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: Quebec

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.

Answer:

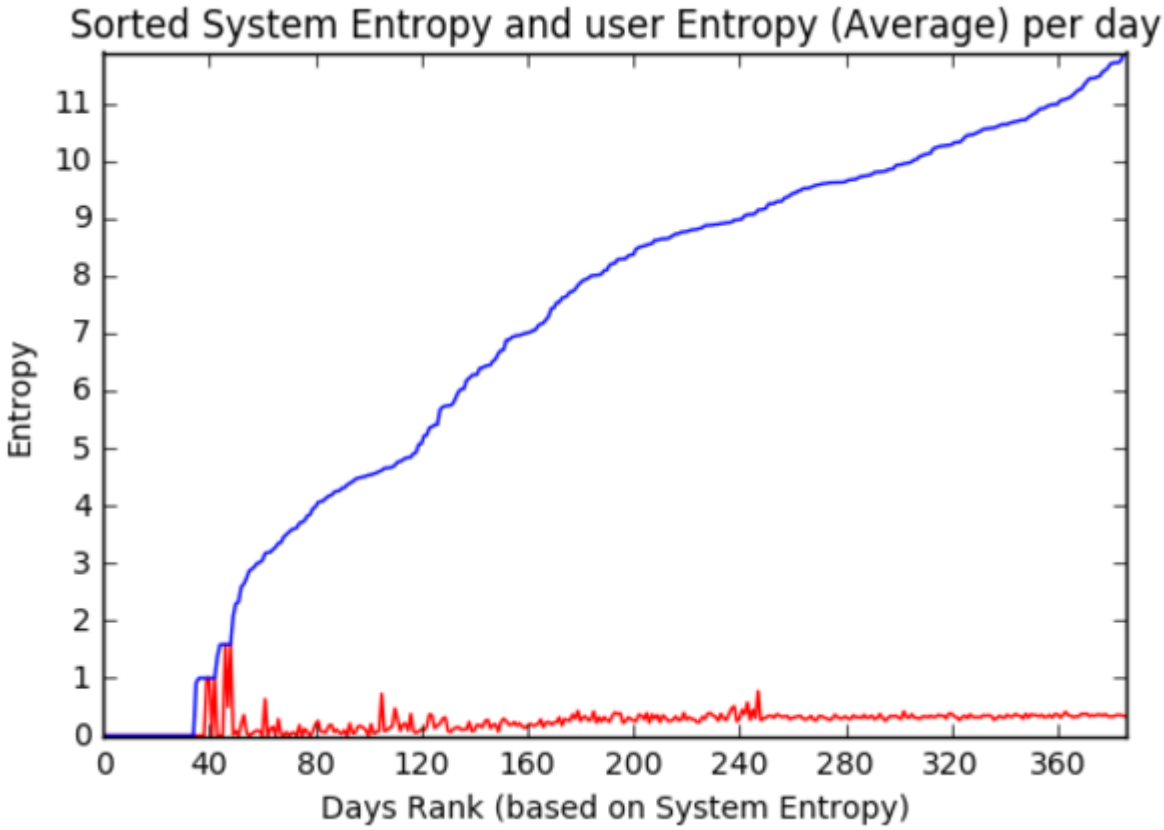
```
1: import numpy as np
2: import pandas as pd
3: from collections import Counter
4: import math
5: import matplotlib.pyplot as plt
6:
7:
8: # In[2]:
9:
10: data = pd.read_table("onlyhash.data",names=["user","date","hashtag"])
11: data.head()
12:
13:
14:
15: # In[4]:
16:
17: data["hashtag_list"] = data.hashtag.apply(lambda x: x.split(" "))
18: # data = data.head(10)
19: data.head(5)
20:
```

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

```
21:
22:
23:
24: # In[9]:
25:
26: users = data.user.unique()
27: #print(len(users))
28: index = [i for i in range(1, len(users)+1)]
29:
30: dates = data.date.unique()
31: index2 = [i for i in range(1, len(dates)+1)]
32:
33:
34: # In[10]:
35:
36: df = pd.DataFrame(users, index=index, columns=['user'])
37: df = df.head(1000)
38:
39:
40: # In[11]:
41:
42: data1 = data[data["user"] == "GetFreelanceJob"]
43: vals = data1["hashtag_list"].values
44: list_has = [val for sublist in vals for val in sublist]
45: #list_has
46:
47:
48: # In[12]:
49:
50: def get_hash(user_name, dataframe):
51:     dataframe = dataframe[dataframe["user"] == user_name]
52:     hashtag_list = dataframe["hashtag_list"].values
53:     list_hash = [val for sublist in hashtag_list for val in sublist]
54:     return list_hash
55:
56:
57: # In[13]:
58:
59: df["hashtags"] = df.user.apply(lambda x: get_hash(x, data))
60:
61:
62:
63: df.head()
64:
65:
66: def entropy(counter_list):
67: #     log2 = lambda x: math.log(x,2)
68:     c = Counter(counter_list)
69:     ent = 0.0
```

```
    for k,v in c.items():
        p = float(v)/len(counter_list)
        ent = ent - p*math.log2(p)
    return ent


# In[17]:

df["user_entropy"] = df.hashtags.apply(lambda x: entropy(x))


# In[18]:

df.head()


# In[19]:

df.user_entropy.mean()


# In[21]:

data["entropy"] = data.hashtag_list.apply(lambda x: entropy(x))


# In[22]:

data.head()


# In[23]:

grp = data.groupby(["date"]).entropy.mean()
user_entropy_by_day = grp.to_frame()
user_entropy_by_day.describe()


# In[24]:

user_entropy_by_day.head()

# In[26]:

df2 = pd.DataFrame(dates, index=index2, columns=['date'])

# In[28]:

def get_hash_date(date, dataframe):
```

```
    dataframe = dataframe[dataframe["date"] == date]
    hashtag_list = dataframe["hashtag_list"].values
    list_hash = [val for sublist in hashtag_list for val in sublist]
    return list_hash

# In[29]:

df2["hashtags"] = df2.date.apply(lambda x: get_hash_date(x, data))

# In[31]:

df2["sys_entropy"] = df2.hashtags.apply(lambda x: entropy(x))

# In[33]:

user_entropy_by_day.head()

# In[35]:

user_entropy_by_day['date'] = user_entropy_by_day.index

# In[37]:

entropy_df = pd.merge(df2, user_entropy_by_day, on='date', how='outer')
len(entropy_df)

# In[39]:

ranked = entropy_df.sort_values(by="sys_entropy")
ranked.head()


# In[40]:

import matplotlib.pyplot as plt
length=len(ranked)
x = [x for x in range(0,length)]
plt.title("Sorted System Entropy and user Entropy (Average) per day")
plt.xticks(np.arange(0,max(x),40))
plt.yticks(range(0,int(max(ranked.sys_entropy)+2)))
plt.xlabel("Days Rank (based on System Entropy)")
plt.ylabel("Entropy")
plt.plot(x,ranked.entropy.values,label='User Entropy',color="r")
plt.plot(x,ranked.sys_entropy.values,label='System Entropy',color="b")
# plt.legend(loc=2)
plt.show()
```



From this graph we can see that the system entropy in our case grows from value 0 to almost 12, unlike the one from the paper in question where it starts from 10.5 and grows until close to 13. The user entropy is also not so constant (from beginning to end) as the one in the paper.

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: We have modified the code to calculate the Gini coefficient after every guest enters the restaurant.

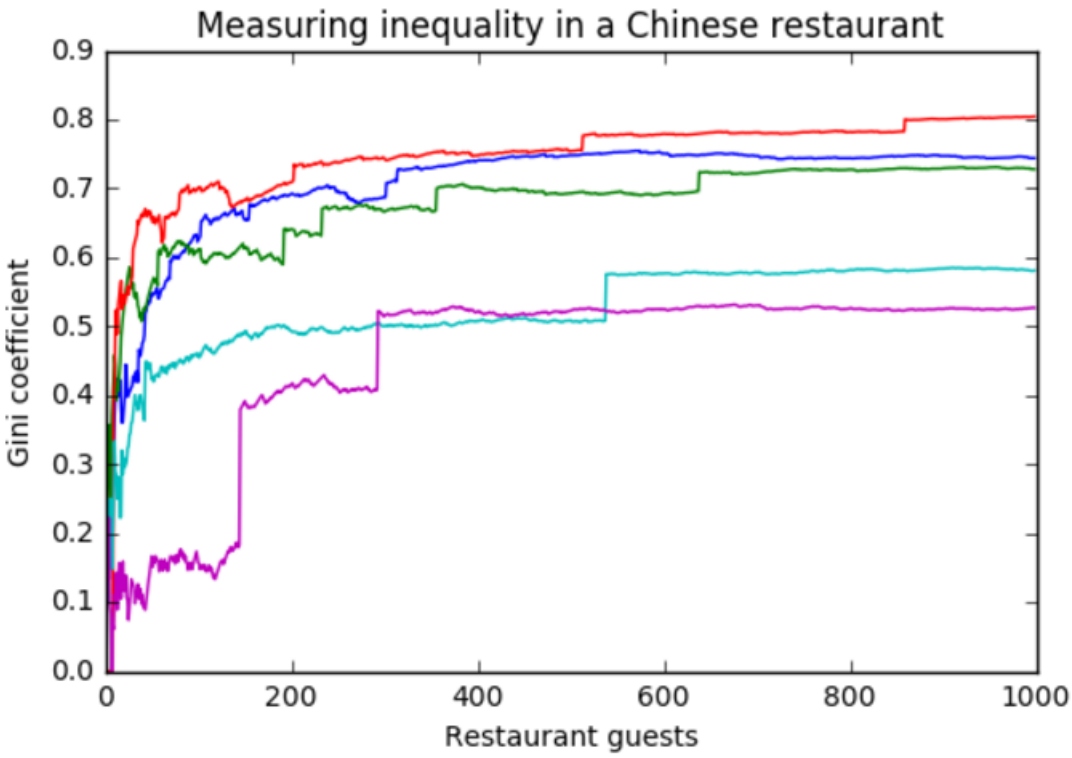
```
1: import random
2: import json
3: import numpy as np
4:
5:
6: def gini(x):
7:     mad = np.abs(np.subtract.outer(x, x)).mean()
8:     rmad = mad/np.mean(x)
9:     g = 0.5 * rmad
10:    return g
11:
12:
13: def generateChineseRestaurant(customers):
14:    # First customer always sits at the first table
15:    tables = [1]
16:    gini_vals = []
17:    #for all other customers do
18:    for cust in range(2, customers+1):
19:        # rand between 0 and 1
```

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

```
20:         rand = random.random()
21:         # Total probability to sit at a table
22:         prob = 0
23:         # No table found yet
24:         table_found = False
25:         # Iterate over tables
26:         for table, guests in enumerate(tables):
27:             # calc probability for actual table and add it to total probability
28:             prob += guests / (cust)
29:             # If rand is smaller than the current total prob., customer will sit
30:             if rand < prob:
31:                 # incr. #customers for that table
32:                 tables[table] += 1
33:                 # customer has found table
34:                 table_found = True
35:                 # no more tables need to be iterated, break out for loop
36:                 break
37:         # If table iteration is over and no table was found, open new table
38:         if not table_found:
39:             tables.append(1)
40:         tables_s = sorted(tables)
41:         gini_vals.append(gini(tables_s))
42:     return tables, gini_vals
43:
44: if __name__ == "__main__":
45:     restaurants = 1000
46:
47:     network = generateChineseRestaurant(restaurants)
48:     with open('network_' + str(restaurants) + '.json', 'w') as out:
49:         json.dump(network, out)
```

```
1: from chinese_restaurant import generateChineseRestaurant as chineseRes
2:
3: customers = 1000
4: for i in range(0,5):
5:     tables, ginis = chineseRes(customers)
6:     plt.plot(ginis)
7: plt.show()
```

The resulting plot for running the script five times:



All the lines stabilize after around 500 people. It seems that if the inequality is higher in the start, it will stabilize on a higher value. Compared to the graphs in the lecture, our values stabilize higher - ranging from 0.5 to 0.8, while in the lecture most are under 0.6 .

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.

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.

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.

Answer:
Results before discussion:

fortress	argument
90	compared to the size of the houses under it, looks like 3-5 of them
250	the height of the hill plus the height of the tower
375	...

```
print("mean: ",estimations.fortress.mean())
print("std: ",estimations.fortress.std())
print("var: ",estimations.fortress.std()**2)
```

```
mean:  238.33333333333334
std:  142.85773809399802
var:  20408.333333333332
```

Results after discussion:

tower	argument
250	compared to the height of trees looks like 20 trees stacked up
200	as it is a regular and non famous TV tower it shouldn't be more than 200m
450	...

```
print("mean: ",estimations.tower.mean())
print("std: ",estimations.tower.std())
print("var: ",estimations.tower.std()**2)
```

```
mean:  300.0
std:  132.28756555322954
var:  17500.000000000004
```

By looking at the mean values, we can see that the group estimates that the tower is higher than the fortress. From the standard deviation, we can see that the mean value in the second part is a bit more representative then the one from the first part. Also, by taking into account the values of variance we can see that the difference in individual estimations is greater in case without discussion.

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`.