
            continue
        success += 1
        len_of_words = []
        contents = text.split(' ')
        #just to get the len of each word
        for i in contents:
            #just getting special char
            i = i.strip('\r')
            i = i.strip('\n')
            i = i.strip('\t')
            len_of_words.append(len(i))
        mean_count.append(mean(len_of_words))
        word_count.append(len(contents))
    print('average word count per book : {0:.2f}'.format(mean(word_count)))
    print('average word length         : {0:.2f}'.format(mean(mean_count)))

Main()
```