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: Golf

Members: Atique Baig, Mtarji Adam, Deepak Garg

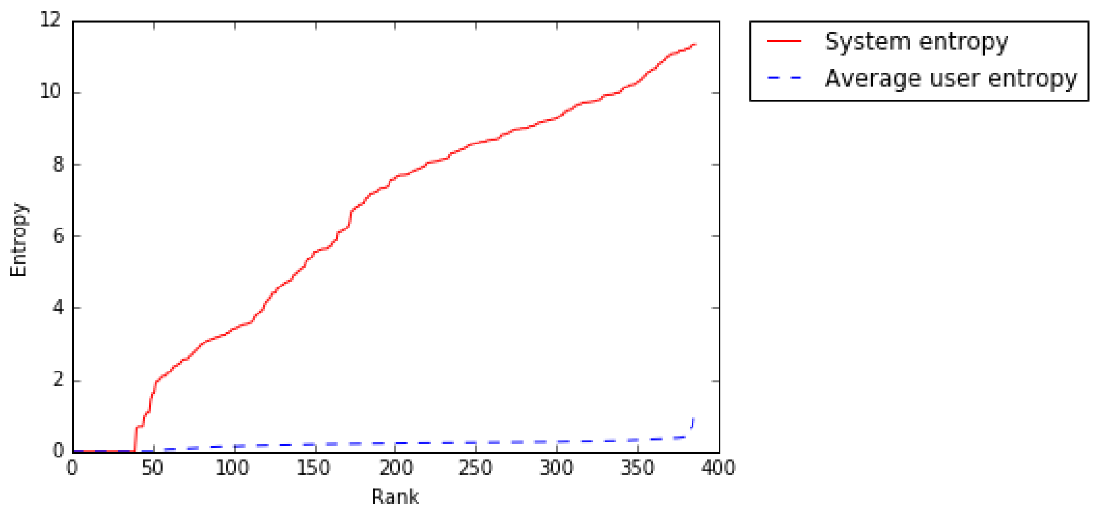
1 Modeling Twitter data (10 points)

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.

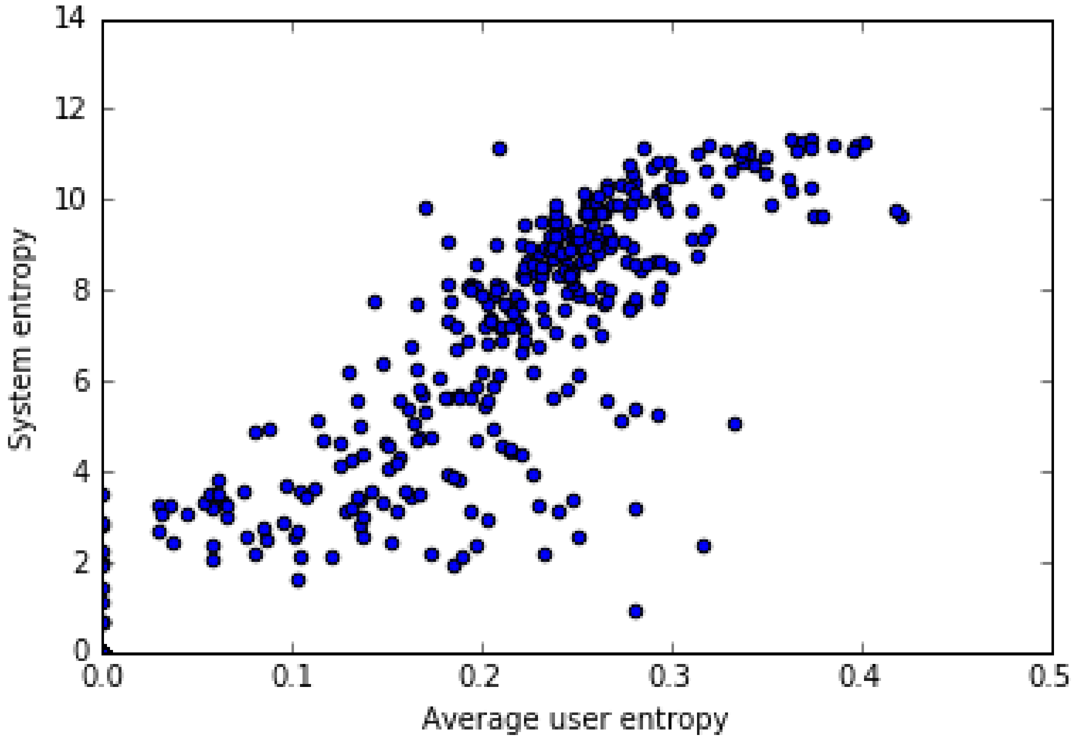
Answer:

Figure 1: Plot of the system entropy and average user entropy per day, ranked by system entropy



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

Figure 2: Scatter plot of the system entropy and average user entropy per day



The scatter plot shows a strong correlation between the system entropy and the average user entropy, When certain memes are strongly represented (low entropy) it induces a a low average user entropy, which supports the authors findings that memes compete for the users limited attention span, and that memes survive by competing with other memes.

```
import pandas as pd
import numpy as np
import math
import matplotlib.pyplot as plt

# we start by reading and processing the data
with open("onlyhash.data", "r",encoding='utf8') as data:
    working_data = []
    for line in data:
        line=line.replace('\n','').split('\t')
        working_data.append(line)
wdata=np.array(working_data)
# we create a dataframe to hold our data
```

```
df=pd.DataFrame(data=wdata[:,:],columns=['user','date','tweets'])
print(df.head())
# every tweet contains at least one meme, so we split by ' ' before comparing
def count_meme(meme,tweets):
    cnt=0
    for t in tweets:
        if meme in t.split(" "):
            cnt+=1
    return cnt
# calculate entropy given a list of tweets
def entropy(tweets):
    N = len(tweets)
    #unique, counts = np.unique(memes, return_counts=True)
    memes=set(meme for t in tweets for meme in t.split(" "))
    counts=[count_meme(meme,tweets) for meme in memes]
    return -sum([C/N*math.log(C/N) for C in counts])
# getting the list of days
days=df.date.unique()
entropies=[]
# we loop on every day
for day in days:
    # getting a dataframe representing the rows of the current day
    df_day=df.loc[df.date == day]
    # lis of all users that tweeted this day
    users=df_day.user.unique()
    # calculating the system entropy
    system_entropy=entropy(df_day.tweets)
    # calculating the average user entropy
    avg_user_entropy=sum([entropy(df_day.loc[df_day.user == u].tweets)
                                        for u in users])/len(users)
    entropies.append([system_entropy,avg_user_entropy])

ranked_data=np.array(entropies)
# saving data to a file
np.savetxt("ranked.out",ranked_data,delimiter=',')

# recovering the data
values=np.genfromtxt("ranked.out",delimiter=",")
# we sort ascending using the system entropy column
ranked_data=np.sort(values,axis=0)
# generate all ranks
ranks=[i for i in range(0,len(ranked_data))]
"""
Ploting a ranked plot of the daily system entropy and the average user entropy
```

```
"""
system_entropies=ranked_data[:,0]
avg_user_entropies=ranked_data[:,1]
plt.plot(ranks, system_entropies, 'r',label="System entropy")
plt.plot( ranks,avg_user_entropies,"b--",label="Average user entropy")
plt.xlabel("Rank")
plt.ylabel("Entropy")
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.show()

"""
Ploting a scatter plot of system entropy and average user entropy
"""

system_entropies=values[:,0]
avg_user_entropies=values[:,1]

plt.scatter(avg_user_entropies,system_entropies)
plt.xlabel("Average user entropy")
plt.ylabel("System entropy")
plt.xlim(0,0.5)
plt.ylim(0,14)
plt.show()
```

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.

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 $p_i = c_i/n$.

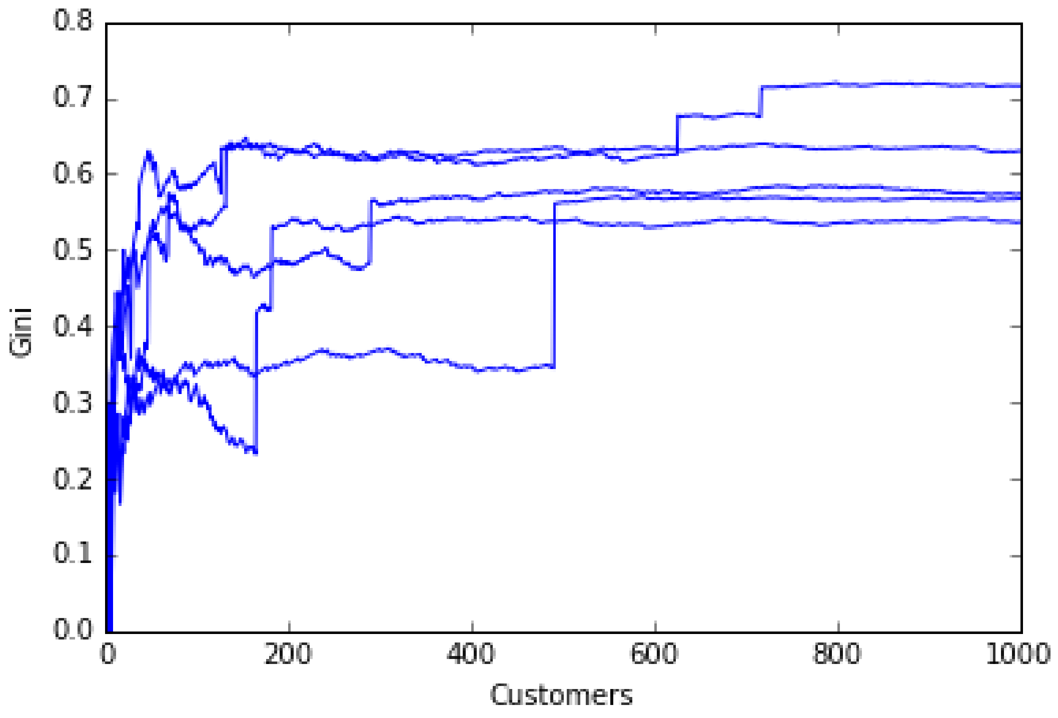
Provided script simulates the process and returns number of people sitting at each table. We will study restaurants for 1000 customers. Now you should modify the code and evaluate how unequal were the customers' choices of tables.

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.

Answer:

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

Figure 3: Plot of the Gini coefficient per number of customer



```
import random
import matplotlib.pyplot as plt

# function to sum nested lists
def sum_nested(l):
    s = 0
    for item in l:
        if type(item) is list:
            s += sum_nested(item)
        else:
            s += item
    return s
# function to calculate the gini given a list of shares
def calculateGini(shares,n):
    return 1/(2*n)*(sum_nested(abs(shares[i] - shares[j]) for j in range(0,n)
                    for i in range(0,n))/sum(shares[i] for i in range(0,n)))

def generateChineseRestaurant(customers):
    # First customer always sits at the first table
    tables = [1]
    ginis=[0]
    #for all other customers do
    for cust in range(2, customers+1):
            # rand between 0 and 1
            rand = random.random()
            # Total probability to sit at a table
            prob = 0
            # No table found yet
            table_found = False
            # Iterate over tables
            for table, guests in enumerate(tables):
                # calc probability for actual table an add it to total
                # probability
                prob += guests / (cust)
                # If rand is smaller than the current total prob.,
                # customer will sit down at current table
                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)
            # we calculate the share of every table
            table_shares=[g/cust for g in tables]
            # then we calculate the gini for the current number of customers
            ginis.append(calculateGini(table_shares,len(tables)))
    return tables,ginis



restaurants = 1000
for i in range(0,5):
    network,gcoefs = generateChineseRestaurant(restaurants)
    plt.plot(range(1,restaurants+1), gcoefs, 'b')
plt.xlabel("Customers")
plt.ylabel("Gini")
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
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.
Answer:

150 m	The average house in Koblenz could be around 20m and the fortress seems to be around 4 houses high, if we add Koblenz altitude (74+20*4)
180 m	A person is a on average 1.7m tall, and the fortress seems to be around 60 or 70 person high
220 m	Looking up from the river and judging from the angle, I would say it's a little over 200 meterif we take into consideration the given Koblenz altitude

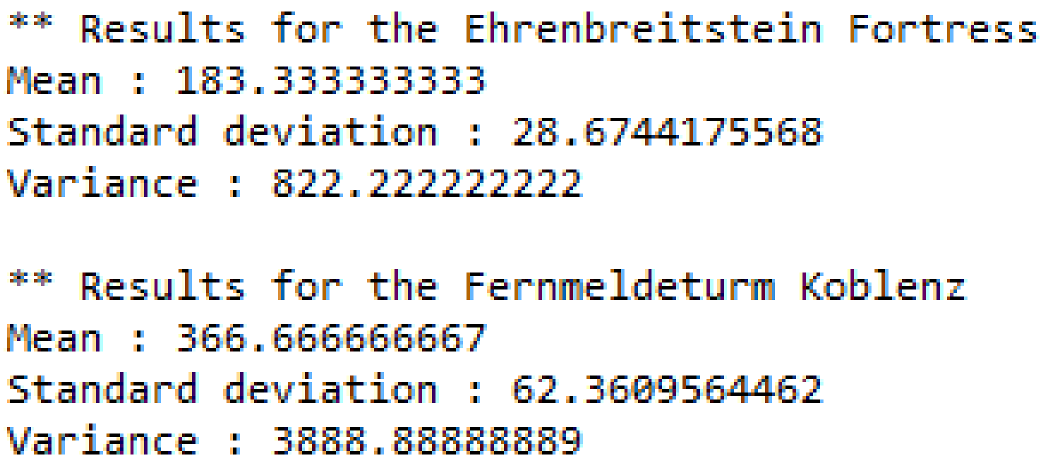
Part II : The Discussion
Discuss amongst yourself with valid reasoning what could be the height of the Fernmeldeturm Koblenz. Only after discussing, each member of the group is asked to arrive at a value and present this value in a tabulated format as was done in Part I.

Calculate the Mean, Standard Deviation and Variance of your noted results for both the cases and explain briefly what you infer from it.
Answer:

350 m	Using the average house height again, the tower seems to be around 12 to 15 houses high
300 m	A football pitch is about 120 m or so, and the tower seems to be around 2 football pitches high
450 m	The Rhein bridge is a little over 300 m, if we lay the tower visually on it it should somewhat be close if we add few dozen meters

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.

Figure 4: Results



```
** Results for the Ehrenbreitstein Fortress
Mean : 183.333333333
Standard deviation : 28.6744175568
Variance : 822.222222222

** Results for the Fernmeldeturm Koblenz
Mean : 366.666666667
Standard deviation : 62.3609564462
Variance : 3888.88888889
```

Due to the small amount of data, we get a high variance on both experiments,other the increasing the size of the data collected, we can get better results by giving a com mun reference to use to estimate the height (football field width for example), we believe this can give a far accurate set of data.

```python
import numpy as np

height_ehrenbreitstein=[150,180,220]

print("** Results for the Ehrenbreitstein Fortress")
print("Mean : %s "%np.mean(height_ehrenbreitstein))
print("Standard deviation : %s "%np.std(height_ehrenbreitstein))
print("Variance : %s "%np.var(height_ehrenbreitstein))
print()
height_fernmeldeturm=[350,300,450]
print("** Results for the Fernmeldeturm Koblenz")
print("Mean : %s "%np.mean(height_fernmeldeturm))
print("Standard deviation : %s "%np.std(height_fernmeldeturm))
print("Variance : %s "%np.var(height_fernmeldeturm))
```

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`.