GAME OF THRONES
Network Analysis

Nazli Begum
CIRPANLI

Introduction

- Game of Thrones is a popular series adapted from George R. R. Martin's fantasy novel series A Song of Ice and Fire.
- It is set in a fantasy world that is ruled by a king and is home to several noble houses.
- This network analysis aims to show how the characters of this series are connected to each other

Dataset contains

- 107 nodes
- 352 edges

- Nodes are characters
- Edges created by connecting two characters whenever their names appeared within 15 words of one another in the books. The edge weight corresponds to the number of interactions.

Preprocessing and reading the dataset

```
got_graph=nx.Graph()
```

```
with open('got_nodes.txt', 'r') as nodecsv: # Open the file
    nodereader = csv.reader(nodecsv) # Read the csv
    # Retrieve the data (using Python list comprehension and list slicing to remove the header row)
    node = [n for n in nodereader][1:]

nodes = [n[0] for n in node] # Get a list of only the node names

with open('got_edges.txt', 'r') as edgecsv: # Open the file
    edgereader = csv.reader(edgecsv) # Read the csv
    edges = [tuple(e[0:3]) for e in edgereader][1:] # Retrieve the edges along with the weights
```

```
got_graph.add_nodes_from(nodes) # Add nodes to the Graph
got_graph.add_weighted_edges_from(edges,weight='Weight') # Add edges and edge weights to the Graph
```

```
# adding attributes to nodes
label_dict={}

for n in node:
    label_dict[n[0]] = n[1]
nx.set_node_attributes(got_graph, label_dict, 'label')
```

General Structure

- Type of the graph: Undirected
- Number of Nodes: 107
- Number of edges:352

- One large connected component
- Diameter: 6
- Average Shortest Path Length:2.904

- Average Degree :6.5794
  - Standard Deviation:6.580965732013492
  - Median: 4
  - Minimum Degree: 1
  - Maximum Degree: 36
- Average Weighted Degree : 80.822
- Density: 0.062

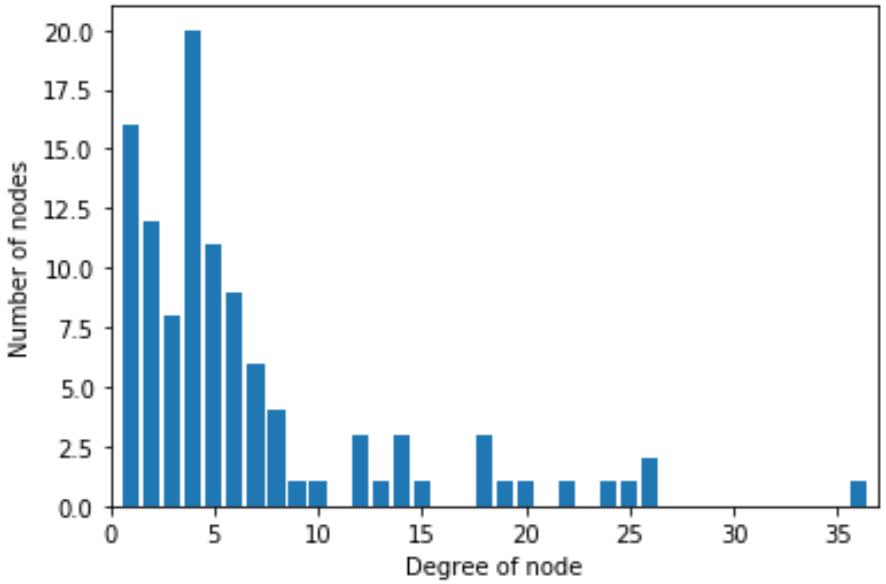
Nodes with high degree :

(Higher than 98% of the nodes)

(**'Tyrion',** 0.3396226415094339),

(**'Jon',** 0.24528301886792453),

(**'Sansa',** 0.24528301886792453)



First Look

Node sizes are ranked by degree

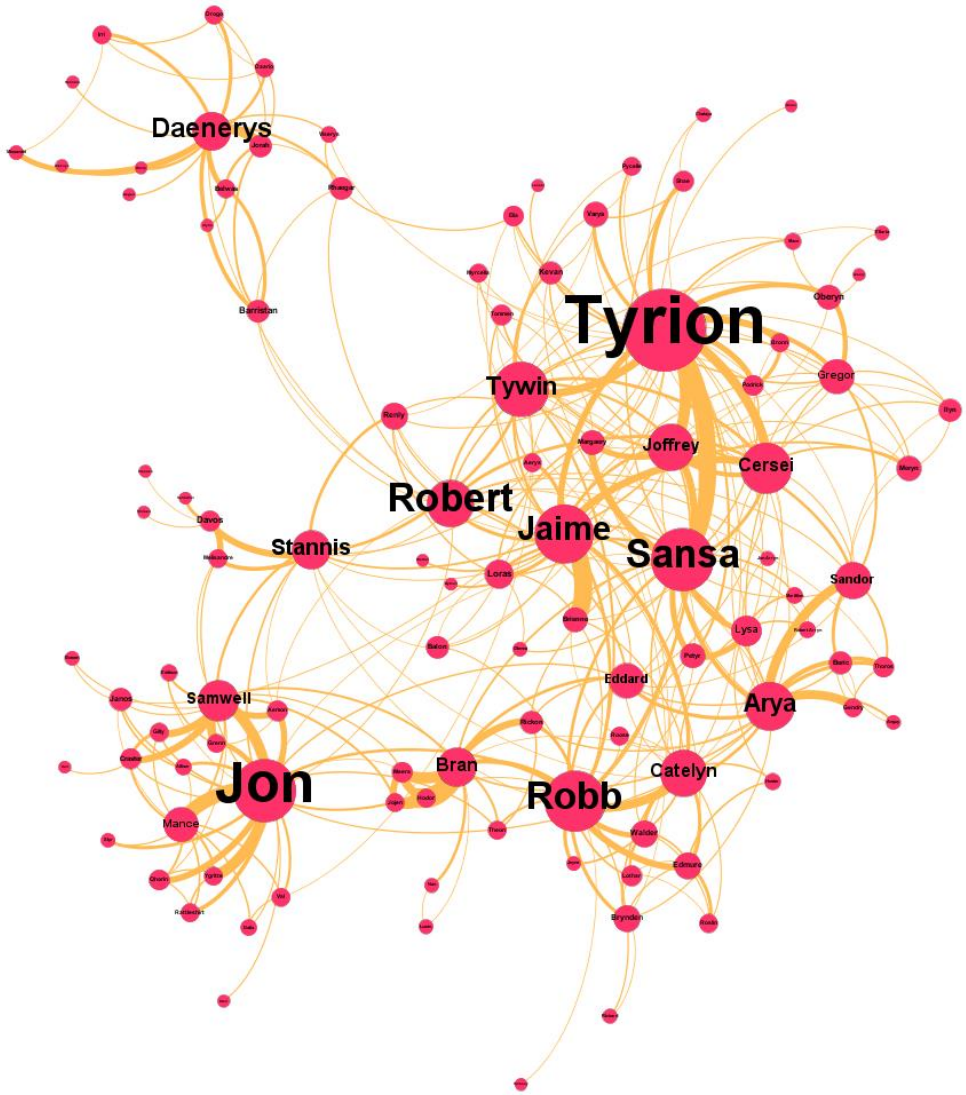
Label sizes are ranked by betweenness centrality

Thicker edges represents interaction strength

Important characters are distinctly visible.

Tyrian has the most connected node, Jon and Sansa follows

Jon has the largest betweenness centrality, Robert and Tyron follows with very close values.

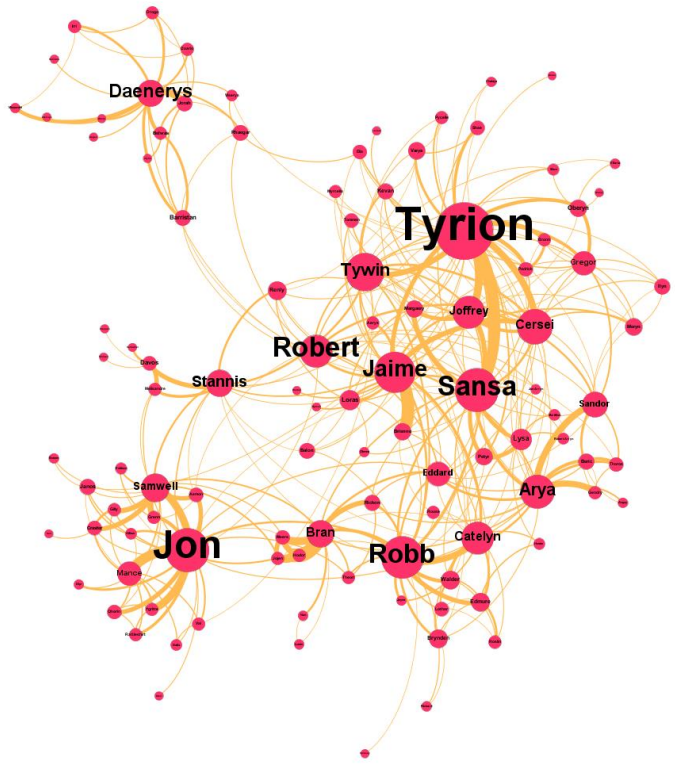


HUBS

```
degree:       ('Tyrion', 0.33962264150943394)
betweenness:  ('Jon', 0.22996466368473173)
closeness:    ('Tyrion', 0.5120772946859904)
eigenvector:  ('Tyrion', 0.33663766254996313)
```

- Tyrian Lannister seems like the most important character in the story. He has not only the largest number of connections but also the important ones. He is also the closest node to the others.

- Jon Snow controls the information flow on the network. He has the greatest betweenness centrality

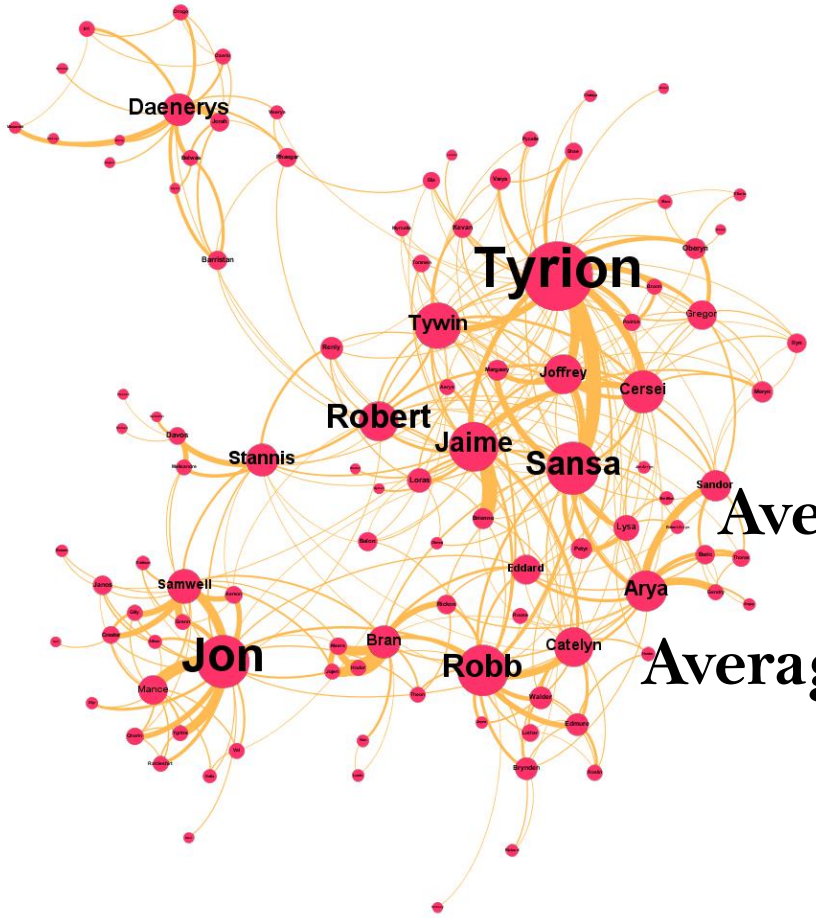


- Connected graph

- No isolates in the network

Global Clustering Coefficient

the degree to which nodes in the network tend to cluster together



Transitivity:

0.32866

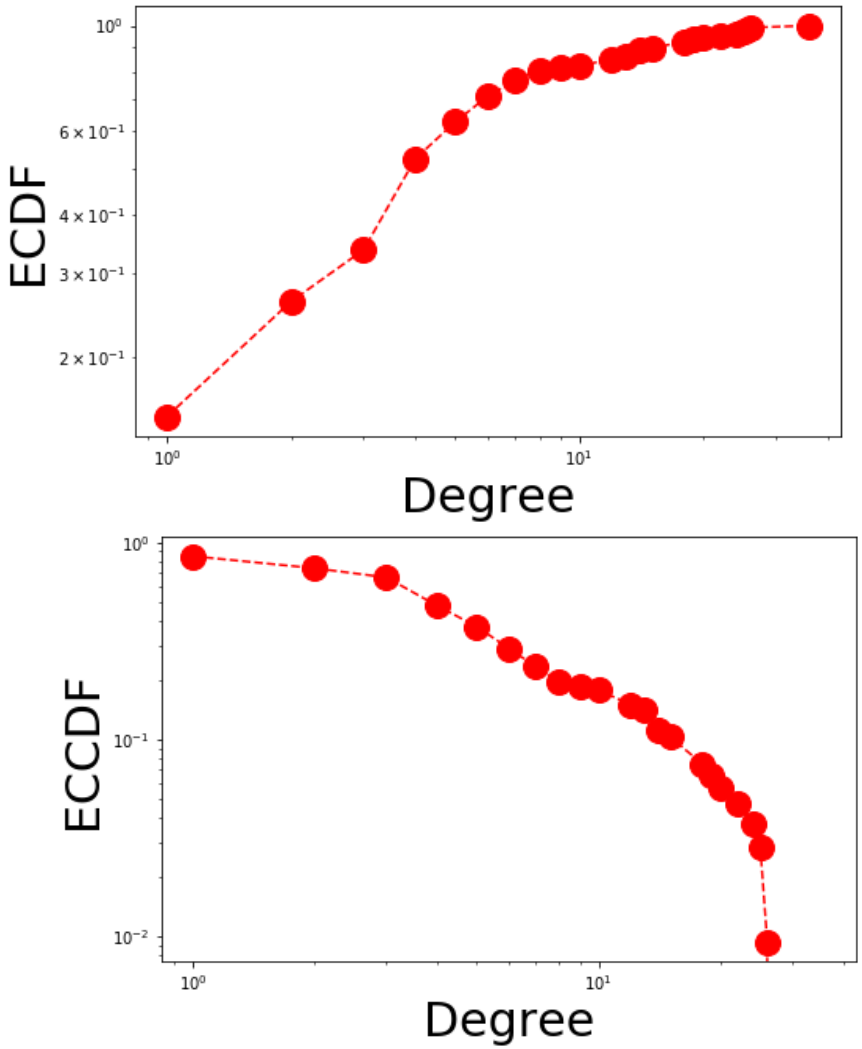
Average Clustering Coefficient:

0.5514

Average Clustering Coefficient of the random graph:

0.0709

Degree distrubition of the network



Comparison with a random graph

Creating a random graph with p is equal to the density of our network

```
mean_degree_got=np.mean(got_graph_degree)
p= mean_degree_got/(got_graph.order()-1)
p #same as the density of the network
```

```
0.062070181625815554
```

```
random_graph = nx.fast_gnp_random_graph(got_graph.order(),p)
```

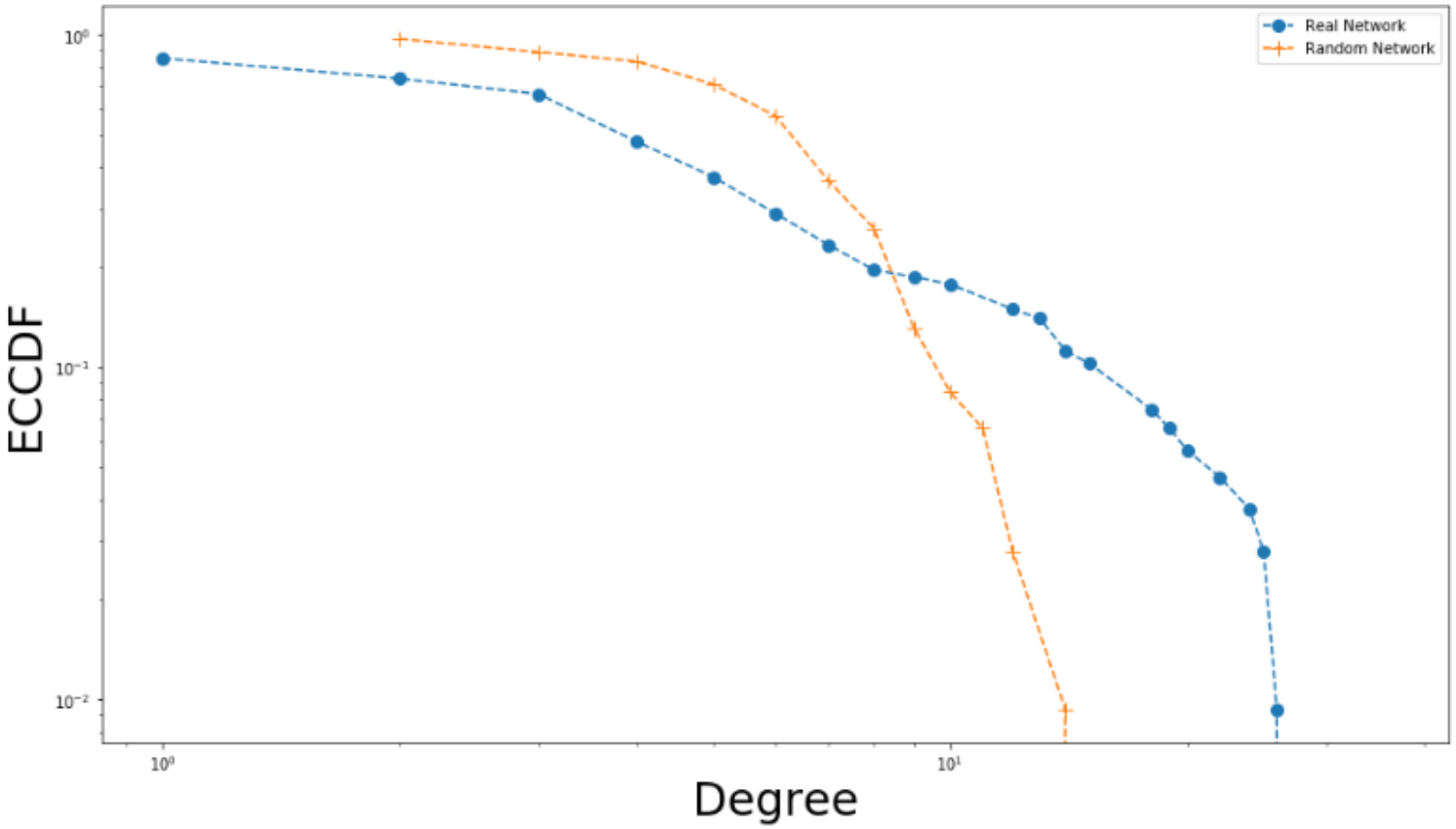
```
print('Number of nodes: {}'.format(random_graph.order()))
print('Number of links: {}'.format(random_graph.size()))
```

```
Number of nodes: 107
Number of links: 314
```

Average degree of the random graph is: **5.8691**

Average degree of our graph: **6.5794**

Comparison with a random graph



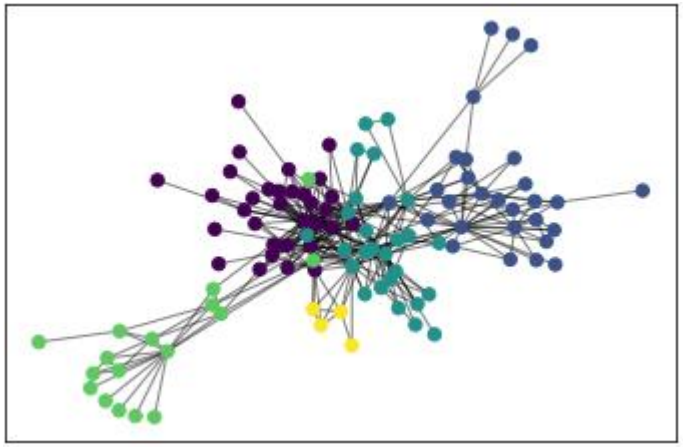
Degree distributions of two networks

Random network with 107 nodes and 314 links, similar node degrees

Game of Thrones network with 107 nodes and 352 links, few nodes with high degrees and many nodes with lower degrees.

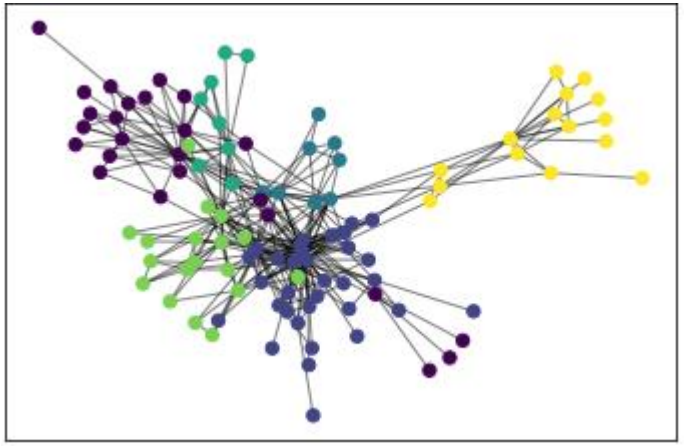
COMMUNITIES

Community detection with Greedy Algorithm



Coverage: 0.7301136363636364
Modularity :0.458431527634295 57
Performance: 0.792100158702 1689

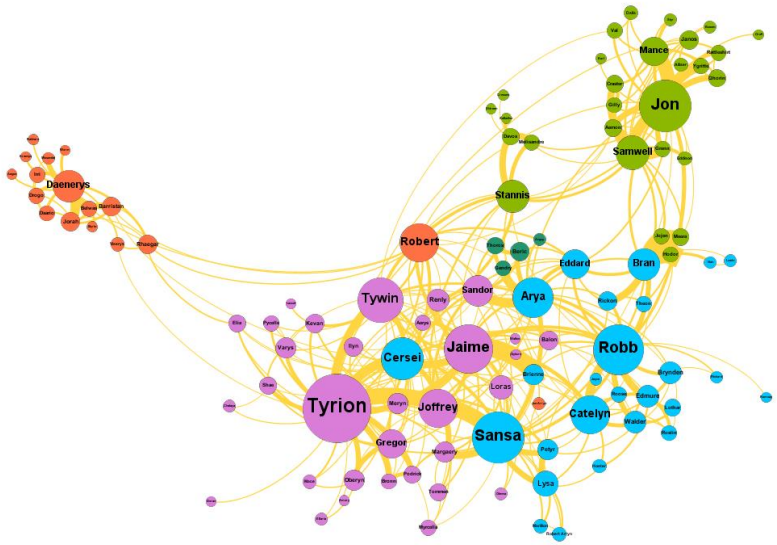
Community detection with Louvain Algorithm



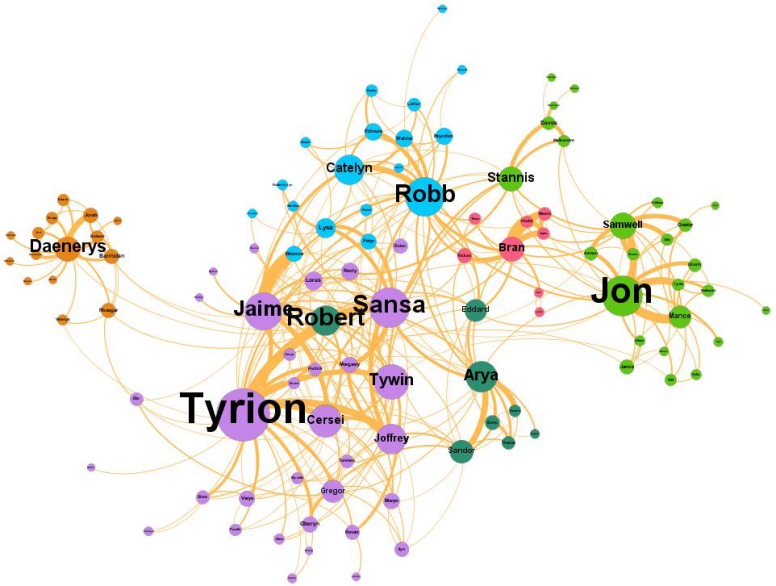
Coverage 0.7301136363636364
Modularity 0.493333548553 7178
Performance 0.827190971609 9454

COMMUNITIES

Greedy Algorithm



Louvain Algorithm

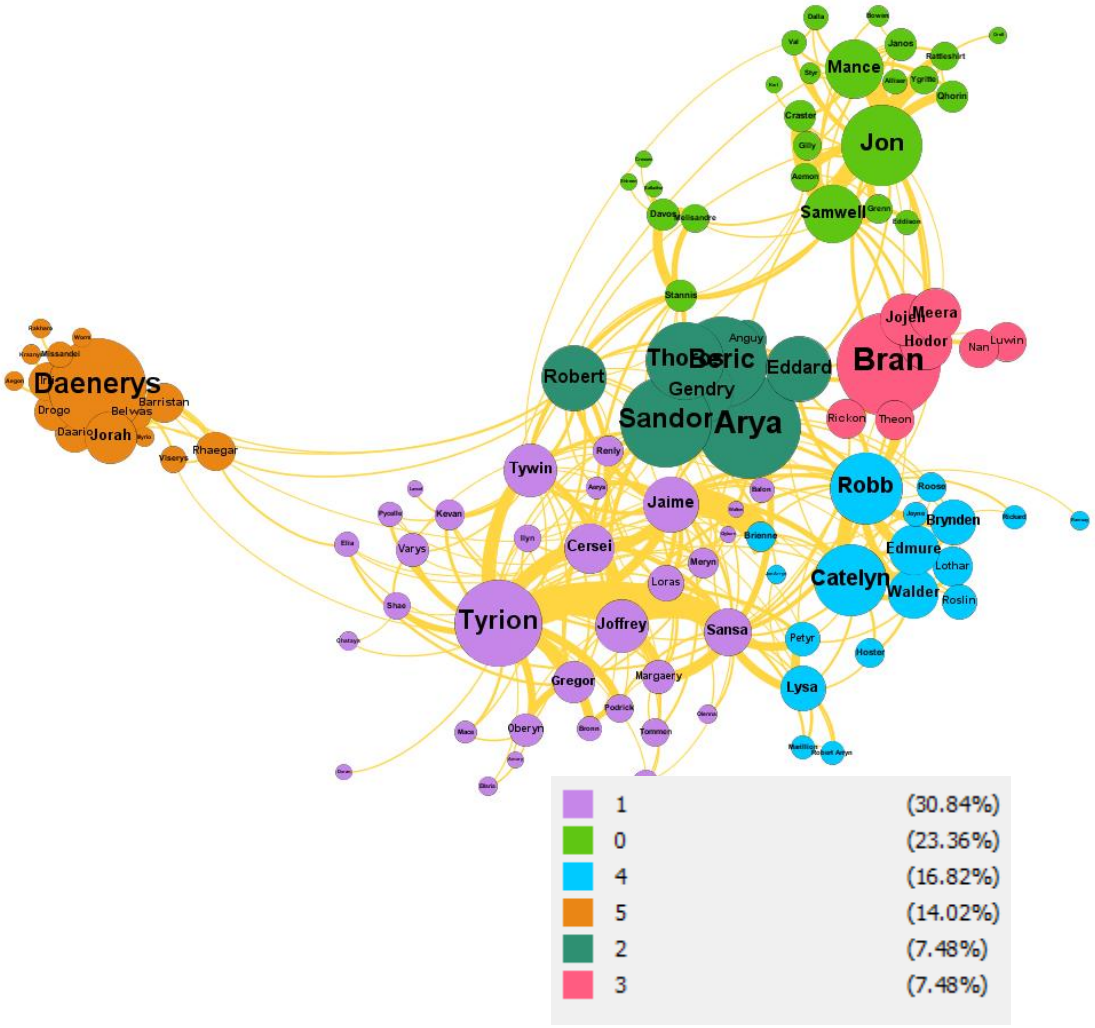


COMMUNITIES

- Six different communities on the network structure

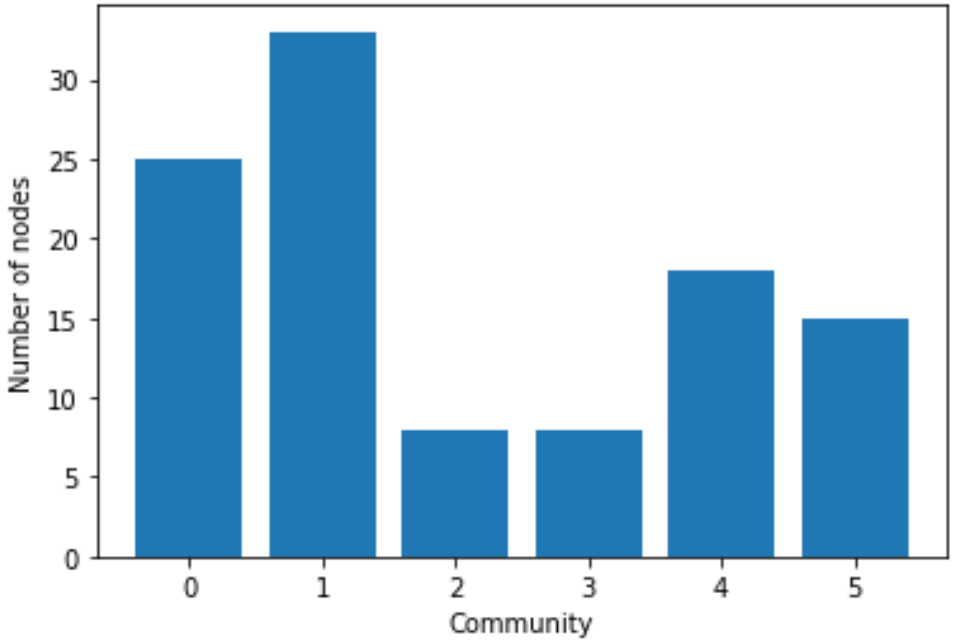
- 0. John Snow and the Wall
- 1. Lannisters/King's Landing
- 2. Arya and friends
- 3. Brand and his friends
- 4. Robb&Catelyn and their army
- 5.Daerenys and the people of Essos

- Node sizes adjusted to reflect the central nodes for each community



COMMUNITIES

Size distribution of the communities
(Louvain Algorithm)

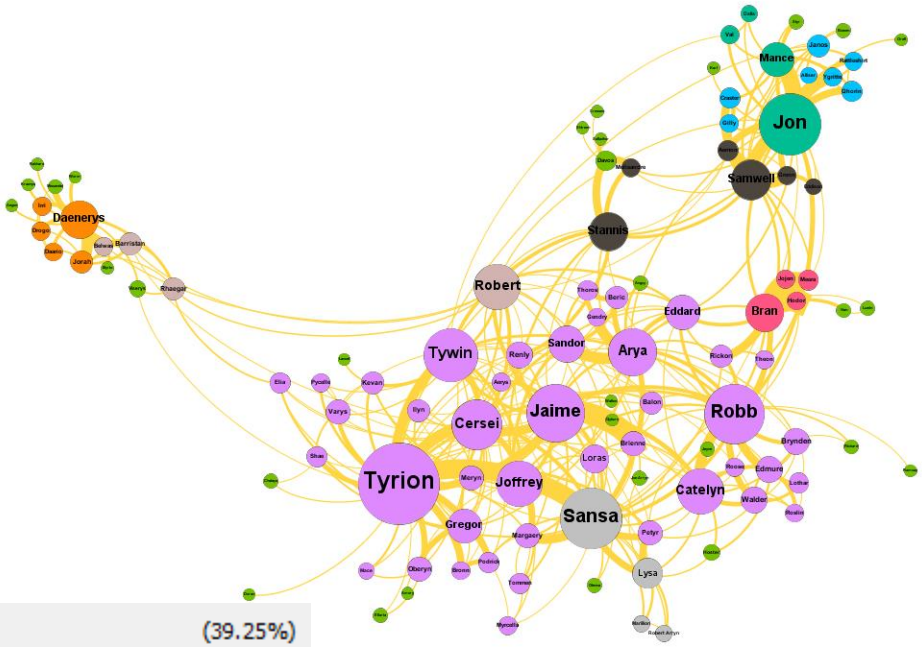


COMMUNITIES

k_4 Clique

8 communities

28.97 % of nodes not assigned to any community



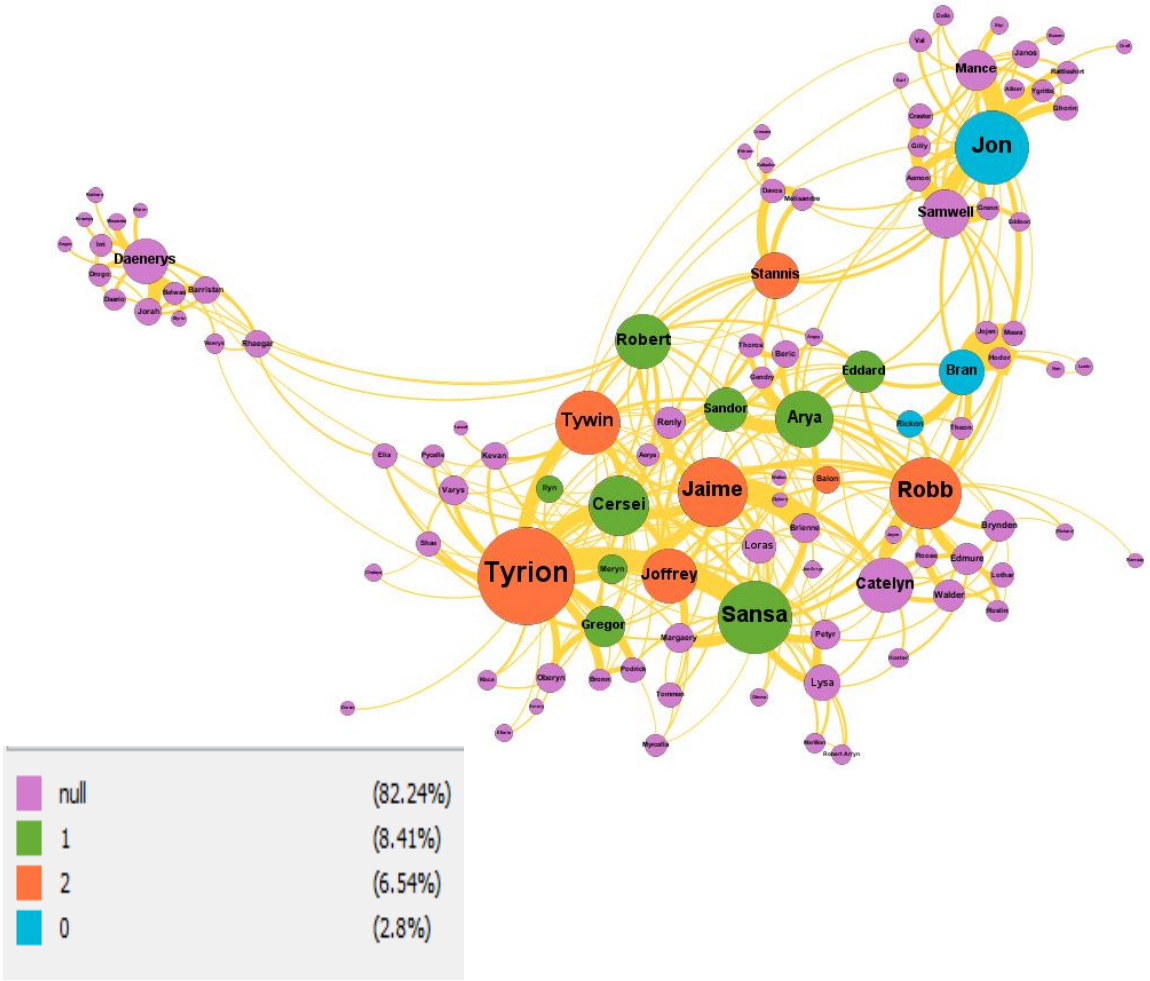
1	(39.25%)
null	(28.97%)
0	(6.54%)
3	(5.61%)
6	(4.67%)
2	(3.74%)
4	(3.74%)
5	(3.74%)
7	(3.74%)

COMMUNITIES

k_6 Clique

3 communities

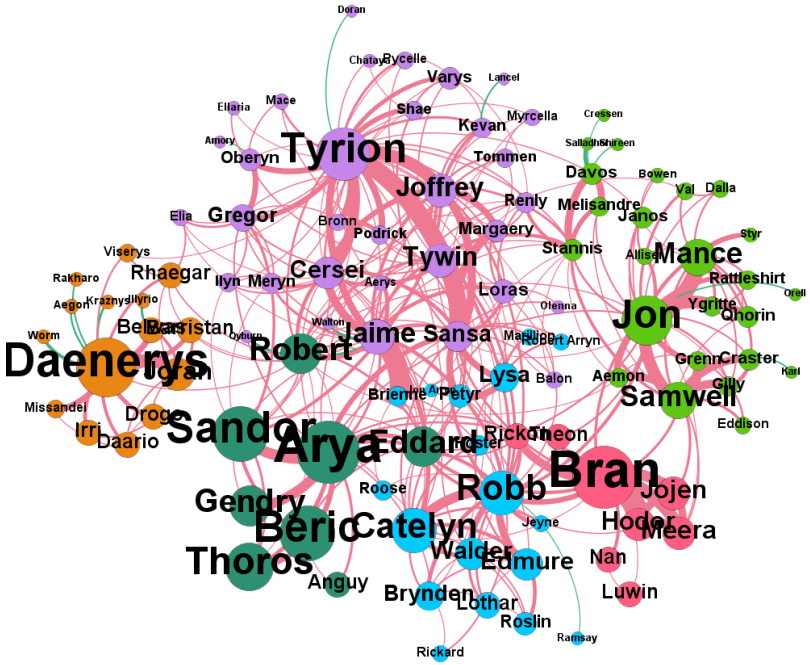
82.24% of nodes not assigned to any community



Bridges and Local Bridges

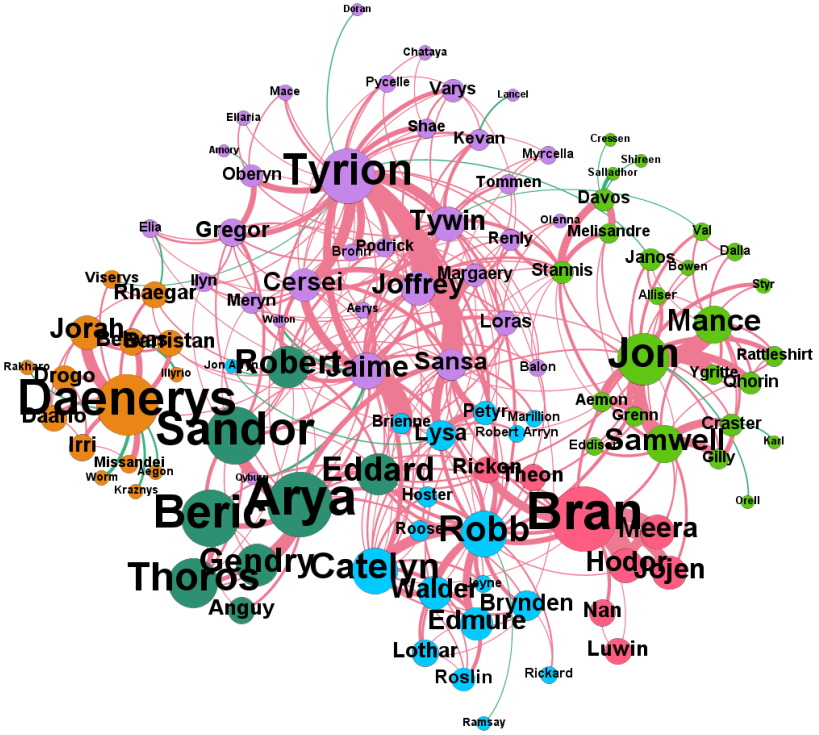
Bridges

An edge in the graph whose removal disconnects the graph



Local Bridges

an edge whose endpoints have no common neighbors.



Conclusion

- Sparse network
- Several important nodes both locally and globally.
- Few high degree nodes, many low degree nodes
- Multiple hubs
- Six communities given by the Louvain algorithm (best partition)