Clustering Real vs Random on snap dataset (Social Network Analysis)

October 23, 2020

By Subhodeep Sinha Dated 23-10-2020
Clustering Real vs Random on snap dataset (Social Network Analysis)
Objective To calculate a particular graph parameter on a real social network dataset and compare it to the value of the same parameter on a randomly generated graph.

In [86]:
```
import networkx as nx;
import numpy as np;
from random import random;
```

In [58]:
```
# Read data from the dataset, and create graph G_fb
G_fb = nx.read_edgelist("facebook_combined.txt", create_using = nx.Graph(), nodetype=
```

In [68]:
```
# Show the number of edges in G_fb
print("edges = " + str(G_fb.number_of_edges()));

edges = G_fb.number_of_edges()
type(edges)
```

```
edges = 88234
```

Out[68]:
```
int
```

In [72]:
```
# Show number of nodes in G_fb
print("nodes = " + str(G_fb.number_of_nodes()));

nodes = G_fb.number_of_nodes()

x = [nodes*(nodes-1)/2]
```

```
nodes = 4039
```

Out[72]:
```
[8154741.0]
```

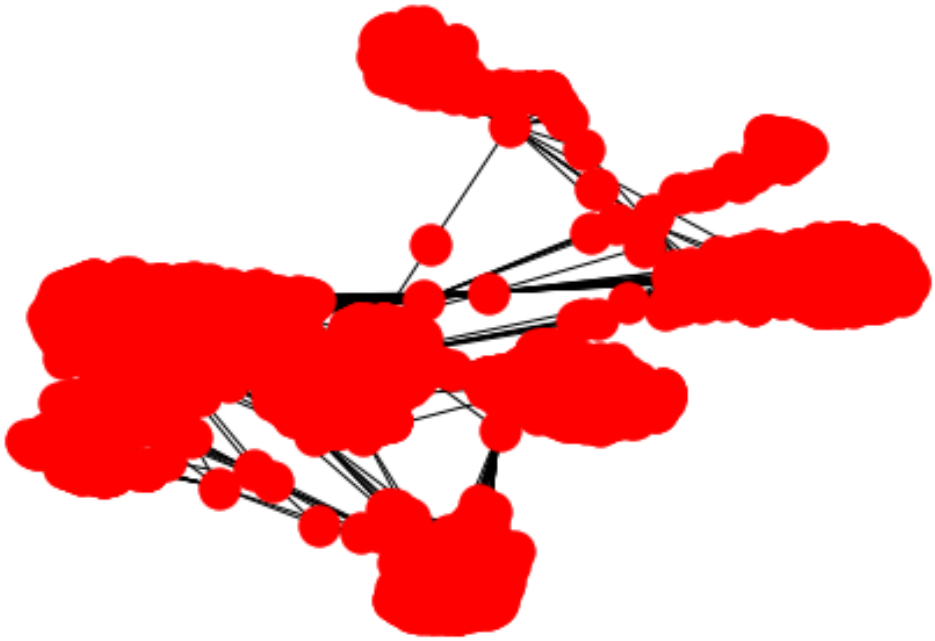
In [76]:
```
# TASK1. Now your task is to compute the probability whether there is an edge between
## edge_probab = ...

edge_probab = (edges / 8154741.0 )

print("Edge probab: " + str(edge_probab))
```

```
Edge probab: 0.010819963503439287
```

In [79]:
```
nx.draw(G_fb)
```



In [80]:
```
G_fb.size()
```

Out[80]:
```
88234
```

In [81]:
```
# TASK2. Compute the ACC (average clustering coefficient) of G_fb
# (consult the NetworkX manual or the video lecture for the correct function which do
## av_clust_coeff =  ...

av_clust_coeff_1 = nx.average_clustering(G_fb)
print(av_clust_coeff_1)
```

```
0.6055467186200876
```

In [82]:
```
# Now we have to generate a random graph. First we initialize it
G_rand = nx.Graph();

'''
# TASK3. generate edges in G_rand at random:
for i in range(0,k) :
    for j in range(0,i) :
        G_rand = nx.random_graphs(4039,0.045776004714735814)
        # Add an edge between vertices i and j, with probability edge_probab (as in G
        # ...
'''

G_rand = nx.erdos_renyi_graph(4039,0.0108199635)

import numpy as np
print(np.average(total_edges))
```

```
374799.8
```

In [83]:
```
nx.draw(G_rand)
```

In [84]:
```
# Now we print out the number of edges and the ACC of the new graph
print("rgraph_edges = " + str(G_rand.number_of_edges()));
```

```
rgraph_edges = 87909
```

```
In [85]: # av_clust_coeff = ...

         av_clust_coeff_2= nx.average_clustering(G_rand)
         print("rgraph_acc = " + str(av_clust_coeff_2));
```

```
rgraph_acc = 0.010682206700308675
```