HW Week6 SNA

짱짱조

[1] Dataset

본 실습에서는 kaggle에 있는 Star Wars Social Network 데이터를 이용하였다. 이 데이터에서 node는 Star Wars에 나오는 캐릭터이고, 두 node 간 edge의 value는 두 캐릭터가 대화를 나눈 장면 수이다. 이를 통해 네트워크를 구성하여 SNA를 실시하면, 영화의 등장인물 간 관계를 보다 객관적인 지표로 살펴보고, 핵심 인물이 누구인지 밝혀낼 수 있다.

```
# load json
import json
with open('starwars-full-interactions-allCharacters-merged.json', 'r') as reader:
    data = json.load(reader)
characters = data['nodes']
relations = data['links']
nodes = [character['name'] for character in characters]
edges = []
weighted_edges = []
no_edge_nodes = nodes.copy()
for relation in relations:
    source_name = nodes[relation['source']]
    target_name = nodes[relation['target']]
    weight = relation['value']
    edges.append((source_name, target_name))
    weighted_edges.append((source_name, target_name, weight))
    try:
        no_edge_nodes.remove(source_name)
        no_edge_nodes.remove(target_name)
    except ValueError:
        pass
for no_edge_node in no_edge_nodes:
    nodes.remove(no_edge_node)
```

데이터의 전처리는 위와 같은 과정을 거쳤다. 기존 json 형식의 데이터를 python list로 변환하였고 edge가 하나도 없는 node는 제거하였다. 또한 edge list는 weight가 포함된 버전과 포함되지 않은 버전 두 가지로 준비하였다.

[2] Degree distribution / Density / Diameter

```
print("diamter")
print(nx.diameter(G))
print("---------------")
print("average shortest path length")
print(nx.average_shortest_path_length(G))
print("---------------")
print("density")
print(nx.density(G))
```

```
diamter
6
---------------
average shortest path length
2.6100083402835694
---------------
density
0.07406171809841534
```

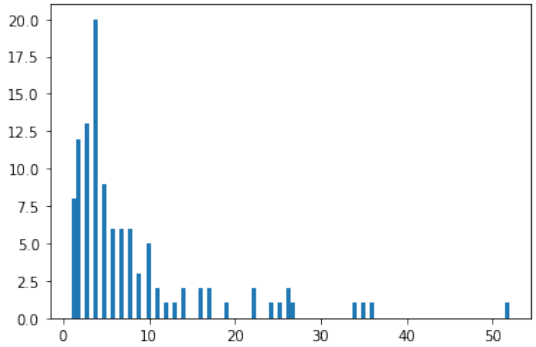
Network Diameter는 네트워크에서 지름을 나타내는 표현으로 Longest shortest pat라고 정의된다.다시 말해 모든 node 쌍에 대하여 shortest path를 계산해보고 그 중 가장 큰 값을 가지는 path의 길이가 diameter가 된다. Networkx 모듈의 diameter 함수를 활용해 얻은 star wars character network의 diameter는 6이고 평균 shortest path의 값은 2.610이다. 이는 노드들 간의 shortest path들은 최대 6개의 링크를 가지며 평균 2.6개의 링크를 가진다는 뜻이다. 다시 말해, 아무리 먼 노드라도 6개의 링크를 거치면 모두 접근 가능하며 평균 2~3개의 링크면 접근 가능하다는 뜻이다.

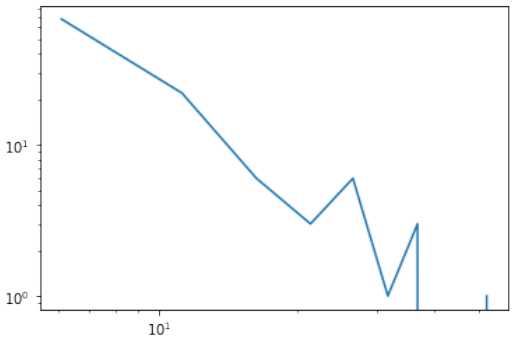
Density는 network내 전체 node들이 서로 얼마나 많은 relation을 맺고 있는 있는지를 표현하기 위한 것으로, 가능한 모든 relation 중 실제로 맺어진 relation 수의 비율을 의미하는 것이다. 이를 수식으로 표현하면 undirected graph에서는 $d = \frac{2m}{n(n-1)}$, directed graph에서는 $d = \frac{m}{n(n-1)}$로 표현할 수 있다(n: node 수, m: edge 수) 이번 과제에서 구축한 네트워크의 density는 0.074로 노드 간 영향력이 일정하게 분할되어 있기보다는 특정 노드의 영향력이 크다는 것을 알 수 있다.

```
degree = nx.degree(G)
deg_dict = dict(degree)
min_val = min(deg_dict.values())
max_val = max(deg_dict.values())
print("Degree 최소값:", min_val, "Degree 최대값:", max_val)

degree_sort = sorted(deg_dict.items(), reverse=True, key=lambda x:x[1])
print("₩nDegree를 많이 가지는 상위 노드 5개")
print(degree_sort[:5])
```

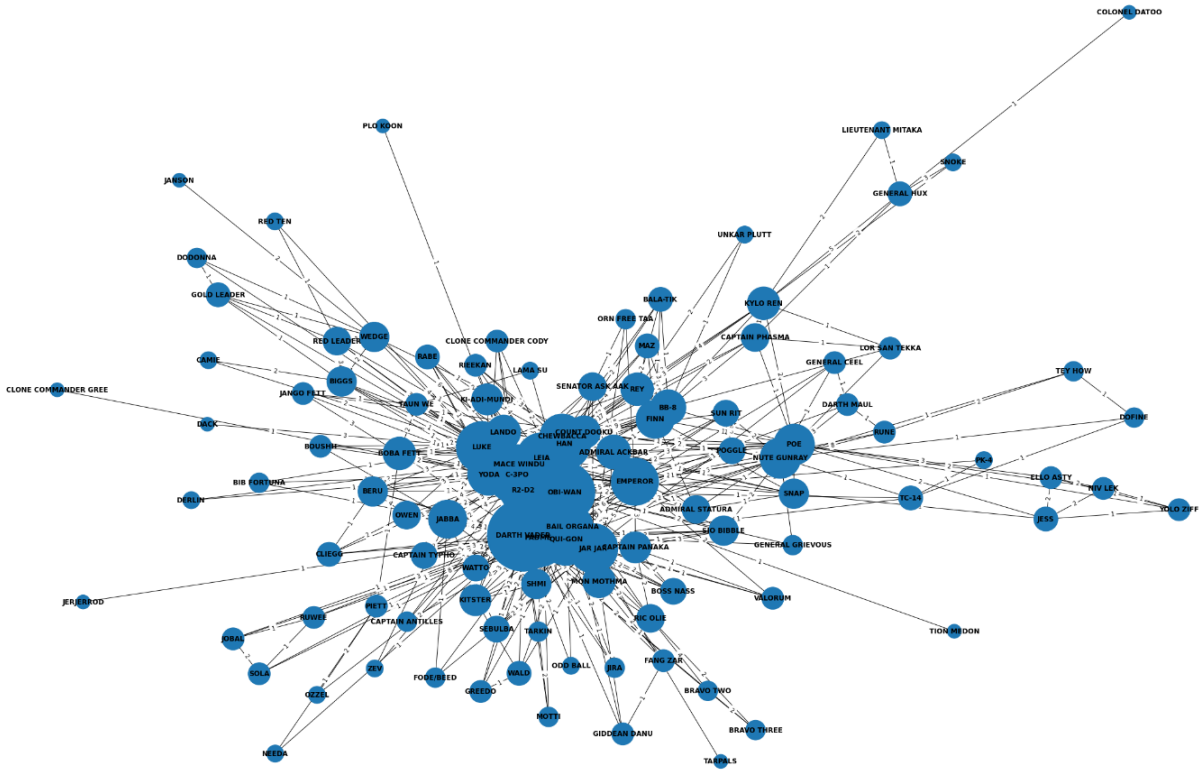
```
Degree 최소값: 1 Degree 최대값: 52

Degree를 많이 가지는 상위 노드 5개
[('DARTH VADER', 52), ('OBI-WAN', 36), ('C-3PO', 35), ('PADME', 34), ('QUI-GON', 27)]
```

Degree는 노드에 연결된 링크의 수로 어떤 노드가 영향력이 큰지를 파악할 수 있는 지표이다. Network의 각 노드의 degree 값 중 최소값은 1, 최대값은 52이므로 최소 한 scene에는 나왔음을 알 수 있다. Degree가 가장 높은 5개의 노드는 'DARTH VADER'(52), 'OBI-WAN'(36), 'C-3PO'(35), 'PADME'(34), 'QUI-GON'(27)로 다른 캐릭터들과 한 Scene에 가장 많이 나온 다섯 캐릭터라고 할 수 있다.





왼쪽 그래프는 노드의 degree 분포를 히스토그램으로 나타낸 것이다. 분포를 보면 대부분 10이하의 degree를 가지고 10 이상의 degree를 가지는 노드의 수는 매우 적은 것을 확인할 수 있다. 오른쪽 그래프는 degree distribution을 loglog 그래프로 나타낸 것으로 우하향하는 형태를 띠고 있다. 이를 통해, 연결고리가 거의 없는 노드가 많고 연결고리가 많은 것은 거의 없는 것을 확인할 수 있다.



위의 그림은 star wars character간의 weighted graph를 network로 표현한 것으로, degree값이 클수록 더 큰 원으로 표시되어 있다. 네트워크를 살펴보면, 대부분의 노드들의 degree값이 작아 작은 원을 가지고 있는 것을 확인할 수 있다. 또, 네트워크의 중심에 degree값이 높은 'DARTH VADER', 'OBI-WAN' 노드 등이 있는 것으로 보아 위의 네트워크 그림이 앞서 설명했던 degree distribution을 잘 설명하고 있는 것을 알 수 있다.

[3] 네트워크의 여러가지 중심성 관점에서 Key Actor에 대한 분석

In [6]:
```
deg_cen = nx.degree_centrality(G)
bet_cen = nx.betweenness_centrality(G)
clo_cen = nx.closeness_centrality(G)
eig_cen = nx.eigenvector_centrality(G)
```

In [7]:
```
# 각각에 대한 Top 3
print('degree centrality')
print(sorted(deg_cen.items(), key=lambda x:x[1], reverse=True)[0:3])
print('--------------------')
print('betweenness centrality')
print(sorted(bet_cen.items(), key=lambda x:x[1], reverse=True)[0:3])
print('--------------------')
print('closeness centrality')
print(sorted(clo_cen.items(), key=lambda x:x[1], reverse=True)[0:3])
print('--------------------')
print('eigenvector centrality')
print(sorted(eig_cen.items(), key=lambda x:x[1], reverse=True)[0:3])
```

```
degree centrality
[('DARTH VADER', 0.4727272727272727), ('OBI-WAN', 0.32727272727272727), ('C-3PO', 0.3181818181818182)]
--------------------
betweenness centrality
[('DARTH VADER', 0.2842705231184945), ('OBI-WAN', 0.16265651299057526), ('C-3PO', 0.13906804110243165)]
--------------------
closeness centrality
[('DARTH VADER', 0.6067926455566905), ('C-3PO', 0.5596325953838908), ('OBI-WAN', 0.5567478912839738)]
--------------------
eigenvector centrality
[('DARTH VADER', 0.337061526604466), ('OBI-WAN', 0.2712079460236518), ('C-3PO', 0.2670150777367724)]
```

그래프에서 각 centrality에 대해서 가장 높은 값을 보이는 세 가지의 node를 추출하여 확인해보았다.

Degree centrality는 하나의 node에 얼마나 많은 node들이 연결되어 있는 지 파악하는 지표이고, weighted graph의 경우에는 weight도 고려하여 판단한다. 이 데이터의 경우 가장 많은 장면에서 출연한 등장인물, 즉 주연이라 생각할 수 있다. DARTH VADER가 약 0.4727로 1위, OBI-WAN이 약 0.3273로 2위, C-3PO가 약 0.3182로 3위로 나타났다.

Betweenness centrality는 어떤 node가 node 간 경로 사이에 있는지 파악하는 지표이고, node 간 경로에 있는 것이 더 centrality가 높다고 판단할 수 있다. 이 데이터의 경우 많은 인물들과 가장 관련 있는 인물, 즉 이야기를 이끌어 나가는 핵심 인물이라 볼 수 있다. DARTH VADER가 0.2843으로 1위, OBI-WAN이 약 0.1627로 2위, C-3PO가 약 0.1391로 3위로 나타났다.

Closeness centrality는 한 node가 다른 node에 얼마나 가까운 지를 파악하는 지표이고, 중요한 node일수록 다른 node까지의 거리가 짧다고 판단할 수 있다. 이 데이터의 경우 주요 핵심 인물들과 가장 가까이 지내는 인물이라 볼 수 있다. DARTH VADER가 0.6068로 1위, C-3PO가 0.5596으로 2위, OBI-WAN이 약 0.5567로 3위로 나타났다.

Eigenvector centrality는 각 node별 가중치를 고려하여 중요도를 파악한 지표로 다른 node의 중심성을 반영하여 계산한다. 이 데이터의 경우 등장인물 중 다른 등장인물들에 가장 많은 영향력을 끼치는 인물이라 볼 수 있다. DARTH VADER가 0.3371로 1위, OBI-WAN이 0.2712로 2위, C-3PO가 0.2670으로 3위로 나타났다.

[4] Reference

Dataset. https://www.kaggle.com/ruchi798/star-wars?select=starwars-full-interactions-allCharacters-merged.json