Introduction to Web Science

Assignment 10

Prof. Dr. Steffen Staab
staab@uni-koblenz.de

René Pickhardt
rpickhardt@uni-koblenz.de

Korok Sengupta
koroksengupta@uni-koblenz.de

Olga Zagovora
zagovora@uni-koblenz.de

Institute of Web Science and Technologies
Department of Computer Science
University of Koblenz-Landau

Submission until: January 25, 2016, 10:00 a.m.
Tutorial on: January 27, 2016, 12:00 p.m.

For all the assignment questions that require you to write code, **make sure to include the code in the answer sheet, along with a separate python file. Where screen shots are required, please add them in the answers directly and not as separate files.**

Team Name: Oscar

1 Modeling Twitter data (10 points)

In the meme paper[1] by Weng et al., in Figure 2[2] you find a plot, comparing the system entropy with the average user entropy. Your task is to reproduce the plot and corresponding calculations.

1. We provide you with the file 'onlyhashtag.data', containing a collection of hashtags from tweets. Use this data to reproduce the plot from the paper. Once you have the values for average user entropy and system entropy calculated per day create a scatter plot to display the values.
2. Interpret the scatter plot and compare it with the authors interpretation from the graph showed in the paper. Will the interpretations be compatible to each other or will they contradict each other? Do not write more than 5 sentences.

1.1 Hints

1. Use formulas from the lecture to calculate the entropy for one user and the system entropy.
2. Do not forget to give proper names of plot axes.

```
 1: import numpy as np
 2: import pandas as pd
 3: import math
 4: with open("onlyhash.data","rb") as f:
 5:     data=[line.split() for line in f]
 6: data=np.array(data)
 7: user=[data[i][0] for i in range(len(data))]
 8: meme = [data[i][2] for i in range(len(data))]
 9: date= [data[i][1] for i in range(len(data))]
10:
11:
12: def system_entropy(meme):
13:     s=np.array([[x,meme.count(x)] for x in set(meme)])
14:     s=s[:,1].astype(np.int)
15:     s = s.tolist()
16:     N=len(meme)
17:     s=[(x/N)*(math.log(x)/N) for x in s]
18:     result= -sum(s)
19:     return result
20:
21: def user_entropy(user,meme):
22:     #total_user=np.array([[x,meme.count(x)] for x in set(meme)])
23:     index=[]
```

[1] http://www.nature.com/articles/srep00335
[2] Slide 27, Lecture Meme spreading on the Web

```
24:     entrop=0
25:     for i in set(user):
26:         index = np.where(np.array(user)==i)[0]
27:         index=index.astype(np.int).tolist()
28:         memes=[meme[j] for j in index]
29:         s=np.array([[x,memes.count(x)] for x in set(memes)])
30:         s=s[:,1].astype(np.int)
31:         s = s.tolist()
32:         n=len(memes)
33:         s=[(x/n)* (math.log(x)/n) for x in s]
34:         result= -sum(s)
35:         entrop= entrop+result
36:     average = entrop/len(user)
37:     return average
38:
39: system_ent=[]
40: average_us_ent=[]
41: for i in set(date):
42:     index= np.where(np.array(date)==i)[0]
43:     index = index.astype(np.int).tolist()
44:
45:     me=[meme[j] for j in index]
46:
47:     us=[user[k] for k in index]
48:
49:     system_ent.append(system_entropy(me))
50:
51:     average_us_ent.append(user_entropy(us,me))
52: print (system_ent)
53: print ("_____________________________________________________________________________
54: print (average_us_ent)
55:
56: x=[]
57: for i in range(0, len(set(date))):
58:     x.append(i)
59:
60: from matplotlib import pyplot as plt
61: plt.scatter(x,average_us_ent,color='red')
62: plt.scatter(x,system_ent,color='blue')
63: plt.ylabel('Entropy')
64: plt.xlabel('Days')
65: plt.show()
```

2 Measuring inequality (10 points)

We provide you with a sample implementation of the Chinese Restaurant Process[3].

Assume there is a restaurant with an infinite number of tables. When a new customer enters a restaurant he chooses an occupied table or the next empty table with some probabilities.

According to the process first customer always sits at the first table. Probability of the next customer to sit down at an occupied table i equals ratio of guests sitting at the table (c_i/n), where n is the number of guests in the restaurant and c_i is the number of guests sitting at table i.
Probability of customer to choose an empty table equals : $1 - \sum_{i=1}^{S} p_i$, where S is the number of occupied tables and $p_i = c_i/n$.

Provided script simulates the process and returns number of people sitting at each table. We will study restaurants for 1000 customers. Now you should modify the code and evaluate how unequal were the customers' choices of tables.

Calculate the Gini- coefficient measuring the inequality between the tables, until the coefficient stabilizes. Do five different runs and plot your results in a similar way that plots in the lecture slides are done, cf. Slide 32 and Slide 33.

```
1: import random
2: import json
3: import matplotlib.pyplot as plt
4:
5:
6: def generateChineseRestaurant(customers):
7:     # First customer always sits at the first table
8:     tables = [1]
9:
10:    # The output list that contains the gini coefficient for each customer
11:    ginis=[calculateGini(tables)]
12:
13:    #for all other customers do
14:    for cust in range(2, customers+1):
15:            # rand between 0 and 1
16:            rand = random.random()
17:            # Total probability to sit at a table
18:            prob = 0
19:            # No table found yet
20:            table_found = False
21:            # Iterate over tables
22:            for table, guests in enumerate(tables):
23:                # calc probability for actual table an add it to total probabilit
```

[3]File "chinese_restaurant.py"; Additional information can be found here: https://en.wikipedia.org/wiki/Chinese_restaurant_process

```
                prob += guests / (cust)
                # If rand is smaller than the current total prob., customer will s
                if rand < prob:
                    # incr. #customers for that table
                    tables[table] += 1
                    # customer has found table
                    table_found = True
                    # no more tables need to be iterated, break out for loop
                    break
            # If table iteration is over and no table was found, open new table
            if not table_found:
                tables.append(1)

            # calculateGini function is called to calculate the gini coefficient f
            ginis.append(calculateGini(tables))

    return ginis

def calculateGini(tables):
    # m is calculated for each table as the example in the lecture
    tables_m=[]
    for table in tables:
        tables_m.append(table/sum(tables))

    gini=0
    denominator=0

    #formula used => Gini=(sum_over_S(sum_over_S(abs(m1-m2)))/
    #                        2*S*sum(m))
    # nested loop to loop on the list of tables twice and calculate the gini
    for table in tables_m:
        for table2 in tables_m:
            gini+=abs(table-table2)
        denominator+=table


    denominator=2*len(tables)*denominator

    gini=gini/denominator

    return gini

restaurants = 1000

ginis1 = generateChineseRestaurant(restaurants)
ginis2 = generateChineseRestaurant(restaurants)
ginis3 = generateChineseRestaurant(restaurants)
ginis4 = generateChineseRestaurant(restaurants)
ginis5 = generateChineseRestaurant(restaurants)
```

```
73:
74:
75: # I changed the colors for better visualization
76: customers=list(range(1,1001))
77: plt.plot(customers, ginis1, 'b', customers, ginis2, 'g',customers,ginis3, 'r',cust
78: plt.xlabel('Customers')
79: plt.ylabel('Gini Coefficient')
80: plt.show()
```

3 Herding (10 points)

Let us consider the altitude of Koblenz to be 74 m above sea level. You are asked to figure out the height of the Ehrenbreitstein Fortress and the Fernmeldeturm Koblenz without googling.
The exercise is split in two parts:

Part 1 : The Secret
In *complete secrecy*, each member of the team will write down their estimated height of the Ehrenbreitstein Fortress without any form of discussion. Please keep in mind that you need to have reasons for your assumption. Once you are done, then openly discuss in the group and present you values in a tabulated format with the reasons each one assumed to arrive at that value.

Ehrenbreitstein Fortress

Person	Assumed height	Reason
1	160m	I know that the Cathedral of Koln is around 160m high, and I think that the fortress is around in that height.
2	130m	Given that the normal hight for a hill is 30-150 meters, I suppose that the height of the Ehrenbreitstein Fortress is 130 meters.
3	100m	On the opposite of the fortress is a statue at the Deutsches Eck. I think that that statue is around 25 meters high and if I compare it to the fortress I guess that the fortress is around 4-5 times higher.

Part II : The Discussion
Discuss amongst yourself with valid reasoning what could be the height of the Fernmeldeturm Koblenz. Only after discussing, each member of the group is asked to arrive at a value and present this value in a tabulated format as was done in Part I.

Fernmeldeturm Koblenz

Person	Assumed height	Reason
1	350m	I think that the architucture of Fernmeldeturm hides a little the height of the tower as it makes it look thiner, and I guess that the height of it is 350 meters.
2	300m	I think that Fernmeldeturm is a communication tower, therefor it must be more than 250 meters.
3	200m	The position of the Fernmeldeturm on a hill makes it to look like it has a bigger height that it actually is, so I think it is around 200 meters.

Calculate the Mean, Standard Deviation and Variance of your noted results for both the

cases and explain briefly what you infer from it.

Mean

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{1}$$

Variance

$$\delta^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2 \tag{2}$$

Standard Deviation

$$\delta = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2} \tag{3}$$

1. Ehrenbreitstein Fortress
   - Personal guesses
     - Mean
     $$\mu = \frac{1}{3}(160 + 130 + 100) = \frac{390}{3} = 130 \tag{4}$$
     - Variance
     $$\delta^2 = \frac{1}{3}[(160 - 130)^2 + (130 - 130)^2 + (100 - 130)^2] = 600 \tag{5}$$
     - Standard Deviation
     $$\delta = \sqrt{600} = 24.49 \tag{6}$$
2. Fernmeldeturm Koblenz
   - Group guesses
     - Mean
     $$\mu = \frac{1}{3}(300 + 280 + 250) = \frac{830}{3} = 276.6 \tag{7}$$
     - Variance
     $$\delta^2 = \frac{1}{3}[(300 - 276.6)^2 + (280 - 276.6)^2 + (250 - 276.6)^2] = 422.22 \tag{8}$$
     - Standard Deviation
     $$\delta = \sqrt{422.22} = 20.54 \tag{9}$$

The variance is a measure of how far each value in the data set is from the mean. As we can see from the calculations the variance in the case when the group discussed together before coming to an assumption is smaller that the variance when the members of the group decided individually. This leads us to the conclusion that in this case the discussion

helped to lower the distance between the values that each of us presented with the mean of the our assumptions. The standart deviation tells us that in the case where we acted individually, our assumptions were off approximatelly 24.5 meters from the mean, while in the case of Fernmeldeturm the average difference was 20.54.

Note: This exercise is for you to understand the concepts of herding and not to get the perfect height by googling information. There is in fact no point associated with the height but with the complete reasoning that you provide for your answers.

Important Notes

Submission

- Solutions have to be checked into the github repository. Use the directory name `groupname/assignment10/` in your group's repository.
- The name of the group and the names of all participating students must be listed on each submission.
- Solution format: all solutions as *one* PDF document. Programming code has to be submitted as Python code to the github repository. Upload *all* `.py` files of your program! Use `UTF-8` as the file encoding. *Other encodings will not be taken into account!*
- Check that your code compiles without errors.
- Make sure your code is formatted to be easy to read.
  - Make sure you code has consistent indentation.
  - Make sure you comment and document your code adequately in English.
  - Choose consistent and intuitive names for your identifiers.
- Do *not* use any accents, spaces or special characters in your filenames.

Acknowledgment

This latex template was created by Lukas Schmelzeisen for the tutorials of "Web Information Retrieval".

LaTeX

Currently the code can only be build using LuaLaTeX, so make sure you have that installed. If on Overleaf, there's an error, go to settings and change the LaTeXengine to `LuaLaTeX`.