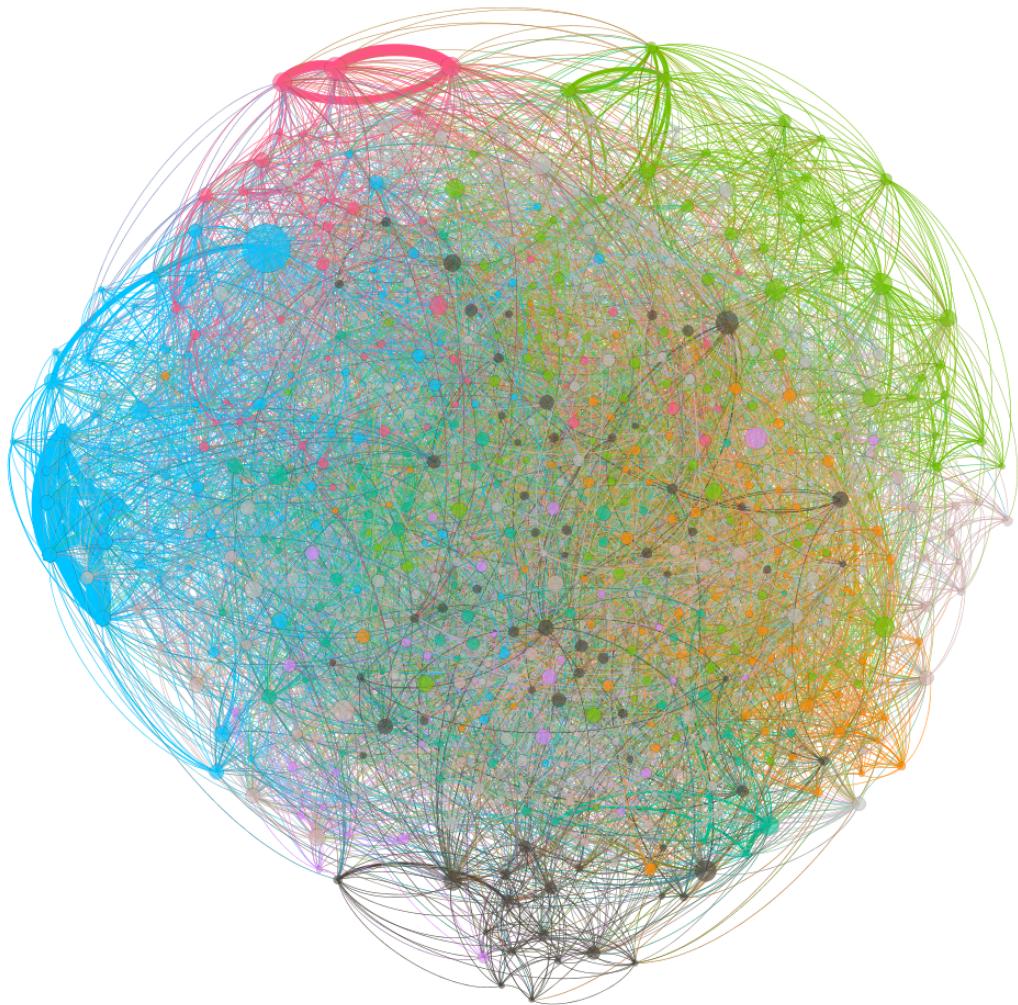


# Práctica 1. Análisis de una red

Marlon Campoverde, Rebeca Herranz

Guillermo Ovejero, Carlos Sánchez

Octubre de 2019



# Índice

	Página
1. Introducción	1
2. Obtención de datos	2
3. Obtención del grafo	2
4. Análisis inicial del grafo	3
5. Análisis por grupos del grafo	12
6. Conclusiones	15
7. Apéndice Código	16

# 1. Introducción

Este documento se ha realizado mediante el estudio de una lista de películas y actores, que hemos tenido que revisar para crear esta red.

Los nodos son las películas y las aristas son los actores que han participado en ambas películas. El grafo entonces está formado por la lista de nodos y aristas obtenida mediante el procesamiento de los datos que tenemos. Debido a la naturaleza de estos datos se crearán diferentes grupos y comunidades.

En este trabajo se estudiarán estas comunidades y cómo se ha obtenido los datos para desarrollar el grafo final mediante Python para procesar los datos y un pequeño análisis inicial, y Gephi para poder visualizar el grafo y poder analizarlo en detalle.

## 2. Obtención de datos

El profesor ha dispuesto un archivo, llamado *actorsMovies.csv* en el que había dos columnas, la primera el nombre del actor y la segundo una lista de películas en las que había actuado, se ha tenido que procesar esta información mediante Python para obtener el formato de grafo que buscamos, mediante un notebook, mostrado en el Apéndice, se han obtenido los dos nuevos ficheros csv. Una tabla de nodos (películas) con 1896 nodos y una tabla de aristas (actores) con 18797 aristas.

## 3. Obtención del grafo

Al tener ya los datos separados en tablas se ha podido importar esta información directamente a Gephi para poder visualizar y analizar el grafo. Una vez importados ambos ficheros, se procede a realizar el calculo de diferentes parámetros que luego nos servirán para poder filtrar y visualizar de forma correcta el grafo. Después se procede a cartografiar el grafo seleccionando como separación del color de los nodos la métrica de *Modularity Class*, y las aristas se quedan por defecto. Los layouts elegidos para las diferentes para analizar el grafo han sido tres principalmente.

- *Fruchterman Reingold*
- *Force Atlas 2*
- *OpenOrd*

Y por ultimos los filtros usados para reducir el tamaño del grafo han sido Igualdad por *Modularity Class* y *k-core*

## 4. Análisis inicial del grafo

Para el análisis inicial se han utilizado los filtros *k-core*, para reducir el número de películas que se visualizan y el filtro por *Modularity Class*, y para realizar un layout que pueda ser bien visualizado se ha utilizado principalmente el algoritmo *Fruchterman Reingold*

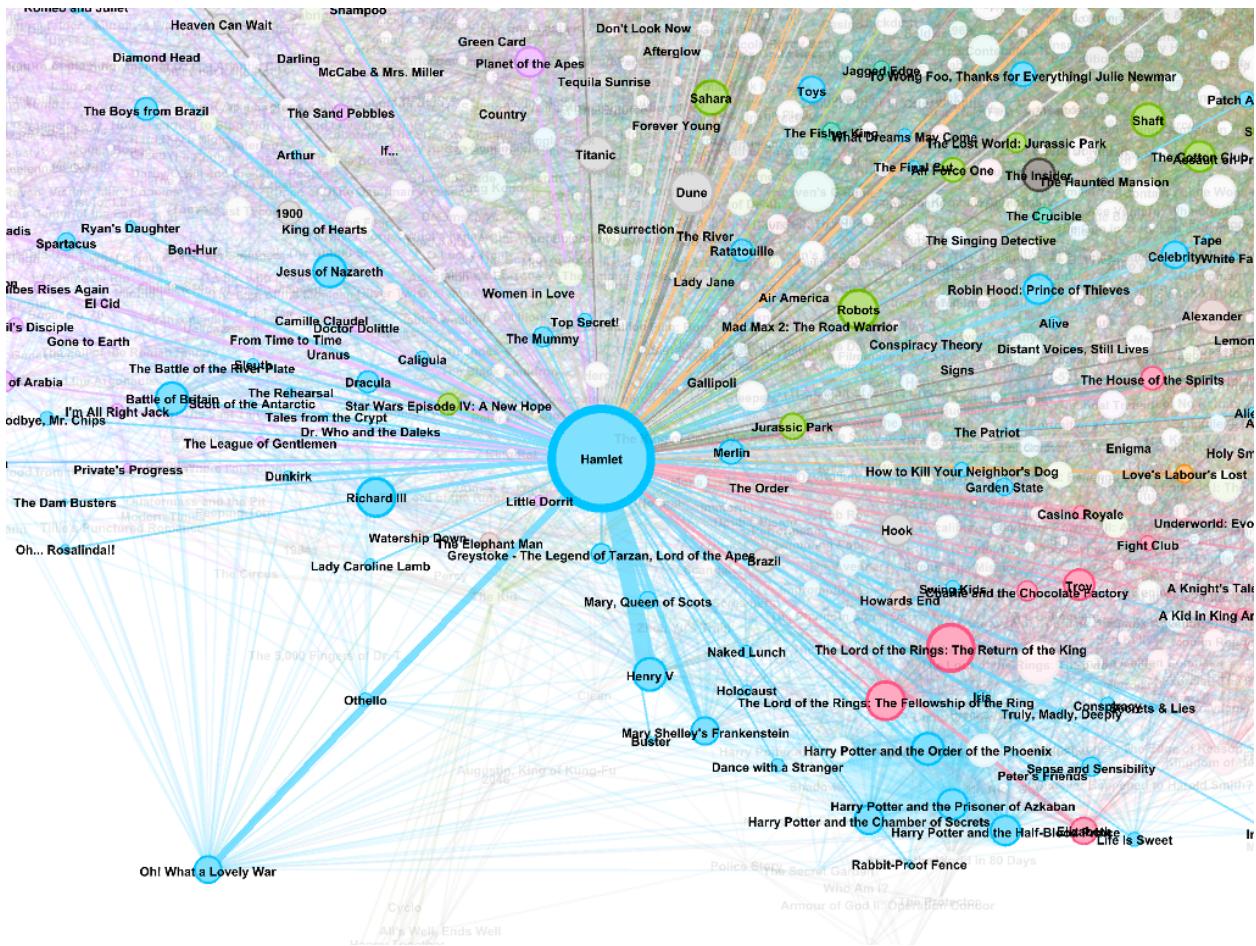


Figura 1: Hamlet Centrality

Esta imagen representa al nodo con mayor grado, que es *Hamlet*, al ser la película que más actores han participado, aumenta el número de películas relacionadas con esa y por lo tanto sus conexiones. También se puede observar una gran conexión entre *Hamlet* y *Henry V*, esto se debe a que casi todo el cast de ambas películas es el mismo y por eso están tan conectadas entre si.

Este parte del grafo total representan películas ficticias y de fantasía.

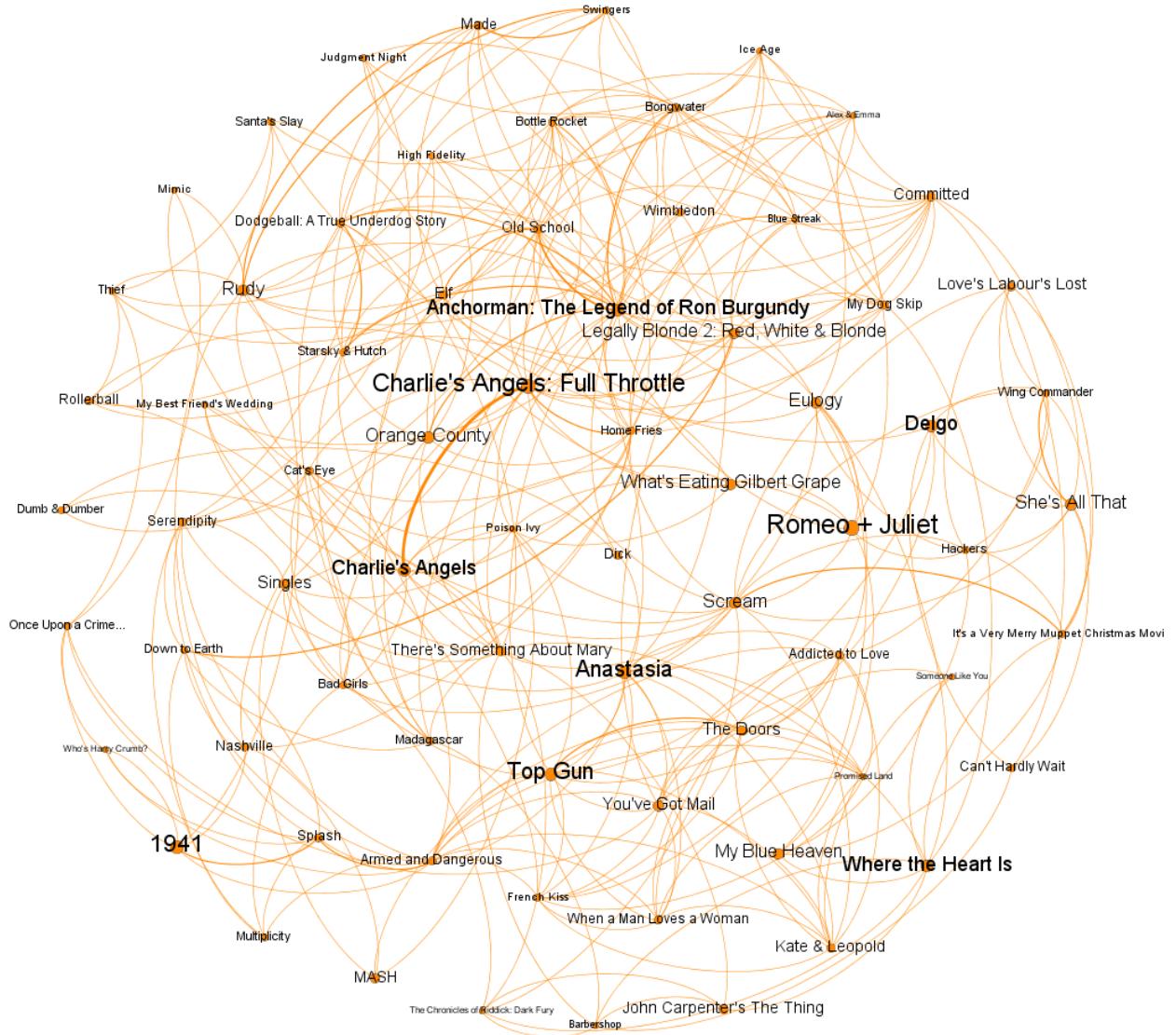


Figura 2: Grupo naranja

Este grupo se caracteriza por la aparición de mujeres en elenco principal de las películas, como ejemplos tenemos: *Charlies's Angels*, *Anastasia*, *Romeo & Juliet*.

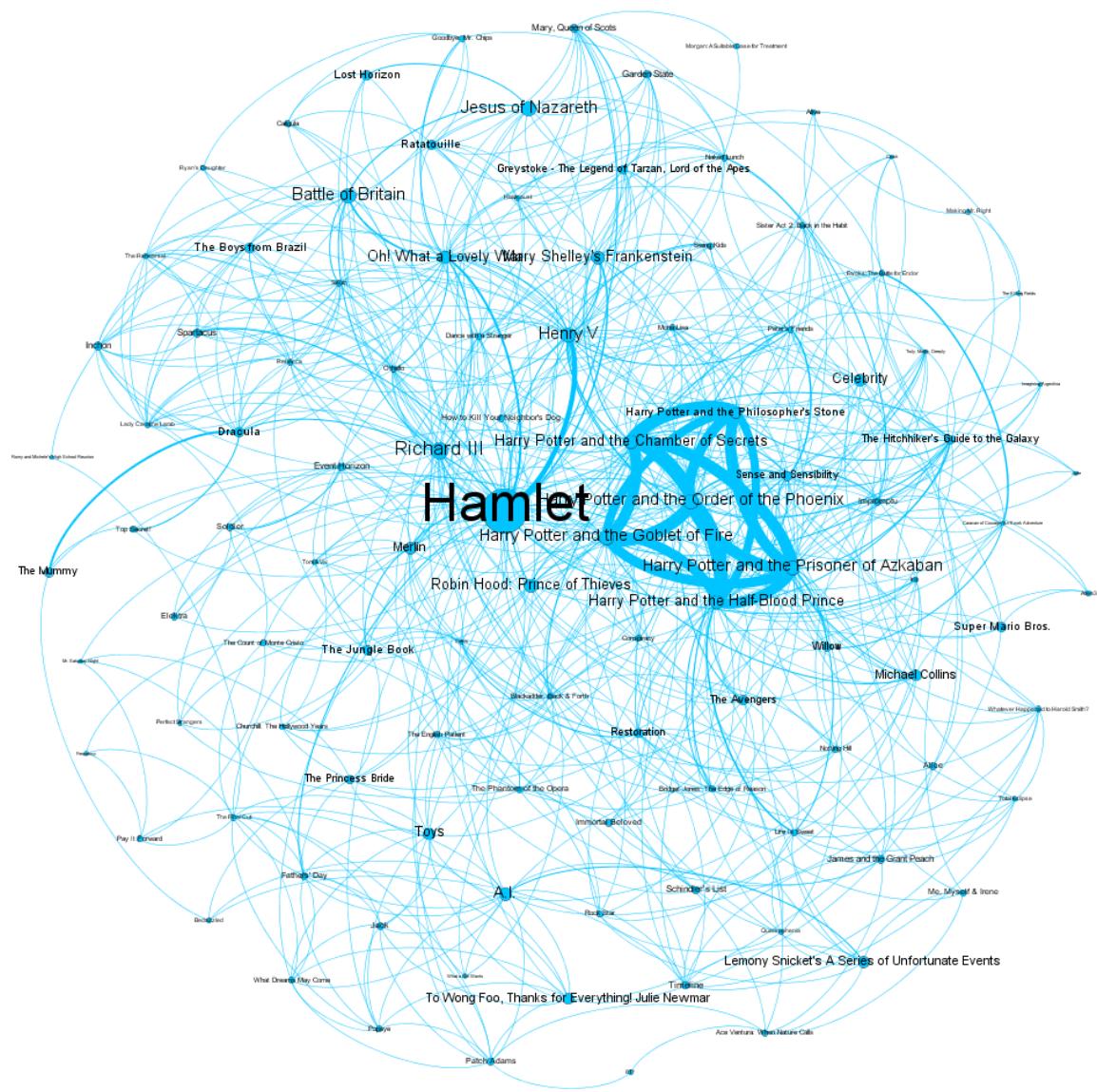


Figura 3: Grupo Azul

En este grupo bastante relacionado con películas de fantasía y ficción, es de los más numerosos, entra las películas con mas grado se encuentra *Hamlet*, la cual se relaciona fuertemente con *Henry V*, ambas comparten un cast similar, también podemos observar un grupo de películas muy relacionadas, las de la saga *Harry Potter*

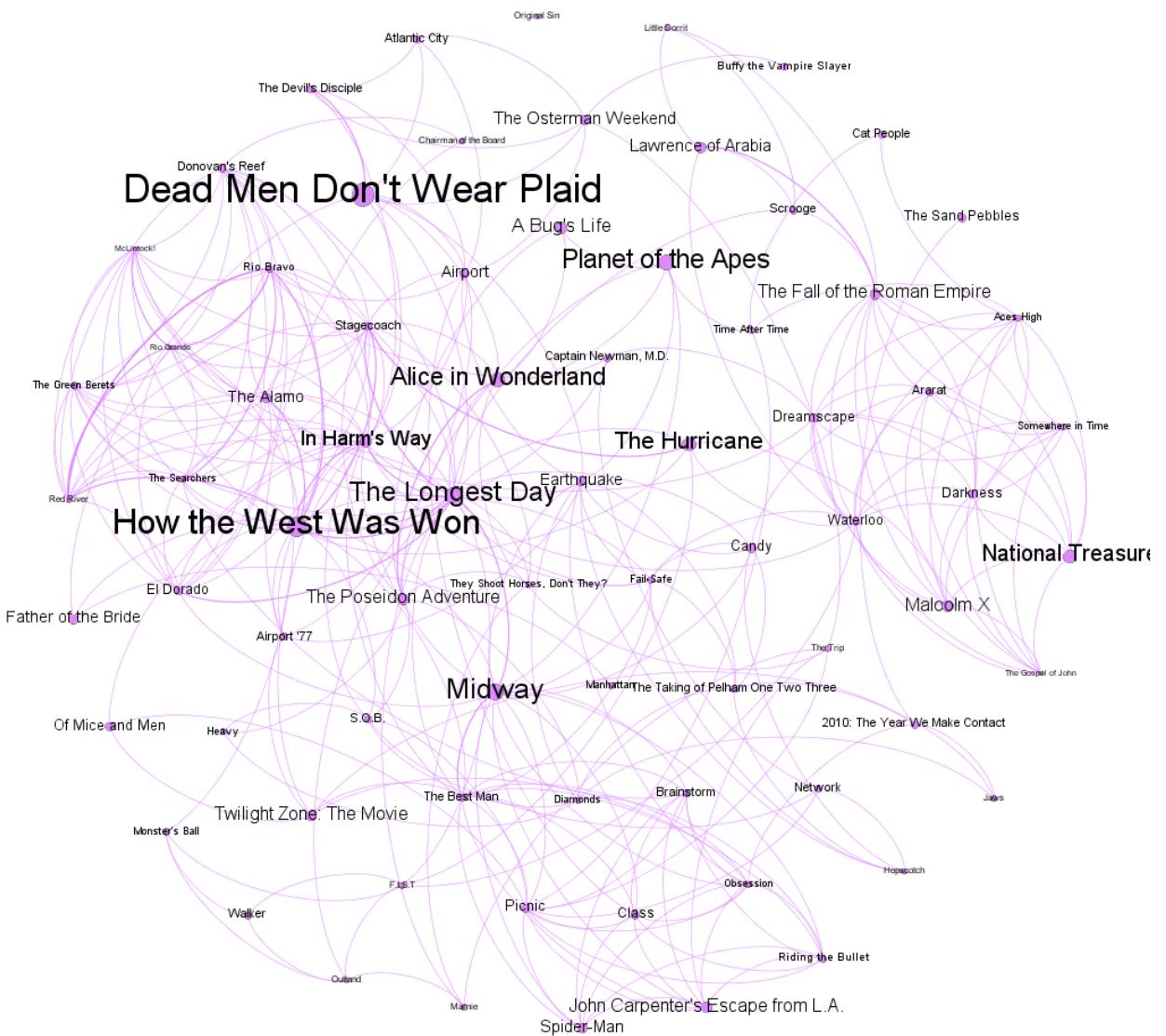


Figura 4: Grupo morado

Las películas representadas en el grafo son principalmente clásicos del cine como *Alice in Wonderland*, *Dead men don't wear plaid* o *How the West Was Won*, aunque también encontramos algunas películas de guerra como *The Longest Day*.

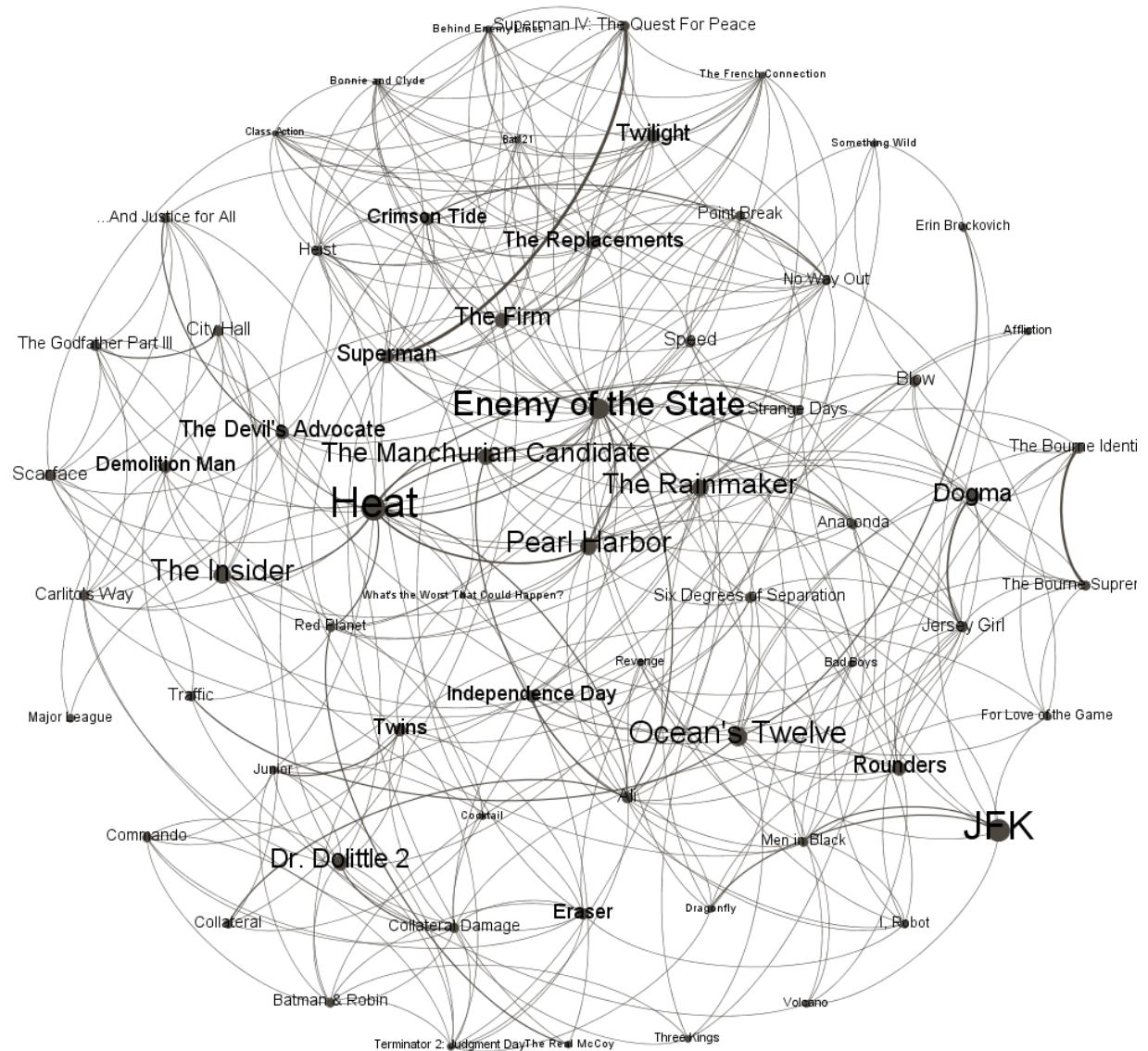


Figura 5: Grupo negro

La temática de estas películas es lo que marca la diferencia con las demás ya que son sobre todo de acción y asesinatos como *Heat*, *Enemy of the State*, *Scarface*, aunque también nos encontramos con algún outlier como *Dr Dolittle 2*.

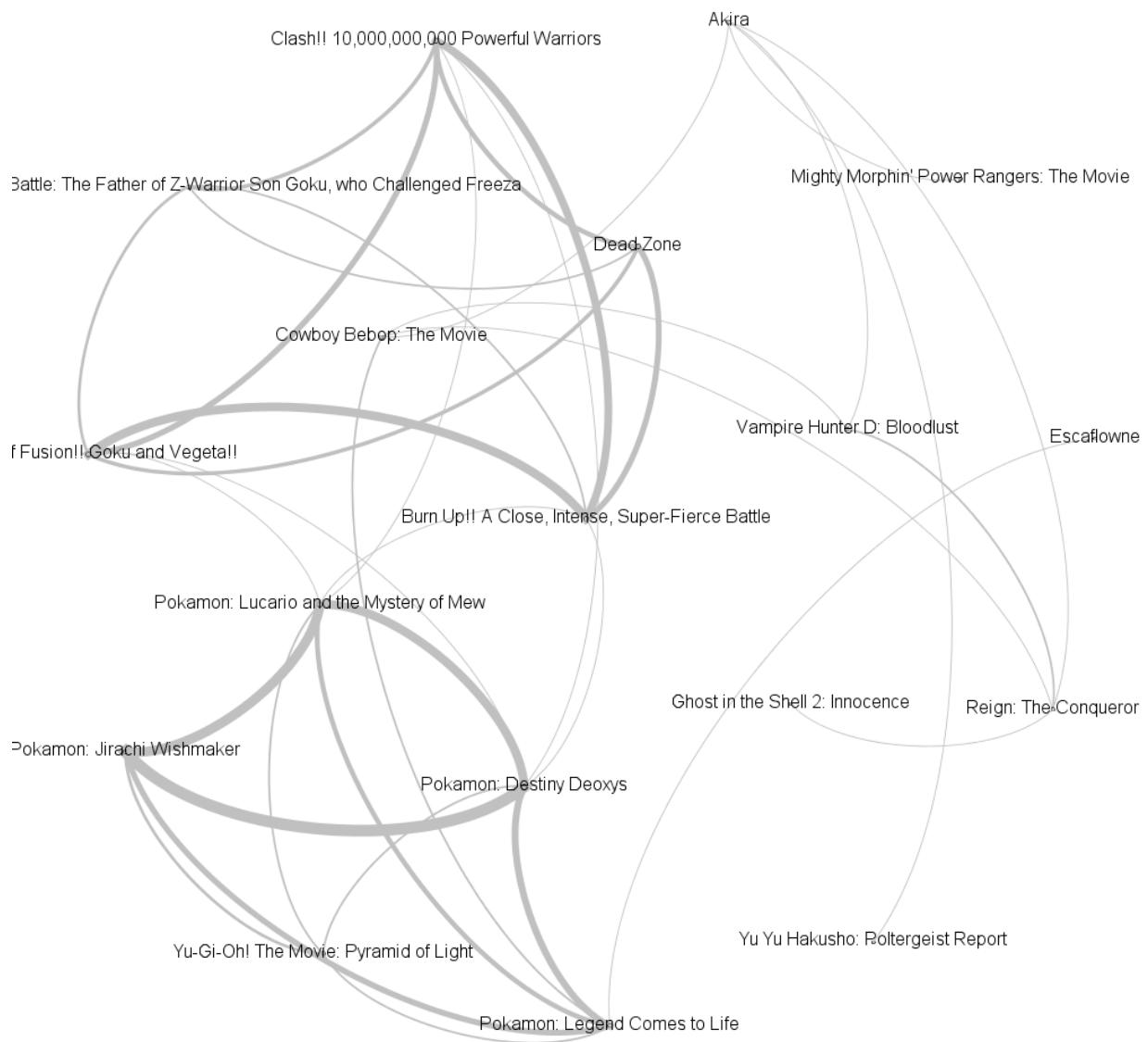


Figura 6: Grupo animación

Este grupo de pequeño tamaño esta relacionado por la procedencia y el estilo de estas, casi todas son japonesas y de animación (anime), tenemos *Pokémon*, *Akira*, por otro lado vemos a los *Power Rangers*, película americana que no es de animación, pero que esta muy inspirada en el cine y la cultura japonesa

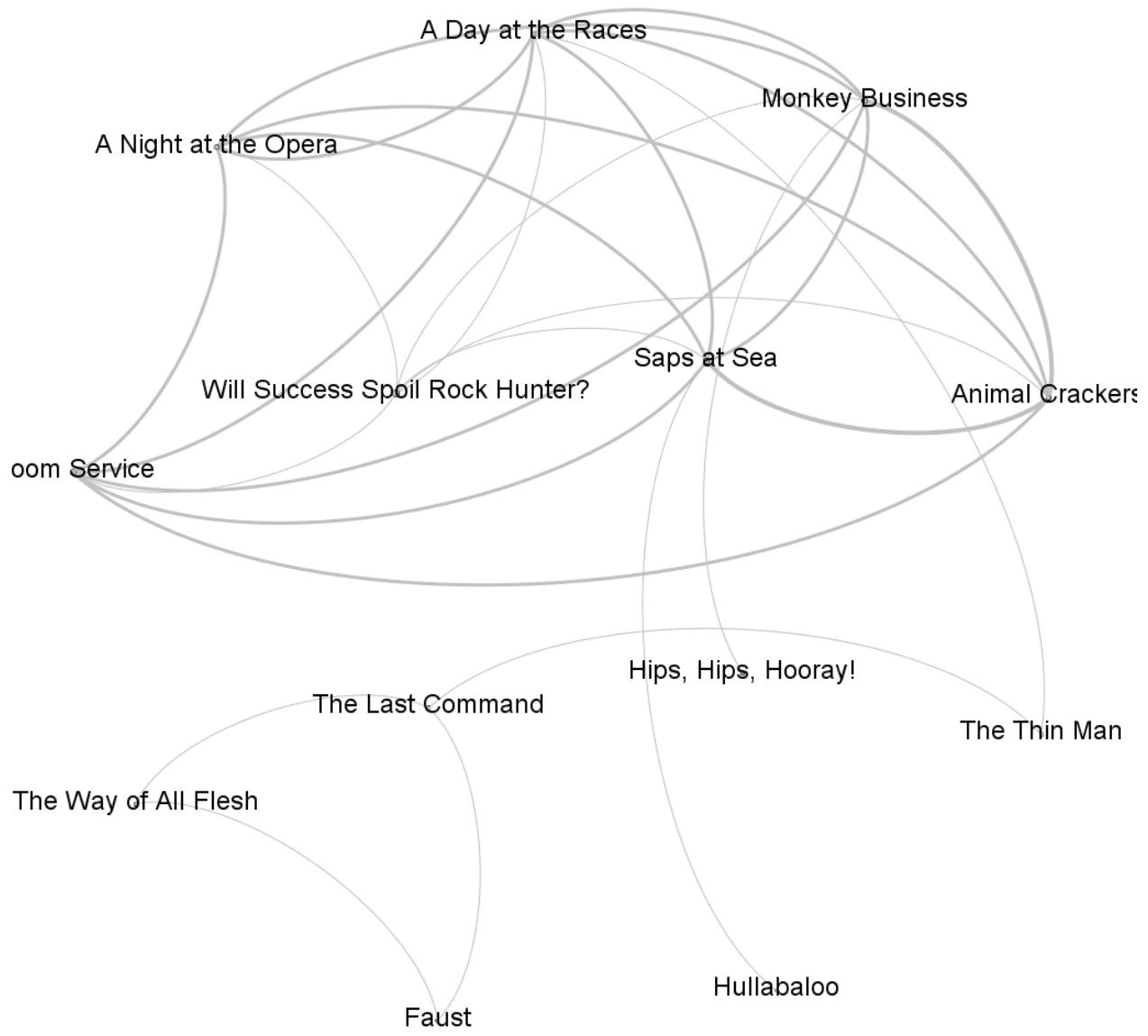


Figura 7: Grupo Hermanos Marx

En este grupo, es fácil analizar ya que en la mayoría de las películas aparecen los Hermanos Marx.



Figura 8: Grupo Disney

Este grupo tiene distintos tipos de películas, pero en su mayoría son de Disney o dirigidos a un público infantil, a excepción de algunas como *IT*. También está presente la saga de Star Trek.

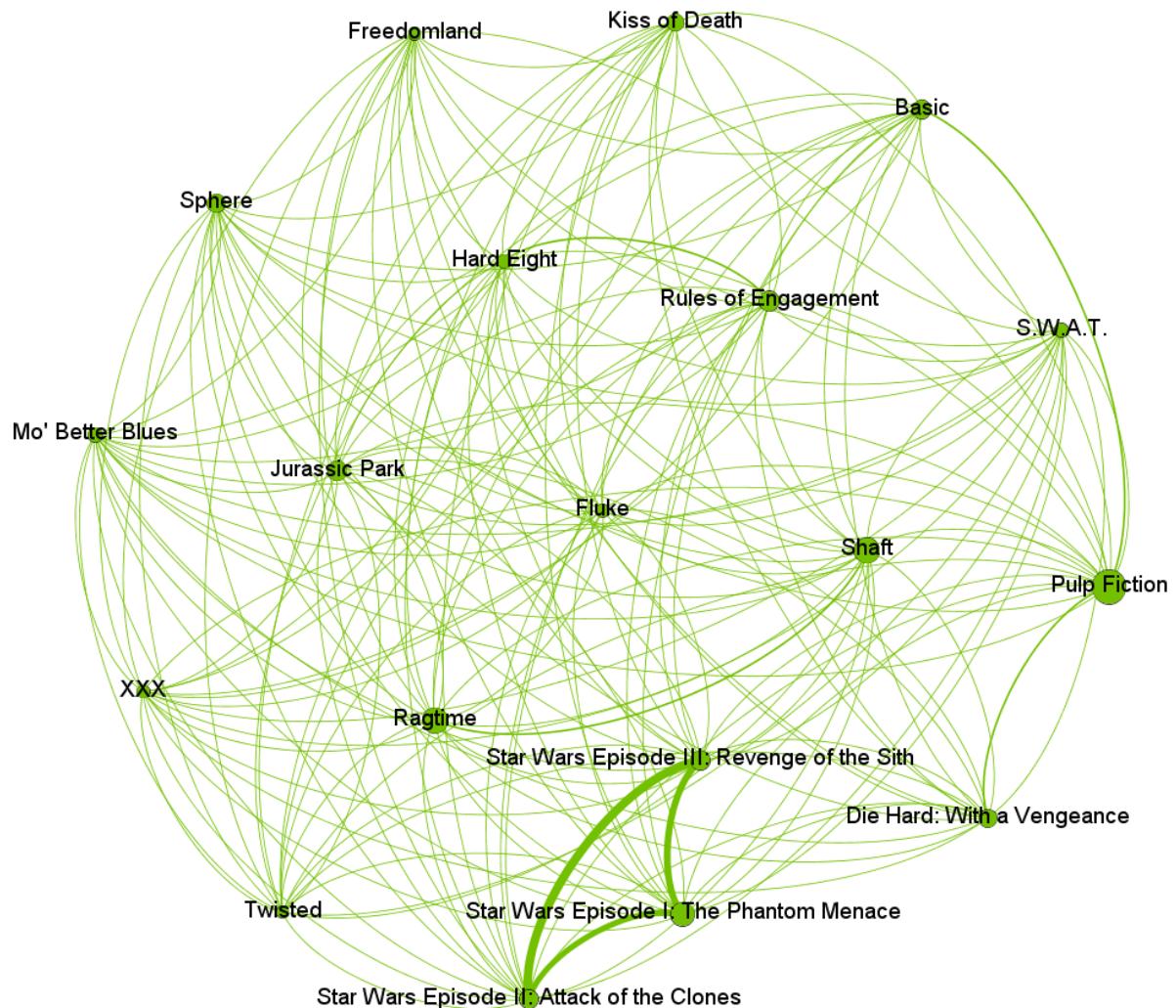


Figura 9: Grupo verde

Este grupo tiene como actores Bruce Willis, Samuel L.Jackson, si aumentamos el filtro del k-core hasta 17, se puede observar que en todas las películas aparece Samuel L.Jackson.

## 5. Análisis por grupos del grafo

Este segundo análisis por grupos del grafo total se ha realizado utilizando los algoritmos *Force Atlas 2* y *Open Ord*, estos algoritmos nos permiten separar muy bien en los grupos de películas para su correcta visualización, aquí no se aplica ningún filtro pues observamos grupos muy pequeños de películas.

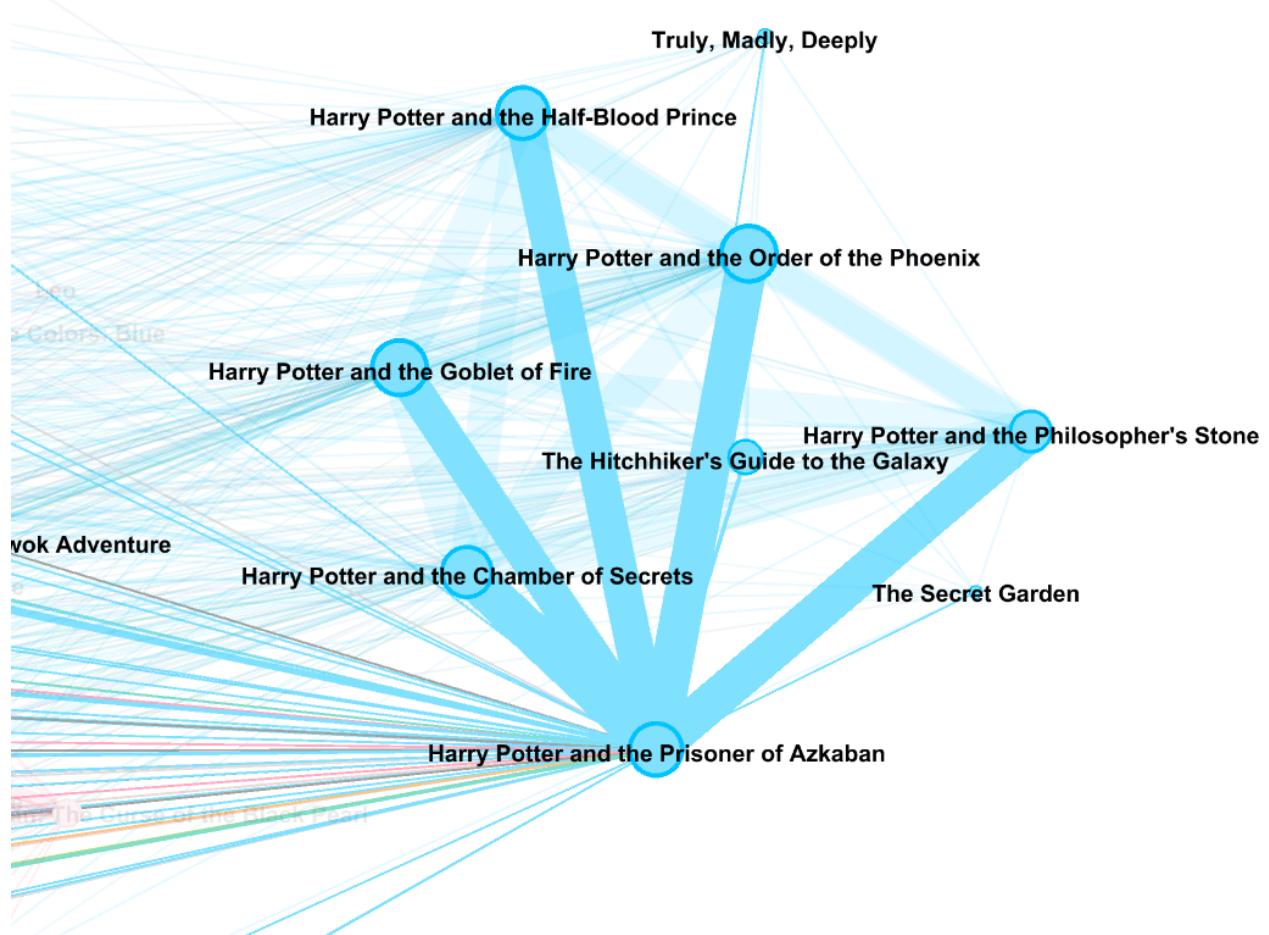


Figura 10: Harry Potter Group

En este grafo se observa como están conectadas las películas de Harry Potter confirmando así que forman una comunidad. Las películas *The Prisoner of Azkaban*, *The Philosopher's Stone*, *The Order of the Phoenix*, *The Half-Blood Prince*, *The Goblet of Fire*, *The Chamber of Secrets* están relacionadas ya que los actores en todas las películas son los mismos.

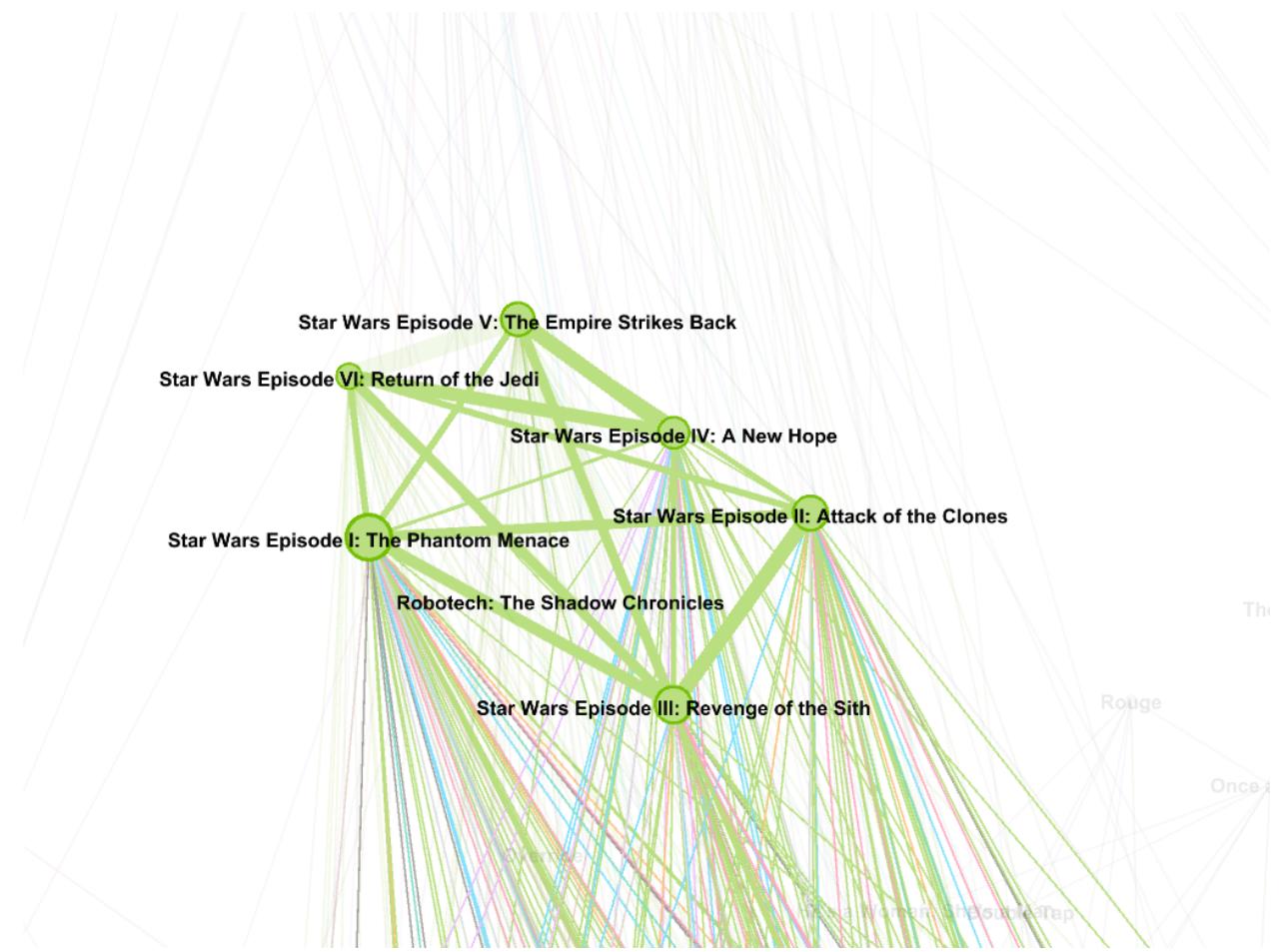


Figura 11: Star Wars Group

En este grafo se observa como están conectadas las películas de Star Wars confirmando así que forman una comunidad. Las películas están relacionadas ya que los actores en todas las películas son los mismos.

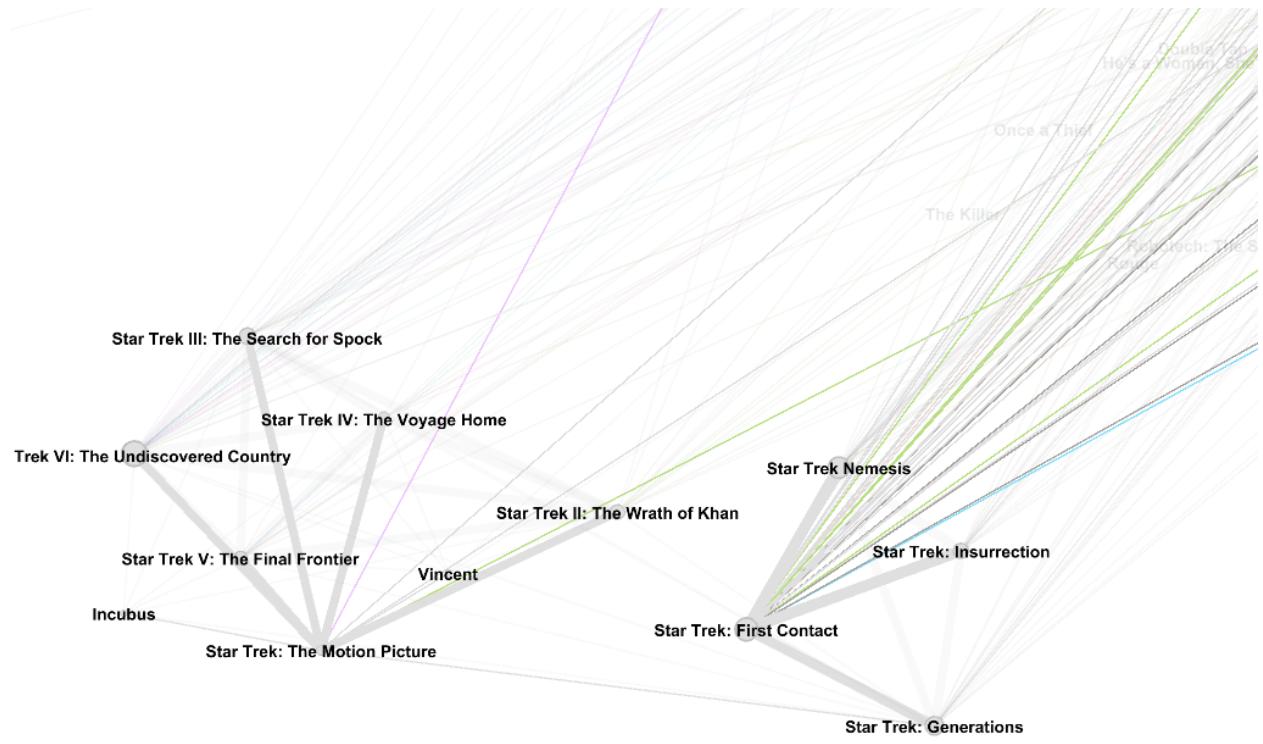


Figura 12: Star Trek Groups

En este grafo se observan como están conectadas las películas de Star Trek confirmando así que forman una comunidad. Las películas están relacionadas ya que los actores en todas las películas son casi los mismos.

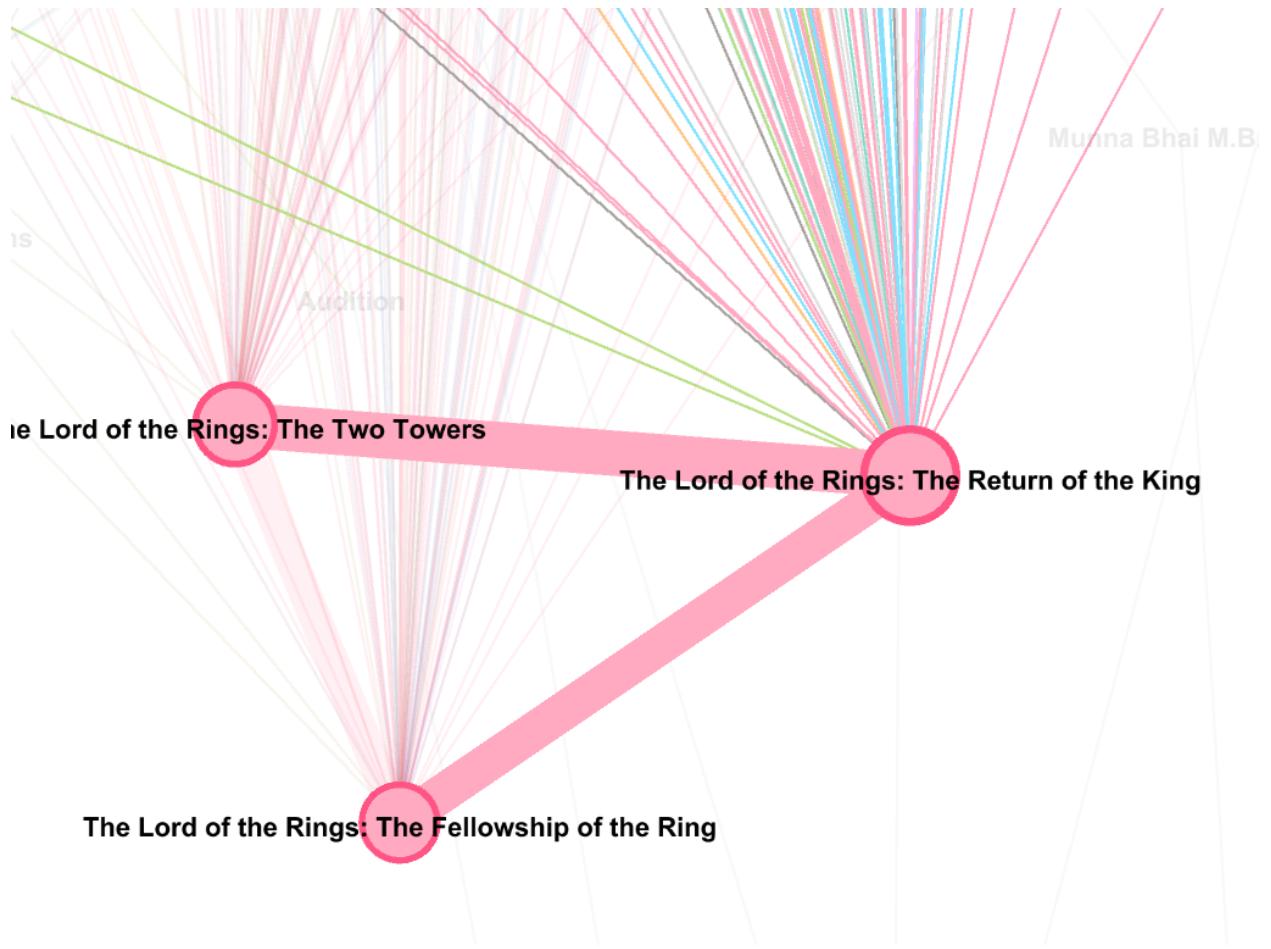


Figura 13: The Lord of the Rings Group

En este grafo se observan como están conectadas las películas de The Lord of the Rings confirmando así que forman una comunidad. Las películas están relacionadas ya que los actores en todas las películas son casi los mismos.

## 6. Conclusiones

Como conclusión a este trabajo es que se ha podido analizar con eficacia una tabla de actores y películas. Los datos más importantes que se ven reflejados son la centralidad dominada por Hamlet y los grupos y comunidades que se forman debido a la aparición de los mismos actores o la diferencia de temática.

Como se puede observar por la separación por clases, podemos ver que los actores tienden a agruparse en un cierto genero parecido de películas, haciendo que se les contrate por ejemplo, sobre todo en películas de acción, o en películas de comedia. Mediante las aplicaciones de Gephi y Python ha sido más fácil este análisis, pudiendo así sacar la cantidad de nodos y aristas que hay en el grafo principal.

## 7. Apéndice Código

Visualización del código realizado en Python, en el entorno de Jupyter Notebooks, aquí se puede observar paso a paso como se ha obtenido la separación de los datos del archivo *actorMovies.csv* y se ha acabado convirtiendo en dos archivos separados.

El archivo *nodes\_list.csv* y el archivo *edges\_list.csv* ambos se corresponden con los nodos y aristas que después se usan en Gephi para obtener los resultados vistos en los anteriores apartados.

### actors\_and\_movies\_notebook.ipynb

```
[1]: import networkx as nx
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import os

SOC_URL = "./actorMovies.csv"
df = pd.read_csv(SOC_URL, sep=";")

for i in range(df["Movies"].count()):
    df["Movies"][i] = df["Movies"][i].split(" | ")

df.head(25)

total = 0
for i in range(df["Movies"].count()):
    total += len(df["Movies"][i])
print(total)
df.head()
```

10247

```
[1]:          Actor                  Movies
0      Karen Allen  [Malcolm X, National Lampoon's Animal House, S...
1      Taye Diggs  [House on Haunted Hill, Go, Basic, Chicago, Eq...
2  Michael Murphy  [Salvador, Cloak & Dagger, Nashville, Salvador...
3   Les Tremayne  [The War of the Worlds, The War of the Worlds]
4 Bridgette Wilson  [Mortal Kombat, Nixon, House on Haunted Hill]
```

First we need to convert the .csv that we are given to the format that we want

Nodes -> Films Edges -> Films with same actor

```
[2]: df["Movies"][18]
```

```
[2]: ['Faces', 'Faces']
```

We have duplicated movies in some actors

```
[3]: for i in range(df["Movies"].count()):  
    df["Movies"][i] = list(set(df["Movies"][i])) #Can't use pandas.  
    ↪unique() cos type of ["Movies"][i] is -> list  
df["Movies"][18]
```

```
[3]: ['Faces']
```

Now we make the nodes list just by appending all the films

```
[4]: movies = df.iloc[:,1].copy() #We copy the actors series  
#movies = movies.to_frame() #Convert it to a dataframe  
nodes = pd.DataFrame(columns=['Id', 'Movie'])  
id = 0  
for i in range(len(movies)):  
    for j in range(len(movies[i])):  
        nodes = nodes.append({'Id': id, 'Movie': movies[i][j]},  
        ↪ignore_index=True)  
        id+=1  
  
print(len(nodes))  
nodes.head()
```

10071

```
[4]:   Id          Movie  
0   0          Starman  
1   1  National Lampoon's Animal House  
2   2          Malcolm X  
3   3          Basic  
4   4  House on Haunted Hill
```

```
[6]: #We eliminate the duplicate movies  
nodes_list = nodes  
nodes_list = nodes_list["Movie"].unique()  
nodes_list = pd.DataFrame({'Movie': nodes_list[:]})  
nodes_list.rename(columns={0:"Id"})  
nodes_list.index.name = "Id"  
nodes_list.head()  
nodes_list.to_csv("nodes_list.csv")
```

```
nodes_list.tail()
```

```
[6]:
```

	Movie
Id	
1891	Omagh
1892	The Great St Trinian's Train Robbery
1893	Hullabaloo
1894	Wasabi
1895	Hawks

```
[7]: edges = pd.DataFrame(columns=["Source", "Target", "Weight", "Type"])
```

```
#is_empty = lambda x,y: edges.loc[(edges["Source"] == x) & (edges["Target"] == y)].count().all() == 0
is_empty = lambda x,y,edges: edges.loc[(edges["Source"] == x) & (edges["Target"] == y)].count()[0] == 0

get_index = lambda movie_name : nodes_list.loc[nodes_list["Movie"] == movie_name, "Movie"].index[0]

def add_weight(x,y,edges):
    edges.loc[(edges["Source"] == x) & (edges["Target"] == y), "Weight"] += 1

def get_edges(edges):
    for i in range(df["Movies"].count()):
        for j in range(len(df["Movies"][i])):
            #Dont go over the full list -> quicker
            #why? -> last element connected in previous iter
            for k in range(j+1,len(df["Movies"][i])): #Dont add equal
                #edges or already added
                source = get_index(df["Movies"][i][j])
                target = get_index(df["Movies"][i][k])
                if(source != target):
                    if(not is_empty(source,target,edges)): #if there
                        #is one in k->j, we increase weight
                        add_weight(source,target,edges)
                    elif(not is_empty(target,source,edges)): #if there
                        #is one in j->k, we increase weight
                        add_weight(target,source, edges)
                    else:
                        #both are empty -> add another edge
                        edges = edges.append({"Source":source, "Target":target, "Weight":1, "Type":"Undirected"}, ignore_index=True)
```

```
    return edges
edges = get_edges(edges)
len(edges)
edges.tail()
```

[7]:

	Source	Target	Weight	Type
18792	83	1267	1	Undirected
18793	83	1272	1	Undirected
18794	906	1272	1	Undirected
18795	1267	1272	1	Undirected
18796	644	82	1	Undirected

[8]:

```
edges.rename(columns={0:"Id"}) #Adding an index column
edges.index.name = "Id"
edges_test = edges.copy()
```

[9]:

```
edges_test.drop_duplicates(subset=["Source","Target"], inplace=True)
print(len(edges))
len(edges_test)
```

18797

[9]:

18797

[10]:

```
edges_test.to_csv("edges_list.csv")
```