**Procesamiento y manejo de la muestra o dataset:**

Partimos de un dataset de alrededor de 19.000.000 millones de líneas el cual estaba muy “sucio”. Constaba de los siguientes campos: **<user,timestamp,artist, song>.** El dataset fue obtenido a partir de la API de Last.Fm, una plataforma que almacena y proporciona mucho contenido musical. Los campos hacen referencia al usuario que ha escuchado la canción, la fecha en la que fueron escuchadas, el nombre del artista y el título de la canción respectivamente.

El primer paso fue hacer un pequeño estudio del dataset, para hacernos una idea de cómo era nuestra muestra y sobre que podríamos sacar de ella. Se trataba de un dataset con muy pocos campos que no daba información ninguna sobre la canción o el artista si no que solo te proporcionaba un medio para poder obtenerla.

Hicimos una limpieza del dataset ya que había numerosos valores que eran nulos en determinados campos o no tenían una codificación correcta.

Es en este punto donde comenzamos a pensar como queríamos mostrar la información que nosotros podíamos ofrecer previa al funcionamiento de la aplicación. Teníamos dudas de ofrecer cualquier objeto del dataset o restringirlo solo a algunos.

En la librería de Wikidata, por motivos obvios se debe poner como input de cualquier función de búsqueda, el código del objeto sobre el que se quiere obtener información. Estos códigos son fijos y estáticos para cada uno de los objetos que están registrados en la página. Con nuestra librería somos capaces de a partir de un string que represente un título de canción o un artista, género… obtener su respectivo código para posteriormente procesar las consultas.

El problema aquí es que para obtener el código del objeto, su nombre título o descripción, debía ser exacto al que aparecía en Wikidata, de cualquier otra forma, se lanzaría una excepción. Debido a este caso es necesario hacer un parseo previo de cualquier String que se vaya a utilizar como input.

Como decisión final, elegimos crear una lista preseleccionada y parseada de las canciones más populares de todo el dataset(limpio). Para ello ordenamos las canciones del dataset por popularidad descendente, contando las veces que aparecían en la muestra. Después, elaboramos un scrip que recorría todas ellas, ejecutaba el parseo y finalmente comprobaba si era posible obtener los datos de Wikidata.

Empezamos con un dataset de las 2500 canciones más populares y fuimos capaces de obtener la información de 1408 canciones con su determinado artista. Cabe señalar que en cada búsqueda de una canción el artista debe ir implementado como parámetro pues hay muchas canciones con el mismo título que no podrían diferenciarse.

Finalmente probaremos la aplicación final con ese dataset limitado pero que cuenta con una seguridad de que se elija la canción que se elija para analizar, vamos a obtener datos fiables. Además, al haber escogido las canciones con mayor popularidad nos vamos a encontrar con mayor cantidad de datos ya que estas eran las que más documentadas estaban , a diferencia de las que estaban en la parte inferior del dataset, que eran muy poco conocidas y a penas se podían sacar datos valioso sobre ellas.

**Código y Librerías**

Como inicio tenemos varios scripts que nos ayudan a organizar y manejar mucho mejor los datos, como por ejemplo el script de limpieza y organización del dataset ,que como ya hemos explicado antes, elimina cualquier tipo de valor no nulo y lo ordena por popularidad o el script de parseo, que nos permite acercarnos más a la sintaxis gramatical de Wikidata.

Como punto troncal, hemos desarrollado una librería que nos permite trabajar con la app de wikidata. Hace uso de SparQL Library la cual no permite hacer uso del lenguaje SPARQL y hacer todo tipo de consultas a un punto. Esta librería se encarga de crear un punto de conexión a la API de Wikidata y de crear y ejecutar queries de consulta o obtención de datos.

Como siguiente punto tendríamos lo que es el modelo de la app, que es una serie de clases que se encargan de ejecutar todo el proceso de obtención de datos, el cual es genérico para todas las canciones.

Cuenta con una lista de propiedades que son el objeto de estudio de cada canción y métodos para añadir o eliminar alguna de estas. Actualmente cuenta con un número de 21 propiedades totales, las cuales pueden tener su vez aún más, de un modo que llega a ser recursivo.

Por último la clase principal, el la que se encarga de hacer todas la llamadas al modelo de cada una de las dos canciones y artistas que son introducidos. También se encarga del manejo de excepciones y creación del csv con las relaciones totales.

**Trabajo futuro:**

El esquema RDF y el uso de SPARQL para su uso y análisis puede ser muy potente y abarcar grandes cantidades de datos. En nuestro caso nos hemos centrado en trabajar con la herramienta y los datos de Wikidata, por un caso de cantidad de información, fiabilidad y resultados obtenidos.

Con Wikidata hemos sido capaces de crear conexiones entre canciones con toda la información posible en cada una de sus respectivas páginas descriptivas. Sin embargo hay muchas más opciones de donde recopilar más información sobre canciones y artistas.

En un principio intentamos obtener información que se alejaba un poco más de lo que hemos creado, que es un análisis musical. Hay otras opciones y datos que se pueden sacar sobre canciones, como por ejemplo las películas o series en las que aparecen como banda sonora, o también canciones que han sido emitidas durante grandes eventos como la superbowl. El problema que obtuvimos es que el proyecto en ese punto se desviaba de su rumbo original ya que en Wikidata no había datos tan específicos y deberíamos navegar en otras fuentes que probablemente no tendrían la opción de una librería con lenguaje sparql.

Además se trataba de información muy específica que relataría conexiones entre canciones o artistas meramente a modo de curiosidad, lo cual puede resultar muy interesante, el incluir relaciones alejadas de las explicaciones musicales e incluir otro ámbito más social o emparejados a movimientos culturales.

Otro aspecto en el que se puede trabajar para llegar a un trabajo mucho más completo y poder hacer un explicación de la relación de dos canciones, sería un estudio a nivel de usuario:

En nuestro proyecto hemos hecho un análisis avanzado de las relaciones que tienen las entidades en su ambiente musical. Sin embargo hay otro estudio posible que es el análisis de los usuarios que han escuchado esas canciones. Partiremos de la base de que dos personas que escuchan una misma canción, pueden compartir parte de sus gustos musicales, y se pueden recomendar y crear relaciones conforme a su historial.

Cada usuario tiene un historial de tracks escuchados, de hecho uno de los datasets que obtuvimos al principio del proyecto eran de este carácter. Si se hiciese un estudio individual de un usuario, se podrían analizar sus gustos musicales, canciones más escuchadas últimos géneros más populares etc…

Como ejemplo podemos seleccionar un usuario cuya mayoría de tracks pertenecen al género pop, pero también cuenta con algunos pertenecientes al género indie durante los últimos meses. Si otro usuario para el cual el género pop también fuese el más popular, podríamos recomendarle tracks de género indie por correlación con el primer usuario.

Este se trata de un ejemplo muy básico, pero con un estudio mucho más amplio se podrían obtener recomendaciones mucho más fiables y coherentes.