UNIVERSITAT DE BARCELONA

Trabajo de fin de grado

GRADO DE INGENIERÍA INFORMÁTICA

Facultad de Matemáticas e Informática

Universitat de Barcelona

Toda la música suena igual

Autor: Cristina Reina León

Director: **Dr. Jordi Nin**
Realizado en: **Departamento de Matemáticas e Informática**

Barcelona, **27 de junio de 2019**

Resumen

En una era donde internet es una herramienta fundamental en la difusión de información, las tendencias musicales pueden verse afectadas por la viralización de canciones y la aparición de artistas emergentes. Esto puede hacer que, géneros musicales antes minoritarios e incluso, en ocasiones, asociados a comunidades marginales, puedan volverse rápidamente populares y quitarse este estigma.

Muchas de estas tendencias, que consiguen surgir gracias a internet, a veces tapan el resto de géneros musicales y dan una falsa sensación de que «toda la música suena igual». Ponemos la radio, oímos la televisión o vamos a discotecas, y la música es siempre la misma. ¿Pero significa eso que solo haya un único estilo musical?

En este estudio vamos a analizar un poco el panorama musical a partir de la creación de una red, un grafo, que nos permitirá ver los artistas escuchados actualmente en la red social *Last.fm*[1] y los géneros que predominan. Nos basaremos, por lo tanto, en los datos que pone a disposición de los usuarios Last.fm para realizar este proyecto.

También realizaremos una adaptación del algoritmo de difusión cultural de Axelrod[2], planteado en 1997, con el objetivo de predecir la evolución musical de la red y las nuevas tendencias que vendrán en el futuro.

Veremos finalmente que, por los motivos comentados al inicio, no podemos saber a ciencia cierta cuál será el futuro de la industria musical, ya que hay factores exógenos a los datos y al algoritmo, que modelaremos de forma estocástica en nuestros experimentos.

Resum

En una era on internet és una eina fonamental a la difusió d'informació, les tendències musicals poden veure's afectades per la viralització de cançons i la aparició d'artistes emergents. Això pot fer que, gèneres musicals abans minoritaris i fins i tot, en ocasions, associats a comunitats marginals, puguin tornar-se ràpidament populars i treure's aquest estigma.

Moltes d'aquestes tendències, que consegueixen sorgir gràcies a internet, a vegades tapen la resta de gèneres musicals i dónen una falsa sensació de que «tota la música sona igual». Posem la ràdio, escoltem la televisió o anem a la discoteca, i la música sempre es la mateixa. Pero, significa això que només hi hagi un únic estil musical?

En aquest estudi analitzarem una mica el panorama musical a partir de la creació d'una xarxa, un graf, que ens permetrà veure quins són els artistes

més escoltats actualment a la xarxa social *Last.fm*[1] i els gèneres que predominen. Ens basarem, per tant, en les dades que posa a disposició dels usuaris Last.fm per a realitzar aquest projecte.

També realitzarem una adaptació de l'algorisme de difusió cultural d'Axelrod[2], plantejat a 1997, amb l'objectiu de predir l'evolució musical de la xarxa i les noves tendències que vindràn en el futur.

Veurem finalment que, pels motius comentats a l'inici, no podem saber amb certesa quin serà el futur de la indústria musical, ja que hi ha factors exògens a les dades i a l'algorisme, que modelarem de forma estocàstica en els nostres experiments.

Abstract

In an age where internet is a fundamental tool in the dissemination of information, musical trends can be affected by the viralization of songs and the appearance of emerging artists. This can mean that musical genres that were previously minority and even, occasionally, associated with marginal communities, can quickly become popular and remove this stigma.

Many of these tendencies, which manage to emerge thanks to the Internet, sometimes cover the rest of the musical genres and give a false sense that«all music sounds the same». We listen the radio, we watch the television or we go to discos, and the music is always the same. But does that mean that there is only one musical style?

In this study we are going to analyze a bit the musical panorama from the creation of a network, a graph, that will allow us to see which are the artists currently heard in the *Last.fm*[1] social network and the genres that predominate. We will base, therefore, on the data Last.fm make available to users to carry out this project.

We have also made an adaptation of the cultural diffusion algorithm of Axelrod [2], proposed in 1997, with the aim of predicting the musical evolution of the network and the new trends that will come in the future.

Finally we will see that, for the reasons discussed at the beginning, we can not know for sure what the future of the music industry will be, since there are factors exogenous to the data and the algorithm, which we will model stochastically in our experiments.

Agradecimientos

Quiero agradecer en primer lugar a Jordi Nin, por toda la ayuda que me ha brindado a lo largo de la realización de este proyecto, pero sobre todo por haberme guiado y haber confiado en mí. Estoy segura que sin su apoyo el trabajo habría sido un desastre.

También quiero agradecer a mi madre, por haber creído en mí todos estos años y por su paciencia conmigo en los momentos de estrés, por cuidarme. Y cómo no, a mi preciosa gata, que me ayuda a relajarme y añade a mis días momentos de risas, ternura y amor.

A Sergio Plans, que se convirtió en una persona muy importante para mí desde casi el primer día de universidad. Creo que sin él no habría podido podido acabar tan rápido la carrera. Por darme apoyo moral y ayudarme en mi pelea con las gráficas (y otras muchas cosas).

Y por último, a mis amigos Dani, Sergio, Lorena, Mar y Mata. Por haberme aguantado todos estos años y haberme acompañado en muchos momentos de mi vida. Por haberme dado, estos últimos meses, esos ratos de desconexión y desahogo que tanto necesitaba. Por acompañarme en las fiestas. Pero sobre todo, por su amistad.

Índice

1. Introducción **1**

2. Estado del arte **3**

2.1. Modelo de Axelrod 4

3. Objetivos **6**

4. Planificación **8**

5. Costes **10**

6. Obtención de los datos **11**

6.1. Obtención de los datos 11

6.2. Limpieza de Datos y Data Enrichment 12

7. Creación del grafo **18**

7.1. Creación del grafo usando una matriz de adyacencia 18

7.2. Carga del grafo en python 19

8. Análisis topológico del grafo **21**

8.1. Estadísticas numéricas 21

8.2. Heatmap 24

8.3. Ley Potencial (Power Law Distribution) 29

9. Adaptación del algoritmo de Axelrod **32**

9.1. Modificaciones introducidas 33

10 Pruebas y resultados **37**

10.1 Aplicación del algoritmo 37

10.2 Homogeneidad musical 37

10.3 Fortaleza de los géneros musicales 41

10.4 Resultados 44

11 Conclusiones y trabajo futuro **45**

1. Introducción

Cada vez hay una mayor variedad musical, pero en ocasiones nos puede dar la sensación de que toda la música suena igual. La música es cada vez más "comercial" – dicen – ya no es *como antes*. Pero, ¿realmente es así o es la percepción que nos llega por escuchar la radio, ir a discotecas o, incluso, a tiendas de ropa?

En este proyecto vamos a tratar de ver si realmente la industria musical cada vez es más comercial. Intentaremos ver si los artistas tienden a evolucionar a los géneros musicales que son más populares o si, por el contrario, el panorama musical es cada vez más diverso y apto para todos los gustos.

El campo de la **data science** es muy útil para solucionar este tipo de problemas. Por lo que vamos a usar varios métodos y procesos de la ciencia de los datos para intentar hallar una respuesta a las preguntas planteadas. Nuestro propósito va a ser crear un **grafo musical** que nos permita ver el estado actual y futuro de los géneros musicales, a partir del estudio de las relaciones e interacciones entre los artistas. El objetivo principal será ver cómo los géneros musicales asociados a los artistas se van propagando por la red, volviéndose con el paso del tiempo más fuertes o desapareciendo por completo. Para ello nos basaremos en las ideas del científico político *Robert Axelrod* y adaptaremos su algoritmo de diseminación cultural.

Pero a pesar de que pueda parecer que solo la creación del grafo y la adaptación del algoritmo son las partes fundamentales del proyecto, todas las fases por las que deberemos pasar para poder llegar a estas, y las que nos permitirán visualizar los resultados, son imprescindibles para el desarrollo y análisis del proyecto.

Las fases del proyecto, que van a ser explicadas en detalle en las siguientes páginas de este documento, son las siguientes:

Lo primero que haremos será obtener los datos. No podemos crear una red musical si no tenemos datos que nos permitan hacerlo. Para ello vamos a usar los datos recogidos en el archivo *hetrec2011-lastfm-2k* [3] que recopiló el grupo IRG de la Universidad Autónoma de Madrid. Este conjunto de datos contiene información recopilada de la red social musical **Last.fm**[1] [1] de 1892 usuarios y 17632 artistas.

Después deberemos procesar y limpiar estos datos. Para esta segunda tarea nos ayudaremos de la API [2][5] que Last.fm pone a disposición de los usuarios para recopilar información de su base de datos.

[1] **Last.fm** es una red social musical, donde los usuarios pueden registrar su actividad musical e interactuar con otros usuarios. También es una radio vía internet e incluye un sistema de recomendación de música basándose en los datos enviados por los usuarios registrados.[4]

[2] La API de Last.fm permite a cualquiera construir sus propios programas usando los datos de Last.fm.

Con los datos ya listos pasaremos a la creación de la red musical, que no es otra cosa que un grafo que implementaremos usando la librería de Python [6] **NetworkX** [7]. Deberemos crear una matriz de adyacencia, donde cada celda represente la relación entre cada par de artistas. En nuestro caso tomaremos esta relación como el número de oyentes comunes de ambos artistas.

Es muy importante también poder visualizar el grafo de manera que podamos analizar los aspectos relevantes de la red. Por ello crearemos inicialmente un par de visualizaciones para observar el estado de la red y, más adelante, veremos que necesitaremos una visualización más para poder comparar el antes y el después de simular la evolución musical.

El siguiente paso será la adaptación del algoritmo de diseminación cultural de Axelrod y su aplicación en el grafo. Con ya todo el proceso sobre nuestra red realizado, procederemos a analizar los resultados obtenidos y buscaremos respuesta a las preguntas planteadas.

Por último visualizaremos de varias maneras los resultados y los analizaremos para sacar conclusiones. Veremos que los resultados obtenidos no nos permiten determinar cuál es el futuro de la música, pero si podemos sacar algunas conclusiones.

Veremos, por lo tanto, el proceso entero que implica realizar una tarea de data science, desde el inicio hasta el final. Desde la búsqueda y obtención de los datos, su limpieza y enriquecimiento, pasando por la creación de la matriz de adyacencia, el grafo, hasta la implementación y aplicación del algoritmo.

No podemos saber con certeza cuáles serán las próximas tendencias musicales, ya que hay un componente que no podemos calcular y es el azar. En una era donde las redes sociales son un pilar muy importante en la difusión de información, la aparición y viralización de canciones en plataformas como *Youtube* puede cambiar por completo el trayecto de la música. Podemos verlo actualmente con la reciente popularización de la música latina, como el *reggaeton* y el *trap*, donde artistas emergentes como Bad Bunny[8] o C. Tangana [9] han conseguido poner de moda géneros musicales antes minoritarios. Lo que sí que sabemos es que las tendencias musicales siempre han existido y van cambiando y evolucionando, pero esto no significa que la diversidad musical deba morir.

El resto de esta documento está organizado de la siguiente forma. Primero, la Sección 2 introducirá las bases de las que parte nuestro proyecto: qué son las redes y su relación con el estudio sobre diseminación cultural de Axelrod. Después, en la Sección 3 detallaremos cuáles son los objetivos principales del proyecto. En la Sección 4 encontramos la planificación del proyecto dividida en tareas y sus posibles costes en la Sección 5. Las siguientes Secciones 6-10 corresponden a las tareas del propio proyecto, ya descritas anteriormente. Por último la Sección 11 describirá las principales conclusiones de este trabajo así como posibles tareas futuras.

2. Estado del arte

Una red es, en términos sencillos, una colección de puntos que están unidos entre ellos, en pares, mediante una línea. A los puntos les llamamos “nodos” y a las líneas “aristas”.

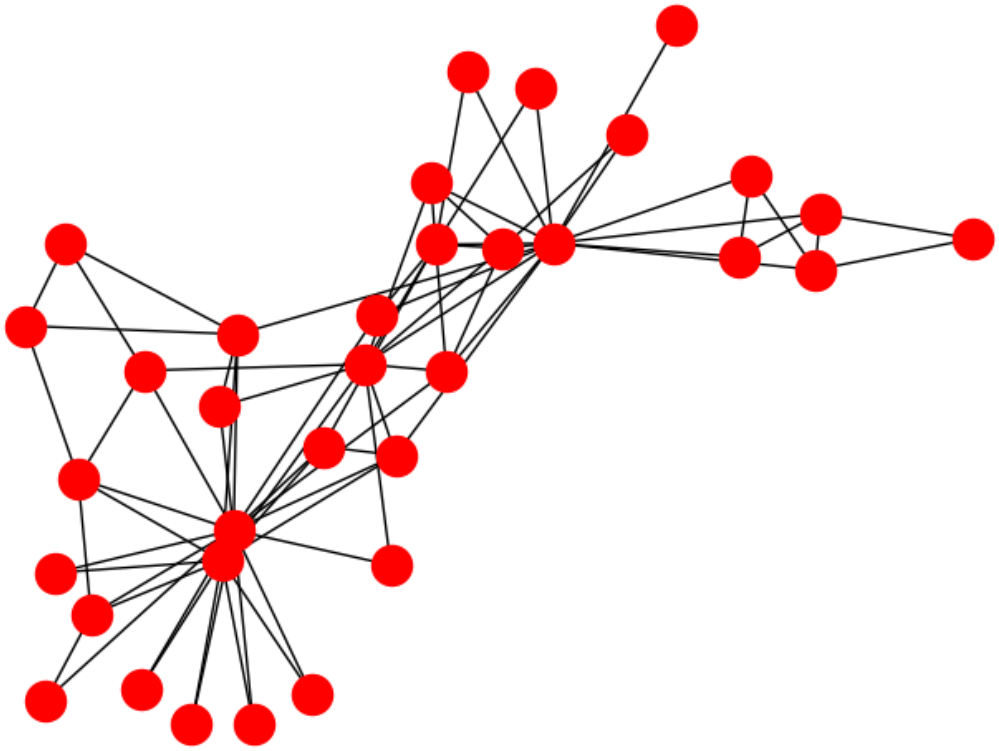


Figura 1: Ejemplo grafo

Existen varios tipos de redes o grafos. Aquellos cuyas aristas indican en qué dirección se puede ir de un nodo a otro se llaman *dirigidos*, mientras que los que no lo indican y, por lo tanto, todas las aristas se pueden cruzar en ambas direcciones, se llaman *no dirigidos*. También los podemos clasificar según si las aristas tienen un valor asociado, siendo *pesados* si lo tienen y *no pesados* en caso contrario.

Hay muchos sistemas que pueden verse como redes en muchas disciplinas distintas: física, biología, ciencias sociales... Y el hecho de representarlas como redes nos permite estudiar las relaciones e interacciones entre los miembros del sistema, en lugar de centrarnos únicamente en los objetos que lo forman, permitiéndonos resolver cuestiones nuevas y útiles [10].

Esto es lo que se entiende como **redes complejas**. En las redes complejas el grado promedio no es un dato identificativo de la red, ya que el propósito de estas redes es representar las conexiones e interacciones de los nodos.

Se necesita mucha más información para ser entendida, ya que lo importante es ver cómo están realizadas estas conexiones, entre quiénes, qué grupos de nodos tienen mayor densidad y cuáles menos... Son redes, por lo tanto, que poseen ciertas propiedades estadísticas y topológicas que no ocurren en las redes simples.

En 1997, el científico político Robert Axelrod construyó un modelo de diseminación cultural haciendo uso de las redes para representar una comunidad de individuos y así poder estudiar sus interacciones. Publicó un famoso artículo que comenzaba con la siguiente pregunta:

«If people tend to become more alike in their beliefs, attitudes and behavior when they interact, why do not all differences eventually disappear?»[2]

Es decir: *Si la gente tiende a irse pareciendo cada vez más al resto en sus creencias, actitudes y comportamiento cuando interactúan, ¿por qué todas estas diferencias no acaban desapareciendo?* Este es un problema que ha estado muy presente a lo largo del tiempo en la rama de las ciencias sociales.

Para intentar resolverlo, Axelrod construyó un modelo de diseminación cultural[3], el cual explica en el mismo artículo, con el cual provee una explicación de por qué la tendencia a converger para antes de completarse. [11]

2.1. Modelo de Axelrod

El modelo construido se basa en dos suposiciones simples [12]:

1. Es más probable que las personas interactúen con otras personas con las que comparten muchos de sus atributos culturales.
2. Estas interacciones tienden a incrementar el número de atributos que comparten, haciendo aumentar a su vez la posibilidad de que vuelvan a interactuar.

En este modelo encontramos una matriz *L*L*, donde cada elemento representa un individuo que pertenece a una cierta cultura. Es, por lo tanto, una matriz de adyacencia donde se representan las uniones entre los nodos, que corresponden a cada individuo. La cultura de los individuos viene representada por una lista *f* de características. Cada una de estas características puede tener un conjunto *v* de valores.

[3]El término *cultura* se usa para indicar el conjunto de atributos individuales que están sujetos a la influencia social.

El modelo ejecuta los siguientes pasos:

1. Un agente k activo es seleccionado aleatoriamente.
2. Un agente *r* pasivo es seleccionado aleatoriamente de entre los vecinos de *k*.
3. Los agentes k y r interactúan con probabilidad igual a su similitud cultural n_{kr}/f, donde n_{kr} es el número de características culturales donde k y r tienen el mismo valor.
4. Cuando la probabilidad se cumple, el agente k cambia aleatoriamente una de las características culturales de r cuyos valores diferían.

El proceso continua hasta que no se pueden realizar más cambios.

3. Objetivos

En este proyecto queremos construir un modelo que nos permita visualizar y trabajar con una red musical. Partiremos de las ideas de Axelrod y del algoritmo que propuso en 1997, explicado en el apartado anterior, para simular la diseminación y la evolución de los géneros musicales.

Vamos a ver cómo los artistas musicales están conectados e interactúan entre ellos, y como los géneros musicales que los representan pueden ganar o perder fuerza en función de estas interacciones, así como evolucionar.

El proyecto consta principalmente de tres partes.

Objetivo 1. Recopilación y tratamiento de los datos

El primer paso y, probablemente, el más importante es encontrar un buen conjunto de datos que nos permita crear el grafo. Necesitamos, por una parte, la información relativa a un gran número de artistas de diversos géneros musicales. Por otra parte necesitamos datos que nos permitan establecer relaciones entre los artistas, para crear los ejes de nuestro grafo. Estos datos pueden ser, por ejemplo, el número de usuarios que escuchan o dan una valoración similar a dos artistas.

Además deberemos limpiar los datos para deshacernos de información incorrecta o ruido.

Objetivo 2. Creación del grafo

Con los datos ya listos pasamos a crear el grafo. Lo haremos en el lenguaje de programación *Python* mediante el uso de la librería *NetworkX* [7]. El código lo vamos a estructurar en notebooks[4], ya que es más cómodo para el tipo de tareas que vamos a realizar.

Este grafo representará los artistas musicales y sus conexiones. Cada artista tendrá asociados un máximo de tres géneros musicales, que lo representarán. Para realizar las conexiones entre cada par de artistas usaremos el número de usuarios que escuchan a ambos.

Antes de crear el grafo como tal, usando la librería *NetworkX*, crearemos la matriz de adyacencia correspondiente con una matriz de *NumPy*[13] y la guardaremos en un fichero de texto, de manera que solo tengamos que calcularla una vez.

[4]Jupyter Notebook es un entorno de trabajo interactivo que permite desarrollar código en Python en un fichero con interfaz. Estos documentos incluyen entradas y salidas de código, texto explicativo y otras representaciones multimedia.

También implementaremos varias funciones para visualizar de manera interesante y útil el grafo.

Objetivo 3. Implementación del algoritmo y resultados

Por último aplicaremos un algoritmo de diseminación culturar para ver cómo los géneros musicales van esparciéndose por la red, muriendo o volviéndose más fuertes. También analizaremos por qué, aún estando toda la red conectada, y siendo lógico pensar que todos los artistas deberían acabar confluyendo a los mismos géneros, los resultados finales reflejan lo contrario.

4. Planificación

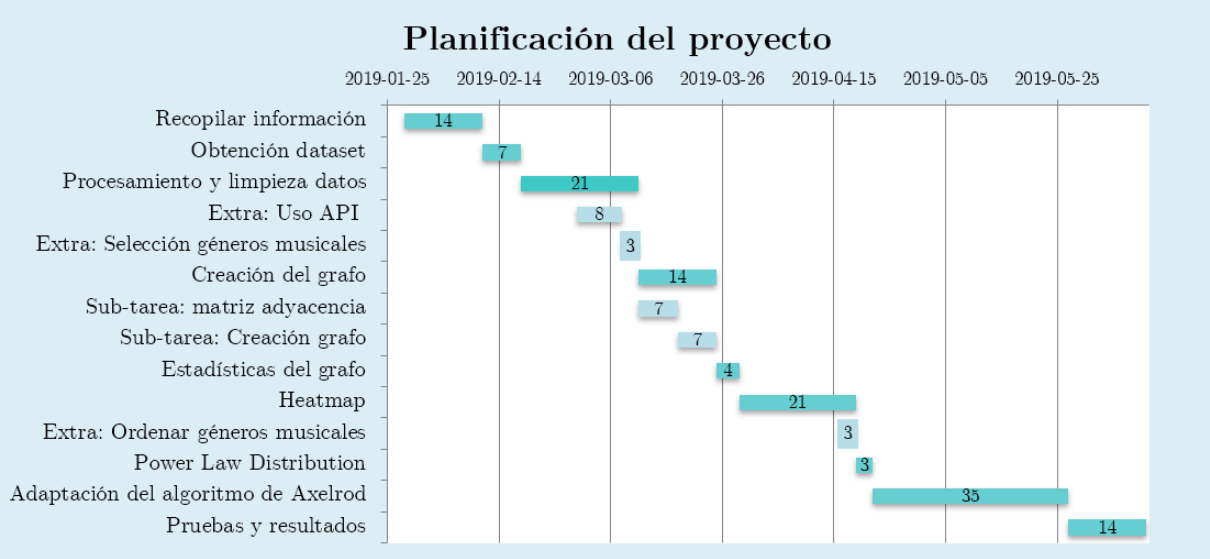


Figura 2: Diagrama de Gantt

Las diferentes tareas de la planificación del proyecto son las siguientes:

1. **Recopilación de información sobre redes y grafos (2 semanas)**
   Principalmente lectura del libro sobre redes *Networks: An Introduction* de M.E.J. Newman.

2. **Obtención del conjunto de datos para la creación de la red (1 semana)**
   Buscar un set de datos que cumpla con los requisitos necesarios para la creación de nuestra red musical.

3. **Procesamiento y limpieza de los datos (2 semanas ⇒ 3 semanas)**
   Debemos procesar y limpiar los datos, de manera que eliminemos artistas incorrectos, datos poco relevantes, ruido, así como elegir un conjunto limitado de géneros musicales y asignar los correspondientes a cada artista.

   - **Extra:** Recopilar datos mediante el uso de la API, para ambas tareas, llevó más tiempo de lo esperado.
   - **Extra:** Realizar la selección manual de los géneros musicales válidos (213 géneros), y el diccionario para reemplazar los géneros mal escritos, también llevo más tiempo del planeado, unos cuatro días en total aproximadamente.

4. **Creación del grafo (2 semanas)** Se divide principalmente en dos tareas.

- **Crear la matriz de adyacencia de los artistas musicales(1 semana)** Aunque es un proceso sencillo, llevó tiempo ya que quisimos implementarlo de la forma más óptima posible.
- **Crear el grafo (1 semana)** usando la librería NetworkX.

5. **Estadísticas del grafo (2-3 días)** Las estadísticas numéricas no llevaron mucho tiempo, son tareas que se consiguen con funciones simples de la librería Networkx de Python.

6. **Heatmap (1 semana $\Rightarrow$ 3 semanas)** La visualización del mapa de calor de los géneros musicales de la red en principio parecía una tarea sencilla, pero finalmente nos llevó bastante. Debimos hacer varias pruebas y distintos planteamientos para que el mapa mostrase algo interesante.

   - **Extra:** Para visualizar correctamente el mapa los géneros musicales deben estar ordenados siguiendo un criterio musical. La ordenación manual de los 213 géneros musicales, según el género principal al que pertenecen y sus orígenes musicales, llevó aproximadamente tres días.

7. **Power Law Distribution (2-3 días)** Crear una visualización del número de apariciones de cada género musical en el grafo.

8. **Adaptación del algoritmo de Axelrod (1 semana $\Rightarrow$ 5 semanas)** El algoritmo de diseminación cultural en principio iba a ser una tarea bastante sencilla de realizar, pero finalmente llevó bastante más tiempo del que esperabamos. Los resultados que daba no parecían correctos y debimos solucionar varios errores en nuestro planteamiento.

9. **Pruebas y resultados (1 semana $\Rightarrow$ 2 semanas)** Para los resultados debimos implementar una nueva visualización que nos permitiera ver los cambios de posición de los géneros musicales, antes y después de la aplicación del algoritmo, ya que las visualizaciones creadas previamente no nos permitían hacerlo de forma rápida. Esto hizo que tardásemos un poco más de lo esperado. Además la ejecución del algoritmo es algo lenta y quisimos hacer varias pruebas.

5. Costes

En este proyecto hemos identificado los siguientes costes:

Mano de obra. En el proyecto sólo han intervenido dos personas, el realizador del proyecto y el supervisor.

Personal	**€/h**	**Horas**	**Total**
Trabajador técnico especializado	60 €/h	270 h	16200 €
Supervisor del proyecto	120 €/h	18 h	2160 €
Coste Total			**18360 €**

Software. Debido a que el programario usado es en su mayoría libre y/o de código abierto, el único coste que encontramos es el correspondiente a *Microsoft PowerPoint*, usado para la realización de la presentación.

Programa	**Precio**	**Cantidad**	**Total**
Ubuntu 16.04 LTS	0 €	1	0 €
Jupyter Notebook	0 €	1	0 €
Overleaf (LaTeX)	0 €	1	0 €
Microsoft PowerPoint	69 €/año	1 año	69 €
Coste Total			**69 €**

Hardware. Para la realización de este proyecto se ha usado únicamente un ordenador portátil. A pesar de ello, para poder realizar un mayor número de ejecuciones y de forma más rápida, habría sido interesante disponer de varios ordenadores con más potencia.

Hardware	**Precio**	**Cantidad**	**Total**
HP Notebook 250 G6	555 €	1	0 €
Coste Total			**555 €**

Coste Total.

Gasto	**Coste**
Mano de obra	18360 €
Software	69 €
Hardware	555 €
Coste Total	**18984 €**

6. Obtención de los datos

En esta sección veremos como hemos obtenido los datos y como han sido procesados antes de generar nuestra red musical.

6.1. Obtención de los datos

Antes de nada, para poder llevar a cabo este proyecto, debimos buscar un set de datos que cumpliera ciertas características.

Los datos tenían que permitir identificar los artistas y debían incluir algún tipo de información que permitiera crear una relación entre artistas. Esta información podría ser, por ejemplo:

- El número de reproducciones (de los artistas) por parte de los usuarios.
- La valoración (de los artistas) por parte de los usuarios.

No fué una tarea sencilla ya que muchos de estos sets de datos, a pesar de ser actuales y con una gran variedad musical, usaban únicamente identificadores para referirse a los artistas musicales, lo que imposibilitaba trabajar con los datos y saber qué significaban los resultados.

Por otra parte, también era importante poder saber a qué géneros musicales pertenecía cada artista

Finalmente encontramos un set de datos que, aunque algo desactualizado (2011), nos brinda información útil para la realización del proyecto.

hetrec2011-lastfm-2k

Tal y como podemos leer en el *README* del archivo .zip *hetrec2011-lastfm-2k*[3], este set de datos contiene información recopilada de la red social musical Last.fm de 1892 usuarios y 17632 artistas.[5]

Los archivos incluidos en este fichero son:

- **artists.dat:** Es un listado de artistas que incluye el id del artista, el nombre, la url a su ficha en Last.fm y la url a una imagen suya.
- **tags.dat:** Listado que relaciona los tags[6] y sus id's.

[5]Este conjunto de datos fué publicado en 2011 por el grupo Information Retrieval Group de la Universidad Autónoma de Madrid junto con otros conjuntos parecidos.

[6]Un tag es una etiqueta que se le asigna a un artista. Puede ser un género musical, nacionalidad o cualquier otra característica que se relacione con el artista.

- **user_artists.dat:** Contiene un listado de las reproducciones realizadas por cada usuario. Esto se representa con tres valores: el id del usuario, el id del artista y el número de reproducciones.
- **user_taggedartists.dat:** Contiene la asignación de tags a los artistas por parte de cada usuario.
- **user_taggedartists_timestamps.dat:** Completa el fichero anterior mostrando cuándo se ha realizado la asignación de cada tag.
- **user_friends.dat:** Muestra las relaciones de amistad entre usuarios.

De estos archivos finalmente solo usamos dos: *artists.dat* y *user_artists.dat*. La información relativa a las etiquetas de los artistas nos resulta de mucha utilidad, pero no la proporcionada en los ficheros ya que, al ser asignadas por los usuarios, son demasiado variadas y poco fiables.
Es por esto que, contando con que Last.fm pone a disposición de los usuarios una API[7] para recopilar información de la red, decidimos ignorar este archivo y obtener las etiquetas mediante el uso de ésta.

Además, el uso de la API también nos permite obtener otra información que nos será útil para la limpieza de datos, tal y como veremos en el siguiente apartado.

6.2. Limpieza de Datos y Data Enrichment

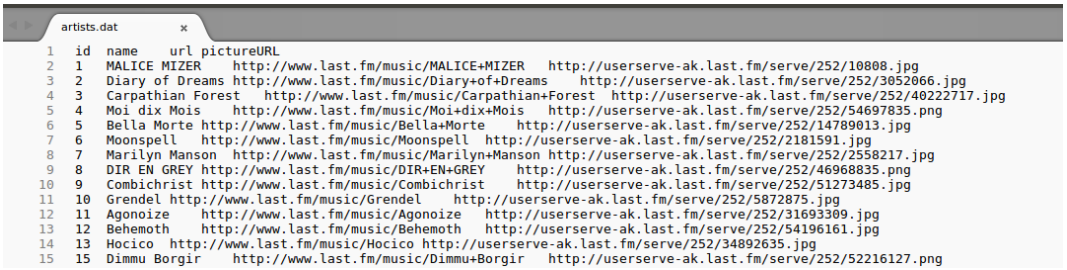
En este apartado vamos a atajar principalmente dos puntos:

- Realizar una **limpieza de artistas**, usando el número de usuarios del que disponemos información y el número de oyentes. Este último lo recopilaremos usando la API de Last.FM.
- Por otra parte, **asignaremos** a cada artista **tags** o etiquetas, que corresponderán principalmente a géneros musicales. Para ello usaremos también la API de Last.FM y realizaremos una selección de tags de manera manual y una posterior limpieza en los datos, de manera que se use un número limitado de etiquetas y además con la seguridad de que son válidas y útiles.

Igual que para el resto del proyecto usaremos Python para implementar las funciones que nos permiten conectarnos a la API.

En este apartado, como lo único que necesitamos es el nombre de los artistas, usaremos principalmente el archivo *artists.dat* (Figura 3), que es el que contiene el nombre de los artistas de los cuales tenemos información.

[7] Una API (Application Programming Interface) es una interfaz que permite realizar funciones de un determinado software.



artists.dat

```
id  name    url pictureURL
1   MALICE MIZER    http://www.last.fm/music/MALICE+MIZER   http://userserve-ak.last.fm/serve/252/10808.jpg
2   Diary of Dreams http://www.last.fm/music/Diary+of+Dreams    http://userserve-ak.last.fm/serve/252/3052066.jpg
3   Carpathian Forest   http://www.last.fm/music/Carpathian+Forest  http://userserve-ak.last.fm/serve/252/40222717.jpg
4   Moi dix Mois    http://www.last.fm/music/Moi+dix+Mois   http://userserve-ak.last.fm/serve/252/54697835.png
5   Bella Morte http://www.last.fm/music/Bella+Morte    http://userserve-ak.last.fm/serve/252/14789013.jpg
6   Moonspell   http://www.last.fm/music/Moonspell  http://userserve-ak.last.fm/serve/252/2181591.jpg
7   Marilyn Manson  http://www.last.fm/music/Marilyn+Manson http://userserve-ak.last.fm/serve/252/2558217.jpg
8   DIR EN GREY http://www.last.fm/music/DIR+EN+GREY    http://userserve-ak.last.fm/serve/252/46968835.png
9   Combichrist http://www.last.fm/music/Combichrist    http://userserve-ak.last.fm/serve/252/51273485.jpg
10  Grendel http://www.last.fm/music/Grendel    http://userserve-ak.last.fm/serve/252/5872875.jpg
11  Agonoize    http://www.last.fm/music/Agonoize   http://userserve-ak.last.fm/serve/252/31693309.jpg
12  Behemoth    http://www.last.fm/music/Behemoth   http://userserve-ak.last.fm/serve/252/54196161.jpg
13  Hocico  http://www.last.fm/music/Hocico http://userserve-ak.last.fm/serve/252/34892635.jpg
15  Dimmu Borgir    http://www.last.fm/music/Dimmu+Borgir   http://userserve-ak.last.fm/serve/252/52216127.png
```

Figura 3: Archivo artists.dat

6.2.1. API de Last.FM

Para poder obtener el número de oyentes de los artistas y sus etiquetas más relevantes creamos un par de funciones en Python que nos permiten conectarnos a su API. Además, es necesario obtener una clave para realizar las peticiones, pero se puede conseguir fácilmente entrando en el enlace https://www.last.fm/api.

artist_getInfo()

Hemos creado la función **artist_getInfo()** para obtener el número de oyentes de cada artista.
Esta función, que realiza una llamada a la función *artist.getInfo* de la API, devuelve información relativa al artista pasado por parámetro. En nuestro caso, como únicamente nos interesa el número de oyentes, simplemente ignoraremos el resto de información.
Con esta función también podemos ver qué artistas no se encuentran en la base de datos de Last.fm. Esto lo podemos saber porque el artista no se encuentra en el cuerpo de la respuesta. Cuando esto sucede simplemente asignamos al artista el número de oyentes “-1”.

Para obtener los oyentes de cada artista simplemente ejecutamos la función creada para cada uno de los artistas de nuestro conjunto de datos. Este proceso puede llevar unas cuantas horas y varias ejecuciones, ya que en ocasiones falla la conexión.

artist_getTopTags()

Hemos creado la función **artist_getTopTags()** para obtener los tags más frecuentes de cada artista.
Esta función realiza una llamada a la función *artist.getTopTags()* de la API que devuelve un número indeterminado de tags más frecuentes del artista. Nosotros hemos optado por poner como parámetro en nuestra función el número

de tags máximo que queremos obtener, fijándolo en 10 para este proyecto.

Del mismo modo que para obtener el número de oyentes, para obtener los tags debemos ejecutar la función para cada artista de nuestro conjunto de datos.

6.2.2. Purga de artistas

Realizar una limpieza de artistas es muy importante, ya que queremos que la información de nuestra red sea lo más fiable posible. Es por eso que debemos tener en cuenta varios aspectos importantes:

1. Que no haya **artistas repetidos** (mismo artista: distinto formato de string, mayúsculas, minúsculas, símbolos o abreviaturas).
2. Que no haya **artistas incorrectos** como fechas, títulos de vídeos de youtube, colaboraciones o símbolos inválidos.
3. Eliminar grupos de los cuales contamos con **poca información**.

Para evitar tener en nuestros datos este tipo de artistas vamos a jugar con dos parámetros asociados a cada artista:

- **frequency** es el número de veces que aparece el artista en nuestros datos, es decir, el número de usuarios de los cuales tenemos información que escuchen ese artista.

  Para obtener la frecuencia de cada artista hemos tenido que contabilizar el número de veces que aparece cada artista en el fichero *user_artists.dat*.

- **listeners** es el número de oyentes del artista en la red social Last.FM.

  Como ya hemos explicado en el subapartado anterior, el número de oyentes lo conseguimos gracias a conectarnos a la API de Last.FM con la función implementada.

Después de probar varias combinaciones y analizar un poco los datos, creemos que lo más sensato es eliminar los artistas de los que no disponemos datos de al menos 3 usuarios y de los que cuentan con menos de 5000 oyentes.

¿Por qué eliminar los artistas que tienen menos de 3 usuarios en nuestros datos? Porque cuando tengamos que establecer las relaciones entre artistas usaremos el número de usuarios en común. En el caso de los artistas con 1 solo usuario, tendrán 100 % de usuarios en común con todo el resto de artistas de ese usuario. En el caso de los artistas con 2 usuarios, tendremos exactamente el mismo problema si ponemos el umbral de unión a

menos del 50 %. Por lo tanto, creemos que tener en cuenta esos artistas puede ensuciar los datos y llevar a resultados equívocos.

¿Por qué eliminar los artistas con un número de oyentes bajo ($<$ 5000**)**? Porque es una manera sencilla de asegurarnos de que no hay artistas con nombres incorrectos: suelen tener entre 0 y 100 oyentes (o -1 en el caso de los artistas no encontrados). No queremos ruido en nuestros datos.

De esta manera pasamos de tener los inicialmente 17632 artistas a tener 4562, aproximadamente una cuarta parte.

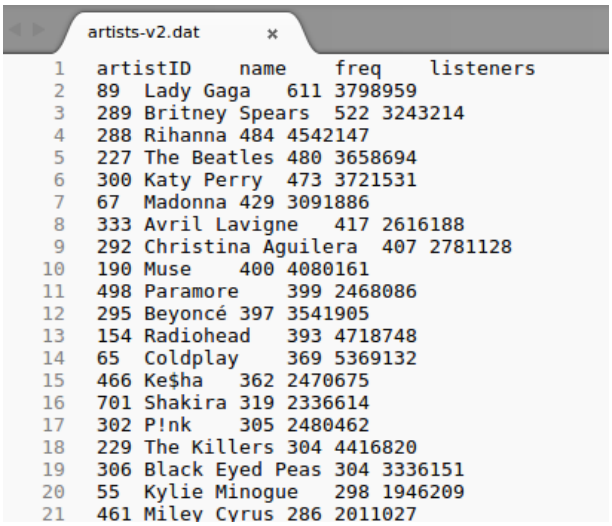


Figura 4: Archivo artists-v2.dat

Después de la limpieza realizada podemos generar un nuevo fichero **artists-v2.dat** (Figura 4) que reemplaza el anterior fichero *artists.dat*, que contiene únicamente los 4562 artistas que cumplen las características requeridas. Además, este fichero ahora incluye la nueva información relevante obtenida, la frecuencia y el número de oyentes, y prescinde de la que no nos interesa. Para guardarlos de manera rápida en un único fichero de texto hemos hecho un join del dataframe que contiene las frecuencias y el que contiene los oyentes, obteniendo un dataframe con los campos nombrados.

6.2.3. Asignación de tags

Como ya hemos dicho antes, en el set de datos usado encontramos el archivo *user_taggedartists.dat* que contiene los tags que cada usuario a asignado a los artistas y, como hemos explicado, esto es una información poco fiable y demasiado variada. El número de tags asignado a un mismo artista es muy amplio y los datos que tenemos de cada artista a veces son escasos. Es por eso que, gracias a que Last.fm pone a disposición de los usuarios una API para obtener datos de su red, vamos a obtener los tags de los artistas mediante

el uso de esta, en lugar de usar el archivo proporcionado en el set de datos. Recordado esto, vamos a proceder a la obtención de los tags.

Antes de nada, cargamos en un dataframe el nuevo archivo *artists-v2.dat*. Hecho esto, lo primero que vamos a hacer es obtener los 10 tags más frecuentes de cada artista. Para ello usaremos la función explicada anteriormente, que nos permite recopilar los tags más frecuentes de cada artista.

Posteriormente deberemos hacer una selección de tags, por lo que nos será de utilidad saber cuál es su número de apariciones. Por este motivo hemos decidido hacer otro archivo complementario, a medida que realizamos el de artistas y etiquetas, que contiene únicamente las etiquetas en forma de lista.

Una vez ya tenemos los dos nuevos archivos, **artist_tags.dat** y **tags.dat**, los cargamos para trabajar con ellos.

Ahora vamos a crear un nuevo fichero **tags_freq.dat** que contendrá el listado de tags junto con su frecuencia. Para ello contamos el número de veces que aparece cada tag en el fichero *tags.dat*. Al haber guardado los tags en una única columna y nada más, podemos hacer esto de manera muy sencilla.

Con los tags ya contabilizados podemos escoger con más criterio con qué tags quedarnos. Los vamos a ordenar de manera descendente y, manualmente, vamos a descartar aquellos que no son géneros musicales aunque, excepcionalmente, vamos a guardar etiquetas que puedan ser interesantes considerar (e.g. disney, instrumental) y vamos a renombrar/corregir las incorrectas o con más de una representación.

Crearemos, por lo tanto, dos nuevos ficheros. Por un lado están los tags a eliminar **tags_to_delete.dat**, que son tags poco útiles o incorrectos. La mayoría corresponden a nacionalidades u otras características del artista no relevantes.
Por otro lado están los tags a sustituir **tags_to_replace.dat**, que son tags incorrectos pero que pueden substituirse por un tag correcto, como géneros repetidos escritos de distinta forma (e.g. hip hop en lugar de hip-hop) o géneros que se han acompañado con una nacionalidad, sin tener distinción alguna del género original (rock brasileño o rock ruso en lugar de simplemente rock).

Inicialmente escogimos los 420 tags más frecuentes, de los que eliminamos 126 y reemplazamos 86, quedándonos finalmente con únicamente 213 tags de los 3943 que contenían nuestros datos. Estos tags, que consideramos como correctos, los guardamos en el fichero **tags-v3.dat**. También tenemos el fichero *tags-v2.dat* contiene los tags correctos pero también los tags a reemplazar

Limpieza de tags

Recordemos que hemos asociado a cada artista un máximo de 10 tags, y hemos seleccionado los ≈ 200 tags más relevantes, pero nuestros datos aún contienen las etiquetas descartadas. Nuestro objetivo es por tanto crear un nuevo fichero **artist_tags-v2.dat** que guarde los artistas con únicamente los tags que hemos considerado correctos.

Para ello vamos a iterar sobre las filas del dataframe creado a partir del fichero *artist_tags.dat* y vamos a generar una lista de listas. Cada lista será una fila del nuevo fichero, que contendrá el ID del artista, el nombre y los tags correctos. Para quedarnos con los tags correctos simplemente miramos cada uno de los tags asociados al artista y:

1. Si se encuentra entre los tags correctos lo guardamos en la lista.
2. Si se encuentra entre los tags a reemplazar miramos a qué tag correcto corresponde y lo guardamos igualmente en la lista.
3. En caso contrario lo ignoramos.

Una vez ya tenemos la lista de listas con los datos tratados podemos guardarlos en el nuevo fichero de manera muy sencilla usando la función *writer()* de la librería *csv*.

7. Creación del grafo

En este apartado nos vamos a centrar en la creación del grafo y la visualización de la red de artistas.

7.1. Creación del grafo usando una matriz de adyacencia

El primer paso para crear nuestro grafo es decidir cómo vamos a establecer la relación entre los artistas. Para esta tarea contamos principalmente con dos archivos: **artists-v2.dat** y **user_artists.dat**. Recordemos que estos ficheros nos proporcionan información relativa a cada arista (ID, nombre, frecuencia y oyentes) y las reproducciones de los artistas por parte de cada usuario. Lo más sencillo para crear las relaciones es usar estas relaciones usuario-artista: podemos contar el número de usuarios comunes que tiene cada par de artista. Esto será lo que creará la unión entre los nodos y el peso de las aristas.

La manera más simple de crear el grafo es generando su matriz de adyacencia. En las filas y columnas encontraremos los artistas, y el valor de las celdas será el número de usuarios comunes.

Para saber el número de usuarios comunes debemos comparar las listas de usuarios que han escuchado los dos artistas y sacar los que comparten. Vamos a cargar en un dataframe el fichero **user_artists.dat** y lo vamos a filtrar para obtener los usuarios que han escuchado cada artista.

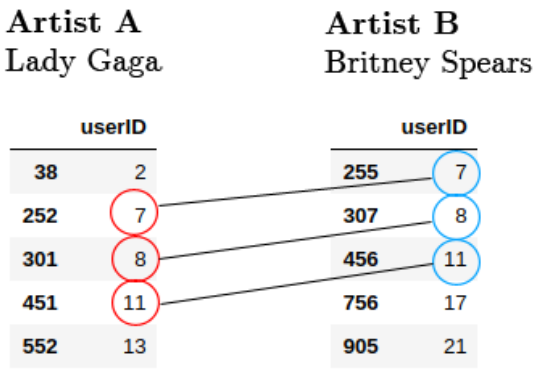


Figura 5: Parte de los dataframes de los artistas A y B

Ahora debemos encontrar las coincidencias entre ambos subdataframes, tal y como se muestra en la Figura 5. Podríamos hacerlo de manera sencilla usando joins, pero es un proceso lento.
Como tenemos los usuarios siempre ordenados de forma ascendente y no nos interesa guardar qué usuarios comparten, sinó únicamente el número de coincidencias, podemos implementar una función que nos solucione esto.

Hemos optado por usar la búsqueda binaria para encontrar qué usuarios del primer subdataframe se encuentran en el segundo subdataframe y sencillamente vamos sumando 1 a nuestro contador cada vez que un usuario de A se encuentra en B. Con esto conseguimos ahorrar mucho tiempo.

Repetimos la búsqueda para cada par de artistas de nuestros datos creando así la matriz de adyacencia.

7.2. Carga del grafo en python

Para crear el grafo vamos a usar la librería de Python[6] **NetworkX** [7]. Los grafos pueden representarse mediante dos listas:

- Una **lista de nodos**: una lista de 2-tuplas donde el primer elemento de cada tupla es la representación del nodo y el segundo elemento es un diccionario con los metadatos asociados al nodo. En nuestro caso la representación del nodo será el nombre del artista y los metadatos asociados serán, por una parte, los tres tags más representativos del artista, y por otra parte el número de usuarios que escuchan al artista, es decir, la frecuencia. Quedando por lo tanto de la siguiente manera:

```
{'lady gaga': {
    'attr_dict': {
        'genre1': 'pop',
        'genre2': 'dance',
        'genre3': 'electronic',
        'freq': 611
    }
}}
```

- Una **lista de aristas**: una lista 3-tuplas donde los dos primeros elementos son los nodos que están conectados y el tercer elemento es un diccionario con los metadatos asociados a la arista.

  En nuestro caso la representación de cada arista contará con el nombre de los dos nodos (artistas) que une y como metadatos únicamente tendremos el peso (número de usuarios comunes) de la arista. La lista queda de la siguiente manera:

```
    ('lady gaga', 'britney spears', {'weight': 436.0})
```

Teniendo claro cómo debemos crear el grafo y qué datos necesitamos, el proceso para hacerlo es muy sencillo.

Hemos implementado un pequeño algoritmo que:

1. Crea un grafo vacío.
2. Recorre las filas del dataframe de artistas para añadir los nodos al grafo.
3. Recorre la matriz de adyacencia para añadir las aristas a los nodos.

Con esto ya tenemos la función necesaria para crear el grafo. A esta función además le podemos pasar un valor umbral a partir del cual se añadirán las aristas (número mínimo de usuarios en común).

8. Análisis topológico del grafo

Una vez creado el grafo nos encontramos con que sus descriptores estadísticos no nos aportan toda la información que nos interesa. Podemos saber cosas útiles de la red como el número de nodos, la asortatividad o la distribución del grado, que nos muestran algunos datos relevantes sobre la estructura del grafo, pero nos falta otro tipo de información referente a sus metadatos, los géneros musicales.

Por eso vamos a buscar varias maneras de visualizar la red, de manera que podamos analizar otros aspectos importantes, como por ejemplo cómo están repartidos los géneros musicales, cuáles predominan o cómo los artistas interactúan unos con otros.

En este apartado vamos a ver, por lo tanto, algunas estadísticas interesantes del grafo pero también un par de representaciones que hemos creado especialmente para visualizar otros aspectos importantes de nuestra red.

8.1. Estadísticas numéricas

- **Número de nodos:** En nuestra red contamos con un total de 4562 nodos y, por lo tanto, artistas musicales.
- **Número de aristas:** El número de conexiones entre nodos es de 696740. El mínimo de usuarios que debe tener una arista para existir es 1.
- **Grado máximo:** Es el número de aristas que tiene el nodo con más aristas del grafo. En nuestro caso este puesto es para The Beatles, que cuenta con 3262 conexiones.
- **Grado promedio:** Es el número de ejes esperado en un nodo cualquiera. Depende de la densidad de la red y se calcula según la fórmula:

$$k = 2 * E/N$$

Que en nuestro caso:

$$k = 2 * 696740/4562 = 305{,}454$$

Por lo que el grado promedio de nuestra red es 305. Este valor por sí solo, pero, no nos da mucha información. No podemos saber cuál es la forma de la red y cómo están estructuradas las conexiones entre artistas, ya que puede ser que todos los nodos estén más o menos igual de conectados (grado parecido) o, por el contrario, que haya un grupo reducido de nodos altamente conectados y el resto estén más aislados.

- **Asortatividad (assortativity):** La asortatividad es la correlación entre dos pares de nodos. Esta correlación no es otra que el coeficiente de correlación de Pearson de los grados de los pares de nodos conectados de la red. Un valor positivo de r indica una correlación entre nodos de grado parecido, mientras que valores negativos indican correlación entre nodos de grado distinto. [14]

  En una red asortativa encontramos un corazón con nodos con gran densidad (elevado número de conexiones) y, cuanto más nos alejamos del corazón, más dispersos encontramos los nodos. En una red disasortativa, por el contrario, los nodos no se encuentran ordenados (ni pueden ser ordenados) según su grado.
  En nuestra red el coeficiente de asortatividad tiene el valor -0.167, que es ligeramente negativo. Esto nos indica que el grado de un nodo no es demasiado decisivo para que dos artistas estén relacionados.

- **Promedio de triángulos formados por un eje (transitivity):** Es una propiedad muy importante y útil en las redes sociales. Una relación x es transitiva si $a * x$ y $b * c$ juntas implican $a * c$. Si la relación *conectado por un eje* fuese transitiva, equivaldría a la famosa frase *los amigos de mis amigos son mis amigos*. [15]
  En nuestro grafo la transitividad es $0{,}28$, lo que indica que hay una tendencia a que la gente suele escuchar normalmente el mismo conjunto de artistas. Es decir, si yo escucho al artista A y al artista B, es probable que escuche también el artista C.

- **Coeficiente global de clustering:** El coeficiente de agrupamiento es una medida del grado en que los nodos de un gráfico tienden a agruparse. Se calcula como el ratio entre triángulos cerrados y triángulos totales, y es una buena medida para determinar como de cerradas o abiertas son las comunidades de nuestra red, o en otras palabras, como de interconectados están mis amigos. Obtendremos un valor grande cuando todos nuestros amigos se conocen entre ellos y pequeño en caso contrario.
  Concretamente, nuestra red tiene un coeficiente global de clustering igual a 0.52 que es un valor intermedio nos indica que en nuestra red tenemos ambos tipos de comunidades.

- **Diámetro:** El diámetro de la red es la longitud del camino geodésico más largo de la red, que es el camino más corto (existente) entre dos pares de nodos. En nuestro grafo su valor es 3, lo que significa que nuestros nodos están muy conectados, ya que el máximo número de aristas que debemos recorrer para llegar de un nodo cualquiera a otro nodo cualquiera es 3.

- **Distribución de grado:** Es una característica que define la estructura de la red. Calcula la probabilidad de que un nodo cualquiera tenga grado $k = 0, 1, 2...n$ (donde n es el número de nodos del grafo) como $P(k) = n_k/n$, siendo n_k el número de nodos del grafo que tienen grado k.[16]

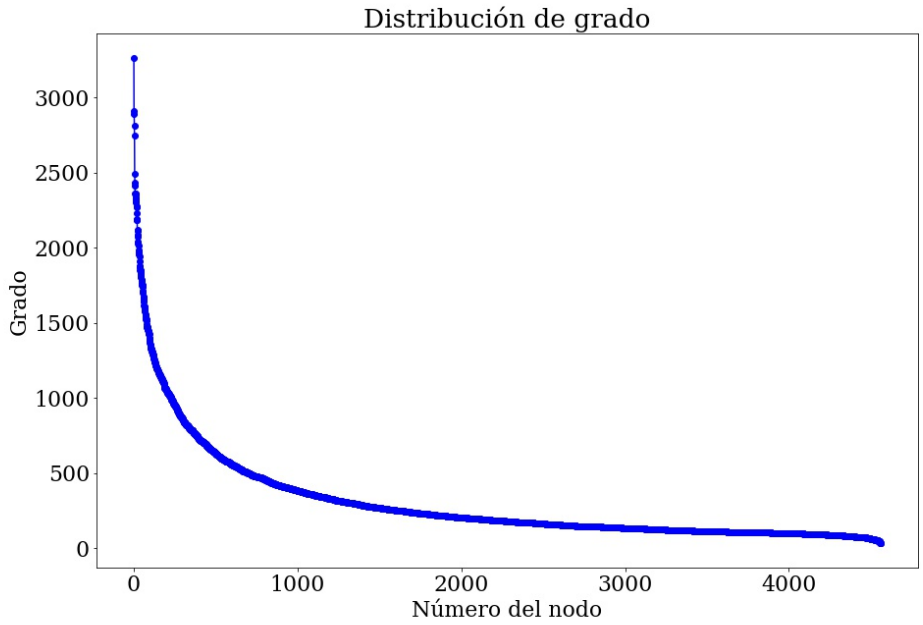


Figura 6: Distribución de grado

Como podemos observar, en nuestro grafo hay un pequeño porcentaje de nodos altamente conectados, con más de 500 aristas, y mientras que el resto se encuentra poco conectado, con entre 500 y 0 aristas. El grado de los nodos nos puede dar información como la capacidad de un artista musical de llegar a diferentes públicos.

- **Diagrama de fuerza:**

  El diagrama de fuerza es parecido a la distribución del grado, solo que en lugar de tener en cuenta únicamente el número de aristas conectadas a cada nodo, tiene en cuenta también su peso. Por lo que el diagrama muestra la suma de los pesos de las aristas de cada nodo. Podemos observar que en este caso la curva de la gráfica es mucho más acentuada que en la distribución del grado. Tenemos un número muy reducido de nodos cuyo peso está entre 5000 y 25000, otro pequeño grupo de nodos (alrededor de 700) con pesos entre 5000 y 200 y el resto de nodos (alrededor de 3800) con pesos muy bajos, entre 200 y 0. Esto sucede porque en los pesos de las aristas se reflejan factores como la compa-

tibilidad entre grupos musicales pero también, de manera indirecta, su popularidad.

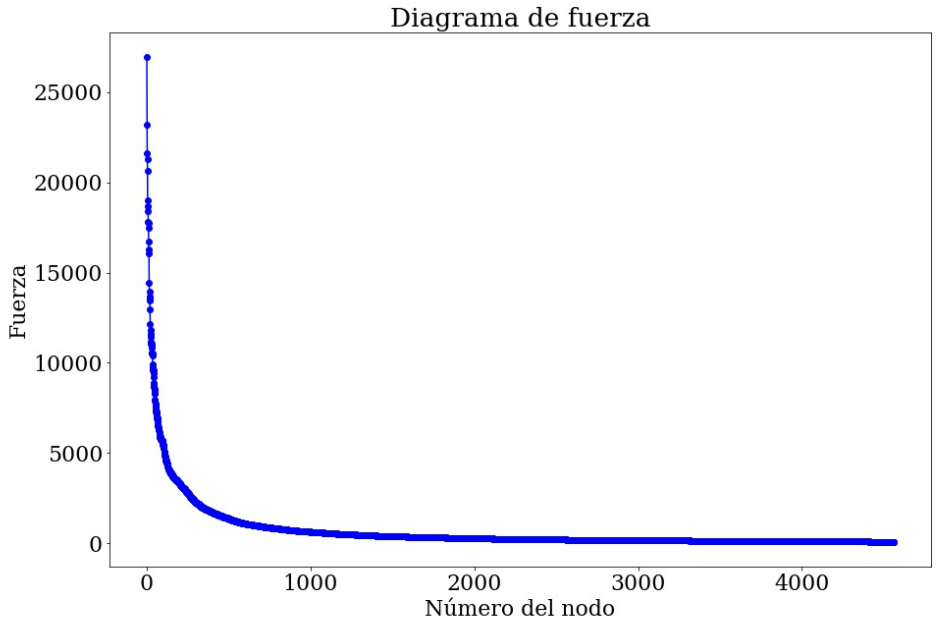


Figura 7: Diagrama de fuerza

8.2. Heatmap

La primera visualización que vamos a realizar es un mapa de calor o *Heatmap*. El heatmap es una representación gráfica de los datos donde los valores individuales, contenidos en una matriz, se representan como colores. [17]

En nuestro caso, el heatmap del grafo representaría a los usuarios comunes de cada par de artistas, pero el número de artistas es demasiado elevado y esta representación tampoco nos daría una información muy útil. Por eso, en lugar de representar en cada celda los usuarios comunes de pares de artistas, lo que vamos a hacer es agrupar los artistas según su género musical, de manera que lo que veamos representado sean las relaciones entre géneros. Deberemos crear, por lo tanto, una nueva matriz de adyacencia que muestre los oyentes comunes entre géneros musicales.

Lo primero que necesitamos es la matriz de adyacencia. Vamos a usar dos, la matriz de adyacencia original, con el número de usuarios comunes entre artistas, y una nueva matriz de adyacencia en porcentajes. Para calcular la matriz de porcentajes simplemente dividimos cada celda entre el número de oyentes que tiene el artista menos escuchado. Este paso extra lo realizamos con la función implementada *matrix_to_percentage(matrix, df)*.

Para crear la nueva matriz de adyacencia con los géneros musicales deberemos realizar los siguientes pasos:

Para cada género a:

1. Filtramos las filas de la matriz tomando el género a.
2. Sumamos el valor de las columnas. Obtenemos una lista con las sumas.
3. (Optativo) Normalizamos según el número de artistas de los géneros que se están contemplando.
4. Para cada género b:
    4.1. Filtramos la lista del paso 2 por columnas según b.
    4.2. Sumamos la lista.
    4.3. El valor obtenido es el número de oyentes comunes de los géneros a y b. Lo asignamos a las celdas $m[a, b]$ y $m[b, a]$ de la matriz

Una vez creada la matriz de adyacencia se la podemos pasar a la función *heatmap()* de la librería ***seaborn*** para crear el mapa de calor.

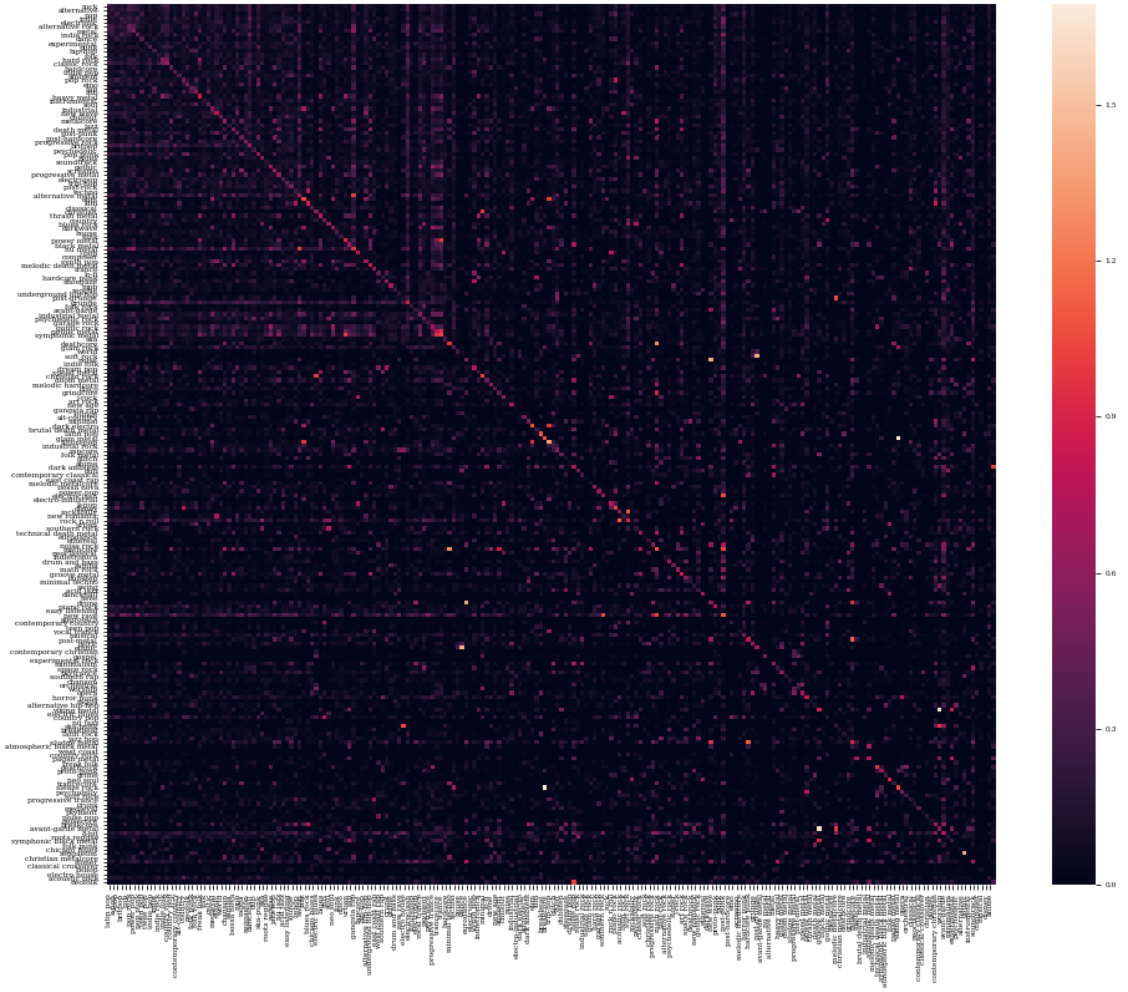


Figura 8: Heatmap de los géneros musicales sin ordenar.

Como podemos ver en la Figura 8 conseguimos una visualización que de primeras no nos dice mucho. Los géneros musicales están todos desordenados y cuesta interpretarlo. Por eso vamos a ordenar los géneros musicales de forma manual, agrupándolos según al género que pertenecen y/o sus orígenes musicales.

Después de ordenarlos, el heatmap queda de la siguiente manera:

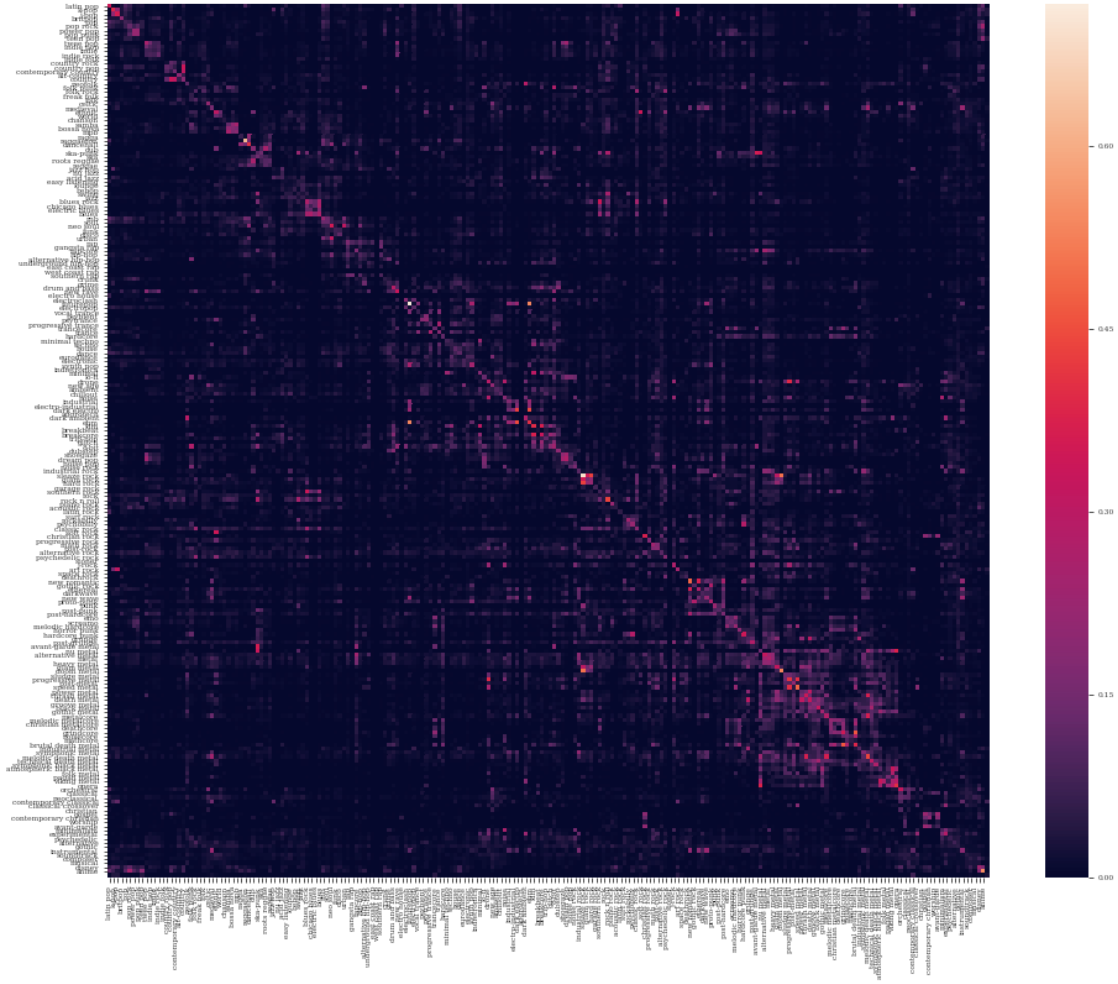


Figura 9: Heatmap de los géneros musicales ordenado.

Ahora sí podemos apreciar en la Figura 9 que hay una clara relación entre subgéneros del mismo género musical y también entre géneros cercanos evolutivamente.

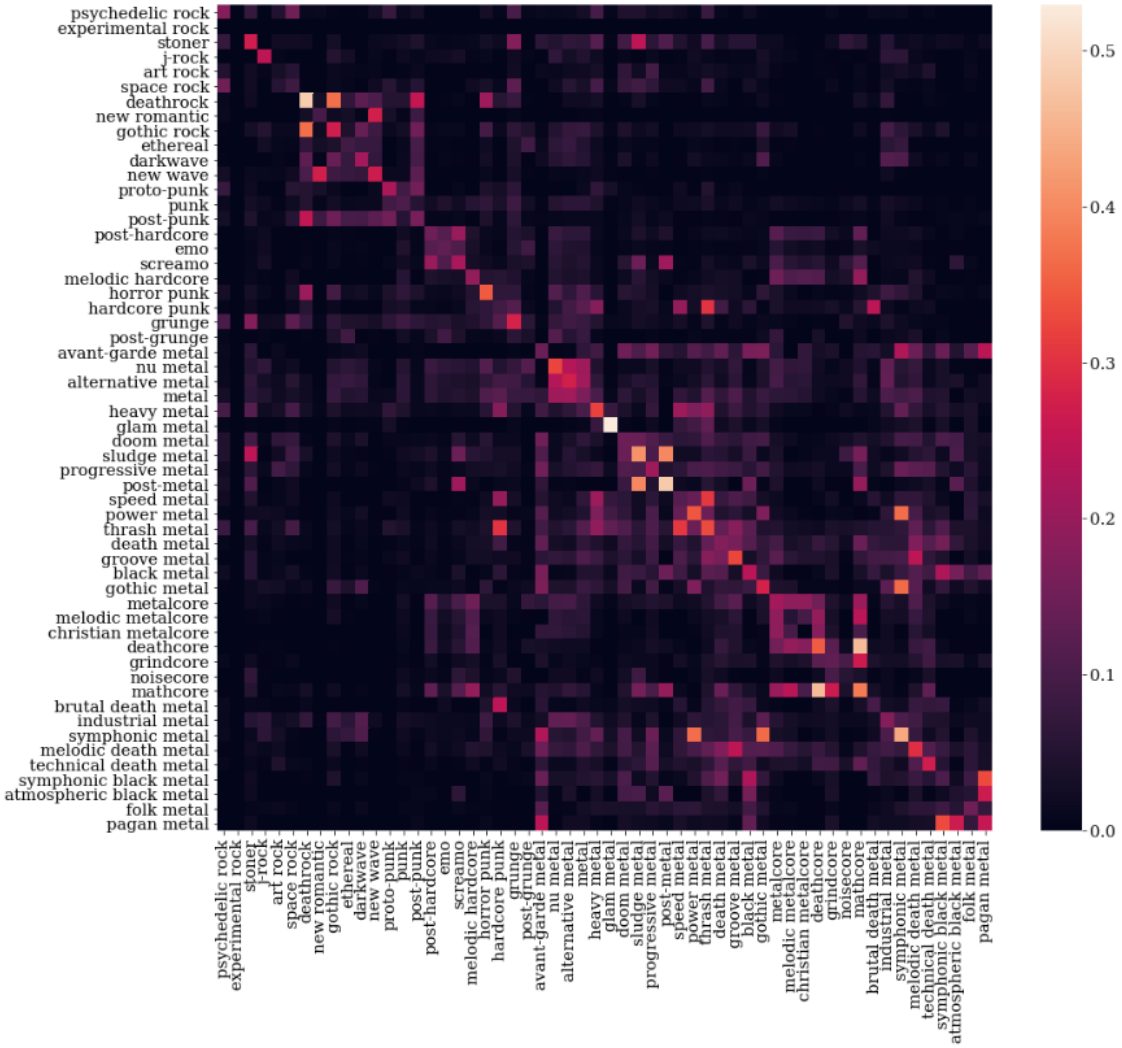


Figura 10: Heatmap Rock, Punk y Metal

Podemos observar en la Figura 10 cómo, por ejemplo, en la esquina inferior derecha del heatmap, hay una gran correlación entre géneros musicales, sobre todo en la parte de abajo. Esta zona corresponde al rock, el punk y el heavy metal. Podemos ver que la zona más altamente correlacionada corresponde a los subgéneros del heavy metal.

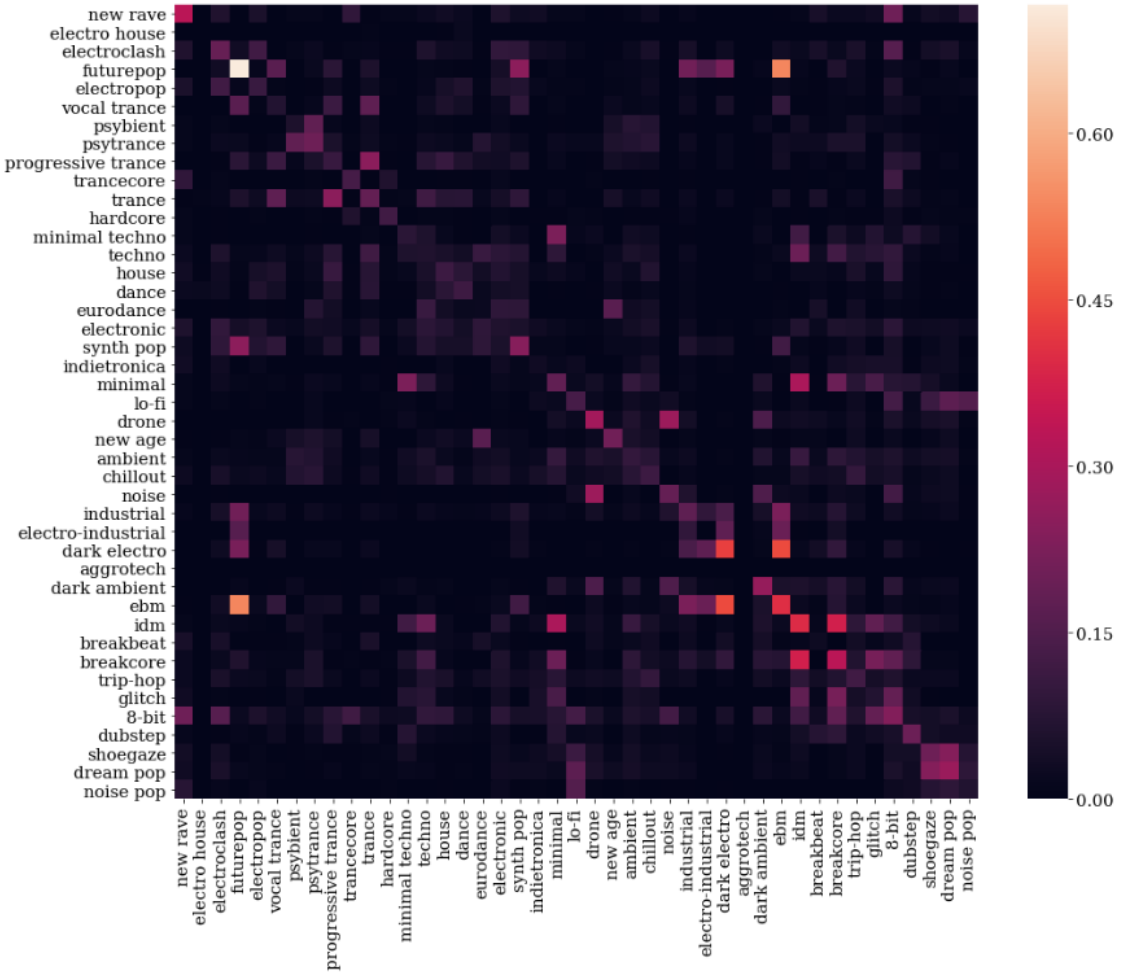


Figura 11: Heatmap Electrónica

Otra zona que también está bastante correlacionada, aunque no tanto como en el caso del heavy metal, es la música electrónica, que la encontramos en la parte central del heatmap. Podemos verlo en detalle en la Figura 11.

8.3. Ley Potencial (Power Law Distribution)

La segunda visualización es más sencilla tanto a nivel de implementación como conceptualmente. Es únicamente el grado de los géneros musicales, es decir, cuántos artistas hay de cada género musical. La vamos a usar por dos motivos: el primero es poder ver qué géneros tienen más popularidad dentro de la red y cuáles menos, y el segundo ver si esto sigue una forma de ley potencial.

Para calcular el peso de cada género musical vamos a sumar el número de veces que aparece cada género. El proceso lo vamos a realizar de forma independiente para cada uno de los 3 géneros que aparecen como atributos. Es

decir, primero vamos a contar las apariciones en el género 1 de cada artista, después en el género 2 y por último en el género 3. Para conseguir esto hemos creado la función ***get_genres_counts()***, que realiza los siguientes sencillos pasos:

Recorremos todos los nodos del grafo y guardamos en una lista el valor del género que estemos mirando esa iteración (1, 2 o 3). Entonces contamos el número de apariciones de cada género, obteniendo dos listas:

- Una lista con los géneros. Ej. $['rock','pop','alternative']$
- Una lista con el número de apariciones (count). Ej. $[10, 12, 8]$

Estas listas las guardamos a su vez en dos listas, una lista *pl_genres* que guardará las 3 listas de los nombres de los géneros y una lista *pl_counts* que guardará las 3 listas del número de apariciones .

De esta manera podremos visualizar tanto las apariciones de los géneros musicales de forma independiente: los que están en primera posición, en segunda o tercera, o realizar una visualización completa combinando las 3 listas con la función ***get_complete_genres_counts()***.

Después de realizar este proceso, podemos obtener, con la última función mencionada, la visualización de la figura 12.

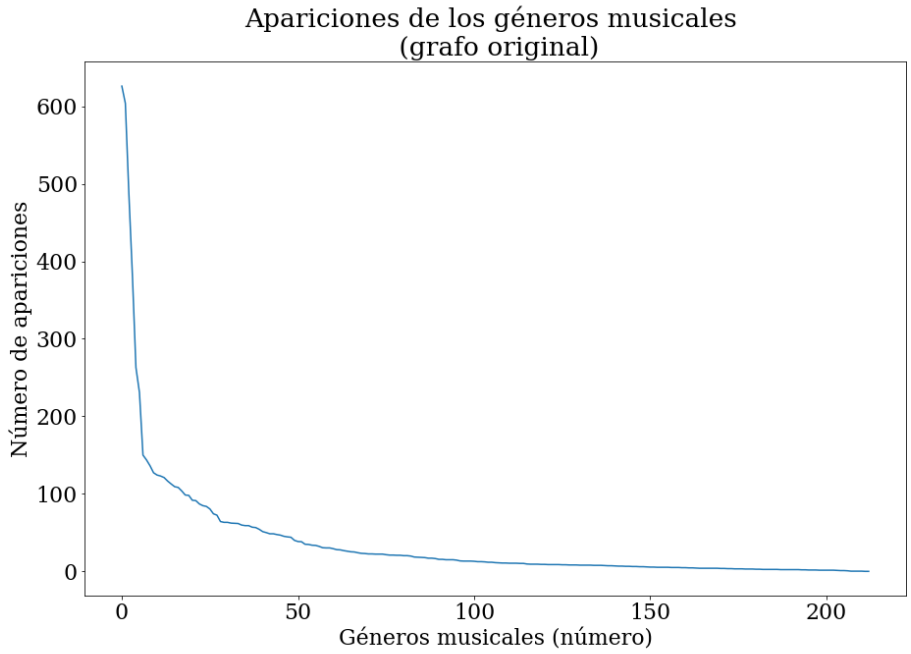


Figura 12: Power Law Distribution

Como podemos ver en el diagrama, observamos que hay unos pocos géneros musicales que concentran la mayoría de apariciones de la red, mientras que un gran porcentaje de los géneros, aproximadamente unos 150, tienen un número muy bajo de apariciones en comparación. Esto es debido a varios factores, pero principalmente a dos. Por una parte, es más fácil que los géneros más genéricos sean usados de forma más frecuente por los usuarios para clasificar a los artistas musicales (rock, pop, heavy metal, electronica, hip-hop). Pero también, por otra parte, no podemos ignorar que hay géneros musicales que son mucho más populares o *comerciales* que el resto, consiguiendo la mayoría del público, mientras que hay una gran diversidad de géneros musicales que son tanto realizados por pocos artistas, como escuchados por menos personas.

9. Adaptación del algoritmo de Axelrod

Una vez ya creada la red musical llegamos al objetivo de nuestro proyecto: ver cómo los artistas se influencian entre ellos, haciendo que los géneros musicales vayan diseminándose y evolucionando. También veremos si los géneros minoritarios consiguen sobrevivir, por ejemplo, al mantenerse más aislados en su propia comunidad.

Para ello realizaremos una adaptación del algoritmo desarrollado por Axelrod (1997) [12] que hemos visto en el apartado 2. Recordemos que este algoritmo trata de construir un modelo para estudiar el proceso de la difusión cultural basándose en dos sencillas suposiciones:

1. Es más probable que las personas interactúen con otras personas con las que comparten muchos de sus atributos culturales.

2. Estas interacciones tienden a incrementar el número de atributos que comparten, haciendo aumentar a su vez la posibilidad de que vuelvan a interactuar.

Los pasos de este algoritmo son bastantes sencillos. La adaptación que hemos realizado cuenta principalmente de los siguientes pasos:

Para empezar debemos elegir el agente activo. El agente activo será el artista que influenciará a otro artista de la red. Este segundo artista será el agente pasivo, que elegiremos seguidamente. Una vez elegidos los dos agentes, activo y pasivo, calcularemos el umbral a partir del cual, con probabilidad X, el agente activo influenciará al agente pasivo. Este umbral será igual al número de atributos culturales que comparten ambos agentes. Entendemos por influenciar el hecho de que uno de los géneros musicales del agente pasivo, que no sea común con el agente activo, sea reemplazado por el género más relevante (no común) del agente activo. Si la probabilidad X es superior a la umbral, se llevará a cabo el reemplazo.

Estos pasos se realizarán de manera indefinida, hasta que no se lleven X iteraciones seguidas sin que se haya producido ningún cambio en los atributos de los artistas.

Recordemos que la información y el formato de nuestro grafo es el siguiente:

- **Nodos:**

```
    {'lady gaga': {
        'attr_dict': {
            'genre1': 'pop',
            'genre2': 'dance',
            'genre3': 'electronic',
            'freq': 611
        }
    }}
```

Siendo el género 1 el más relevante y el género 3 el menos relevante.

- **Aristas:**

```
        ('lady gaga', 'britney spears', {'weight': 436.0})
```

9.1. Modificaciones introducidas

Vamos a ver en detalle los pasos y modificaciones realizados en nuestra adaptación. Nuestro algoritmo solo requiere que le pasemos por parámetro el grafo que hemos creado.

Selección del nodo activo

El primer paso de nuestro algoritmo es elegir el agente activo, es decir, el artista que influenciará a otro. Pero para poder elegirlo primero deberemos realizar un paso extra. Como queremos elegir el agente activo de manera aleatoria, pero dando mayor peso a los artistas que están más conectados, vamos a necesitar crear dos listas que nos permitan guardar, por un lado, el nombre del artista y, por otro, el peso total de sus aristas. Para ello hemos implementado una pequeña función **calculate_active_weights** que realiza esta tarea, y que llamamos antes de entrar al bucle principal de nuestro algoritmo.

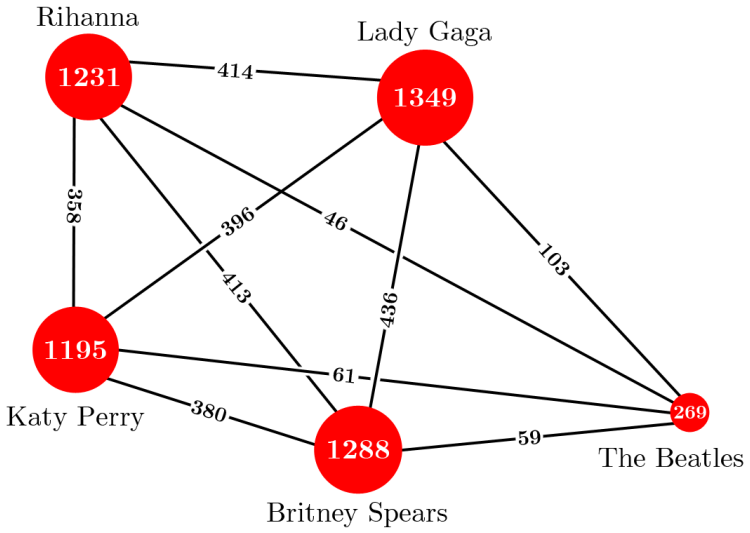


Figura 13: Subgrafo - Ejemplo selección nodo activo

Una vez creadas las listas sí que podemos pasar al bucle principal de nuestro algoritmo en el que realizaremos los pasos explicados anteriormente. Toca, por lo tanto, escoger el agente activo. Para poder elegirlo de manera aleatoria, pero dando más peso a los artistas con mayor grado, lo que hacemos es:

1. Sacar un número aleatorio del 0 al X (que es el peso total de todos los nodos).

2. Recorrer la lista de pesos en orden e irle restando al número obtenido en el paso 1 el valor de cada elemento. Actuará, por lo tanto, de contador.

3. Cuando el contador llegue a 0, habremos llegado al artista elegido. Miramos en qué elemento nos hemos quedado y a qué artista se corresponde en la lista de nombres.

Estos pasos se realizan en la función **get_random_node** pero, para que el código fuese más entendible, hemos hecho una función **get_random_active** que únicamente llama a la función anterior.

Selección del nodo pasivo

Lo siguiente es elegir el agente pasivo, que se escoge usando el mismo método que el agente activo, solo que en lugar del peso total del nodo lo que vamos a mirar es el peso de las aristas que están conectadas a nuestro agente activo. El nodo deberá ser, por lo tanto, un vecino del nodo activo.

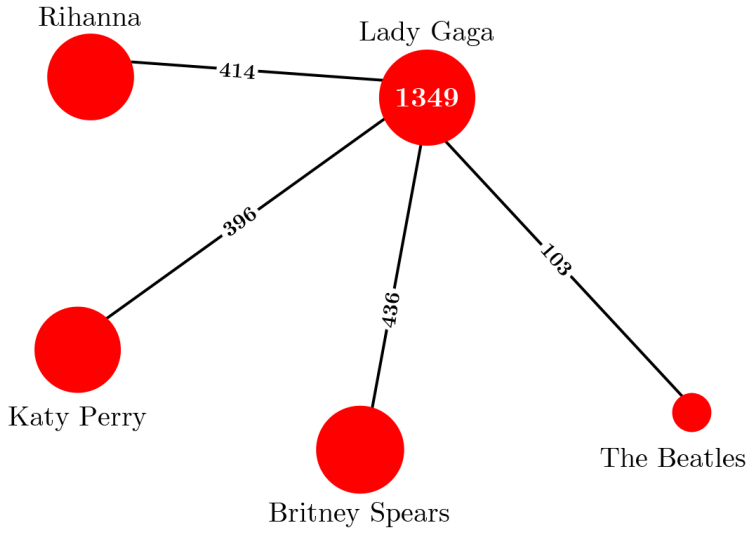


Figura 14: Subgrafo - Ejemplo selección nodo pasivo

Con la información de las aristas de nuestro nodo activo crearemos dos nuevas listas con el mismo formato que en el caso anterior (nombre - peso) usando la función **get_random_passive**. Con las nuevas listas creadas llamaremos a la función que usábamos en el caso del agente activo **get_random_node** para elegir el agente pasivo.

Selección de los géneros musicales: reemplazado y a reemplazar

Ahora que ya tenemos nuestro agente activo y nuestro agente pasivo, debemos ver si el primero influenciará al segundo. Primero vamos a guardar para cada nodo una lista de sus géneros musicales, para poder dejar de lado los 'NaN'.

Entonces miraremos, y guardaremos, cuáles de los 3 géneros musicales del agente activo no se encuentran en el nodo pasivo, haciendo la diferencia de los sets:

$$missing\_actives = |(active\_genres \setminus passive\_genres)|$$

Si falta algún género musical, entonces comprobaremos cuántos géneros musicales (que no sean 'NaN') comparten, haciendo la intersección de los sets:

$$num\_common\_genres = |(active\_genres \cap passive\_genres)|$$

Determinación de la probabilidad de interacción

El número de géneros que comparten será el que determine la probabilidad de que el nodo activo influencie al pasivo. Cuanto mayor sea, mayor será la probabilidad, por lo que reduciremos el umbral.

$$threshold = 1 - \frac{num\_common\_genres}{3}$$

Lo último que nos queda por hacer es lanzar una moneda. Si el valor del número aleatorio entre 0 y 1 da superior al valor umbral, entonces el agente activo influenciará al pasivo. En caso contrario no se hará ningún cambio y la variable 'not_change' se incrementará una unidad.

Para reemplazar cogeremos siempre el género activo más relevante (de los que no comparte con el nodo pasivo) y lo reemplazaremos por el género con menos peso (el tercero), a menos que sea uno de los géneros que se comparten con el nodo activo, en cuyo caso miraremos el segundo con menos peso y, sinó, el primero. Se reemplazan 'NaN's.

Fin del algoritmo: criterio de parada

Cuando la variable 'not_change' llega al valor indicado X, es decir, no se ha producido ningún cambio durante X iteraciones seguidas, el algoritmo se da por finalizado.

10. Pruebas y resultados

Ya hemos obtenido, procesado y limpiado los datos. Hemos creado nuestra red musical y analizado un poco algunas estadísticas interesantes de ésta, así como creado dos visualizaciones especiales para ver algunos aspectos importantes de la red. Por último, hemos realizado la adaptación del algoritmo de diseminación cultural de Axelrod, que nos permite simular el paso del tiempo, con el fin de ver la evolución y expansión de los géneros musicales. Por lo que podemos pasar al último paso de nuestro proyecto: aplicar este algoritmo a nuestra red y ver qué sucede.

Para ver y analizar los resultados vamos a usar principalmente la visualización creada en la sección 8.3, que muestra el número de veces que aparece cada género musical en la red. Como esta visualización se nos queda un poco corta, porque es difícil ver los cambios de posición de los géneros musicales, hemos creado una nueva visualización que funciona a modo de ranking. Lo veremos en detalle un poco más adelante.

10.1. Aplicación del algoritmo

Hemos aplicado nuestra adaptación del algoritmo de Axelrod varias veces, para ver si el resultado final es siempre el mismo o puede cambiar. Verememos las dos ejecuciones más frecuentes, en cuatro estados del proceso de difusión distintos: con valor de parada 200, 2000, 20000 y 200000. Esto quiere decir que veremos el estado de la red cuando lleva 200 iteraciones sin realizarse ningún cambio, 2000 iteraciones, 20000 iteraciones y 200000 iteraciones.

A medida que aumentamos el criterio de parada, el tiempo de ejecución cambia sustancialmente, sobre todo en el último caso, y el número de interacciones aumenta considerablemente. También podemos observar, gracias a las visualizaciones creadas, que los géneros musicales se equilibran más y se reordenan a medida que aumenta la interacción.

10.2. Homogeneidad musical

Uno de los aspectos más importantes a analizar después de aplicar el algoritmo es ver cómo han cambiado el número de apariciones de los géneros musicales.

Como vimos en el apartado 8.3 y en la Figura 15, los géneros musicales en nuestra red siguen la forma de una ley potencial. La mayoría de los artistas presentes en la red contienen un pequeño porcentaje del total de géneros musicales disponibles, lo que hace que la mayoría de apariciones se concentren en unos pocos géneros, mientras que el resto (la mayoría) aparecen de forma más o menos igual, pero en mucha menor medida.

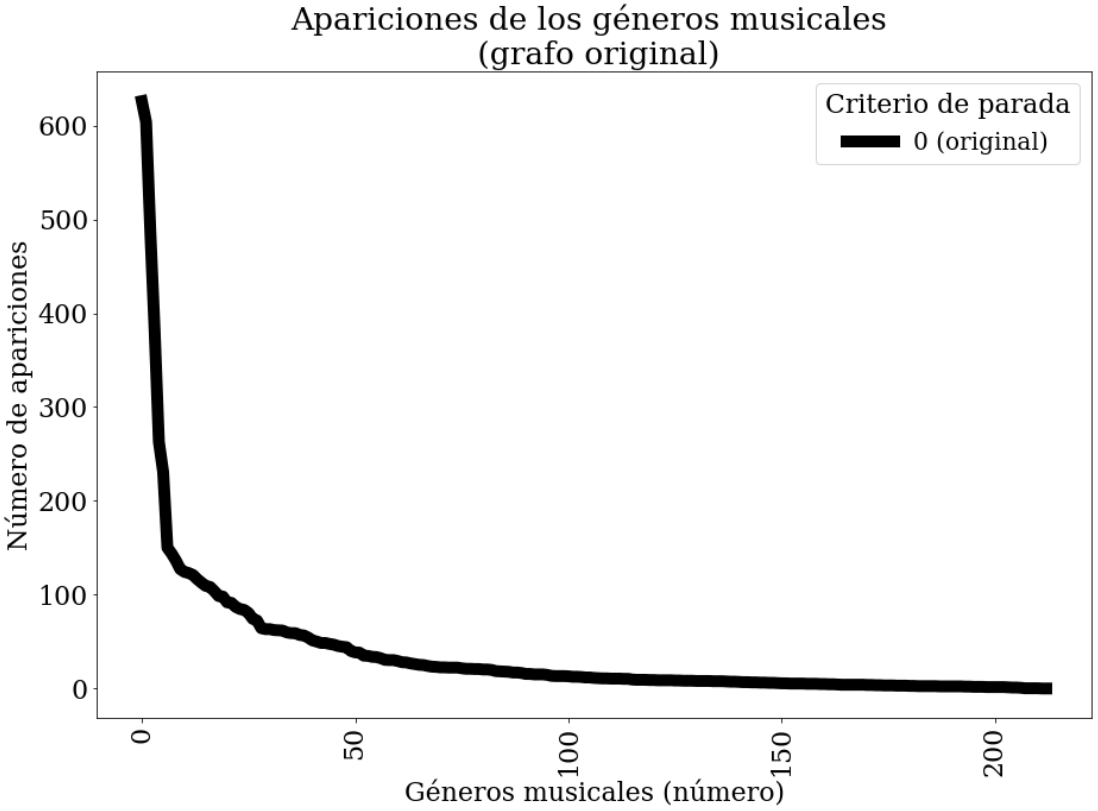


Figura 15: Ley potencial antes de la aplicación del algoritmo

Vamos a usar esta visualización para analizar el estado de los géneros musicales después de las distintas ejecuciones del algoritmo. Pueden ocurrir varias cosas:

1. **La forma de la ley potencial se mantiene tal y como estaba**: los géneros musicales están bien arraigados a los artistas, creando comunidades fuertes, y es difícil que estos se vean afectados por otras comunidades.

2. **La forma de la ley potencial se acentúe**: los géneros musicales más fuertes se imponen a los menos populares, haciendo que la industria musical se vuelva más comercial y parecida.

3. **La forma de la ley potencial se suaviza**: los artistas están abiertos a cambiar y evolucionar, y se ven afectados por las interacciones de aquellos artistas que están más relacionados con ellos, volviendo la red musical cada vez más variada.

Escenario final A

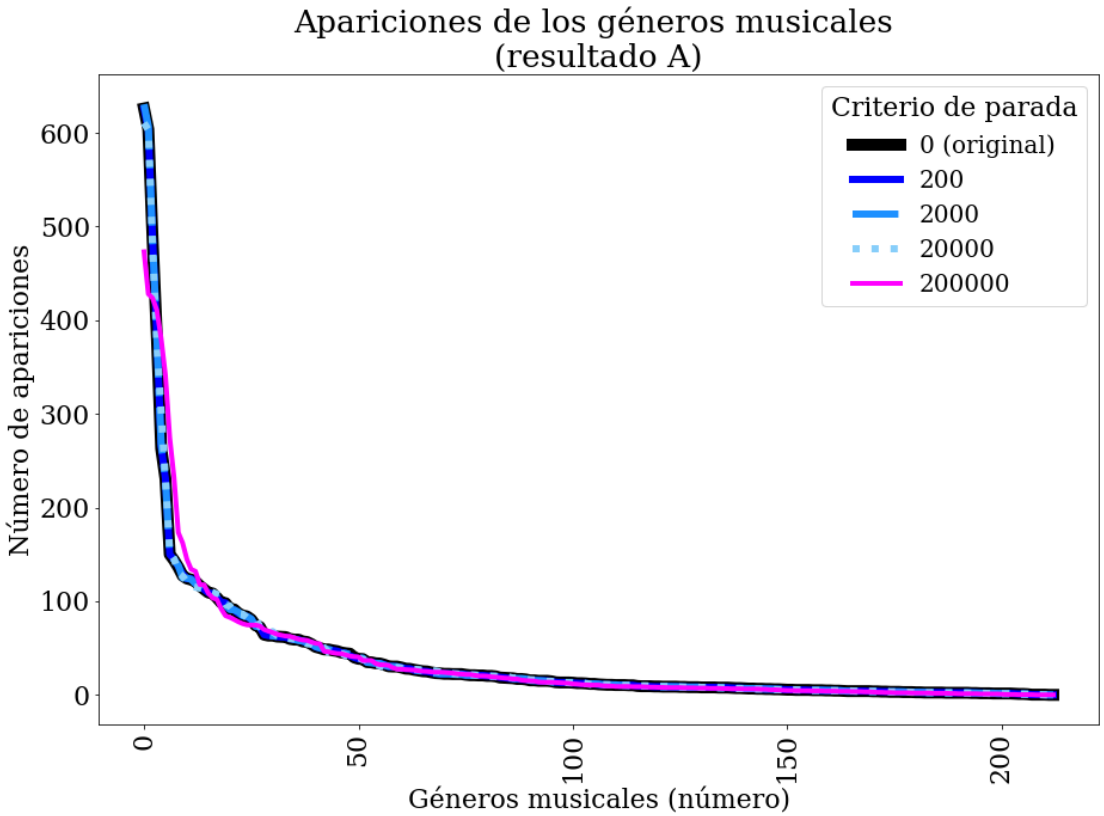


Figura 16: Ley potencial del escenario A

En la figura 16 vemos cinco líneas distintas, cuyos colores corresponden a los siguientes valores de criterio de parada: Original: negro, 200: azul marino, 2000: azul, 20000: azul claro y por último 200000: fucsia.

Como podemos ver, la línea fucsia, que corresponde al estado final de la red, es más suave que la original. Esto significa que en este caso la tendencia de la red es a volverse cada vez más variada. La forma de la ley potencial se va suavizando, haciendo que la parte de más a la izquierda se baje. En las primeras ejecuciones (azules) las líneas tienen prácticamente la misma forma que la original negra, lo que hace que la línea negra quede tapada en la mayoría de sus puntos por el resto, pero al aumentar el criterio de parada a 200000 los artistas realizan muchas más interacciones y los géneros se propagan más. La línea queda bastante más suavizada, y pasa de empezar en más de 600 a menos de 500, es decir, que no encontramos ningún género musical que cuente con más de 500 apariciones, mientras que en las anteriores ejecuciones sí los había.

Es decir, cuanto más aumentamos el criterio de parada, más interactúan los artistas y más cambios de géneros musicales se realizan. Los géneros mu-

sicales no van concentrándose en unos pocos que cada vez son más populares, sino lo contrario: se van repartiendo, volviéndose la red más variada musicalmente.

Escenario final B

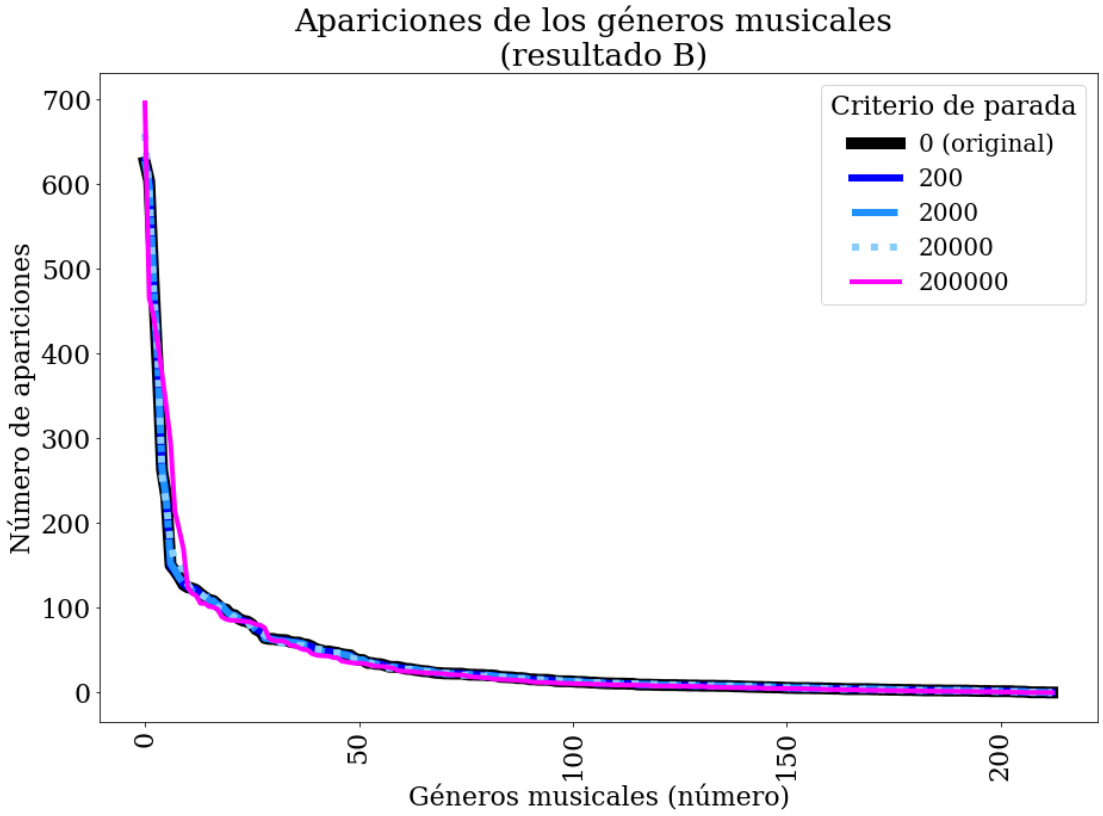


Figura 17: Ley potencial del escenario B

En la figura 17, en cambio, vemos que la tendencia de la red es la contraria. La forma de la ley potencial se va acentuando cada vez más, hasta llegar algunos géneros a casi las 700 apariciones. Lo curioso es que, además, algunos de los géneros donde se están concentrando la mayoría de artistas no se encontraban antes entre los más populares. Esto último no se ve reflejado en esta gráfica, por eso hemos creado una nueva visualización, que se encuentra en el siguiente subapartado, donde lo veremos en detalle.

Encontramos, por lo tanto, dos tendencias: la red puede irse cada vez equilibrando más, haciendo que algunos géneros muy populares pierdan algo de fuerza, o todo lo contrario. Esto sucede principalmente por culpa de dos factores: no tenemos suficientes datos, lo que puede hacer que los resultados sean menos rigurosos de lo deseado, pero sobre todo, hay algo que no podemos controlar y es el factor azar.

10.3. Fortaleza de los géneros musicales

En el apartado anterior podemos ver dos tendencias distintas: por un lado encontramos una red donde hay géneros que consiguen obtener un número mayor de apariciones, agravando la concentración de la audiencia, y por otro una red mucho más equilibrada. Pero nos faltan varios datos importantes: ¿cuántos géneros han obtenido un número mucho mayor de apariciones de los que ya tenían y, son estos géneros los mismos que al inicio? ¿Hay géneros impenetrables, que no se han visto afectados por el resto? ¿O han conseguido géneros minoritarios volverse más populares? No sabemos si las apariciones de los géneros musicales que estamos mirando corresponden a los mismos géneros en todas las ejecuciones, o si hay cambios de posición importantes.

Por este motivo hemos creado una visualización muy simple para poder ver los cambios en las posiciones de los tags. El objetivo es visualizar el listado de los tags, ordenados según su popularidad después de aplicarle el algoritmo. Para ello hemos creado la función ***get_top_tags()*** a la cual hay que pasarle las dos listas que se obtienen con la función explicada en el apartado anterior ***get_complete_genres_counts()*** antes de aplicar el algoritmo y después, es decir, en total cuatro listas.

Esta función devuelve e imprime (si lo deseamos) una lista que contiene:

1. posición final
2. apariciones finales
3. género musical
4. diferencia de posiciones
5. diferencia de apariciones
6. apariciones iniciales (no se muestra)
7. posición inicial (no se muestra)

Con esto podemos hacernos una idea de los cambios que han habido en la red. Si los géneros musicales se han quedado todos bastante estables de manera individual, o si, por el contrario, ha habido bastantes cambios, volviéndose algunos más fuertes y otros debilitándose. Además, vamos a pintar los cambios de posiciones de manera sencilla usando *Matplotlib*.

Ranking Inicial

#. (freq.)	género	Δposición	Δfreq.
1. (626)	rock	–	–
2. (603)	pop	–	–
3. (486)	indie	–	–
4. (382)	electronic	–	–
5. (263)	alternative	–	–
6. (230)	hip-hop	–	–
7. (150)	folk	–	–
8. (143)	classic rock	–	–
9. (136)	punk	–	–
10. (127)	indie rock	–	–

Escenario final A

#. (freq.)	género	Δposición	Δfreq.
1. (473)	alternative	⇑ +4	⇑ +210
2. (428)	indie	⇑ +1	⇓ -58
3. (423)	pop rock	⇑ +30	⇑ +361
4. (410)	pop	⇓ -2	⇓ -193
5. (380)	rock	⇓ -4	⇓ -245
6. (341)	electronic	⇓ -2	⇓ -41
7. (274)	new wave	⇑ +11	⇑ +170
8. (232)	hip-hop	⇓ -2	⇑ +1
9. (173)	trip-hop	⇑ +29	⇑ +116
10. (162)	indie rock	–	⇑ +35

Escenario final B

#. (freq.)	género	Δposición	Δfreq.
1. (696)	rock	–	⇑ +70
2. (466)	hard rock	⇑ +13	⇑ +353
3. (444)	electronic	⇑ +1	⇑ +62
4. (412)	pop	⇓ -2	⇓ -191
5. (376)	indie	⇓ -2	⇓ -110
6. (295)	classic rock	⇑ +2	⇑ +193
7. (214)	pop punk	⇑ +23	⇑ +231
8. (192)	industrial	⇑ +16	⇑ +129
9. (263)	hip-hop	⇓ -3	⇓ -38
10. (124)	alternative	⇓ -5	⇓ -93

Cuando miramos estas nuevas visualizaciones podemos observar nuevos aspectos que antes no podíamos apreciar. Realmente, a pesar de que inicial-

mente parecía que había dos tendencias muy distintas, una en la que unos pocos géneros conseguían la mayoría de oyentes y otra en la que la red se volvía más estable, la realidad es que si eliminamos el primer género musical del escenario B la red está prácticamente igual a nivel de número de apariciones.

Observamos, pero, que en el primer escenario los géneros más populares han perdido fuerza. El rock ha bajado de 626 a 380 oyentes, el pop de 603 a 410, y la electrónica de 382 a 341. En el segundo escenario, en cambio, se han conseguido mantener un poco más los géneros populares, ganando el rock y la electrónica 70 y 62 oyentes respectivamente, y quedándose igualmente el pop en 412. Además, si miramos con más detalle, vemos que a pesar de que en el primer escenario el pop ha perdido bastantes oyentes, el pop rock ha ganado muchísimos oyentes, 361.

En ambos escenarios, géneros que antes eran muy poco escuchados han ganado un gran número de oyentes, entre 100 y 200 aproximadamente. Pero estos géneros no coinciden en las dos ejecuciones.

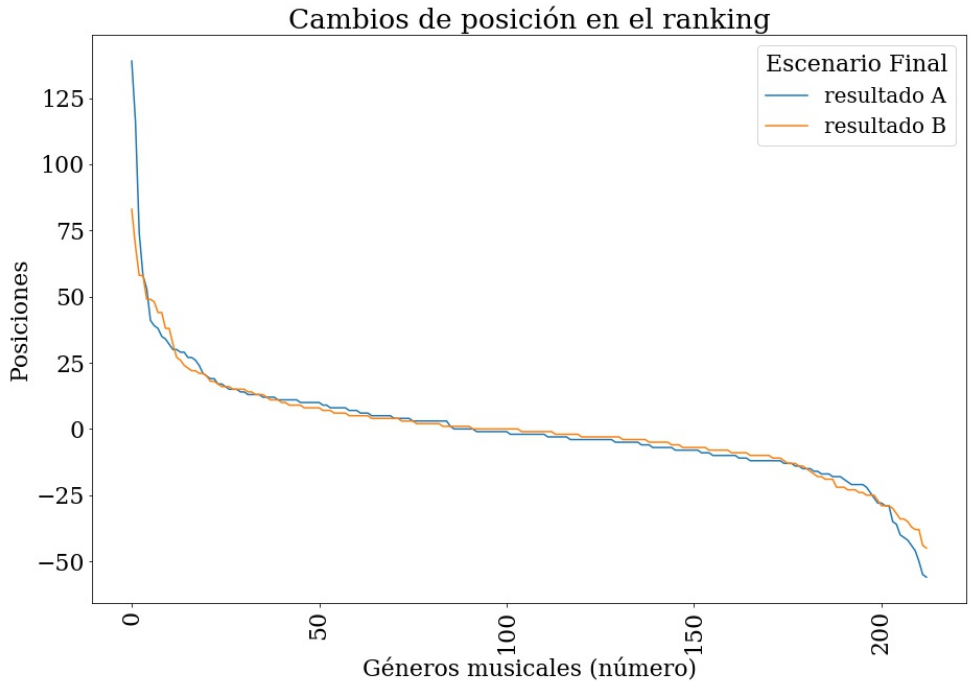


Figura 18: Cambios de posición en ambos escenarios

En la figura 18 vemos que en ambos casos hay un gran número de cambios de posición, algunos de ellos muy elevados, sobre todo cuando se trata de subir en el ranking. En el primer escenario, que es el que está más equilibrado, hay varios géneros que consiguen subir hasta 100 posiciones, pero en cambio

la gran mayoría de géneros no cambian apenas de posición. En el segundo caso, la curva es algo más suave pero hay más géneros que se mueven de posición.

10.4. Resultados

Los resultados del proyecto no son únicos, pero nos permiten ver algunos aspectos importantes e interesantes del estado actual y futuro de la música.

Observamos que, a pesar de que hay una sensación de que la música está muy marcada y sigue una línea, donde no hay sitio para la diversidad ni la música alternativa, la realidad es que el número de géneros musicales existentes, más o menos populares, es muy elevado. En el mundo de la música hay mucho desconocimiento, del cual son responsables, en parte, los medios y la propaganda. En los datos vemos que hay muchos géneros considerados minoritarios y/o poco conocidos que realmente tienen un número bastante elevado de oyentes.

Después de la aplicación del algoritmo, vemos que encontramos diferentes posibles tendencias futuras, pero todas acaban más o menos en el mismo lugar. Hay un pequeño grupo de géneros musicales que puede ganar muchos oyentes, y otro pequeño grupo que puede perderlos, aunque en menor medida, pero la gran mayoría se mantienen estables. Algunos géneros actualmente populares seguirán teniendo fuerza, aunque ocasionalmente puedan perder oyentes, y géneros antes minoritarios que ganarán mucha popularidad, pero no podemos saber a ciencia cierta qué géneros serán los que se vean afectados.

Actualmente es difícil prever qué género se convertirá mañana en tendencia y marcará el futuro musical de los próximos años. Internet se ha convertido en una herramienta esencial para la popularización y el consumo de música, ya que está al alcance de la mayoría. Esto facilita que cualquiera pueda disponer de toda la música que desee con un sólo clic, pero también que artistas que están empezando puedan darse a conocer de manera sencilla. Este último factor creo que es esencial para interpretar los resultados del proyecto, ya que añade un componente sorpresa que puede alterar rápidamente la red.

Lo que sí podemos sacar en claro es que la música está en un constante cambio, donde van apareciendo géneros musicales nuevos, cogiendo fuerza otros, y a veces, perdiendo, pero no debemos preocuparnos porque la música se quede estancada.

11. Conclusiones y trabajo futuro

Gracias a este proyecto he aprendido a realizar el proceso completo de un proyecto (aunque pequeño) de ciencia de los datos. He visto cómo manipular grandes volúmenes de datos para poder usarlos de forma correcta y dar solución a problemas que resulten de interés. Cómo abordar un problema complejo del mundo real a uno modelado usando la teoría de grafos al que poder buscar una solución. He adaptado un algoritmo con el fin de resolverlo y por último he tenido que buscar maneras de visualizar correctamente los aspectos importantes de la red.

Muchas de estas tareas he podido realizarlas gracias a los conocimientos previos adquiridos a lo largo del grado. En *Algoritmia y Algoritmia Avanzada* los estudiantes aprendimos las bases de la programación en Python y a realizar pequeños algoritmos. En *Inteligencia Artificial* pudimos profundizar más en los algoritmos y vimos los grafos y su implementación. Y en *Taller de los Nuevos Usos de la Informática* aprendimos algunos conceptos básicos de la ciencia de los datos y la manipulación de datos en *Python*. Además, a lo largo de los estudios he aprendido a ser autónoma y ser capaz de resolver problemas nuevos.

Este trabajo supone el inicio de un estudio sobre tendencias musicales y es por esto que aun queda mucho trabajo por delante. Se han realizado los pasos iniciales: hemos generado un set de datos nuevo. A partir de este set de datos hemos podido crear y analizar la red musical con distintas métricas y se han propuesto cambios en el algoritmo de diseminación cultural de Axelrod con el fin de adaptarlo a nuestro problema. Por último se ha aplicado el algoritmo final a nuestra red, pero nos habría gustado poder disponer de más tiempo para analizar los resultados con un mayor rigor estadístico.

Es por esto que las conclusiones finales del proyecto no han sido muy esclarecedoras, pero hemos podido ver el estado actual de la música y los posibles escenarios futuros de la escena musical.

En el futuro nos gustaría poder construir la red con un volumen de datos mucho mayor, de manera que los resultados fuesen más fiables. También sería interesante analizar la red más en detalle, viendo cuáles son los principales focos de la red, qué artistas o grupos de artistas hacen que el resto se vea más afectado.

Otro aspecto curioso sería ver cuál es el flujo de los géneros musicales: qué géneros son los que más tienden a evolucionar y hacía dónde. Si, por ejemplo, artistas con géneros musicales más “génericos”, como el rock, se van poco a poco volviendo más específicos, evolucionando a subgéneros del rock, como el rock progresivo, o al contrario, o los artistas van evolucionando de un género a otro que no era de su misma familia, por ejemplo de rock a pop. Por último sería interesante intentar prever el surgimiento de nuevos géneros musicales.

Referencias

[1] Last.fm
https://www.last.fm/home

[2] Axelrod R.: The Dissemination of Culture, *Journal of Conflict Resolution, Vol. 41 No. 2*, pág. 203, Abril 1997.
https://www.jstor.org/stable/174371

[3] Information Retrieval Group de la Universidad Autónoma de Madrid (http://ir.ii.uam.es), 2011.
https://grouplens.org/datasets/hetrec-2011/

[4] https://es.wikipedia.org/wiki/Last.fm

[5] Last.fm Web Services
https://www.last.fm/api/

[6] https://www.python.org/

[7] NetworkX developers: Software for complex networks, 2014-2019.
https://networkx.github.io/documentation/stable/

[8] https://www.mundodeportivo.com/elotromundo/lifestyle/20180513/443210713312/trap-musica-rap-hip-hop-bad-gyal-kidd-keo-dellafuente-drake-kendrik-lamar-post-malone-bad-bunny.html

[9] LOS40: ¿Qué es el trap y cuáles son sus máximos exponentes?
https://los40.com/los40/2019/01/16/musica/1547645386_647009.html

[10] Newman M.E.J.: Networks: An Introduction, *Oxford*, pág. 1-2, 2010.

[11] Axelrod R.: The Dissemination of Culture, *Journal of Conflict Resolution, Vol. 41 No. 2*, pág. 204, Abril 1997.
https://www.jstor.org/stable/174371

[12] Leydesdorff L.: Technology and Culture: The Dissemination and the Potential 'Lock-in' of New Technologies - Axelrod's model of dissemination of culture, *Journal of Artificial Societies and Social Simulation vol. 4, no. 3*, 2001.
http://jasss.soc.surrey.ac.uk/12/1/6/appendixB/Axelrod1997.html

[13] https://www.numpy.org/

[14] Newman M.E.J.: Networks: An Introduction, *Oxford*, pág. 235(7.80), 2010.

[15] Newman M.E.J.: Networks: An Introduction, *Oxford*, pág. 200 (7.9), 2010.

[16] Degree Distribution, *Università di Chieti-Pescara*.
https://www.sci.unich.it/ francesc/teaching/network/distribution.html

[17] Heatmap,
https://en.wikipedia.org/wiki/Heat_map