Tutorial 1. Constructing and analyzing a social network: Twitter retweet and reply data

Tutorials

The purpose of the tutorials is to guide you through a hands-on experience with Social Media Analysis. All exercises in the tutorials are to be solved using Python and `jupyter notebook` (see the instalation and usage instructions (https://wattlecourses.anu.edu.au/pluginfile.php/1412128/mod_resource/content/10/installation-and-usage-instructions.html) for more details). Please go through the exercises in order, as most exercises depend on the previous ones.

We recommend creating a separate notebook for each tutorial. Read the instructions and then **copy paste** the Python code in your own notebook. Execute it and observe the results. In order to note observations or comments, insert a new cell and change its type to "markdown" or "raw". Address your tutor if you have any questions.

Assignment

The questions for the Social Media Analysis assignment are closelly related to the tutorial and they are scattered in the tutorial's text. Your assignment will be graded in the second lab (Grading Lab).

The assignment also needs to be submitted to Wattle as a `jupyter notebook`, containing the text and the analysis required. Please start from this notebook (https://wattlecourses.anu.edu.au/pluginfile.php/1412193/mod_resource/content/12/assignment-submission-notebook-bare.ipynb), which contains already the structure of the questions. Save it in the same folder from which you called "`jupyter notebook`". It should be visible in the Jupyter Dashboard. Fill in the cells indicated to answer each questions.

We encourage you to fill in the questions of the assignment as you go through the tutorial. This notebook represents your answer for the assignment and the corresponding "*.ipynb*" file must be submitted on Wattle.

1.1 Importing and visualizing a social network dataset into a Python graph

This exercise assumes you have Python and the corresponding NetworkX package installed, if not, go back to installation instructions (https://wattlecourses.anu.edu.au/pluginfile.php/1412128/mod_resource/content/10/installation-and-usage-instructions.html). This rest of this tutorial will guide you through a hand-on construction of a graph representation of a social network.

Step 1: A real-life dataset. The data you will be manipulating is issued from a real-life crawl of Twitter posts containing references to a particular Youtube video (https://www.youtube.com/watch?v=iS1g8G_njx8). The tweets were collected during a short period in July 2014. You can either download the **CSV file** (https://wattlecourses.anu.edu.au/pluginfile.php/1412186/mod_resource/content/2/dataset.csv) (Comma Separated Values) into your local folder and load it from there, or you can use it directly from the specified URL using the `urllib2` library.

Step 2: Load and examine the social network dataset.

- Start an jupyter notebook in the same directory, by typing "jupyter notebook".
- Next, we are going to load the aforementioned dataset into NetworkX graph representing the social network. In this social graph, each node represents a Twitter user. Each line of the dataset contains two values (node labels) separated by a comma, signifying that there is an arc between the two nodes. The meaning of an arc is that the second user either retweeted or replied to a tweet of the first user.

In [1]:

```
# necessary imports
import csv
import urllib2
import networkx as nx

# opening a local CSV file
to_read = file("./dataset.csv")  #use this line for a locally downloaded file
# or reading it directly from the specified URL
# url = 'http://rizoiu.eu/sna-lab-ipython/dataset.csv'
# to_read = urllib2.urlopen(url)
reader = csv.reader(to_read)

#  construct the networkx graph
G = nx.Graph()
for line in reader:
    if line[0] not in G:  G.add_node(line[0])
    if line[1] not in G:  G.add_node(line[1])
    G.add_edge(line[0], line[1])
```

Step 3: Visualizing the resulted social graph.

IPython works with the Matplotlib (http://matplotlib.org/) plotting library, which integrates Matplotlib with IPython's display system and event loop handling. To make plots using Matplotlib, you must first enable Jupyter's matplotlib mode. To do this, run the `%matplotlib` magic command to enable plotting in the current Notebook. This magic takes an optional argument that specifies which Matplotlib backend should be used. Most of the time, in the Notebook, you will want to use the `inline` backend, which will embed plots inside the Notebook:

In [2]:

```
%matplotlib inline
```

Next, we will plot the resulted graph, using the a spring layout.

In [3]:

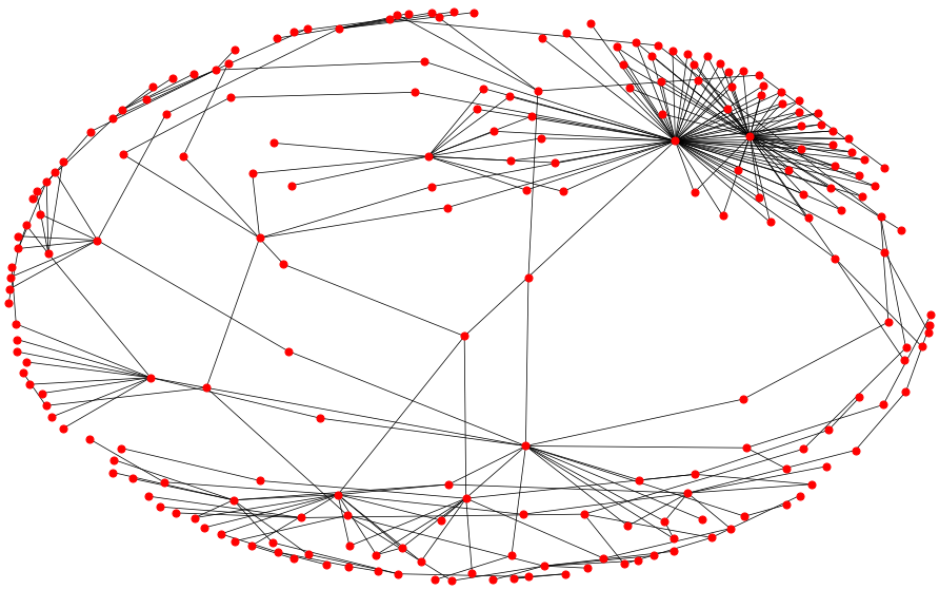
```
import matplotlib.pyplot as plt

# determine the spring layout
pos = nx.spring_layout(G)

# draw the graph
plt.figure(figsize=(19,12))
plt.axis('off')
nx.draw_networkx_nodes(G, pos, node_size=50)
nx.draw_networkx_edges(G, pos, width=0.75, arrows=True)
```

Out[3]:

```
<matplotlib.collections.LineCollection at 0x7f2391cde610>
```



1.2 Exploring the network: basic metrics, centrality, diameter, radius.

Now that we loaded and visualized the graph, lets compute some basic statistics. First of all, how many users do we have in our network?

In [4]:

```
print "The social graph contains %s users." % len(G)
```

```
The social graph contains 217 users.
```

(1 point) Assignment question #1.1: Find another way of determining the number of users, by applying a (`networkx`) graph method.
Similarly, let's see how many relations (edges) are there between our users.

In [5]:

```
print "The social graph contains %s relations." % G.number_of_edges()
```

```
The social graph contains 296 relations.
```

Next, we want to know the relations between given users. Does user `'2568857825'` know the users `'1584106808'` and `'2532489048'`? If not, who can introduce them? What is the optimum path that leads one user to the other?

In [6]:

```
print nx.shortest_path(G, source = '2568857825', target = '1584106808')
print nx.shortest_path(G, source = '2568857825', target = '2532489048')
```

```
['2568857825', '1584106808']
['2568857825', '457728637', '85452649', '78598378', '109670729', '29
1972556', '2260701138', '54837666', '2532489048']
```

So, it seems that, while `'2568857825'` and `'1584106808'` are direct acquaintances, `'2568857825'` needs a long chain of introductions (7 intermediary users, 8 introductions) to get to `'2532489048'`. We define an introduction as an interaction between 2 users in a chain, therefore an introduction is an edge on a shortest path between 2 users.
How many introductions does `'2568857825'` need to reach anyone in the graph? (this means, what is the longest **shortest path** from `2568857825` to anyone in the network).

In [7]:

```
no_hops = max([len(nx.shortest_path(G, source = '2568857825', target = x)) - 1 f
or x in G.nodes()])
print "Using %d introductions, '2568857825' can reach anyone in the network" % n
o_hops
```

```
Using 13 introductions, '2568857825' can reach anyone in the network
```

(0.5 points) Assignment question #1.2: What is the minimum number of introductions required for the user `'137056623'` to reach any other user?
(0.5 points) Assignment question #1.3: What is the minimum number of introductions required for the any user to reach any other user in the network?
HINT: study the `shortest_path_length` method description and compute the shortest distances between all pairs of nodes as shown previously (*e.g.*, when calculating `no_hops`).

The key concepts in the next exercises are **Eccentricity, Diameter and Radius**, which are based on the notion of *shortest distance* that we have just seen here above.

Let $d(u, v)$ be the shortest distance, or graph geodesic, between two nodes u and v.

The *eccentricity* $\epsilon(v)$ of a node v is the greatest geodesic distance between v and any other node. *i.e.*, $\max_{u \neq v} d(u, v)$.
It can be thought of as how far a node is from the node most distant from it in the graph wikipedia (http://en.wikipedia.org/wiki/Distance_%28graph_theory%29).

`Networkx` provides a very convenient method of computing the *eccentricity* of nodes in our social network:

In [8]:

```
ec = nx.eccentricity(G)
print ec
```

```
{'2568857825': 13, '218342185': 11, '984586015': 11, '2532489048': 1
3, '634964760': 13, '2570186883': 13, '1895195947': 14, '273216391':
13, '2513030970': 13, '477534590': 13, '86827273': 11, '109670729':
 9, '2567515295': 15, '78598378': 10, '2438593561': 15, '180502452':
11, '2304891050': 14, '121811715': 16, '230421055': 10, '436229017':
18, '2324917405': 16, '2443017820': 12, '2381587875': 13, '22137991
2': 12, '2260701138': 11, '277180205': 18, '2518865954': 15, '303291
881': 18, '54837666': 12, '270957052': 15, '2231764297': 11, '271475
828': 17, '2531272770': 13, '849520898': 12, '19405774': 13, '534122
115': 13, '91274231': 13, '2269421607': 12, '112878476': 10, '862277
491': 13, '1715125316': 14, '79963913': 18, '914854934': 15, '256942
6702': 13, '553613442': 14, '154519365': 11, '2299265379': 13, '1442
3603': 13, '82243231': 13, '2301616502': 13, '100220864': 14, '21684
15412': 13, '155999283': 14, '41397576': 13, '925042686': 14, '25063
05612': 13, '2531194285': 13, '157883615': 13, '24681779': 13, '1627
50476': 13, '252104868': 11, '137056623': 10, '1331009185': 13, '124
7000905': 12, '547897508': 13, '1493478936': 12, '46896918': 11, '26
8408823': 15, '51589111': 11, '430885477': 12, '964880227': 17, '257
0263948': 13, '122764720': 11, '2570337129': 15, '1181059135': 14,
 '98977944': 18, '2531817618': 13, '115833820': 11, '17941804': 15,
 '2264549296': 13, '30869959': 13, '1135161234': 13, '134552406': 1
3, '78461093': 11, '2163620169': 11, '549417679': 13, '2455207552':
 13, '18979962': 11, '718022762': 16, '785539429': 13, '2570401007':
13, '52096506': 13, '458710122': 13, '2532487142': 15, '426088047':
 13, '1068847219': 13, '36483808': 18, '1176702475': 13, '216733686
1': 13, '108280622': 11, '21758143': 12, '464141254': 13, '62908593
8': 14, '812727967': 13, '2488974288': 13, '2506839308': 11, '167680
0584': 15, '2570031521': 13, '75870003': 9, '1337915606': 12, '61375
153': 11, '414667836': 13, '85452649': 11, '1949817667': 15, '265545
240': 13, '398714008': 11, '1415776814': 13, '2569636498': 13, '8119
1380': 11, '39340804': 13, '350884062': 13, '600126522': 12, '383272
345': 17, '2449922472': 16, '194333435': 12, '560305806': 13, '22931
9409': 13, '2367911': 11, '18222378': 12, '1172235157': 13, '2345903
05': 11, '1205773280': 17, '2367637476': 12, '2231749961': 13, '1546
777802': 12, '768227683': 13, '2531372664': 13, '2312215918': 11, '3
45159720': 13, '453723348': 13, '2461880769': 14, '2532354840': 13,
 '265547360': 13, '163075938': 12, '187429274': 15, '1275810296': 1
2, '53207221': 11, '908956538': 13, '192629085': 13, '1584106808': 1
4, '212084981': 13, '562912303': 13, '2532406316': 15, '311623106':
 13, '245816235': 11, '930456511': 14, '353514773': 12, '229622310
7': 13, '424787224': 12, '2570247160': 13, '57384804': 13, '2441631
6': 16, '1387419408': 18, '1230063518': 18, '2529758732': 17, '40744
4317': 13, '2350341051': 12, '2558451016': 13, '135628775': 13, '253
2533341': 15, '2356886552': 11, '2532392647': 13, '2528800812': 13,
 '269681736': 14, '2392021854': 13, '945116653': 13, '242946772': 1
5, '145429546': 11, '344998189': 15, '2532409500': 15, '2570262069':
15, '147562565': 12, '1262622745': 13, '479245046': 12, '257035455
3': 13, '1411424238': 13, '2532430730': 13, '1199191586': 13, '22352
07248': 17, '510241616': 11, '19559692': 11, '1079404754': 15, '1537
445646': 14, '70021987': 11, '138886151': 12, '79293791': 14, '63484
7847': 17, '748288704': 13, '35518242': 13, '710108896': 13, '524667
128': 12, '291972556': 10, '1266468632': 13, '193234037': 14, '39593
9764': 18, '96687193': 12, '460910822': 15, '297124905': 11, '320742
970': 14, '226515407': 12, '409300519': 13, '457728637': 12, '865843
90': 11, '292837433': 16, '142696793': 10, '591732724': 14, '1318229
22': 11}
```

So let's find out which seems the most central user in our network (the user with the lowest *eccentricity*). Note that, if there are multiple users with the same cetrality, this will output only one of them.

In [9]:

```
import operator

ec = nx.eccentricity(G)
print "The most eccentricity-wise central user is '%s'" % min(ec.iteritems(), ke
y=operator.itemgetter(1))[0]
```

```
The most eccentricity-wise central user is '109670729'
```

The *radius* r of a graph is the minimum eccentricity of any node, $r = \min_v \epsilon(v)$. It can be computed as:

In [10]:

```
print min(ec.values())
## OR
print nx.radius(G)
```

```
9
9
```

(0.5 points) Assignment question #1.4: The *diameter* d of a graph is the maximum eccentricity of any node, $d = \max_v \epsilon(v)$. Give two ways to compute the diameter of the social network G (one using the calculated eccentricity values calculated earlier and another one using the dedicated `networkx` function).

Recall that the betweenness centrality of a *node* v is defined as:

$$c_B(v) = \sum_{s,t \in V} \frac{\sigma(s,t|v)}{\sigma(s,t)}$$

Now define the betweenness centrality of an *edge* e as

$$c_B(e) = \sum_{s,t \in V} \frac{\sigma(s,t|e)}{\sigma(s,t)}$$

here V is the set of nodes in an undirected graph, $\sigma(s,t)$ is the number of shortest paths between node s and node t, $\sigma(s,t|v)$ is the number of those paths passing through node v, and $\sigma(s,t|e)$ is the number of those paths passing through *edge* e.

(0.5 points) Assignment question #1.5: Determine ALL most central node(s) and ALL most central edge(s), with respect to the node and edge betweenes centrality. Remember that two or more nodes/edges may have the same centrality score and they ALL need to be determined for this assignment.
Hint: `networkx` already contains dedicated function to compute node and edge betweenness centrality scores.

Optional assignments

The purpose of the optional assignments here after is to guide you into a more in depth analysis of the social network. These assignments are optional and **they will not be graded**. Whatsoever, if you have questions regarding them, you may discuss them with your tutor or post them on the forum.

Optional assignment question: Determine the number of cliques (http://en.wikipedia.org/wiki/Clique_%28graph_theory%29) and the maximum size of cliques in our social network.

Optional assignment question: What is the meaning of *cliques* in a social network? Interpret based on the definition of cliques in graphs.