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.**

Group Name : Yankee
Candidates :

1. Sabin Bhattarai - 216203590
   sbhattarai@uni-koblenz.de
2. Biplov K.C. - 216203865
   biplov@uni-koblenz.de
3. Syed Salman Ali. - 216203923
   salmanali@uni-koblenz.de

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:

```
import math
import operator
import pandas as pd
import numpy as np
import pylab as plt

df = pd.read_table(r'onlyhash.data')

# renaming data frame with meaningful column name
df.columns = ['user', 'date', 'hash_tag']


# returns all the unique values from a given column
def get_unique_values(data_frame, column):
    return data_frame[column].unique()


# removes all the rows whose given column contains term value
def remove_noise(data_frame, column, term):
    unwanted_data_frame = data_frame[data_frame[column].str.contains(term)]
    for unwanted_value in unwanted_data_frame[column].unique():
        data_frame = data_frame[data_frame[column] != unwanted_value]

    return data_frame


dataFrame = remove_noise(df, 'date', r'\\')
unique_dates = get_unique_values(dataFrame, 'date')
```

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

```
29:
30: system_entropy_dict = {}
31: user_entropy_dict = {}
32:
33:
34: def get_data_frame_with_matching_values(data_frame, column, value):
35:     return data_frame[data_frame[column] == value]
36:
37:
38: def convert_hash_tags_to_list_of_hash_tags(hash_tags):
39:     list_of_hash_tags = []
40:     if isinstance(hash_tags, str):
41:         split_hash_tags = hash_tags.split()
42:         for split_hash_tag in split_hash_tags:
43:             list_of_hash_tags.append(split_hash_tag)
44:         return list_of_hash_tags
45:     for row_hash_tag in hash_tags:
46:         if row_hash_tag.count('#') > 1:
47:             split_row_hash_tags = row_hash_tag.split()
48:             for split_row_hash_tag in split_row_hash_tags:
49:                 list_of_hash_tags.append(split_row_hash_tag)
50:         else:
51:             list_of_hash_tags.append(row_hash_tag)
52:     return list_of_hash_tags
53:
54:
55: def calculate_entropy_for_a_day(hash_tags):
56:     entropy = 0
57:     hash_tag_set = set(hash_tags)
58:     for unique_hash_tag in hash_tag_set:
59:         f_u_i = hash_tags.count(unique_hash_tag) / (len(hash_tags))
60:         entropy_of_one_hash_tag = (f_u_i * math.log2(f_u_i))
61:         entropy += entropy_of_one_hash_tag
62:     return entropy
63:
64:
65: def calculate_average_entropy_of_a_day(total_entropy, total_count):
66:     return total_entropy / total_count
67:
68:
69: count = 0
70:
71: # for each unique date calculate system entropy and user entropy
72: for unique_date in unique_dates:
73:     # df_for_uniqueDate = dataFrame[dataFrame.date == unique_date]
74:     df_for_uniqueDate = get_data_frame_with_matching_values(dataFrame, \
75:                                                             'date', unique_date)
76:     all_possible_hash_tags = df_for_uniqueDate['hash_tag']
77:     total_users_entropy = 0
```

```
78:     # each iteration of this loop deals with a user's hashtags
79:     for user_hash_tags in all_possible_hash_tags:
80:         user_hash_tag_list = convert_hash_tags_to_list_of_hash_tags(user_hash_tags
81:         user_entropy_for_a_day = calculate_entropy_for_a_day(user_hash_tag_list)
82:         total_users_entropy = (total_users_entropy + (-1 * user_entropy_for_a_day
83:
84:     average_users_entropy = calculate_average_entropy_of_a_day\
85:                                 (total_users_entropy, len(user_hash_tag_list))
86:
87:     # add average user entropy for each day as dictionary value
88:     user_entropy_dict[count] = average_users_entropy
89:
90:     all_individual_hash_tag_in_a_day = convert_hash_tags_to_list_of_hash_tags\
91:                                                     (all_possible_hash_tags)
92:     total_system_entropy_for_a_day = -1 * calculate_entropy_for_a_day\
93:                                             (all_individual_hash_tag_in_a_day)
94:
95:     average_system_entropy_for_a_day = calculate_average_entropy_of_a_day\
96:                                       (total_system_entropy_for_a_day, \
97:                                        len(all_individual_hash_tag_in_a_day))
98:
99:     # add average user entropy for each day as dictionary value
100:    system_entropy_dict[count] = average_system_entropy_for_a_day
101:    count += 1
102:
103: sorted_system_entropy = sorted(system_entropy_dict.items(), \
104:                                                 key=operator.itemgetter(1))
105:
106: sorted_user_entropy_dict_based_on_system_entropy = {}
107: count = 0
108: new_data_frame = pd.DataFrame()
109: # new_data_frame.columns = ['sno', 'system_entropy', 'user_entropy']
110: for key, value in sorted_system_entropy:
111:     sorted_user_entropy_dict_based_on_system_entropy[count] = \
112:                                                     user_entropy_dict.get(key)
113:     count += 1
114:     temp = pd.DataFrame([[count, value, user_entropy_dict.get(key)]])
115:     new_data_frame = new_data_frame.append(temp)
116:
117: plt.xlabel("Day(0-384): 0 being the lowest system entropy and 384 the highest.")
118: plt.ylabel("Entropy")
119: system_entropy_plot = plt.scatter(new_data_frame[0], new_data_frame[1], \
120:                       color='r', marker='o', label='System entropy', alpha=0.5)
121: user_entropy_plot = plt.scatter(new_data_frame[0], new_data_frame[2], \
122:                         color='k', marker='x', label='User entropy', alpha=0.5)
123: plt.legend(handles=[system_entropy_plot, user_entropy_plot])
124: plt.show()
```

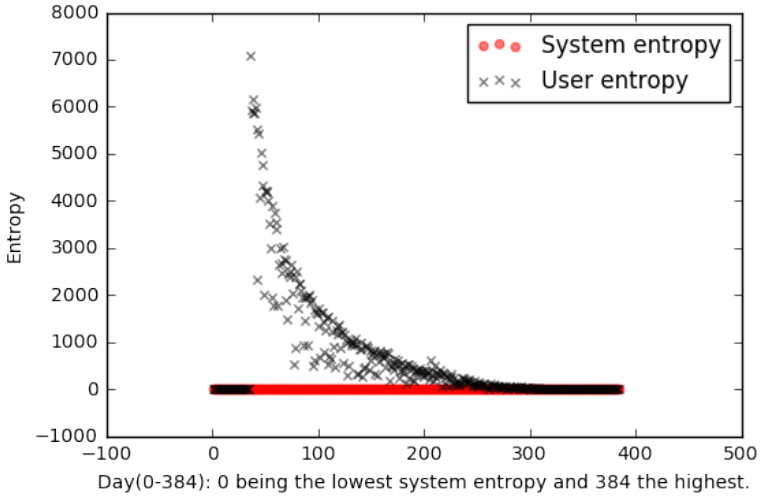

Figure 1: Average system entropy vs average user entropy for a day.

For us the plot was completely different from the graph shown in paper. In paper system entropy's range was between 10 to 13 and average user entropy was between 0 to 1. In contrast, average system entropy for us was between 0 to 1 and while average user entropy for a day widely varied. This difference could be because of different data set for us compared to that of 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:

```
 1: import random
 2: import math
 3: import operator
 4: import pandas as pd
 5: import numpy as np
 6: import pylab as plt
 7:
 8:
 9: def gini_coefficient(tables, guests):
10:     size_of_table = len(tables)
11:     table_shared = {}
12:
13:     for table in tables:
14:         table_shared[table] = tables[table] / guests
15:
16:     count = 0
17:     for table in table_shared:
18:         count += table_shared[table]
```

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

```
19:
20:     count = count / size_of_table
21:     count = 2 * count
22:
23:     temp = 0
24:     for i in table_shared:
25:         for j in table_shared:
26:             temp += abs(table_shared[i] - table_shared[j])
27:
28:     temp = temp / (math.pow(size_of_table, 2))
29:     gini = temp / count
30:
31:     return gini
32:
33:
34: def draw_plot(list1, list2, list3, list4, list5):
35:     x = len(list1) * [0]
36:     for i in range(0, len(x)):
37:         x[i] = i + 1
38:
39:     plt.xticks(np.arange(0, 1001, 100))
40:     plt.ylim(0, 1)
41:     plt.title("Gini Coeficient plot")
42:     plt.xlabel("Subjects")
43:     plt.ylabel("Gini Coefficient")
44:
45:     plt.plot(x, list1, label='simulation1', color="b")
46:     plt.plot(x, list2, label='simulation2', color="r")
47:     plt.plot(x, list3, label='simulation3', color="g")
48:     plt.plot(x, list4, label='simulation4', color="y")
49:     plt.plot(x, list5, label='simulation5', color="k")
50:     plt.legend(loc=0, fontsize='small')
51:     plt.show()
52:
53:
54: def sitting_at_tables(number_of_customers):
55:     tables_dict = {}
56:     all_gini = 1000 * [0]
57:     tables_dict[1] = 1
58:     number_of_tables = 1
59:     total_number_of_guests = 1
60:     re = gini_coefficient(tables_dict, total_number_of_guests)
61:     all_gini[total_number_of_guests - 1] = re
62:
63:     for i in range(2, number_of_customers + 1):
64:
65:         total_number_of_guests += 1
66:         found = 0
67:         rand = random.random()
```

```
68:         for table in tables_dict:
69:
70:             prob = tables_dict[table] / total_number_of_guests
71:             if rand < prob:
72:                 tables_dict[table] += 1
73:                 found = 1
74:                 re = gini_coefficient(tables_dict, total_number_of_guests)
75:                 all_gini[total_number_of_guests - 1] = re
76:                 break
77:
78:         if found == 0:
79:             number_of_tables += 1
80:             tables_dict[number_of_tables] = 1
81:             re = gini_coefficient(tables_dict, total_number_of_guests)
82:             all_gini[total_number_of_guests - 1] = re
83:
84:     return all_gini
85:
86:
87: number_of_customers = 1000
88:
89:
90: gini_1 = sitting_at_tables(number_of_customers)
91: gini_2 = sitting_at_tables(number_of_customers)
92: gini_3 = sitting_at_tables(number_of_customers)
93: gini_4 = sitting_at_tables(number_of_customers)
94: gini_5 = sitting_at_tables(number_of_customers)
95:
96: draw_plot(gini_1, gini_2, gini_3, gini_4, gini_5)
```

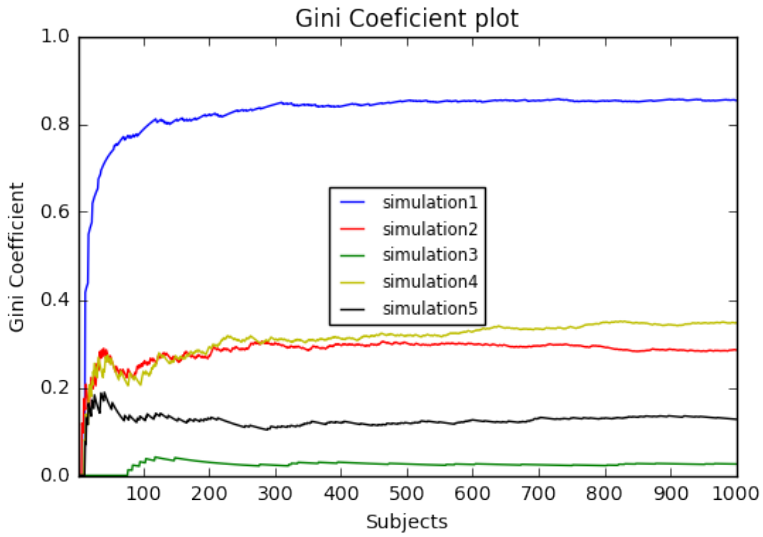

Figure 2: gini co-efficient for 5 different variation

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:

Biplov KC (team member):

I believe height of Ehrenbreitsein is around 125 m above sea level. Imagining average human height is 170 cms, I think if 30 mens are placed on top of each other, one could reach the fortress.

Furthermore, I believe the height of Fernmeldenturm Koblenz is 174 m above sea level. I pictured how long 1 m would be then I thought height of Fernmeldenturm is 100 times the height of 1 m.

Syed Salman Ali (team member):

For me, height of Ehrenbreitsein is around 140 m above sea level. I imagined length of 1 meter then expected the height of fortress to be 140 m.

Similarly, I think height of Fernmeldenturm Koblenz is 200 m above sea level. I calculated the height same as above.

Sabin Bhattarai (team member):

I think the height of Ehrenbreitsein is around 180 metres above the sea level. I play the game pokemon and the last time I went to Ehrenbreitsein, I believe I walked my pokemon about 0.34km which is approx 340 metres. This means the vertical height could approximately be around 180 meters.

Similarly I think the height of Fernmeldenturm Koblenz is around 40 m above sea level. I have 20 stairs in my house for two floors i.e. 10 stairs per floor. I assume Fernmeldenturm to be around 20 house tall i.e. 40 floors. And each stairs is around 20cm. This means we have 40*10*20cm = 8000cm i.e. 80 meters.

Calculating **mean**, **standard deviation**, and **variance**

for Ehrenbreitsein

Mean for Ehrenbreitsein = (125 + 140 + 180)/3 = 148.33

Standard deviation
We will be using the following formula for

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N}(X_i - \mu)^2}{N}}$$

Thus, Standard deviation for Ehrenbreitsein = 28.4312

Variance for Ehrenbreitsein (σ * σ) = 808.33333

for Fernmeldenturm

Mean

$= (174 + 200 + 40)/3 = 138$

Standard deviation

$= 85.86035$

Variance $(\sigma * \sigma) = 7372$

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`.