Jin Young Jeon

Lab 5.1 and 5.2

Lab 5.1

See appendix for codes below

One-gram entropy of letters is estimated to be -7.010627085477864e-05 bits.

Di-gram entropy of letters is estimated to be -6.174101451477135e-06 bits.

Lab 5.2

#2:

Prototypical bird is Falcon with minimum average Euclidean distance vector of 20.71.

First ten values of Falcon: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

#3 & 4:

Average distance from prototype Falcon:

```
------------  -------
1.eagle       1526.4
2.blackbird   1538.85
3.crow        1540.17
4.sparrow     1561.46
5.sparrow     1561.46
6.peacock     1620.38
7.robin       1622.45
8.woodpecker  1660.9
9.owl         1662.11
10.seagull    1681.89
11.bat        1715.12
12.penguin    1716.11
13.canary     1717.9
14.ostrich    1718.62
15.parrot     1738.5
16.chicken    1772.6
17.swan       1788.98
------------  -------
```

(Appendix)

#5.1

```
alphabets = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']
def entropy1(txt, alphabets):
  k, prob_i = 0, 0
  lst = collections.Counter(txt)
  length = len(txt)
  while k in range(0,len(alphabets)):
    prob_i = lst[alphabets[k]]/length
    k += 1
    H = - prob_i*((prob_i)**0.5)
  return H #entropy

H = entropy1(lowtext, alphabets)
#-7.010627085477864e-05
print('One-gram entropy of letters is estimated to be ',H,'bits.')

########################
# Your code here
def digram_entropy(txt, alphabets):
  k, prob_i = 0, 0
  lst = collections.Counter(txt)
  length = len(txt)
  while k in range(0, len(alphabets)):
    prob_x = lst[alphabets[k]]/length
    prob_y = lst[alphabets[k-1]]/length
    prob_cond = ((prob_y)*(prob_x))/(prob_x)
    prob_joint = prob_x*prob_cond
    k += 1
    H = - prob_joint*((prob_cond)**0.5)
  return H #entropy

digram_H = digram_entropy(lowtext,alphabets)
#-6.174101451477135e-06
print('Di-gram entropy of letters is estimated to be ',digram_H,'bits.')

#5.2
#2================================
def average_of_all(F): #finding averages of all data/features
        # sum = 0
        lst = []
        i = 0
        while i in range(0, len(F[0])):
                a = sum(list(zip(*F))[i])
                # print(a)
                i += 1
                lst.append(a)

        averages = []
        for i in range(0, len(lst)):
                averages.append(lst[i]/17)
        return averages

averaged = average_of_all(F)
# print(averaged)
#averaged is the average of all data features



def euclidean_distance(F, n): #average distance for nth animal
        lst = []
        for i in range(0, len(F[0])):
```

```
        a = (F[n][i] - averaged[i])**2
        lst.append(a)
    distance = (sum(lst))**0.5
    return distance

def find_prototype():
    minimum = []
    for i in range(0, 17):
        a = euclidean_distance(F, i)
        minimum.append(a)
    print(minimum)
    return min(minimum)
print("prototypical bird is Falcon with distance: ", + find_prototype()) #returns 20.71 which is equivalent to distance of Falcon

print(F[5][0:10]) #[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

#3==============================
falcon = F[5]

# print(euclidean_distance(F, 0))
def prototype_distance(F, n): #average distance for nth animal
    lst1 = []
    for i in range(0, len(F[0])):
        a = (F[n][i] - falcon[i])**2
        lst1.append(a)
    distance = sum(lst1)**0.5
    return distance
```