Untitled

December 15, 2017

```
In [1]: #import dataset
        import csv
        with open('Persons.csv', 'rb') as f:
            reader = csv.reader(f)
            nodes = list(reader)

        with open('EmailReceivers.csv', 'rb') as f:
            reader = csv.reader(f)
            edges = list(reader)

        num_of_people=len(nodes)
        num_of_emails=7945

        #CLEAN UP DATA
        person_nodes=['']*len(nodes)
        count=0
        for n in nodes:
            person_nodes[count]=nodes[count][1]
            count+=1

        #generate edge data
        count=0
        email_nodes=[0]*len(edges)
        elist=[]*len(edges)
        tup=()
        for e in edges:
            email_nodes[count]=edges[count][2]
            tup=(int(edges[count][1])+num_of_people,int(edges[count][2]))
            elist.append(tup)
            count+=1
```

Create Networkx graph from data

```
In [6]: import networkx as nx
        G=nx.Graph()
        for i in range(num_of_people+num_of_emails):
            G.add_node(i)
        G.add_edges_from(elist)
        #nx.write_gml(G,"test.gml")
        print nx.info(G)
```

```
Name:
Type: Graph
Number of nodes: 8459
Number of edges: 9206
Average degree:   2.1766
```

```
In [3]: #code to add Email alias to nodes
        mapping={}
        mapping[0]='0'
        for num in range(len(nodes)):
                mapping[num+1]=person_nodes[num]
```

```
In [4]: #H=nx.relabel_nodes(G,dict(zip(G.nodes(),person_nodes)))
        H=nx.relabel_nodes(G,mapping)
        nx.write_gml(G,"final.gml")
```

```
In [5]: #get largest connected component for analysis
        connected=nx.is_connected(G)
        graphs = list(nx.connected_component_subgraphs(G, copy=True))
        # the connected component with the most nodes
        graph_max = sorted([(len(gn.nodes()), gn) for gn in graphs], key=lambda x: x[0], reverse
        print nx.info(graph_max)
        G2=graph_max
```

```
Name:
Type: Graph
Number of nodes: 7834
Number of edges: 9009
Average degree:   2.3000
```

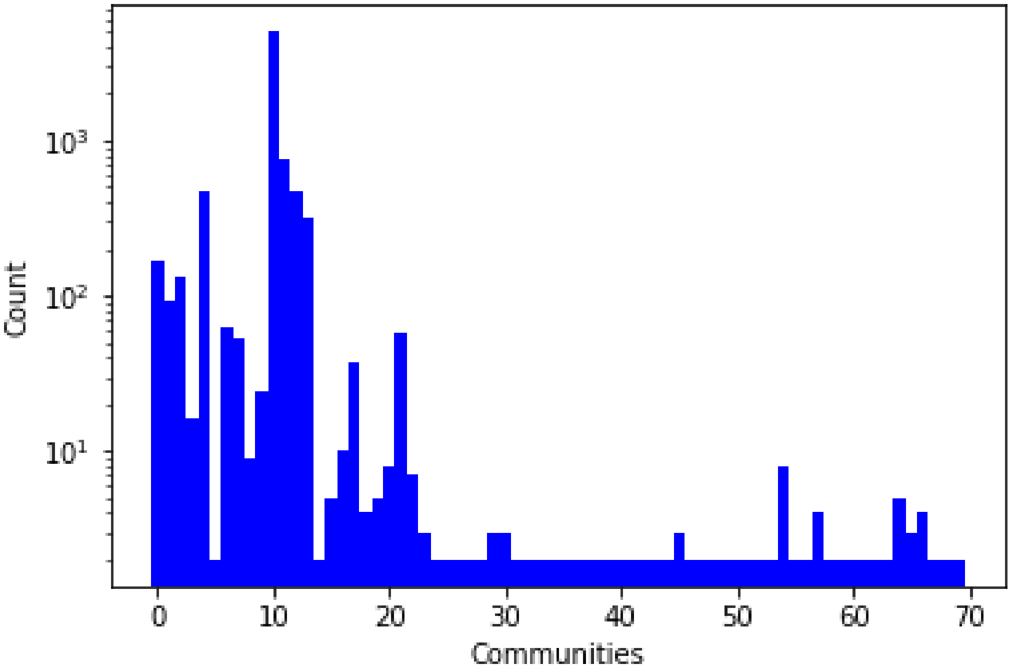
Community Detection

```
In [8]: import community
        part=community.best_partition(G2)
        nx.set_node_attributes(G2, part,"com")
        nx.write_gml(G2,"finalg2.gml")
```

```
In [9]: import matplotlib.pyplot as plt
        com={}
        for p in range(len(part.items())):
            c=part.items()[p][1]
            if c not in com:
                com[c]=0
```

```
    com[c]+=1

items=sorted(com.items())
fig=plt.figure()
ax=fig.add_subplot(111)
ax.bar([k for (k,v) in items], [v for (k,v) in items],1.0,color='b')
#ax.set_xscale('log')
ax.set_yscale('log')
plt.ylabel('Count')
plt.xlabel('Communities')
plt.savefig('com.png', bbox_inches='tight')
plt.show()
print items
```



```
[(0, 170), (1, 94), (2, 134), (3, 16), (4, 474), (5, 2), (6, 63), (7, 53), (8, 9), (9, 24), (10,
```

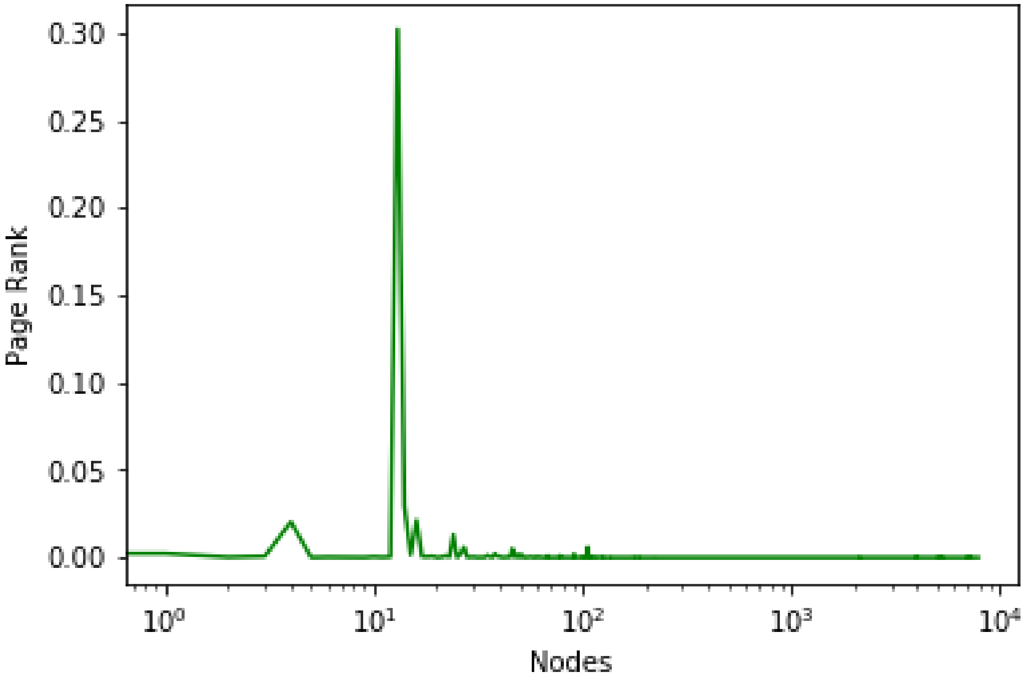
```
In [13]: sorted(items)
```

```
Out[13]: [(0, 170),
          (1, 94),
          (2, 134),
          (3, 16),
          (4, 474),
          (5, 2),
(6, 63),
(7, 53),
(8, 9),
(9, 24),
(10, 5004),
(11, 760),
(12, 468),
(13, 315),
(14, 2),
(15, 5),
(16, 10),
(17, 37),
(18, 4),
(19, 5),
(20, 8),
(21, 58),
(22, 7),
(23, 3),
(24, 2),
(25, 2),
(26, 2),
(27, 2),
(28, 2),
(29, 3),
(30, 3),
(31, 2),
(32, 2),
(33, 2),
(34, 2),
(35, 2),
(36, 2),
(37, 2),
(38, 2),
(39, 2),
(40, 2),
(41, 2),
(42, 2),
(43, 2),
(44, 2),
(45, 3),
(46, 2),
(47, 2),
(48, 2),
(49, 2),
(50, 2),
(51, 2),
(52, 2),
(53, 2),
 (54, 8),
 (55, 2),
 (56, 2),
 (57, 4),
 (58, 2),
 (59, 2),
 (60, 2),
 (61, 2),
 (62, 2),
 (63, 2),
 (64, 5),
 (65, 3),
 (66, 4),
 (67, 2),
 (68, 2),
 (69, 2)]
```

Page Rank Analysis

```
In [12]: import matplotlib.pyplot as plt

         pr=nx.pagerank(G2) #calc page rank value of each node in graph
         lists = sorted(pr.items())
         x, y = zip(*lists)
         #plt.plot(range(len(x)),y,'g')
         plt.plot(y,'g')
         plt.xscale('log')
         #plt.yscale('log')
         plt.ylabel('Page Rank')
         plt.xlabel('Nodes')
         plt.savefig('pr.png', bbox_inches='tight')
         plt.show()
```

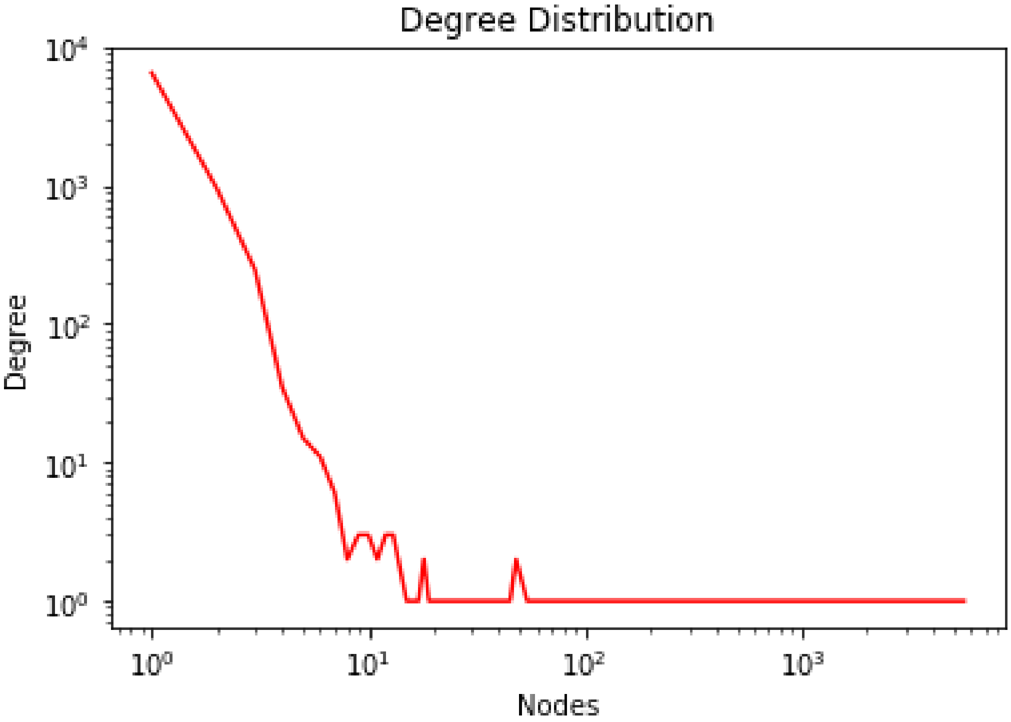


```
In [24]: degs = {}
         for n in G2.nodes():
             deg = G2.degree(n)
             if deg not in degs:
                 degs[deg] = 0
             degs[deg] += 1

         items = sorted(degs.items())
         fig = plt.figure()
         ax = fig.add_subplot(111)
         ax.plot([k for (k,v) in items], [v for (k,v) in items],'r')
         ax.set_xscale('log')
         ax.set_yscale('log')
         plt.ylabel('Degree')
         plt.xlabel('Nodes')
         plt.title("Degree Distribution")

         #ab = fig.add_subplot(212)
         #ab.plot([k for (k,v) in items], [v for (k,v) in items],'r')
         #ab.set_xscale('log')
         #ab.set_yscale('log')
         #plt.ylabel('Degree')
         #plt.xlabel('Nodes')
         #plt.title("Degree Distribution")
```

```
#fig.tight_layout()
fig.savefig("degree_distribution.png")
plt.show()
```



```
In [ ]:

In [ ]:
```