Spotify

Abraham Corta Ramírez y José Calcedo Vázquez

2021-10-01

Índice

Índice de figuras

1. Introducción a Spotify

Para este trabajo de programación la plataforma que vamos a utilizar es Spotify, esta aplicación es empleada para la reproducción de música vía streaming. Actualmente Spotify es uno de los líderes del sector y contiene millones de canciones y cientos de miles de artistas de todos los géneros.

Spotify ofrece una Web API con la que nos permite acceder a numerosos datos como canciones, artistas, playlists, etc. Y no solo eso, sino que de una canción en concreto se pueden obtener datos como su tempo, la duración, su grado de «instrumentalidad», de energía, etc. Esta API se puede utilizar con diferentes lenguajes como PHP, Java, JavaScript o Python entre otras mediante el uso de librerías.

Nosotros hemos decido hacerlo con Python debido a que es mas ligero que Java y más simple de utilizar.

La API se compone de muchos componentes, a continuación, se nombran algunos de estos:

- **Peticiones**
- **Spotify URIs e IDs**
- **Respuestas: en formato JSON**
- **Paginación**
- **Autenticación**

Nuestro objetivo será manipular los datos de la API sobre un artista en concreto, y partiendo de ahí estudiaremos sus conexiones, los artistas relacionados y las playlists.

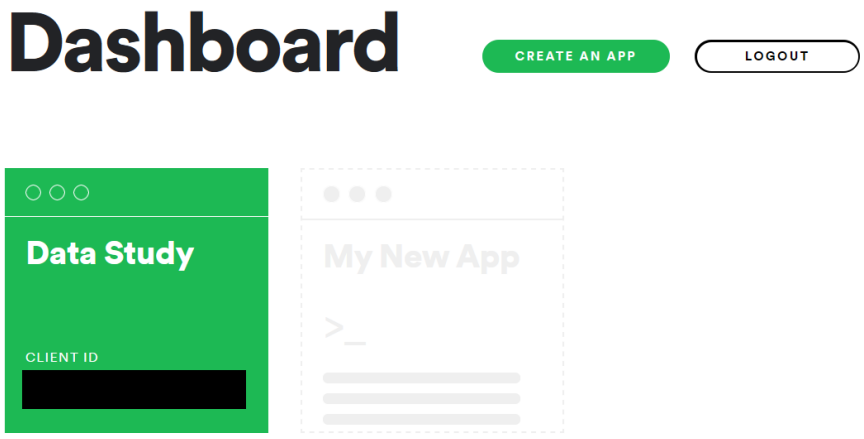


Figura 1: Spotify dashboard

El primer paso es crearnos una cuenta como desarrollador Spotify, si ya tenemos cuenta de Spotify solo necesitaremos iniciar sesión en la web de desarrolladores. Será necesario dar de alta una aplicación, de esta manera obtendremos un Client ID y un Client Secret. Veasé las siguientes figuras: [1]

[1] Se han ocultado los client ID de las figuras ya que este puede ser usado por cualquier persona.

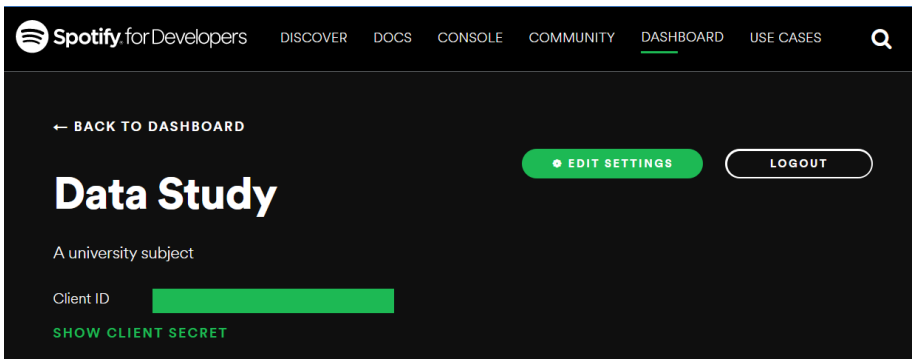


Figura 2: Spotify application

Una vez tengamos nuestra aplicacion creada necesitaremos el client ID y el client secret para poder autenticarnos y recuperar datos de la API.

Nuestro trabajo se compone de dos programas, el primero escrito en python utilizando Visual Studio Code que nos permite recuperar datos de la API de Spotify y exportarlos a JSON. Y el segundo escrito en SageMath que nos permite representar y estudiar esos datos con teoría de grafos.

2. Obtención de datos

El codigo python se compone de tres secciones

- **Conexion**
- **Recolección de datos**
- **Exportación**

Para que el codigo funcione solo se han necesitado 3 librerias:

- **spotipy.oauth2-SpotifyClientCredentials:** nos permite hacer la autenticación.
- **pandas:** que nos permite crear los dataframes que vamos a exportar a SageMath.
- **spotipy:** nos permite interactuar con la API.

A continuación se explicaran las imagenes presentes en la figura 3.

Lo primero que hacemos es obtener todas las playlists relacionadas con el artista. Para ello hemos tomado la URL de cada una de las playlists de 50 Cent y las hemos almacenado para su posterior uso. Tras ello, para cada playlist llamamos al comando sp.user_playlist, que nos permite obtener los datos de una playlist dada su URL, para más tarde almacenar las canciones que contiene cada una.

Una vez obtenidas las canciones, creamos un bucle con el que para cada canción que hayamos obtenido, obtenemos primeramente un dataset de las playlists, donde metemos los nombres de las canciones junto al nombre de sus artistas. Tras hacer eso, creamos un dataset con los artistas, obteniendo su nombre, géneros musicales asociados, popularidad y seguidores con el comando sp.artist.

```python
from spotipy.oauth2 import SpotifyClientCredentials
import pandas as pd
import spotipy

# Conexion con la API
sp = spotipy.Spotify()
cid = "************"
secret = "**********"
client_credentials_manager = SpotifyClientCredentials(
    client_id=cid, client_secret=secret)
sp = spotipy.Spotify(client_credentials_manager=client_credentials_manager)
sp.trace = False
```

(a) conexion API

```python
# Usuario 50 cent
# https://open.spotify.com/user/1217753577

# Playlists de 50 cent
# The Singles Collection: https://open.spotify.com/playlist/1hBQZ8nCtIHrEByxICx8tU
# Featuring 50cent: https://open.spotify.com/playlist/3pDliBuh3MwiSfjOKo5mKl
# 50 cent best of: https://open.spotify.com/playlist/49RoQF55lyRSgSZwRAHh5K
# 50 my playlist: https://open.spotify.com/playlist/1s7ipxTB42mCucqQUWVMP4
```

(b) Urls que utilizaremos con la API

```python
#lista donde tenemos todas las playlists anteriores
pp = ["3pDliBuh3MwiSfjOKo5mKl", "1hBQZ8nCtIHrEByxICx8tU",
      "49RoQF55lyRSgSZwRAHh5K", "1s7ipxTB42mCucqQUWVMP4"]

for i in pp:
    # obtenemos una playlist
    playlist = sp.user_playlist("1217753577", i, fields="tracks,next")
    tracks = playlist["tracks"]
    songs = tracks["items"]

    ids = []
    song = []
    artist = []

    # obtenemos las canciones
    for k in range(len(songs)):
        s = songs[k]["track"]
        ids.append(s["id"])
        artists = []

        for j in range(len(s["artists"])):
            # dataset de playlists
            artists.append(s["artists"][j]["name"])
            song.append([s["name"], s["popularity"], artists])

            # dataset de artistas
            a = sp.artist(s["artists"][j]["id"])
            artist.append(
                [a["name"], a["genres"], a["popularity"], a["followers"]["total"]])

    # Exportamos a Json los datos recabados
    dataArtists = pd.DataFrame(artist)
    dataSongs = pd.DataFrame(song)
    outputArtists = "api/50cent-dataArtists"+str(pp.index(i)+1)+".json"
    output = "api/50cent-dataPlaylist"+str(pp.index(i)+1)+".json"
    dataSongs.to_json(output, orient='records')
    dataArtists.to_json(outputArtists, orient='records')
```

(c) Recolección y exportación de los datos

Figura 3: Código Python

Finalmente, convertimos los datasets obtenidos anteriormente en Data Frames usando pandas (en el código lo usamos como pd) que posteriormente son procesados en formato JSON con el comando .to_json, al cual le pasamos como argumento la ubicación de donde va a terminar el fichero con los datos y el dataframe correspondiente.

3. Creación del grafo y otros cálculos

Una vez hemos extraído los datos de la API es turno de importar los datos Json, para ello utilizaremos la librería *json*.

Una vez importado utilizaremos dos listas que englobaran los 4 datasets, uno para las playlist y otro para los artistas.

```
import json
USER_DIR = 'C:/Users/abram/Documents/MATI/TrabajoDeCurso/Spotify/api'
```

(a) librerias json

```
jArt1 = os.path.join(USER_DIR,'50cent-dataArtists1.json')
jArt2 = os.path.join(USER_DIR,'50cent-dataArtists2.json')
jArt3 = os.path.join(USER_DIR,'50cent-dataArtists3.json')
jArt4 = os.path.join(USER_DIR,'50cent-dataArtists4.json')
dataArt1 = json.loads(open(jArt1).read())
dataArt2 = json.loads(open(jArt2).read())
dataArt3 = json.loads(open(jArt3).read())
dataArt4 = json.loads(open(jArt4).read())
dataArtists = [dataArt1, dataArt2, dataArt3, dataArt4]
```

(b) importación de los datos de los artistas

```
jPlay1 = os.path.join(USER_DIR,'50cent-dataPlaylist1.json')
jPlay2 = os.path.join(USER_DIR,'50cent-dataPlaylist2.json')
jPlay3 = os.path.join(USER_DIR,'50cent-dataPlaylist3.json')
jPlay4 = os.path.join(USER_DIR,'50cent-dataPlaylist4.json')
dataPlay1 = json.loads(open(jPlay1).read())
dataPlay2 = json.loads(open(jPlay2).read())
dataPlay3 = json.loads(open(jPlay3).read())
dataPlay4 = json.loads(open(jPlay4).read())
dataPlayLists = [dataPlay1, dataPlay2, dataPlay3, dataPlay4]
```

(c) importación de los datos de las playlists

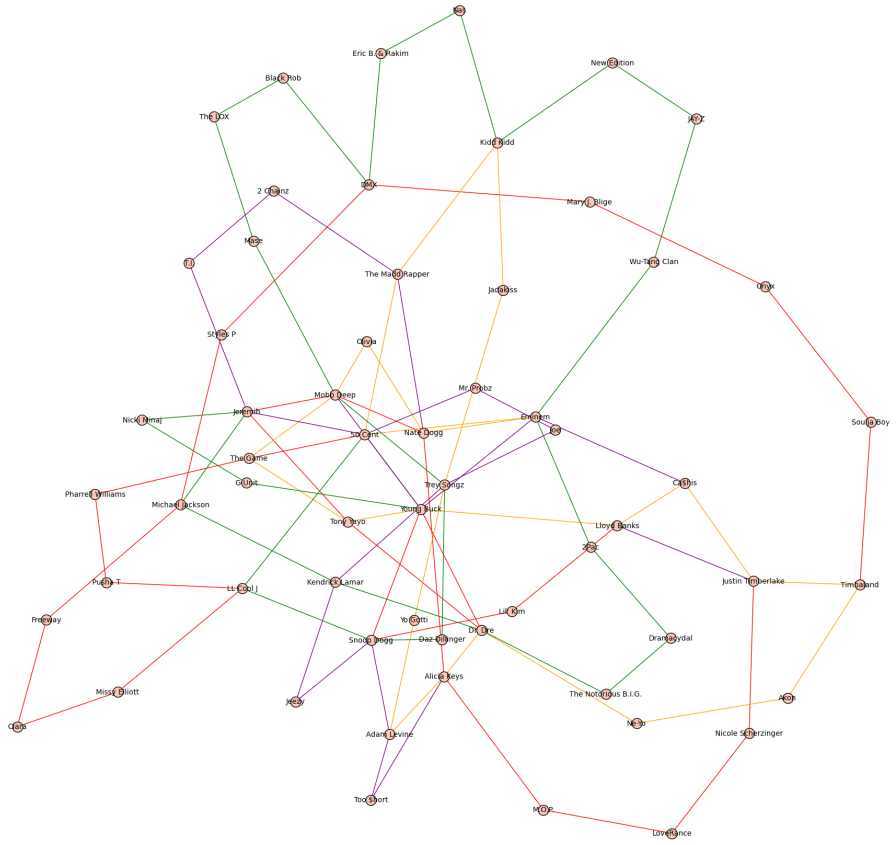
Figura 4: Código SageMath 1

```sage
def anadeVertices(G, a): #añade vertices al grafo en funcion de la lista de reproduccion
    G.add_vertices(a)
    return G
def anadeArista(G, a, z): #añade aristas al grafo en funcion de los artistas
    for i in range(0,len(a)):
        if(i==len(a)-1):
            G.add_edge(a[-1],a[0],z)
        else:
            G.add_edge(a[i], a[i+1], z)
    return G
```

(a) metódos auxiliares para crear el grafo

```sage
G=Graph()
b = len(dataPlayLists)
for i in range(0,b):
    artistas = [] # todos los artistas de todas las listas
    a = dataPlayLists[i]
    z = len(a)
    for j in range(0, z):
        c = a[j]
        artistas.append(c['2'])

    l = [val for sublist in artistas for val in sublist] #añade los artistas

    l = list(dict.fromkeys(l))
    G=anadeVertices(G,l)
    G=anadeArista(G,l,i)


d={0: "red", 1: "purple", 2:'orange', 3:'green'}

G.plot(edge_colors=G._color_by_label(d), edge_style='solid').show(figsize=25 ,fontsize=20)    #artistas que aparecen en
una playlist
```

(b) creación del grafo



(c) grafo

Figura 5: Código SageMath 2

Para crear el grafo hemos recorrido cada Playlist y enlazado aquellos artistas que aparecen en dicha playlist, el resultado es un grafo en donde cada color representa una playlist y cada vértice un artista.

De los artistas que participan en las playlists se han obtenido los mas populares por cada playlists, del mas popular se puede ver que Eminen es el ganador. Figura 6.

Se pueden observar las comunidades segun los generos por artista, de las playlist anteriores podemos enconrar que los generos raiz son el hiphop y el rap y podemos encontrar generos derivados como el nashville hip hop o el electropop. Figura 6.

La centralidad de intermediacion nos permite detectar aquellos nodos que hacen de puente para otros nodos de tal forma que el camino se hace mas corto. En este grafo tras aplicar la centralidad de intermediacion y coger los valores mas altos podemos observar que los mas importantes son Young Buck y 50 cent, curioso ya que 50 cent esta en todas las playlists. Figura 7.

Practicamente coincide con el centro del grafo(cercania) Figura 7.

```
for i in range(0,len(dataArtists)):

    a = dataArtists[i]
    popular=dict()
    popularidadTotal=list()
    for j in range(0,len(dataArtists[i])):
        c=a[j]
        popular.update({c['0']:c['2']})


    p={k: v for k, v in sorted(popular.items(), key=lambda item: item[1])} #sort dict
    print(p)
    for key, value in p.items():
        s=key+":"+str(value)
        popularidadTotal.append(s)

    print(popularidadTotal[20:len(popularidadTotal)])
    print('-----------')
```

(a) Popularidad total según playlist

```
{'LoveRance': 41, 'Tony Yayo': 52, 'Freeway': 52, 'CaShis': 54, 'Onyx': 55, 'M.O.P.': 57, 'Young Buck': 63, 'Nicole S
cherzinger': 63, 'Lloyd Banks': 64, 'Styles P': 64, 'LL Cool J': 65, "Lil' Kim": 68, 'Mobb Deep': 69, 'Soulja Boy': 6
9, 'DMX': 72, 'Pusha T': 72, 'Ciara': 73, 'Mary J. Blige': 74, 'Missy Elliott': 74, 'The Game': 75, 'Nate Dogg': 78,
'Timbaland': 78, 'Dr. Dre': 80, 'Jeremih': 80, 'Pharrell Williams': 80, '2Pac': 81, 'Alicia Keys': 82, 'Justin Timber
lake': 82, '50 Cent': 84, 'Michael Jackson': 84, 'Snoop Dogg': 85, 'Eminem': 94}
['Nate Dogg:78', 'Timbaland:78', 'Dr. Dre:80', 'Jeremih:80', 'Pharrell Williams:80', '2Pac:81', 'Alicia Keys:82', 'Ju
stin Timberlake:82', '50 Cent:84', 'Michael Jackson:84', 'Snoop Dogg:85', 'Eminem:94']
----------
{'Joe': 31, 'The Madd Rapper': 32, 'CaShis': 54, 'Young Buck': 63, 'Lloyd Banks': 64, 'Olivia': 65, 'Too $hort': 67,
'Mr. Probz': 68, 'Mobb Deep': 69, 'Adam Levine': 73, 'Jeezy': 75, 'Trey Songz': 77, 'T.I.': 78, 'Nate Dogg': 78, 'Tim
baland': 78, 'Jeremih': 80, 'Ne-Yo': 80, 'Dr. Dre': 80, 'Akon': 81, 'Justin Timberlake': 82, 'Alicia Keys': 82, '2 Ch
ainz': 83, '50 Cent': 84, 'Snoop Dogg': 85, 'Kendrick Lamar': 87, 'Eminem': 94}
['Alicia Keys:82', '2 Chainz:83', '50 Cent:84', 'Snoop Dogg:85', 'Kendrick Lamar:87', 'Eminem:94']
----------
{'The Madd Rapper': 32, 'Tony Yayo': 52, 'CaShis': 54, 'Kidd Kidd': 55, 'Young Buck': 63, 'Lloyd Banks': 64, 'Olivia
': 65, 'Jadakiss': 67, 'Mobb Deep': 69, 'Adam Levine': 73, 'The Game': 75, 'Yo Gotti': 75, 'Trey Songz': 77, 'Nate Do
gg': 78, 'Timbaland': 78, 'Ne-Yo': 80, 'Dr. Dre': 80, 'Akon': 81, 'Justin Timberlake': 82, 'Alicia Keys': 82, '50 Cen
t': 84, 'Eminem': 94}
['50 Cent:84', 'Eminem:94']
----------
{'Dramacydal': 49, 'Black Rob': 54, 'Eric B. & Rakim': 55, 'Kidd Kidd': 55, 'The LOX': 57, 'New Edition': 59, 'Daz Di
llinger': 62, 'G-Unit': 63, 'Young Buck': 63, 'LL Cool J': 65, 'Maze': 67, 'Mobb Deep': 69, 'Wu-Tang Clan': 69, 'DMX
': 72, 'Nas': 75, 'Trey Songz': 77, 'The Notorious B.I.G.': 79, 'Dr. Dre': 80, 'Jeremih': 80, '2Pac': 81, '50 Cent':
84, 'Michael Jackson': 84, 'Snoop Dogg': 85, 'JAY-Z': 85, 'Kendrick Lamar': 87, 'Nicki Minaj': 89, 'Eminem': 94}
['50 Cent:84', 'Michael Jackson:84', 'Snoop Dogg:85', 'JAY-Z:85', 'Kendrick Lamar:87', 'Nicki Minaj:89', 'Eminem:94']
----------
```

(b) Salida popularidad total según playlist

```
for i in range(0,len(dataArtists)):
    generos=[]
    a = dataArtists[i]
    genres=dict()
    for j in range(0,len(dataArtists[i])):
        c=a[j]
        generos.append(c['1'])

    g = [val for sublist in generos for val in sublist] #añade los generos de esta playlist

    #cuenta generos que aparecen
    for v in range(0,len(g)):
        if g[v] in genres:
            genres[g[v]]=genres[g[v]]+1
        else:
            genres[g[v]]=1

    genres={k: v for k, v in sorted(genres.items(), key=lambda item: item[1])} #sort dict
    print(genres,"\n")
    print("El genero mas popular es",list(genres)[-1])
    print('-------------------------------')
```

(c) Género mas popular según playlist

```
{'trap queen': 1, 'chicago rap': 1, 'electropop': 1, 'europop': 1, 'atl hip hop': 1, 'soul': 1, 'alternative hip hop
': 1, 'philly rap': 1, 'new jack swing': 1, 'old school hip hop': 1, 'underground hip hop': 1, 'crunk': 2, 'nashville
hip hop': 2, 'neo soul': 2, 'pop dance': 2, 'post-teen pop': 2, 'battle rap': 2, 'virginia hip hop': 2, 'dirty south
rap': 3, 'detroit hip hop': 6, 'g funk': 6, 'west coast rap': 6, 'hip pop': 6, 'r&b': 7, 'urban contemporary': 8, 'po
p': 8, 'dance pop': 9, 'hardcore hip hop': 10, 'trap': 12, 'southern hip hop': 13, 'east coast hip hop': 40, 'queens
hip hop': 41, 'pop rap': 54, 'gangster rap': 55, 'rap': 64, 'hip hop': 67}

El genero  es hip hop
-------------------------------
{'nashville hip hop': 1, 'hip pop': 1, 'neo soul': 1, 'cali rap': 1, 'hyphy': 1, 'oakland hip hop': 1, 'pop rock': 1,
'conscious hip hop': 1, 'edm': 1, 'tropical house': 1, 'chicago rap': 2, 'crunk': 2, 'pop dance': 2, 'atl hip hop':
3, 'dirty south rap': 3, 'hardcore hip hop': 3, 'detroit hip hop': 3, 'r&b': 5, 'g funk': 5, 'urban contemporary': 6,
'pop': 7, 'southern hip hop': 7, 'trap': 7, 'west coast rap': 7, 'dance pop': 9, 'east coast hip hop': 44, 'queens hi
p hop': 45, 'gangster rap': 55, 'pop rap': 55, 'rap': 59, 'hip hop': 62}

El genero  es hip hop
-------------------------------
{'pop dance': 1, 'neo soul': 1, 'pop rock': 1, 'memphis hip hop': 1, 'tennessee hip hop': 1, 'battle rap': 1, 'new or
leans rap': 1, 'g funk': 2, 'west coast rap': 2, 'crunk': 2, 'nashville hip hop': 2, 'hip pop': 2, 'dirty south rap':
3, 'r&b': 3, 'hardcore hip hop': 4, 'southern hip hop': 4, 'urban contemporary': 4, 'detroit hip hop': 5, 'dance pop
': 5, 'pop': 5, 'trap': 7, 'east coast hip hop': 37, 'queens hip hop': 39, 'gangster rap': 45, 'pop rap': 45, 'rap':
50, 'hip hop': 52}

El genero  es hip hop
-------------------------------
{'crunk': 1, 'dirty south rap': 1, 'nashville hip hop': 1, 'west coast trap': 1, 'alternative hip hop': 1, 'bboy': 1,
'electro': 1, 'turntablism': 1, 'new orleans rap': 1, 'boy band': 1, 'funk': 1, 'quiet storm': 1, 'detroit hip hop':
1, 'soul': 1, 'chicago rap': 1, 'pop dance': 1, 'post-teen pop': 1, 'new jack swing': 2, 'old school hip hop': 2, 'ha
rlem hip hop': 2, 'conscious hip hop': 2, 'trap': 5, 'g funk': 5, 'dance pop': 5, 'pop': 5, 'urban contemporary': 6,
'west coast rap': 6, 'r&b': 6, 'hip pop': 7, 'southern hip hop': 8, 'queens hip hop': 9, 'hardcore hip hop': 12, 'eas
t coast hip hop': 14, 'pop rap': 16, 'gangster rap': 20, 'rap': 24, 'hip hop': 26}

El genero  es hip hop
-------------------------------
```

(d) Salida Género mas popular según playlist

Figura 6: Código SageMath 3

```
x=G.centrality_degree()
#print(x)
sorted_x = sorted(x.items(), key=operator.itemgetter(1))
print(list(sorted_x)[50:-1])
```

```
[('The Madd Rapper', 1/15), ('Nate Dogg', 1/12), ('Alicia Keys', 1/12), ('Snoop Dogg', 1/10), ('Trey Songz', 1/10),
('Eminem', 1/10), ('Dr. Dre', 1/10), ('Jeremih', 1/10), ('50 Cent', 7/60), ('Mobb Deep', 7/60)]
```

(a) Centralidad de grado

```
x=G.centrality_betweenness()
sorted_x = sorted(x.items(), key=operator.itemgetter(1))
print(list(sorted_x)[55:-1])
```

```
[('Michael Jackson', 0.12827145547484528), ('Eminem', 0.13137386781454582), ('Dr. Dre', 0.14458277284548465), ('Mobb
Deep', 0.1491168953457089), ('50 Cent', 0.17672450901264455)]
```

(b) Centralidad de intermediación

```
x=G.centrality_closeness()
sorted_x = sorted(x.items(), key=operator.itemgetter(1))
print(list(sorted_x)[55:-1])
```

```
[('Jeremih', 0.3409090909090909), ('Dr. Dre', 0.3468208092485549), ('Eminem', 0.3488372093023256), ('50 Cent', 0.3592
814371257485), ('Mobb Deep', 0.3592814371257485)]
```

(c) Centralidad de cercanía

Figura 7: Código SageMath 4

Lo sorpredente de esta centralidad de grado es que a pesar de que 50cent es nuestro autor de estudio Mobb Deep tiene la misma centralidad de grado. Figura 7.

```
def similitud(G,u,v):
    #distancia euclidea
    AM = G.adjacency_matrix()[G.vertices().index(u)] - G.adjacency_matrix()[G.vertices().index(v)]
    #excluimos a u y v
    AM[G.vertices().index(u)] = 0
    AM[G.vertices().index(v)] = 0
    return AM.norm()
```

(a) Similitud

```
def similitudentrecomunidades(G,C1,C2):
    r=0
    for i in C1:
        for j in C2:
            if similitud(G,i,j) > r:
                r = similitud(G,i,j)
    return r
```

(b) Similitud entre comunidades

```
def unircomunidades(G,C):
    similitud_referencia = 100000000
    union = [0,0]
    res=C
    for i in range(len(C)-1):
        for j in range(i+1,len(C)):
            if similitudentrecomunidades(G,C[j],C[i]) < similitud_referencia:
                union = [i,j]
                similitud_referencia = similitudentrecomunidades(G,C[j],C[i])

    res.append(C[union[0]]+C[union[1]])
    res.remove(C[union[1]])
    res.remove(C[union[0]])

    return res
```

(c) Unir comunidades

```
def comunidades_modularidad(G):
    vector=[]
    for v in G.vertices():
        vector.append([v])
    dendograma=[]
    vector_final=[]
    modularidad_referencia=-999
    while(len(vector) >1):
        modularidad = modularidat(G,vector)
        if modularidad > modularidad_referencia:
            vector_final = vector[:]
            modularidad_referencia = modularidad
        vector = unircomunidades(G, vector)
        dendograma.append([vector[:], modularidad])
    modularidad=modularidat(G,vector)
    dendograma.append([vector[:], modularidad])
    if modularidad > modularidad_referencia:
        vector_final=vector[:]
        modularidad_referencia=modularidad
    return modularidad_referencia,dendograma, vector_final
```

(d) Comunidades aplicando modularidad

Figura 8: Código SageMath 5

Ahora veremos el estudio de comunidades para ello hemos utilizado los metódos realizados en la práctica de comunidades de la asignatura. Figura 8.

```
mod,dend,vec=comunidades_modularidad(G)
print("-----Modularidad: ----")
print(mod)
print("-----Dendograma: ----")
print(dend)
print("-----Vector: ----")
print(vec)
```

(a) Salida comunidades

(b) Vector de comunidades

Figura 9: Código SageMath 6

La modularidad nos permite enconrar aquellos nodos que tienen una conexion muy fuerte con sus vecinos, y el algoritmo Hierarchical clustering nos permite encontrar esas 8 comunidades.Figura 9.

4. Estudio de comunidades

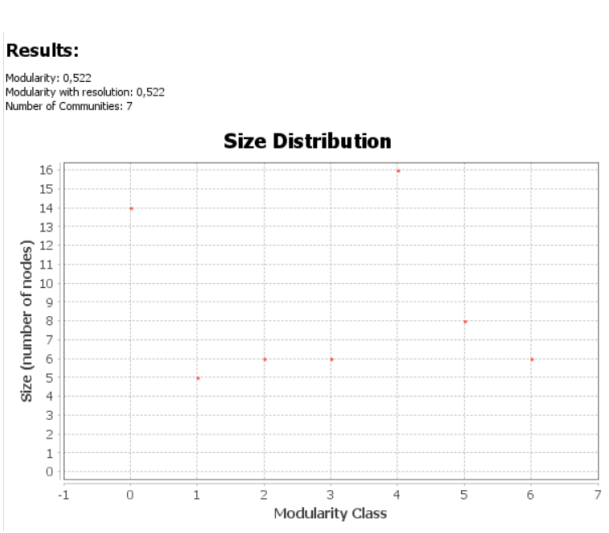
Para el estudio de comunidades utilizaremos la herramienta Gephi. Para ello es necesario tener los vertices y las aristas en formato CSV. Esto se ha hecho a mano con un poco de paciencia, por ende es posible que haya alguna errata, aunque lo ideal es hacer algun tipo de algoritmo que dado un grafo de sage a partir de las listas de vertices y aristas, haga la conversion a CSV y luego eso lo importaríamos en Gephi.

Una vez obtenidos los arhivos CSV el siguiente paso es importarlo en la herramienta, para ello seguiremos los siguientes pasos:

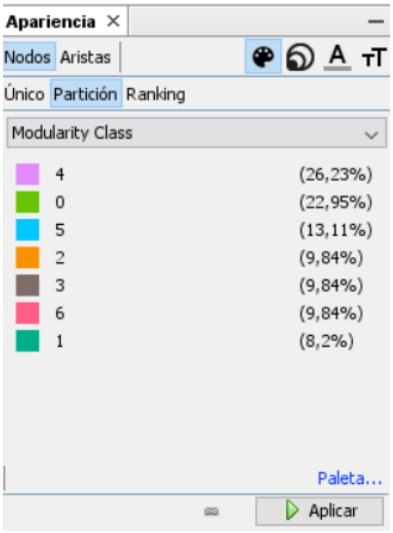
1. **Archivo, importar hoja de cálculo.**
2. **Seleccionamos el archivo CSV que contiene los vértices.**
3. **Le damos siguiente, terminar y añadir a espacio de trabajo existente.**
4. **Repetimos el proceso pero esta vez con el archivo de aristas**
5. **Le damos siguiente a todo y no olvidemos al acabar de añadir las aristas al espacio de trabajo existente.**

A continuación en el panel de estadísticas ejecutaremos el algoritmo de Modularidad que nos dará las estadísticas. Podemos observar que se han detectado 7 comunidades, que para representarlas en el grafo y poder verlas visualmente tenemos que configurar los siguientes elementos: [Figura 10].

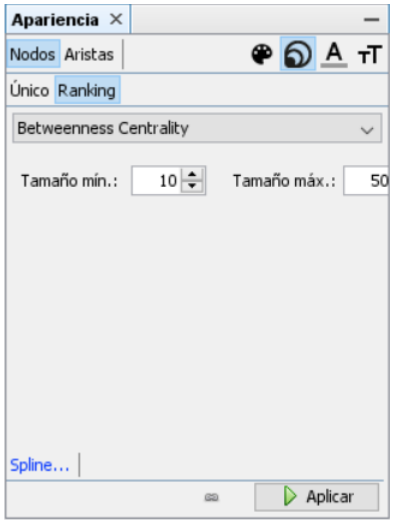
1. **Apariencia(icono de la paleta de colores), nodos, particion y seleccionamos Modularity Class. De esta forma los nodos se colorearan del color correspondiente a su comunidad.**
2. **Apariencia(icono del tamaño), ranking, Betweenness Centrality. Para que el tamaño de los nodos represente la centralidad de cercanía..**
3. **En distribución, seleccionamos Force Atlas y ajustamos la fuerza de repulsión en 100000.0 y marcamos ajustar por tamaños.**
4. **Obtenemos este grafo.**



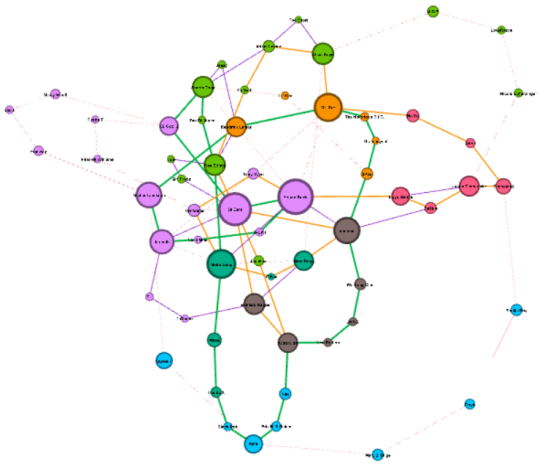
(a) Resultado de la modularidad



(b) Apariencia color



(c) Apariencia tamaño



(d) Grafo por comunidades

Figura 10: Modularidad en Gephi

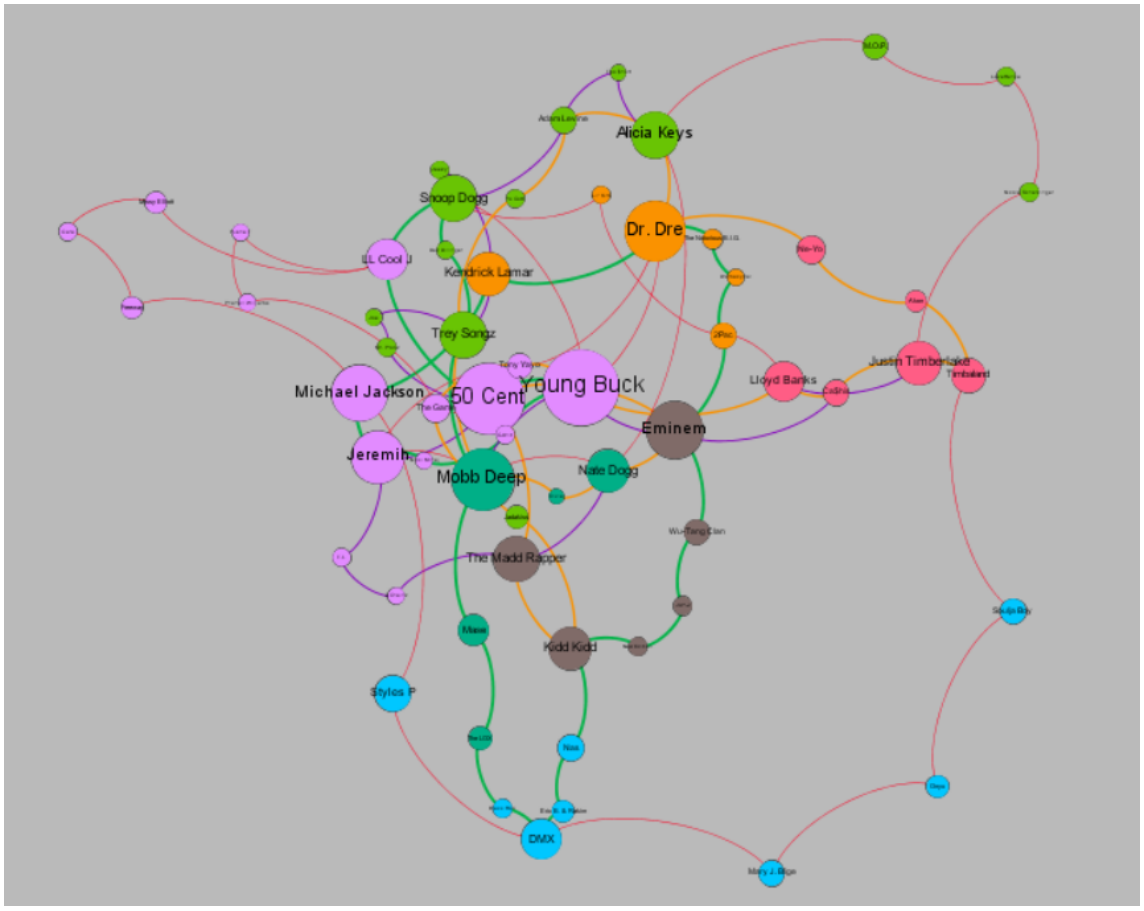


Figura 11: Gephi grafo versión grande

Si utilizamos la pestaña de previsualización obtendremos la Figura 11 y podremos ver las 7 comunidades un poco más de cerca. Dónde los nodos más grandes representan una centralidad de cercania mayor, que a su vez coinciden con el ranking de popularidad de cada artista.

5. Conclusiones

A diferencia de otras redes como Twitter o Facebook, Spotify es un tanto más difícil de estudiar ya que no es tan susceptible de que un ojo novato cree con poco esfuerzo un boceto de como funcionan las relaciones entre artistas, no como en Facebook que con ir diciendo premisas como "Fulanito es amigo de Pepito, así que puedo unirlos con una arista, y entonces...", o "Naranjito le dio retweet a Manolito, así que puedo relacionarlos con una arista, y entonces...", obtener una red es asequible. Spotify te puede ofrecer en cambio listas de reproducción con ciertos artistas, álbumes de esos artistas e incluso podcasts, lo que hace que el ojo inexperto tenga problemas para buscar una forma de estudiar la red.

Sin embargo, al estudiar a un artista en concreto como 50 Cent a partir de sus listas de reproducción, hemos obtenidos resultados interesantes, como que por ejemplo él no sea el artista más famoso de su grupo, sino Eminem, o que por ejemplo un artista Mobb Deep sea un artista de paso entre todas las listas de reproducción. Todos estos resultados fueron gratificantes y cuanto menos informativos del comportamiento de Spotify.