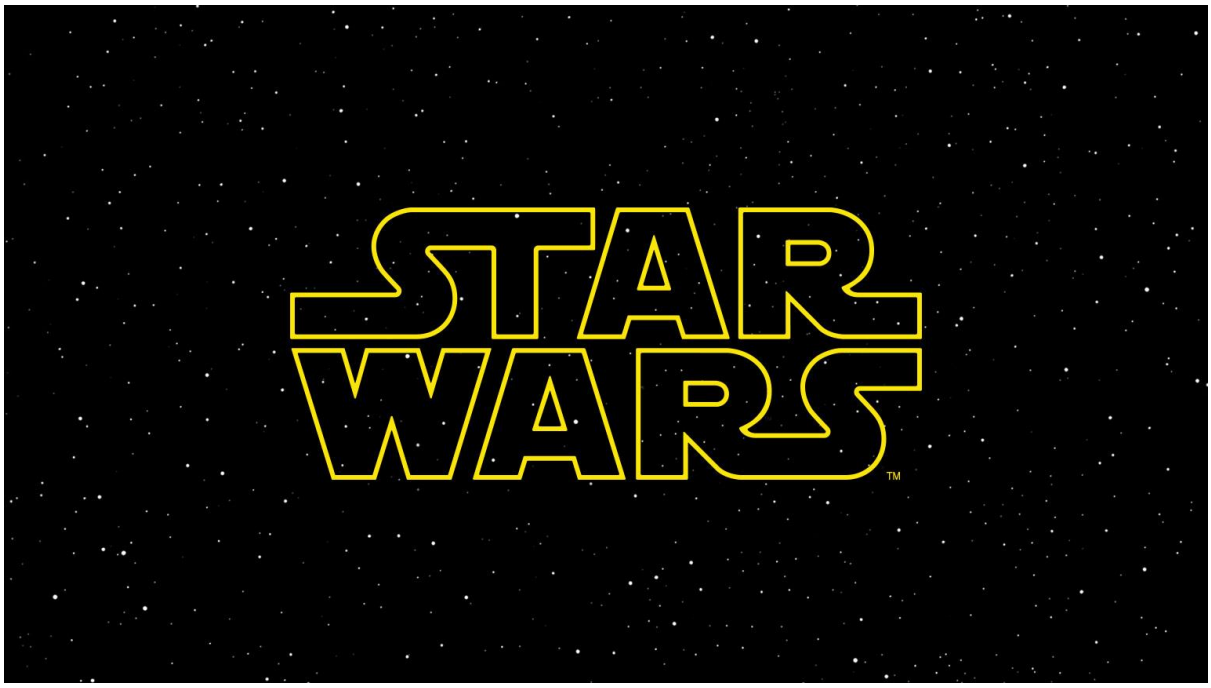
SOCIAL NETWORK ANALYSIS COURSE

STAR WARS

FIRST SEVEN (7) EPISODES



Vougias Ioannis, 8160017

ioannisvougias@hotmail.com

Professor: Dimitrios Pournarakis

Department of Management Science and Technology 2021-2022

Contents

1. Idea

I remember in one of our lectures, where we discussed about the social network of the Trojan War, giving examples on who is the most successful warrior, who is the most famous hero etc. What excited me the most is, the fact that in the beginning of the lecture, I couldn't even imagine that depending on the researchers view, you could extract different results and hypothesis on which character is the most successful, impactful and famous. That's why I wanted to do something similar and make an analysis on another well-known franchise, Star Wars.

Star Wars is a science-fiction franchise comprising movies, books, comics, video games, toys, live action shows, and animated shows. It is a fictional universe created by George Lucas.

The Star Wars story employs archetypal structures common to science fiction, political climax and classical mythology, as well as musical ideas of those aspects. As one of the foremost examples of the space genre of science fiction, Star Wars has become part of mainstream popular culture, as well as being one of the highest-grossing series of all time.

Since the movies are the most well-known of the franchise, I decided to analyze the first seven episodes of the franchise. I found the proper data available and accessed it through Kaggle.com.

Although its undoubtful how popular this franchise is, even the biggest fans of it can't decide on several debates regarding the characters. Who is the most famous character of them all? Who is the character that is most appeared on the movies? Is there any character with less screen time but with more influence on other characters? These are some simple questions that triggered my motivation to analyze Stars Wars and its first seven episodes. Nonetheless, I am a huge fan of George Lucas and his work, so it is more exciting and familiar for me to work on something I have knowledge of and respect.

In the next pages we can see the explanation of the dataset, how I decided to process and clean it, how I visualized and represented some graphs that can show different metrics and results on the characters of the first seven Star Wars movies.

2. Dataset

Through Kaggle.com I found the Star Wars Social Network Dataset which contains several json files.

For example, there are files with the following explanation as pointed out clearly by the author:

“starwars-episode-N-interactions.json contains the social network extracted from Episode N, where the links between characters are defined by the times the characters speak within the same scene.”

“starwars-episode-N-mentions.json contains the social network extracted from Episode N, where the links between characters are defined by the times the characters are mentioned within the same scene.”

“starwars-episode-N-interactions-allCharacters.json is the interactions network with R2-D2 and Chewbacca added in using data from mentions network.”

Regarding the Nodes of the Network:

name: Name of the character

value: Number of scenes the character appeared in

Regarding the Links of the Network that represent connections between characters:

source: zero-based index of the character that is one end of the link, the order of nodes is the order in which they are listed in the “nodes” element

target: zero-based index of the character that is the other end of the link.

value: Number of scenes where the “source character” and “target character” of the link appeared together.

There are files for the seven episodes (movies) that I wanted to visualize and analyze.

The network is undirected and which character represents the source and the target is random, they correspond only to two ends of the link.

3. Data Processing

To begin the Data processing, I downloaded the json file named “starwars-full-interactionsallCharacters.json”. This file contained nodes and links for the characters of all seven (7) episodes that I wanted to analyze. The nodes (screenshot 1) included the name of the character, how many times this character appeared in different scenes and a color for graph representation which I didn’t take into consideration because I made my own visualization with different colors and point of view. The links (screenshot 2) had three variables, “source”, “target” and “value” which helped identify which character relates to whom and how many times.


```
{
  "nodes": [
    {
      "name": "R2-D2",
      "value": 171,
      "colour": "#bde0f6"
    },
    {
      "name": "CHEWBACCA",
      "value": 145,
      "colour": "#A0522D"
    },
    {
      "name": "BB-8",
      "value": 40,
      "colour": "#eb5d00"
    },
    {
      "name": "QUI-GON",
      "value": 62,
      "colour": "#4f4fb1"
    },
    {
      "name": "NUTE GUNRAY",
      "value": 25,
      "colour": "#808080"
    },
    {
      "name": "PK-4",
      "value": 4,
      "colour": "#808080"
```


“Screenshot 1, the nodes of the dataset”


```
  {
    "source": 17,
    "target": 0,
    "value": 24
  },
  {
    "source": 3,
    "target": 0,
    "value": 13
  },
  {
    "source": 20,
    "target": 0,
    "value": 30
  },
  {
    "source": 0,
    "target": 19,
    "value": 3
  },
  {
    "source": 24,
    "target": 0,
    "value": 55
  },
```


“Screenshot 2, the links of the dataset”

To further process the files I used Python 3, Anaconda Navigator, Jupyter Notebook and Atom Editor. To visualize the network, I used Pandas and Networkx libraries since they have a broad and useful toolset that I am familiar with. In the next pages I present some screenshots from Jupyter Notebook of how I began my initial data processing.

```
In [1]: #Import our modules for data analysis, numpy pandas, json

        import numpy as np
        import pandas as pd
        import json

        #set pandas options to print dataframes in full scale
        pd.set_option('display.max_rows', None)
        pd.set_option('display.max_columns', None)
        pd.set_option('display.width', None)
        pd.set_option('display.max_colwidth', None)
```

"Screenshot 3, import modules and set pandas options to prin full scale the dataframes"

```
In [2]: #Open the json file with the full interactions
        with open('starwars-full-interactions-allCharacters.json') as f:
            data = json.load(f)


        Characters_df = pd.DataFrame(columns=['Name', 'Value'])
        #Create a dictionary with the name of the character and the values of how many scenes they appeared
        colour_values = {}
        counter = 0

        #populate the dictionary
        for char in data['nodes']:
            colour_values[char['name']] = char['value']

            counter += 1

        colour_values
        for i,j in enumerate(colour_values):
            print(i+1,j)
```

"Screenshot 4, open the jason file and create a dictionary with the name of the character and how many scenes the appreared in to help me with the visualization"

```
1 R2-D2
2 CHEWBACCA
3 BB-8
4 QUI-GON
5 NUTE GUNRAY
6 PK-4
7 TC-14
8 OBI-WAN
9 DOFINE
10 RUNE
11 TEY HOW
12 EMPEROR
13 CAPTAIN PANAKA
14 SIO BIBBLE
15 JAR JAR
16 TARPALS
17 BOSS NASS
18 PADME
19 RIC OLIE
```

"Screenshot 5, the dataframe with the movie characters"

4. Graph Representation

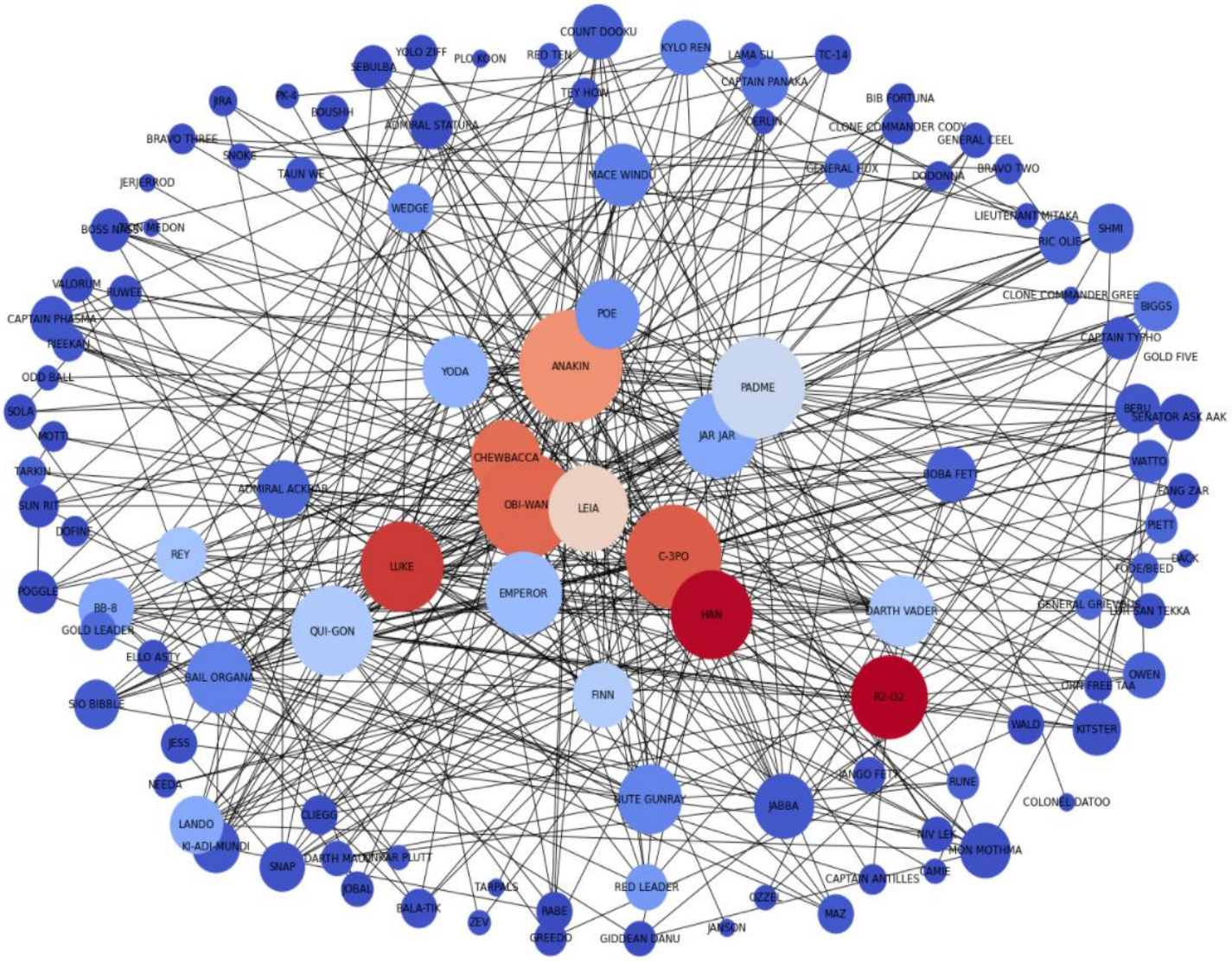
Using the NetworkX library I wanted to create a visualization that would include the characters, how many scenes they appeared in and how many other characters are connected to them during the scenes. I decided to visualize with color the number of scenes a character has appeared. The **redder** a character is, the more scenes he/she/it has appeared. The **bigger** the circle of a character is, more other characters have appeared with him/her/it on the same scene. The theory behind this is that I wanted to show that the "bigger" a character is, the more famous or at least connected with others would be. As well as the "redder" a character is, the "hotter" he is as he has played more scenes in the movies. Taking into consideration the previous notes and theory I started to plot the visualization based on the following code.

In [3]:
```
#Create the visualization
import networkx as nx
G = nx.Graph()

#build graph nodes
for node in data['nodes']:
    G.add_node(node['name'])

#build graph edges
for edge in data['links']:
    G.add_edge(data['nodes'][edge['source']]['name'], data['nodes'][edge['target']]['name'])
```

In [4]:
```
import matplotlib as mpl
from matplotlib import pyplot as plt
from pylab import rcParams

low, *_, high = sorted(colour_values.values())
norm = mpl.colors.Normalize(vmin=low, vmax=high, clip=True)
mapper = mpl.cm.ScalarMappable(norm=norm, cmap=mpl.cm.coolwarm)

rcParams['figure.figsize'] = 21, 15
pos = nx.spring_layout(G, scale=20, k=19/np.sqrt(G.order()))
d = dict(G.degree)
print(d)
nx.draw(G, pos,node_color=[mapper.to_rgba(i)
    for i in colour_values.values()],
        with_labels=True,
        nodelist=d,
        node_size=[d[k]*350 for k in d])
```

"Screenshot 6, coding the visualization of the Star Wars Network "

Based on this, I present the visualization of the **Star Wars Network**.



As you can see, **R2-D2** and **HAN** are the **reddest** characters because they have played in the most scenes (171 and 170 repeatedly). While the **biggest** one or in other words the most famous is **ANAKIN** with a total of 42 links (other characters he has played with in a scene).

5. Basic Topological Properties

To continue with our analysis, we should define some basic topological Properties or our network. First, we have 112 nodes in our dataset, meaning we have 112 characters in the seven movies of Star Wars.

```
In [5]: #Number of Nodes in our Database
        len(colour_values)

Out[5]: 112
```

"Screenshot 7, total Nodes in our Dataset "

Moreover, we have 450 edges, meaning we have 450 connections between characters in the scenes of the seven movies.

```
In [6]: #Number of edges in our Database
        len(data['links'])

Out[6]: 450
```

"Screenshot 8, total edges in our Dataset "

The Network Diameter is 6 which means the longest short path connecting two characters is 6.

```
In [7]: #Find Network Diameter
        diameter = max([max(j.values()) for (i,j) in nx.shortest_path_length(G)])
        print(diameter)

        6
```

"Screenshot 9, Diameter of the Network "

The average of the shortest paths for all pairs of nodes, the average path length, is 2.59 which means that on average two characters have a distance of 2.59 between them.

```
In [8]: #Find Average Path Length
        x=[sum(j.values()) for (i,j) in nx.shortest_path_length(G)]
        avg = sum([sum(j.values()) for (i,j) in nx.shortest_path_length(G)])
        avg_sum=(len(x)*len(x))

        avg_path= avg/avg_sum
        print(avg_path)

        2.5931122448979593
```

"Screenshot 10, Average path length "

6. Component Measures

Measuring the components, I was surprised to find out that the network is disconnected.

```
In [9]: #Component Measures
        #Is the graph connected?
        nx.is_connected(G)

Out[9]: False
```

"Screenshot 11, the graph is not connected"

For my analysis to be more accurate, I wanted to find the reason why the network is not connected.

```
In [10]: #Number of connected components
         nx.number_connected_components(G)

Out[10]: 2
```

"Screenshot 12, the graph has 2 connected components"

It seems like my network has 2 connected components. But as I did a further analysis, I found out that one character has played a solo scene for about 10 seconds and then died on an aircraft explosion as you can see from the video below.

https://www.youtube.com/watch?v=NnP5iDKwuwk

"Screenshot 13, GOLD FIVE, the character that made the graph disconnected"

So as this man (GOLD FIVE in his name), almost ruined my network and the analysis of it, this is how I found about him.

```
In [11]: #Find the Character that makes the graph disconnected
         for name in data['nodes']:
             print(len(nx.node_connected_component(G, name['name'])))
             if (len(nx.node_connected_component(G, name['name']))) == 1:
                 print(name['name'])
```

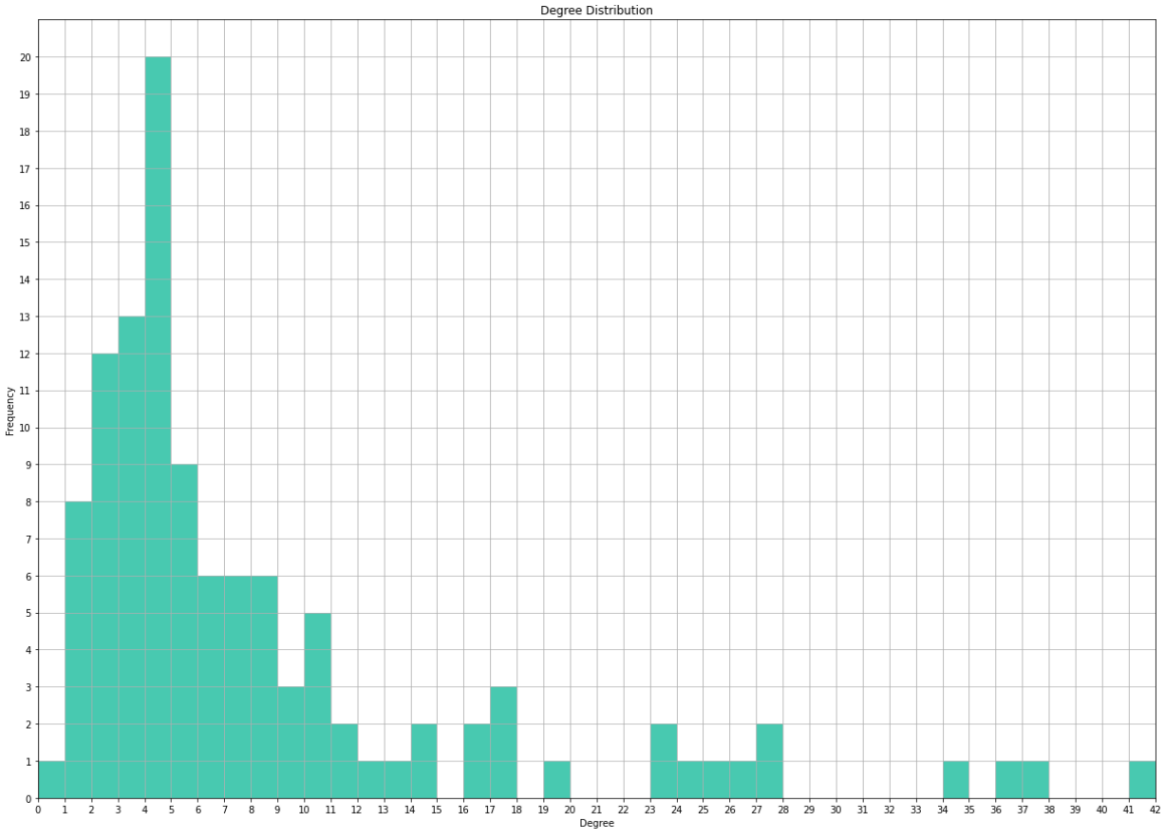
```
1
GOLD FIVE
111
111
111
111
111
111
```

"Screenshot 14, code to find out who is the character that made the graph disconnected"

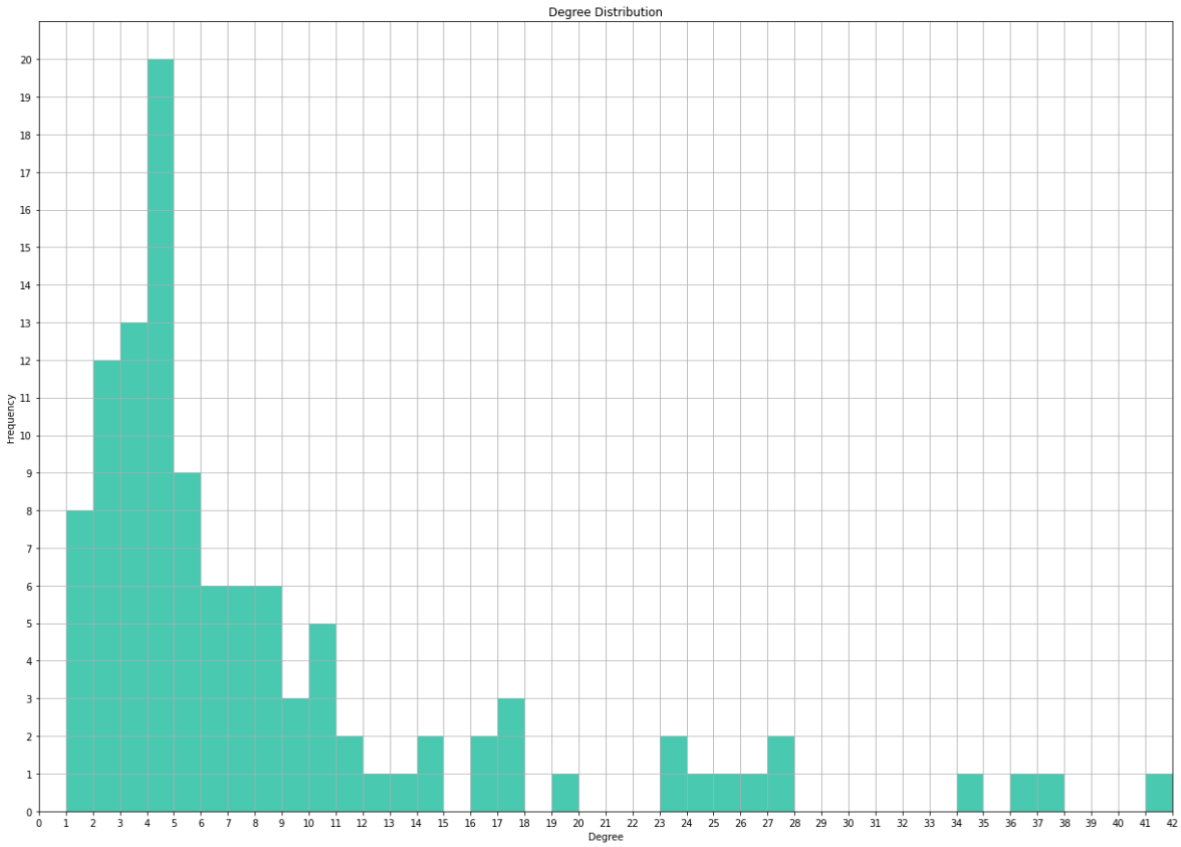
That's why we removed GOLD FIVE and we concluded that the network has one giant component because a significant proportion of the nodes are connected.

7. Degree Measures

The average degree of the network is **8.03** and the character with the highest degree is **ANAKIN** with a degree of **42**.



"Screenshot 15, Degree Distribution with GOLD FIVE"



"Screenshot 16, Degree Distribution without GOLD FIVE"

```
In [14]: #Average Degree
         avg_degree=sum(frequency) / 112
         print(avg_degree)
```

```
8.035714285714286
```

```
In [15]: #Check if ANAKIN (based of the diagram) is the charachter with the maximun degree)
         c_neighbors=[]
         for x in(nx.neighbors(G, 'ANAKIN')):
             c_neighbors.append(x)
         print(len(c_neighbors))
```

```
42
```

"Screenshot 17, Average and Maximum Degree"

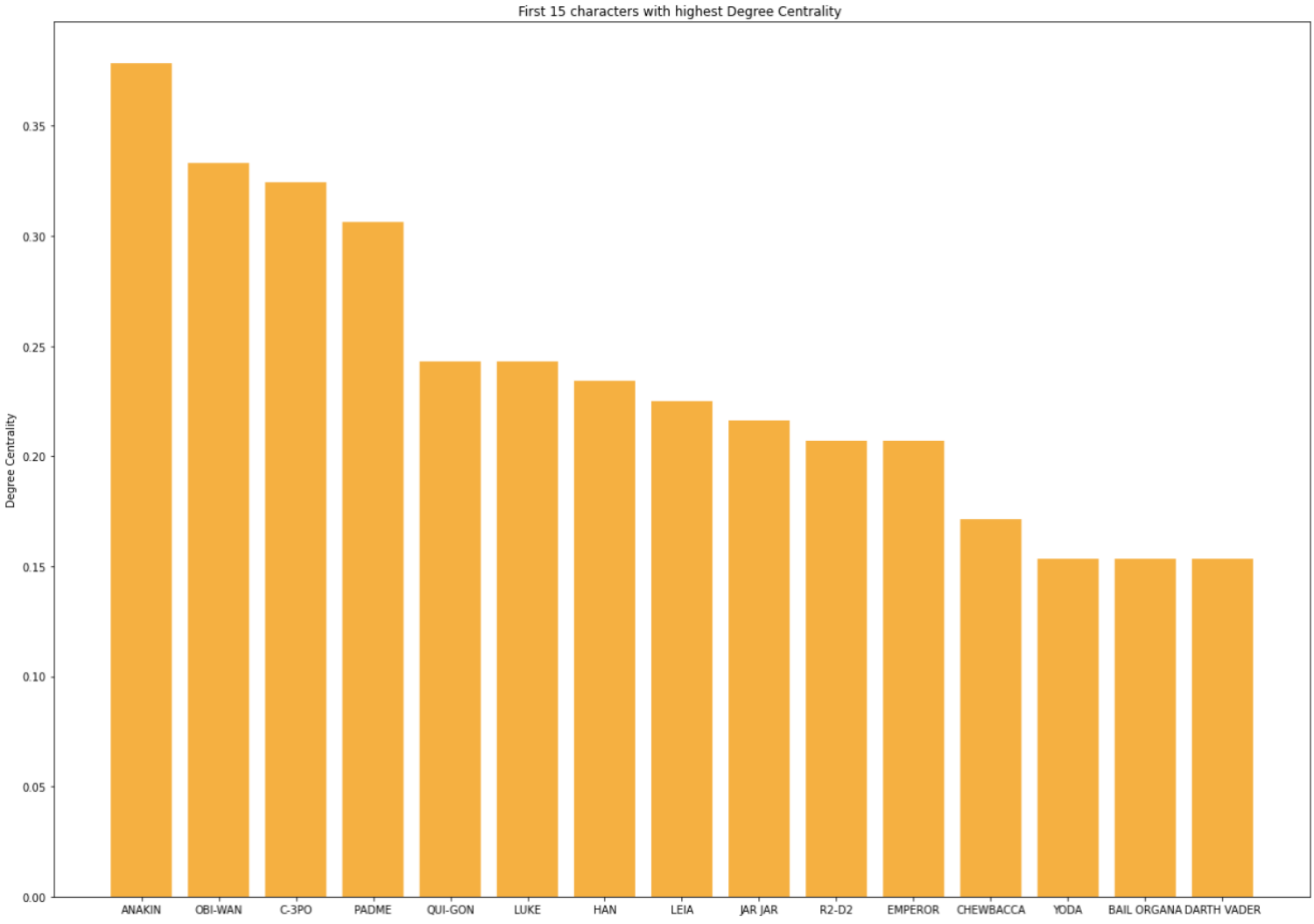
8. Centrality Measures

8.1 Degree Centrality

Degree centrality assigns an importance score based simply on the number of links held by each node.

Here we can see the top 15 characters based on their Degree Centrality.

ANAKIN, OBI-WAN and C-3PO are the top 3 characters with the highest degree centrality as it is also demonstrated by the network visualization. They are the 3 biggest circles as they have the most links of the network.



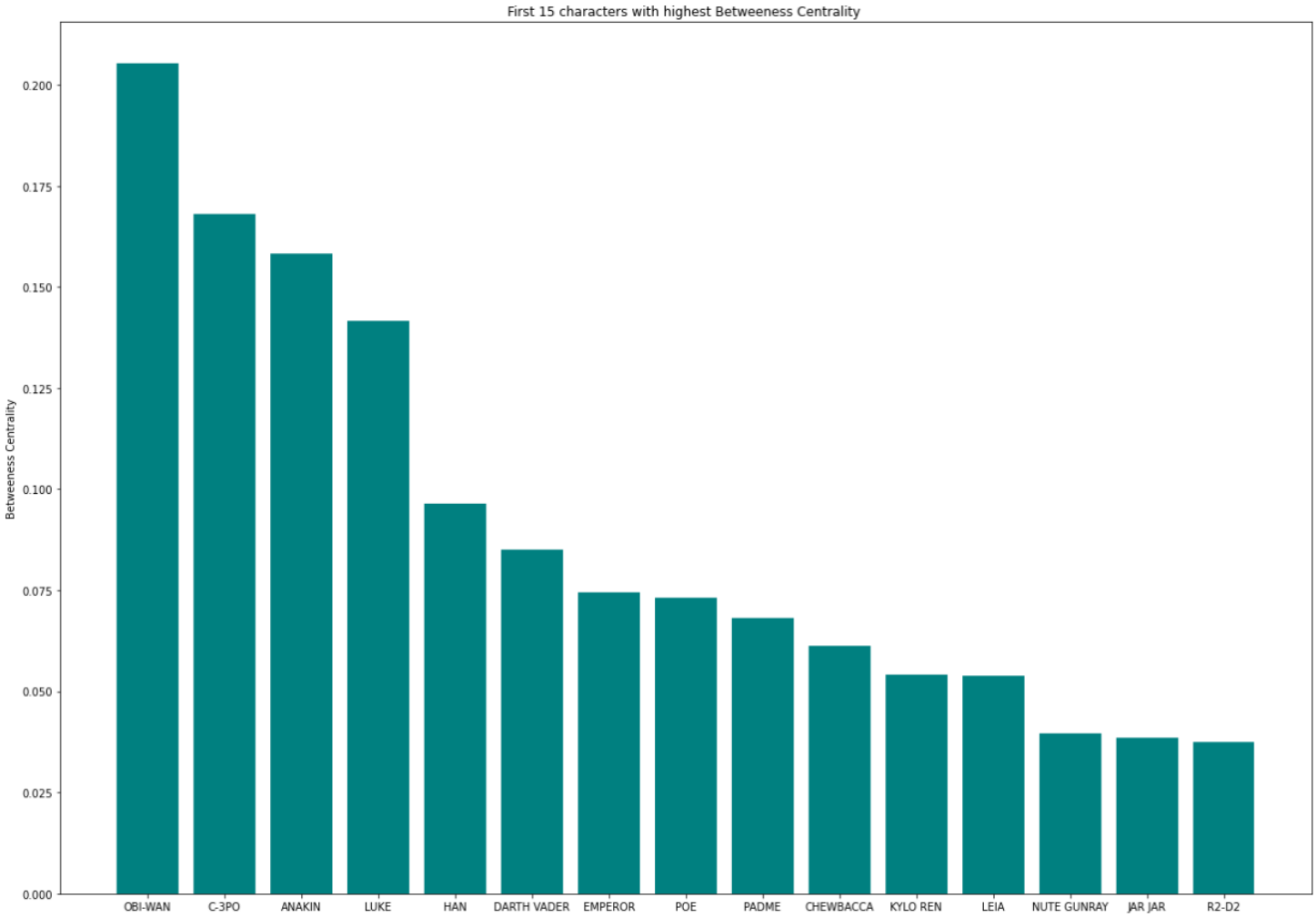
"Screenshot 18, First 15 characters with highest Degree Centrality"

8.2 Betweenness Centrality

Betweenness Centrality shows how important a node is in terms of connecting other nodes. In other words which characters lie on shortest paths between two other characters.

Here we can see the top 15 characters based on their Betweenness Centrality.

Surprisingly, OBI-WAN has the highest Betweenness Centrality, which means he has a bigger influence on the flow of information in the movies that ANAKIN or C-3PO. We also see that the first 4 characters have a much bigger influence that the other 11 which shows us the most influential characters on the movies are.



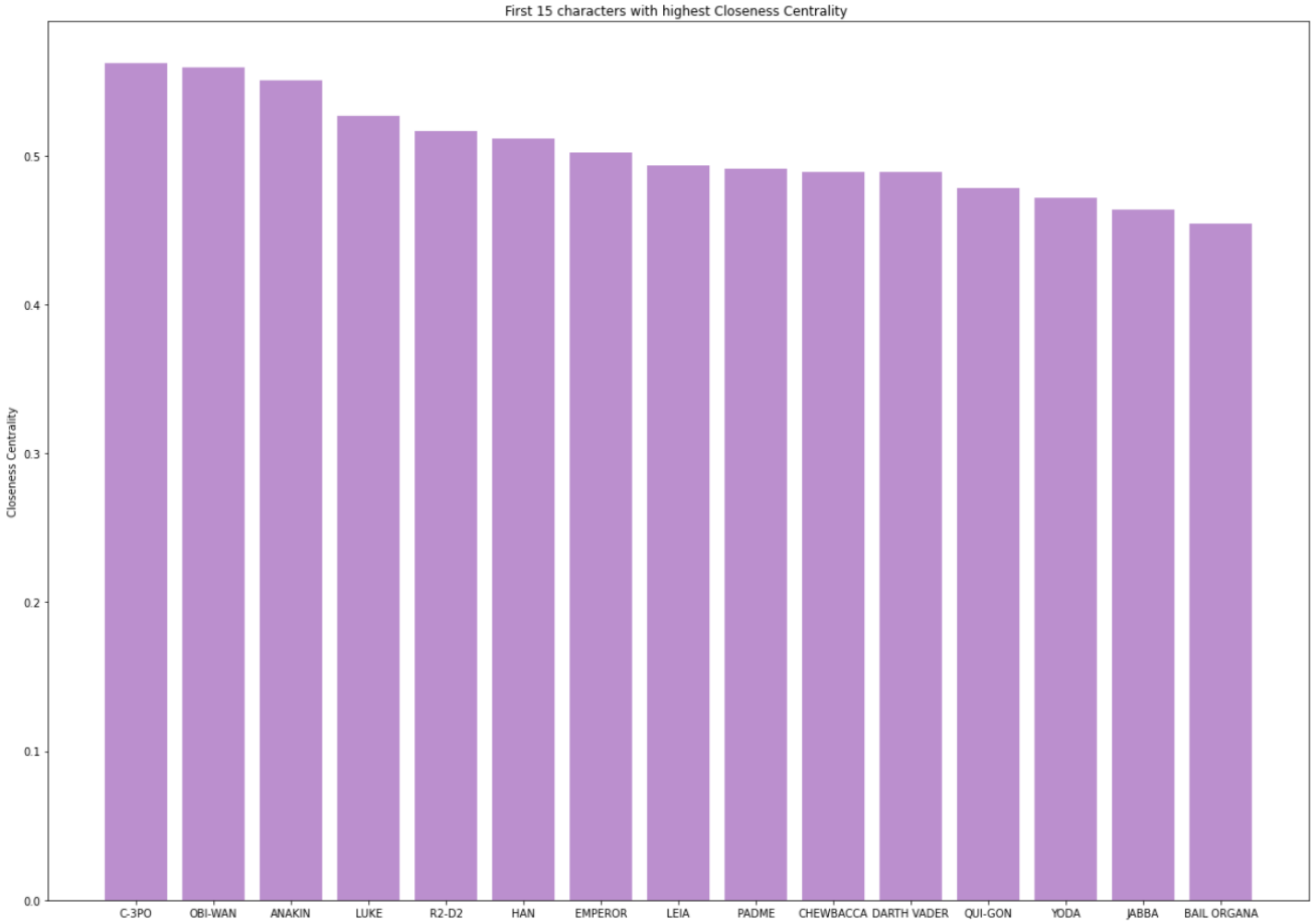
"Screenshot 19, First 15 characters with highest Betweenness Centrality"

8.3 Closeness Centrality

Closeness centrality indicates how close a node is to all other nodes in the network. It also shows how easily a node can reach other nodes or, in other words, how close a node is to the center of the network.

Here we can see the top 15 characters based on their Closeness Centrality.

C-3PO takes the lead on this graph with OBI-WAN and ANAKIN following it. The difference is small but still shows us the different impact these 3 characters have on the movies depending on how we measure it.



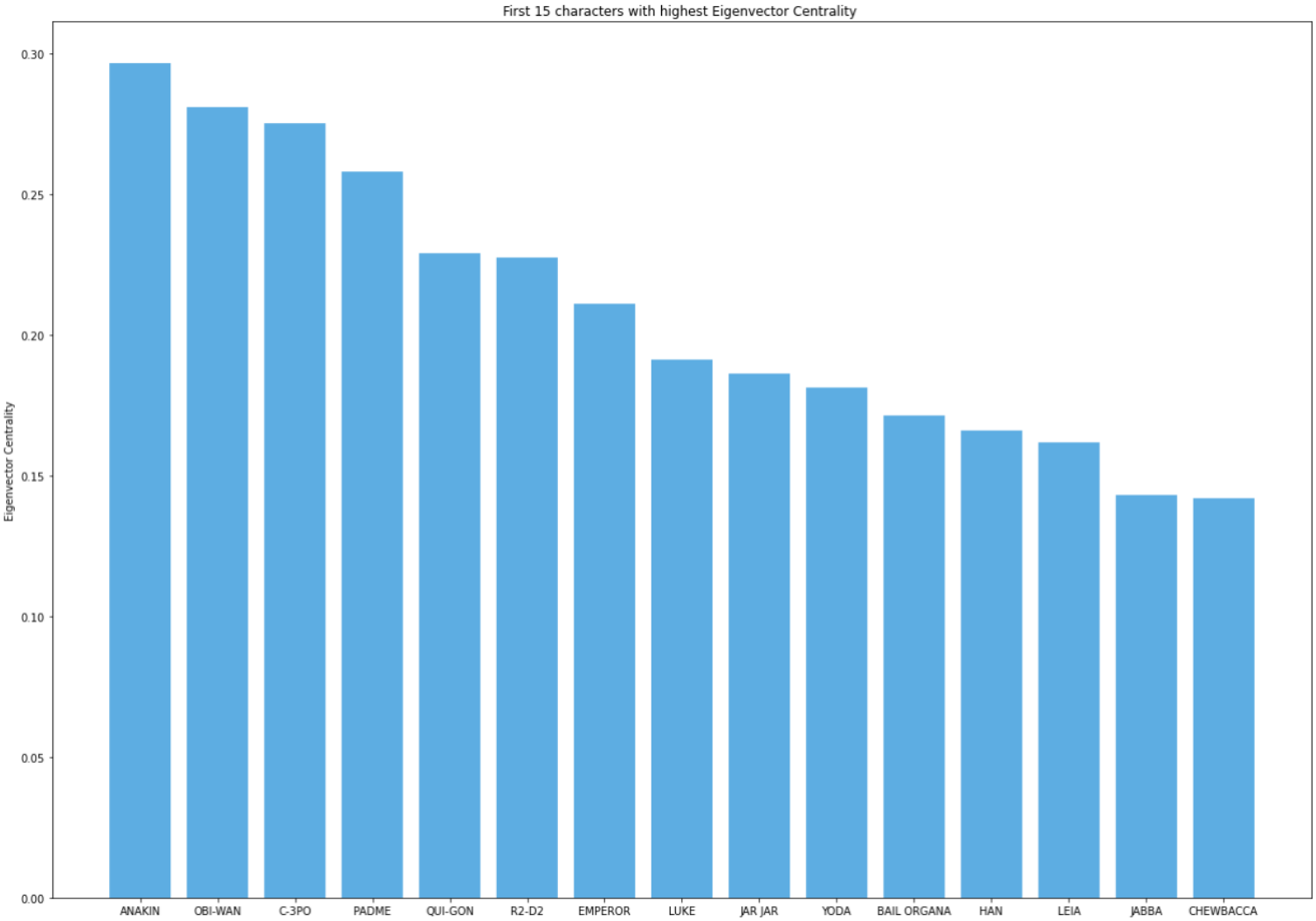
"Screenshot 20, First 15 characters with highest Closeness Centrality"

8.4 Eigenvector Centrality

Eigenvector centrality shows how much a node is connected to other important nodes in the network.

Here we can see the top 15 characters based on their Eigenvector Centrality.

ANAKIN takes the lead here with OBI-WAN and C-3PO following. As it seems these 3 are at the top of all the measurements meaning they are the most impactful and important characters of the Star Wars movies.



"Screenshot 21, First 15 characters with highest Eigenvector Centrality"

9. Clustering Effects

The average clustering coefficient in this network is 0.68 which means that the average probability that two characters that have played in the same scene with another character X, to play together in another scene is 68%.

```
In [38]: #Clustering coefficient
         nx.average_clustering(G)

Out[38]: 0.6868025421339279
```

"Screenshot 22, Clustering Coefficient"

Below there is a list of the triangles of the network.

```
In [48]: #Triangles
         triangles = nx.triangles(G)
         print(triangles)

{'R2-D2': 98, 'CHEWBACCA': 74, 'BB-8': 33, 'QUI-GON': 114, 'NUTE GUNRAY': 40, 'PK-4': 1, 'TC-14': 5, 'OBI-WAN': 145,
'DOFINE': 2, 'RUNE': 4, 'TEY HOW': 2, 'EMPEROR': 95, 'CAPTAIN PANAKA': 30, 'SIO BIBBLE': 26, 'JAR JAR': 84, 'TARPAL
S': 0, 'BOSS NASS': 15, 'PADME': 136, 'RIC OLIE': 13, 'WATTO': 15, 'ANAKIN': 168, 'SEBULBA': 15, 'JIRA': 3, 'SHMI': 2
3, 'C-3PO': 153, 'DARTH MAUL': 5, 'KITSTER': 25, 'WALD': 8, 'FODE/BEED': 3, 'JABBA': 38, 'GREEDO': 3, 'VALORUM': 6,
'MACE WINDU': 51, 'KI-ADI-MUNDI': 25, 'YODA': 69, 'RABE': 10, 'BAIL ORGANA': 64, 'GENERAL CEEL': 4, 'BRAVO TWO': 3,
'BRAVO THREE': 3, 'CAPTAIN TYPHO': 13, 'SENATOR ASK AAK': 17, 'ORN FREE TAA': 3, 'SOLA': 6, 'JOBAL': 6, 'RUWEE': 6,
'TAUN WE': 4, 'LAMA SU': 1, 'BOBA FETT': 24, 'JANGO FETT': 4, 'OWEN': 18, 'BERU': 21, 'CLIEGG': 10, 'COUNT DOOKU': 2
9, 'SUN RIT': 15, 'POGGLE': 15, 'PLO KOON': 0, 'ODD BALL': 1, 'GENERAL GRIEVOUS': 3, 'FANG ZAR': 6, 'MON MOTHMA': 23,
'GIDDEAN DANU': 6, 'CLONE COMMANDER GREE': 0, 'CLONE COMMANDER CODY': 6, 'TION MEDON': 0, 'CAPTAIN ANTILLES': 3, 'DAR
TH VADER': 48, 'LUKE': 86, 'CAMIE': 1, 'BIGGS': 15, 'LEIA': 91, 'MOTTI': 3, 'TARKIN': 3, 'HAN': 100, 'DODONNA': 3, 'G
OLD LEADER': 8, 'WEDGE': 11, 'RED LEADER': 12, 'RED TEN': 1, 'GOLD FIVE': 0, 'RIEEKAN': 6, 'DERLIN': 1, 'ZEV': 1, 'PI
ETT': 3, 'OZZEL': 1, 'DACK': 0, 'JANSON': 0, 'NEEDA': 1, 'LANDO': 37, 'JERJERROD': 0, 'BIB FORTUNA': 3, 'BOUSHH': 5,
'ADMIRAL ACKBAR': 35, 'LOR SAN TEKKA': 4, 'POE': 47, 'KYLO REN': 18, 'CAPTAIN PHASMA': 13, 'FINN': 50, 'UNKAR PLUTT':
1, 'REY': 25, 'GENERAL HUX': 3, 'LIEUTENANT MITAKA': 1, 'BALA-TIK': 10, 'SNOKE': 1, 'MAZ': 10, 'SNAP': 22, 'ADMIRAL S
TATURA': 21, 'YOLO ZIFF': 6, 'COLONEL DATOO': 0, 'ELLO ASTY': 6, 'JESS': 7, 'NIV LEK': 6}
```

"Screenshot 23, Triangles"

We found that we have 2577 triangles. Since we are computing triangles for the entire graph each triangle is counted three times, once at each node. As a result, we need to divide by 3 to get the actual number without having duplicates.

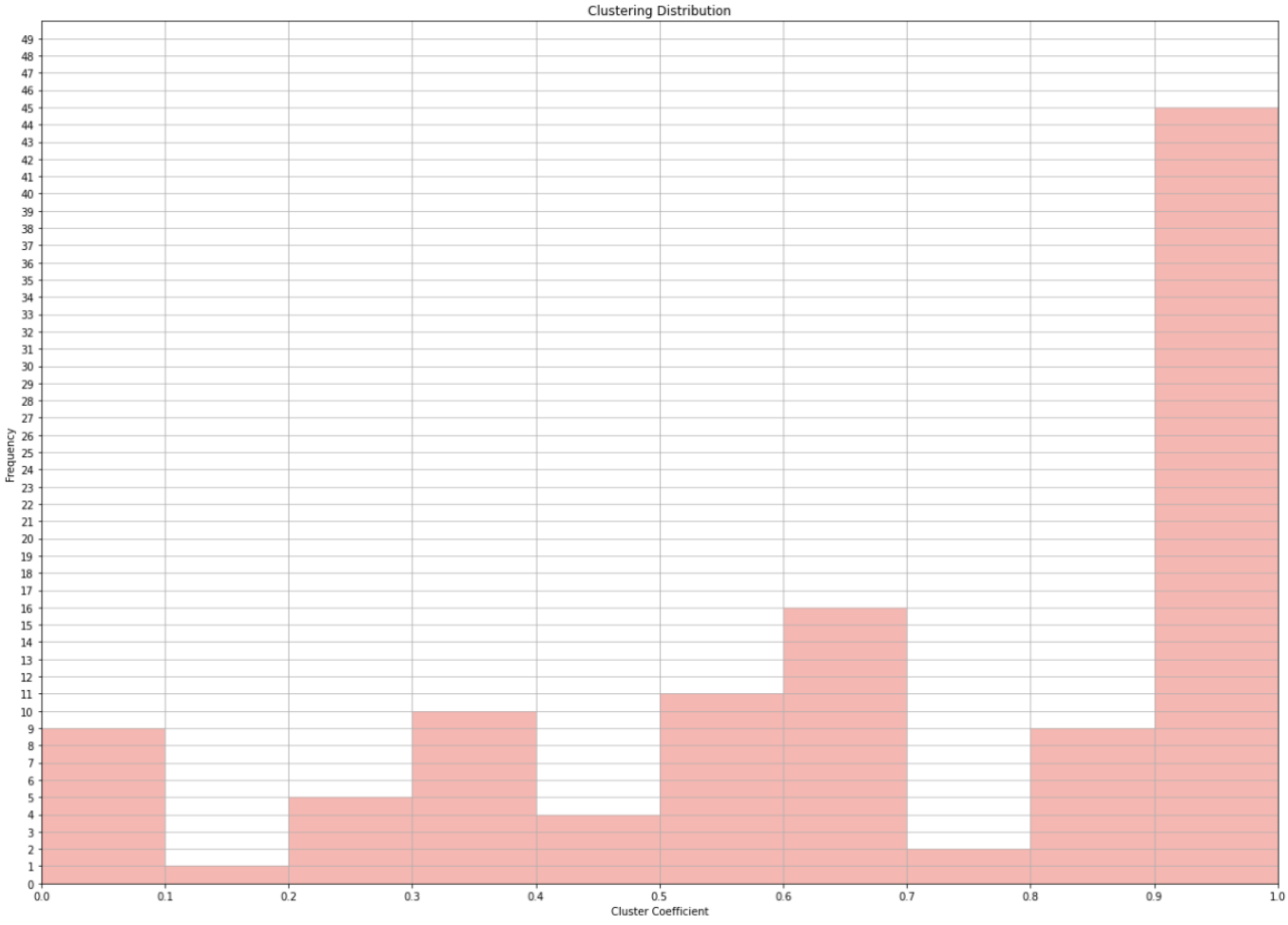
```
In [45]: sum(triangles.values())

Out[45]: 2577

In [46]: sum(triangles.values())/3

Out[46]: 859.0
```

"Screenshot 24, Total Triangles"

Finally, below we can see the clustering coefficient distribution of the Star Wars Database. Most of the characters belong between the range of (0.9, 1].

```
#Clustering coefficient Distribution
histogram_initial = nx.clustering(G)
print(histogram_initial)

degree = pd.Series(histogram_initial.values())
xscale = [i/10 for i in range(0,11)]
yscale = [i for i in range(0,50)]
degree.plot.hist(grid=True, rwidth=1,bins=10,
                 color='#F5B7B1',yticks=yscale ,ylim=(0,50), xlim=(0,1), xticks=xscale)
plt.title('Clustering Distribution')
plt.xlabel('Cluster Coefficient')
plt.ylabel('Frequency')
plt.grid(axis='y', alpha=0.9)
```


{'R2-D2': 0.38735177865612647, 'CHEWBACCA': 0.4327485380116959, 'BB-8': 0.5, 'QUI-GON': 0.3247863247863248, 'NUTE GUNRAY': 0.3333333333333333, 'PK-4': 1.0, 'TC-14': 0.5, 'OBI-WAN': 0.21771771771771772, 'DOFINE': 0.6666666666666666, 'RUNE': 0.6666666666666666, 'TEY HOW': 0.6666666666666666, 'EMPEROR': 0.37549407114624506, 'CAPTAIN PANAKA': 0.8333333333333334, 'SIO BIBBLE': 0.9285714285714286, 'JAR JAR': 0.30434782608695654, 'TARPALS': 0, 'BOSS NASS': 1.0, 'PADME': 0.24242424242424243, 'RIC OLIE': 0.6190476190476191, 'WATTO': 1.0, 'ANAKIN': 0.1951219512195122, 'SEBULBA': 1.0, 'JIRA': 1.0, 'SHMI': 0.8214285714285714, 'C-3PO': 0.24285714285714285, 'DARTH MAUL': 0.8333333333333334, 'KITSTER': 0.6944444444444444, 'WALD': 0.8, 'FODE/BEED': 1.0, 'JABBA': 0.4175824175824176, 'GREEDO': 0.5, 'VALORUM': 1.0, 'MACE WINDU': 0.6538461538461539, 'KI-ADI-MUNDI': 0.6944444444444444, 'YODA': 0.5073529411764706, 'RABE': 1.0, 'BAIL ORGANA': 0.47058823529411764, 'GENERAL CEEL': 0.6666666666666666, 'BRAVO TWO': 1.0, 'BRAVO THREE': 1.0, 'CAPTAIN TYPHO': 0.8666666666666667, 'SENATOR ASK AAK': 0.8095238095238095, 'ORN FREE TAA': 1.0, 'SOLA': 1.0, 'JOBAL': 1.0, 'RUWEE': 1.0, 'TAUN WE': 0.6666666666666666, 'LAMA SU': 1.0, 'BOBA FETT': 0.5333333333333333, 'JANGO FETT': 0.6666666666666666, 'OWEN': 0.8571428571428571, 'BERU': 0.75, 'CLIEGG': 1.0, 'COUNT DOOKU': 0.6444444444444445, 'SUN RIT': 1.0, 'POGGLE': 1.0, 'PLO KOON': 0, 'ODD BALL': 1.0, 'GENERAL GRIEVOUS': 1.0, 'FANG ZAR': 1.0, 'MON MOTHMA': 0.5111111111111111, 'GIDDEAN DANU': 1.0, 'CLONE COMMANDER GREE': 0, 'CLONE COMMANDER CODY': 1.0, 'TION MEDON': 0, 'CAPTAIN ANTILLES': 1.0, 'DARTH VADER': 0.35294117647058826, 'LUKE': 0.245014245014245, 'CAMIE': 1.0, 'BIGGS': 0.5357142857142857, 'LEIA': 0.30333333333333334, 'MOTTI': 1.0, 'TARKIN': 1.0, 'HAN': 0.3076923076923077, 'DODONNA': 1.0, 'GOLD LEADER': 0.8, 'WEDGE': 0.39285714285714285, 'RED LEADER': 0.5714285714285714, 'RED TEN': 1.0, 'GOLD FIVE': 0, 'RIEEKAN': 1.0, 'DERLIN': 1.0, 'ZEV': 1.0, 'PIETT': 0.5, 'OZZEL': 1.0, 'DACK': 0, 'JANSON': 0, 'NEEDA': 1.0, 'LANDO': 0.6727272727272727, 'JERJERROD': 0, 'BIB FORTUNA': 1.0, 'BOUSHH': 0.8333333333333334, 'ADMIRAL ACKBAR': 0.6363636363636364, 'LOR SAN TEKKA': 0.6666666666666666, 'POE': 0.3916666666666667, 'KYLO REN': 0.4, 'CAPTAIN PHASMA': 0.6190476190476191, 'FINN': 0.5494505494505495, 'UNKAR PLUTT': 1.0, 'REY': 0.5555555555555556, 'GENERAL HUX': 0.3, 'LIEUTENANT MITAKA': 1.0, 'BALA-TIK': 1.0, 'SNOKE': 1.0, 'MAZ': 1.0, 'SNAP': 0.7857142857142857, 'ADMIRAL STATURA': 1.0, 'YOLO ZIFF': 1.0, 'COLONEL DATOO': 0, 'ELLO ASTY': 1.0, 'JESS': 0.7, 'NIV LEK': 1.0}



10. Bridges

A bridge is a connection between two nodes that if it were to be deleted it would cause the component to get divided into two different components. Our Network has bridges, and we need to find them to know which node is important and makes the component stay as it is.

```
In [24]: #Find the bridges of the Network
         nx.has_bridges(G)

Out[24]: True
```

"Screenshot 22, Network has bridges"

We found out that the list below contains all the bridges in our network (8 in total). The links between OBI-WAN and TION MEDON, JAR JAR and TARPALS, KI-ADI-MUNDI and PLO KOON, YODA and CLONE COMMANDER GREE, DARTH VADER and JERJERROD, LUKE and DACK, WEDGE and JANSON, GENERAL HUX and COLONEL DATOO.

```
In [32]: #Which nodes are the bridges of our network?
         list(nx.bridges(G))

Out[32]: [('OBI-WAN', 'TION MEDON'),
          ('JAR JAR', 'TARPALS'),
          ('KI-ADI-MUNDI', 'PLO KOON'),
          ('YODA', 'CLONE COMMANDER GREE'),
          ('DARTH VADER', 'JERJERROD'),
          ('LUKE', 'DACK'),
          ('WEDGE', 'JANSON'),
          ('GENERAL HUX', 'COLONEL DATOO')]
```

```
In [29]: #Find local Bridges
         list(nx.local_bridges(G))

Out[29]: [('OBI-WAN', 'TION MEDON', inf),
          ('JAR JAR', 'TARPALS', inf),
          ('GREEDO', 'HAN', 3),
          ('KI-ADI-MUNDI', 'PLO KOON', inf),
          ('YODA', 'CLONE COMMANDER GREE', inf),
          ('DARTH VADER', 'JERJERROD', inf),
          ('LUKE', 'DACK', inf),
          ('WEDGE', 'JANSON', inf),
          ('GENERAL HUX', 'COLONEL DATOO', inf)]
```

"Screenshot 22, Network total and local bridges"

We have and one local bridge between GREEDO and HAN which means that the removal of this local bridge would increase the shortest path length between these 2 characters to 3 (span).

11. Gender and Homophily

Positive values of r indicate a correlation between nodes of similar degree, while negative values indicate relationships between nodes of different degree. In general, r lies between −1 and 1.

In our case the assortative coefficient is equal to -0.1856. It's a negative number close to 0 that's why we can conclude that our network tends to be disassortative.

```
In [50]: #Homophily
         r = nx.degree_assortativity_coefficient(G)
         print(r)

         -0.1856
```

"Screenshot 22, Assortativity coefficient"

This makes sense because mostly 10-15 characters are the main ones, while the others have a secondary role in the movies. Therefore, the secondary characters tend to have relationships mostly with the protagonists who normally have a high degree of connectivity. That's why we have relationships between characters with low degree levels with characters with high degree levels.

12. Graph Density

The Graph Density of our network is 0.07. The graph density is simply the ratio of actual edges in the network to all possible edges in the network. Therefore, our network is not dense.

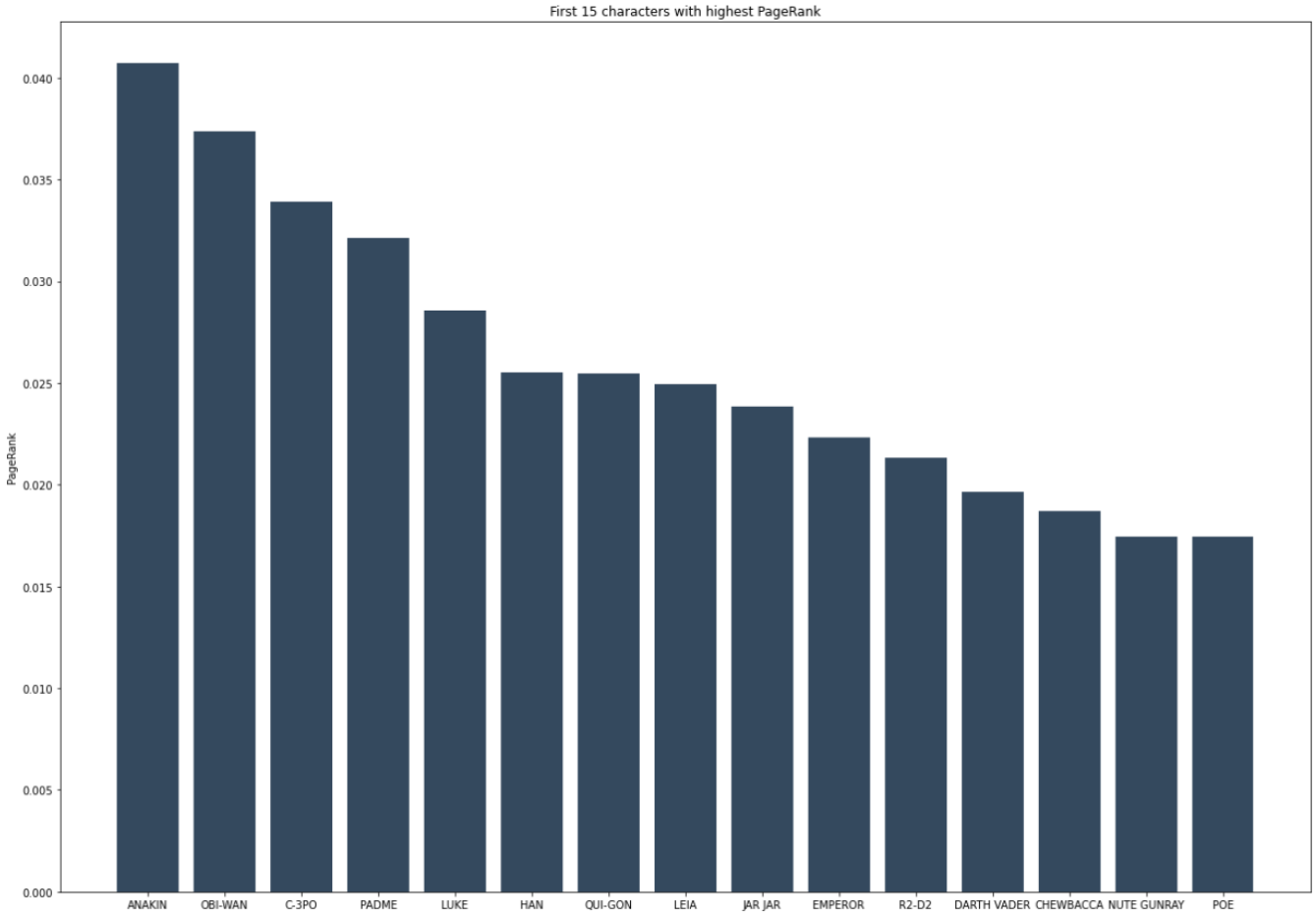
```
In [21]: #Find the Density of the Graph
         nx.density(G)

Out[21]: 0.07239382239382239
```

"Screenshot 22, Graph Density"

13. PageRank

PageRank computes a ranking of the nodes in the graph based on the structure of the incoming links. It was originally designed as an algorithm to rank web pages.

Here we can see the top 15 characters based on their PageRank.We can now say that probably the PageRank has done the best job in outlining the most impactful characters in the Star Wars movies as it has in top 5, ANAKIN, OBI-WAN, C3-PO, PADME, LUKE and HAN.



"Screenshot 21, First 15 characters with highest PageRank"

14. Conclusion

Concluding, after analyzing the Star Wars network and its first episodes, we come to realize that some characters, despite being appeared in most of the scenes (HAN, R2-D2, LUKE) have less influence and connection with the rest of the characters. After computing metrics like Betweenness Centrality, Degree Centrality etc., we can see that ANAKIN, OBI-WAN and C3-PO play a much more important role on the Star Wars universe than we already knew. Lastly, because it's a movie, it was almost obvious that the graph is mostly centered in the protagonists as the connectivity between the secondary ones and the protagonists is high, making the network disassortative. The density of the network is low because the total amount of nodes and edges is relatively low.

15. Tools Used

- Atom: Editor
- Pandas & Jupyter Notebook: Data Preprocessing
- Gephi: Some Network Representation & Analysis
- NetworkX: Python Library used for calculation of some measures in the graph.

16. Sources

- Barabási A.L. (2012). Network Science. Cambridge, United Kingdom: Cambridge University Press.
- Easley, D., & Kleinberg, J. (2010). Networks, crowds, and markets. New York, NY: Cambridge University Press.