Universidad de los Andes
Ingeniería de Sistemas y Computación
Estructuras de Datos y Algoritmos

Reto 3: Soundtrack Your Timeline

Documento de análisis de los requerimientos propuestos en el Reto 3 del curso

María Paula Almériga Moreno (Estudiante B)
m.almeciga@uniandes.edu.co
202023369

Andrés Felipe Vargas Cuadros (Estudiante A)
af.vargasc1@uniandes.edu.co
202013817

Bogotá, Colombia
Mayo 2021

Índice

Descripción

En el tercer reto del curso, **Soundtrack Your Timeline**, se presenta una serie de datos sobre los análisis de canciones para entender el comportamiento de los usuarios, y saber qué tipo de música puede ofrecérseles en un momento determinado, la cual fue provista en Kaggle, y se titula Context-Aware Music Reccomender System. Tiene cerca de 11 millones de eventos de escucha, asociados a publicaciones realizadas por distintos usuarios en Twitter. Dichos eventos y canciones pueden ser analizados y clasificados de distintas formas teniendo en cuenta parámetros numéricos que los describen; puede así determinarse el tipo de música, los estados de ánimo que representan, estimar la cantidad de voz o instrumentos musicales presentes, y demás, de ese modo recomendando una buena opción dependiendo del contexto, por ejemplo, canciones para dormir, estudiar, celebrar, etc.

Para el presente reto se consideran los siguientes objetivos:

- Utilizar árboles binarios de búsqueda y árboles balanceados, en conjunto con las demás estructuras de datos del curso (listas y tablas de Hash) para solucionar los requerimientos del reto.
- Utiilizar adecuadamente el patrón Modelo, Vista, Controlador.
- Utilizar adecuadamente el ambiente de trabajo (IDE, GIT y GitHub).

Para la implementación del reto se han propuesto los siguientes cuatro requerimientos:

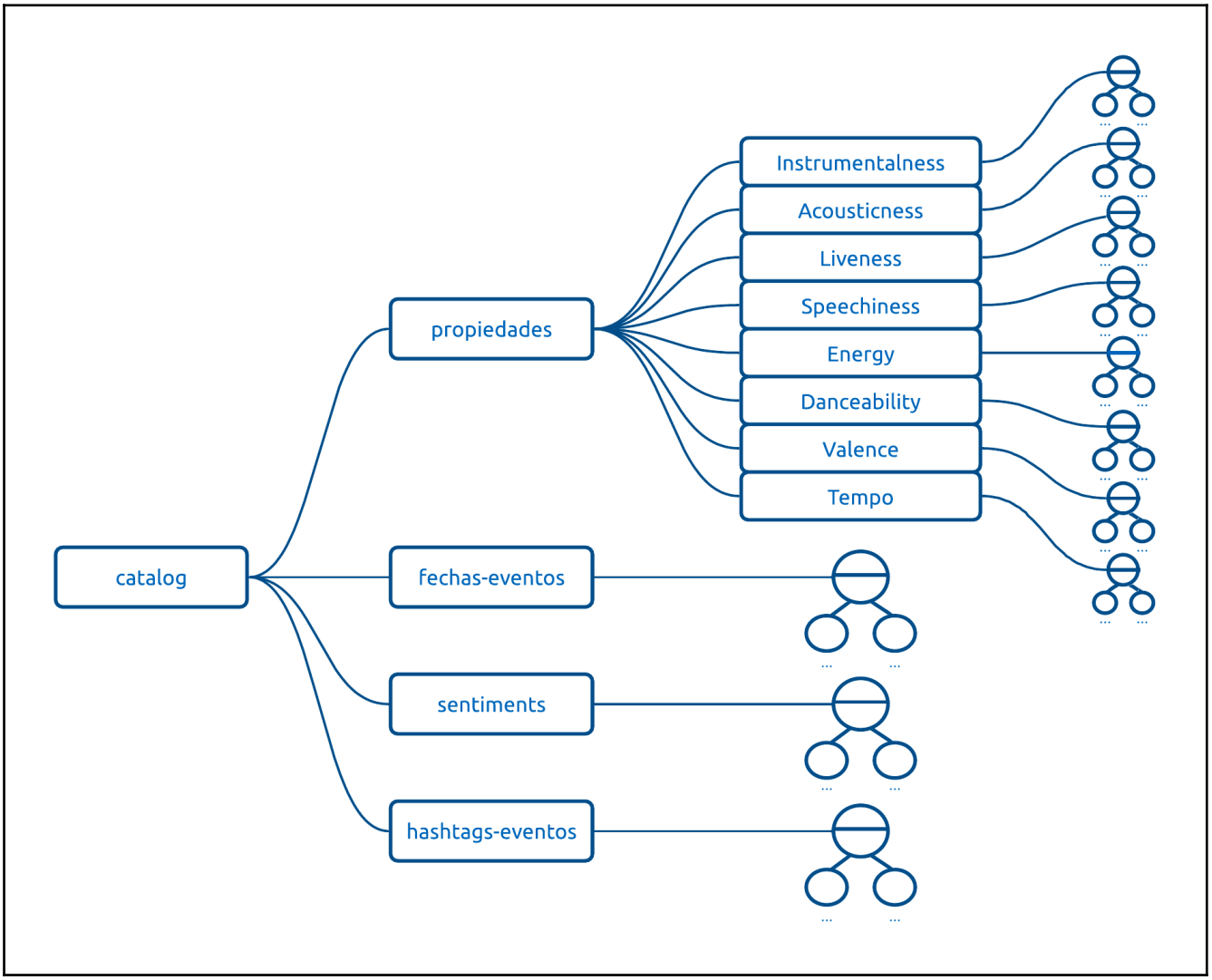
1. Caracterizar las reproducciones (Grupal)
2. Encontrar música para festejar (Estudiante A)
3. Encontrar música para estudiar (Estudiante B)
4. Estudiar los géneros musicales (Grupal)
5. Indicar el género musical más escuchado en el tiempo (Grupal)

A continuación se presenta información más detallada sobre cada uno de los requerimientos y un análisis de su rendimiento en términos de tiempo de ejecución y consumo de memoria, y análisis de complejidad.

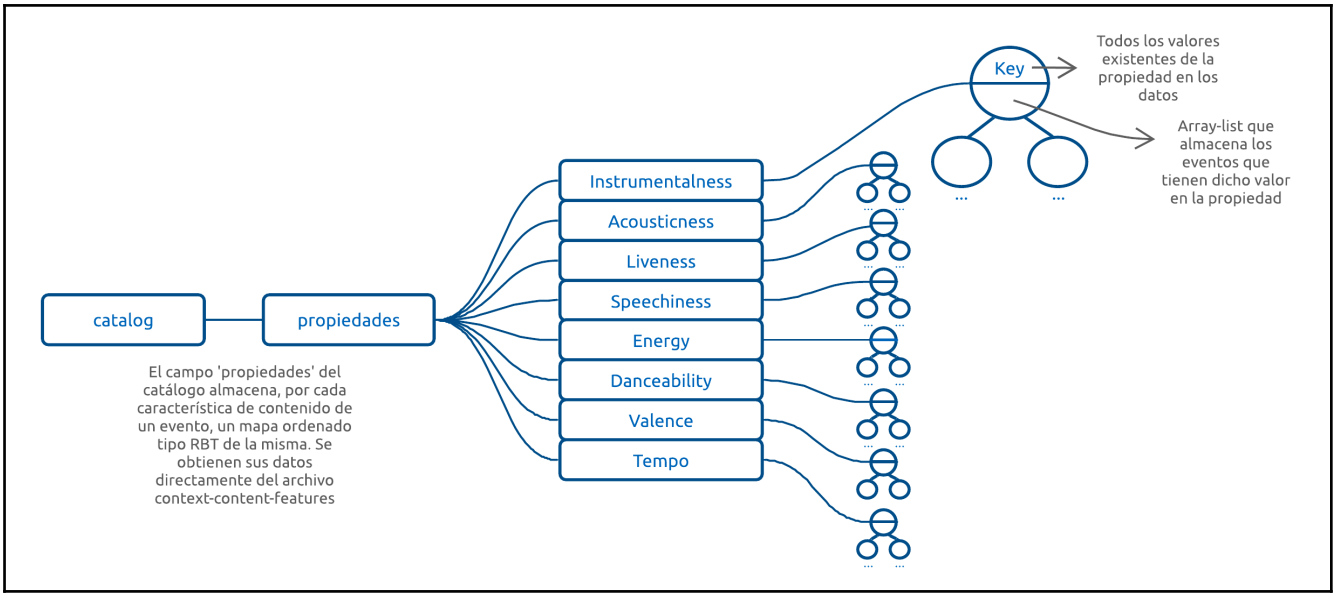
Diagramas

Se presentan los siguientes diagramas para describir las estructuras de datos utilizadas para la implementación de los requerimientos en general:

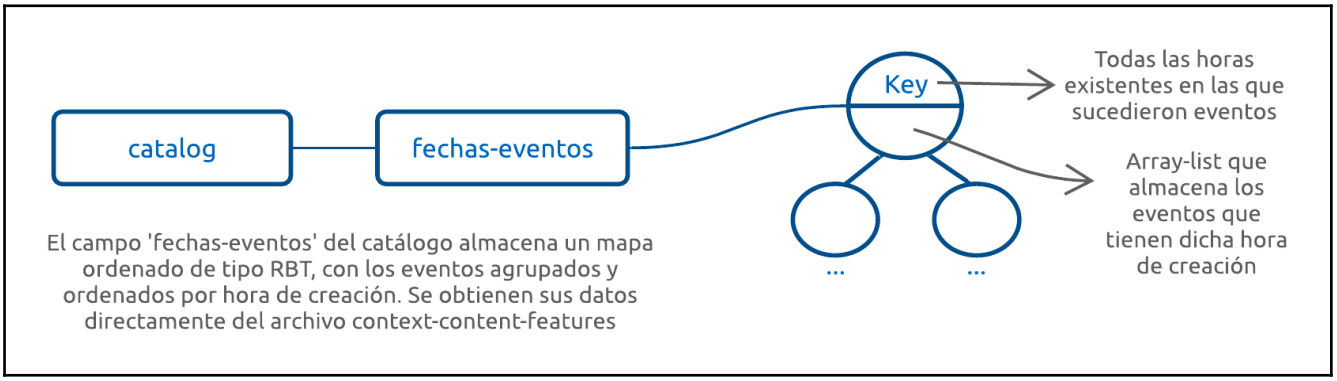
- Diagrama general (catalog): La estructura general que almacena todos los datos es llamada catalog, y es un diccionario nativo de Python. Tiene como llaves ‘propiedades’, ‘fechas-eventos’, ‘hashtags-eventos’ y ‘sentiments’, las cuales tienen como valor una o más estructuras de datos que los agrupan y ordenan por distintos criterios. Más adelante se explicará con mayor detalle cada una de las llaves.



- Propiedades: Almacena, por cada característica de contenido de los eventos, un mapa ordenado de tipo RBT. Se obtienen sus datos directamente del archivo context-content-features. Como se puede apreciar, se tienen las características de contenido Instrumentalness, Acousticness, Liveness, Speechiness, Energy, Danceability, Valence, y Tempo. En cada una de ellas, como se dijo anteriormente, se almacena un mapa que tiene como llaves todos los valores existentes de la propiedad correspondiente dentro de los datos, y como valor un array-list que almacena los eventos que tienen dicho valor en la propiedad. Esta estructura es útil cuando se quiere filtrar los eventos por rangos de valor dentro de características específicas.

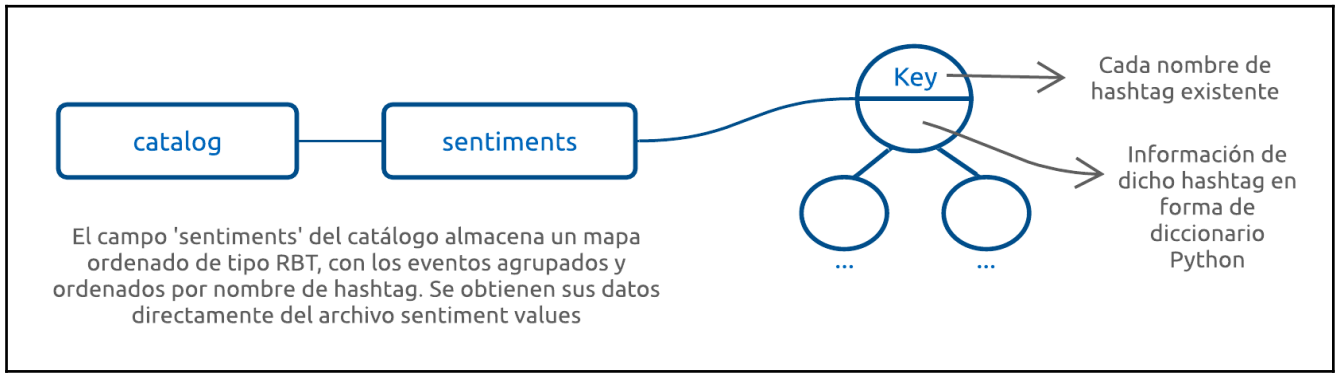


- Fechas-eventos: Almacena un mapa ordenado de tipo RBT, con los eventos agrupados y ordenados por hora de creación. Se obtienen sus datos directamente del archivo context-content-features. Cada nodo del mapa tiene como llave todas las horas existentes en las que sucedieron eventos, y como valor un array list con los eventos que ocurrieron en dicha hora. Esta estructura es útil para filtrar los eventos por rangos de fecha de creación.

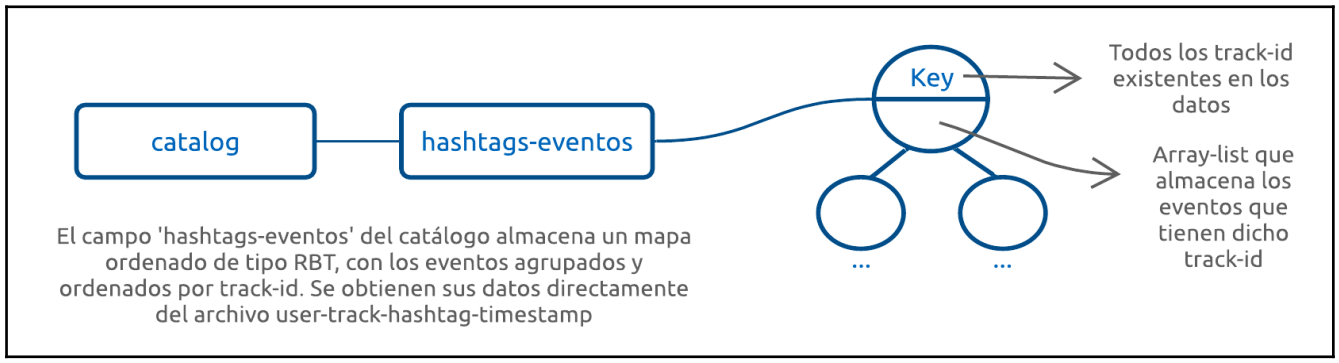


- Sentiments: Almacena un mapa ordenado de tipo RBT, con los eventos agrupados y ordenados por nombre de hashtag. Se obtienen sus datos directamente del archivo sentiment-values. Cada

nodo del mapa tiene como llave cada nombre de hashtag existente, y como valor la información de dicho hashtag almacenada en un diccionario nativo de Python. Esta estructura es útil para filtrar los eventos por su hashtag, que pueden ordenarse según criterios variados tales como su órden alfabetico, en caso de ser necesario.



- Hashtags-eventos: Almacena un mapa ordenado de tipo RBT, con los eventos agrupados y ordenados por track-id. Se obtienen sus datos directamente del archivo user-hashtag-timestamp. Cada nodo del mapa tiene como l track-id lave todos los track-id existentes en los datos, y como valor un array-list que almacena los eventos con dicho track-id. Esta estructura es útil para agrupar los eventos por canción, es decir, aquellos que tengan el mismo track-id.



Requerimiento 1

Se quiere conocer cuántas reproducciones (eventos de escucha) se tienen en el sistema de recomendación basado en una característica de contenido y con un rango determinado.

Entradas:

- La característica de contenido.
- El valor mínimo de la característica de contenido.
- El valor máximo de la característica de contenido.

Salidas:

- Total de eventos de escucha o reproducciones.
- El número de artistas únicos (sin repeticiones).

Análisis de complejidad:

Se determinó que el análisis de complejidad en notación O es:

$$O(value \cdot event)$$

Donde value y event son variables del siguiente ciclo en el model:

```
for value in lt.iterator(carac_filtrado):
    for event in lt.iterator(value['eventos']):
        total_events += 1
        key = event['artist_id']
        val = 0
        om.put(artists_map, key, val)
```

Pruebas:

A continuación se presentan los resultados de dos pruebas ejecutadas con subconjuntos de datos de distintos tamaños (-small.csv y -5pct.csv respectivamente). Adicionalmente, se incluyen los resultados de consumo de memoria y tiempo en la ejecución.

```
----------------------------------------------  Requerimiento #1

Ingrese la característica de contenido: instrumentalness
Ingrese el valor mínimo para la característica: 0.75
Ingrese el valor máximo para la característica: 1.0

Para la característica instrumentalness, en el rango 0.75 a 1.0:
> Total de eventos o reproducciones: 4191
> Total de artistas únicos: 1769
------------------------------------------------------
> Tiempo y memoria consumidos:
> Tiempo [ms]:  97.108  ||  Memoria [kB]:  27.947
```

Prueba con los archivos -small.csv

```
----------------------------------------------  Requerimiento #1

Ingrese la característica de contenido: instrumentalness
Ingrese el valor mínimo para la característica: 0.75
Ingrese el valor máximo para la característica: 1.00

Para la característica instrumentalness, en el rango 0.75 a 1.00:
> Total de eventos o reproducciones: 41104
> Total de artistas únicos: 5287
------------------------------------------------------
> Tiempo y memoria consumidos:
> Tiempo [ms]:  1038.159  ||  Memoria [kB]:  29.689
```

Prueba con los archivos -5pct.csv

Requerimiento 2

Se quiere encontrar las pistas en el sistema de recomendación que puedan utilizarse en una fiesta que se tendrá próximamente. Se desea encontrar las canciones que tengan en cuenta las variables Energy y Danceability. El usuario podrá indicar los valores (rangos) para esos parámetros.

Implementado por estudiante A (Andrés Vargas).

Entradas:

- El valor mínimo de la característica Energy.
- El valor máximo de la característica Energy.
- El valor mínimo de la característica Danceability.
- El valor máximo de la característica Danceability.

Salidas:

- El total de pistas únicas (sin repeticiones).
- La información de 5 pistas seleccionadas aleatoriamente (incluya los valores Energy y Danceability para cada uno).

Análisis de complejidad:

Se determinó que el análisis de complejidad en notación O es:

$$O(elementos \cdot canciones)$$

Donde elementos y canciones son variables del siguiente ciclo en el model:

```
for elementos in lt.iterator(lista):
    for canciones in lt.iterator(elementos['eventos']):
        if (float(canciones["energy"])>= min_Energy and float(canciones["energy"])<=max_Energy):
            track_ID=canciones["track_id"]
            if int(lt.isPresent(lista_tracks, track_ID))== 0:
                lt.addLast(lista_tracks, track_ID)
                lt.addLast(cancion_completa, canciones)
            else:
                lt.addLast(cancion_completa, canciones)
```

Pruebas:

A continuación se presentan los resultados de dos pruebas ejecutadas con subconjuntos de datos de distintos tamaños (-small.csv y -5pct.csv respectivamente). Adicionalmente, se incluyen los resultados de consumo de memoria y tiempo en la ejecución.

```
-------------------------------------------------  Requerimiento #2  ------------------------------------------------------

Ingrese el valor mínimo para Energy: 0.5
Ingrese el valor máximo para Energy: 0.75
Ingrese el valor mínimo para Danceability: 0.75
Ingrese el valor máximo para Danceability: 1.0
Energy is between  0.5  and  0.75
Energy is between  0.75  and  1.0
Total of unique tracks in events:  1703
Track  1  :  4d5350356d40422fc420ad611a84be46 with energy of  0.683  and danceability of  0.75
Track  2  :  ad75b9fb76f0c746e8032f2bd9512c13 with energy of  0.574  and danceability of  0.75
Track  3  :  8b1eb9a23863b95bdceef884e65b73a1 with energy of  0.506  and danceability of  0.75
Track  4  :  e6a39acef1b25259da38b6cb9f89b85b with energy of  0.562  and danceability of  0.75
Track  5  :  81c6b4992f06535ad5a5284995d0d604 with energy of  0.699  and danceability of  0.75
```

Prueba con los archivos -small.csv

```
-------------------------------------------------  Requerimiento #2  ------------------------------------------------------

Ingrese el valor mínimo para Energy: 0.5
Ingrese el valor máximo para Energy: 0.75
Ingrese el valor mínimo para Danceability: 0.75
Ingrese el valor máximo para Danceability: 1.0
Energy is between  0.5  and  0.75
Energy is between  0.75  and  1.0
Total of unique tracks in events:  5823
Track  1  :  e6a39acef1b25259da38b6cb9f89b85b with energy of  0.562  and danceability of  0.75
Track  2  :  6453cd03461ded519fc4cb0f89167bd6 with energy of  0.741  and danceability of  0.75
Track  3  :  e56e8b8156a125cca7078f4dee4e7266 with energy of  0.723  and danceability of  0.75
Track  4  :  d398630293291a52915878935fd6f002 with energy of  0.558  and danceability of  0.75
Track  5  :  6c01c2a2541bb598cb8a8a2f8b7ebbb2 with energy of  0.69  and danceability of  0.75
```

Prueba con los archivos -5pct.csv

Requerimiento 3

Se desea conocer las pistas en el sistema de recomendación que podrían ayudar a los usuarios en su periodo de estudio. Para este fin se debe tener en cuenta las variables Instrumentalness y Tempo.

Implementado por estudiante B (María Alméciga).

Entradas:

- El valor mínimo del rango para Instrumentalness.
- El valor máximo del rango para Instrumentalness.
- El valor mínimo del rango para el Tempo.
- El valor máximo del rango para el Tempo.

Salida:

- El total de pistas únicas (sin repeticiones).
- La información de 5 pistas seleccionadas aleatoriamente (incluyendo sus valores de Instrumentalness y Tempo).

Análisis de complejidad:

Se determinó que el análisis de complejidad en notación O es:

$$O(value \cdot event)$$

Donde value y event son variables del siguiente ciclo en el model:

```
for value in lt.iterator(instr_filtrado):
    for event in lt.iterator(value['eventos']):
        if ((float(event["tempo"])) >= min_tempo) and ((float(event["tempo"])) <= max_tempo):
            elem = [event['track_id'], event['instrumentalness'], event['tempo']]
            contiene = lt.isPresent(tracks_list, elem)
            if contiene:
                pass
            else:
                lt.addLast(tracks_list, elem)
```

Pruebas:

A continuación se presentan los resultados de dos pruebas ejecutadas con subconjuntos de datos de distintos tamaños (-small.csv y -5pct.csv respectivamente). Adicionalmente, se incluyen los resultados de consumo de memoria y tiempo en la ejecución.

```
--------------------------------------------------  Requerimiento #3  ------------------------

Ingrese el valor mínimo para Instrumentalness: 0.6
Ingrese el valor máximo para Instrumentalness: 0.9
Ingrese el valor mínimo para Tempo: 40.0
Ingrese el valor máximo para Tempo: 60.0

Para un Instrumentalness entre 0.6 y 0.9, y un Tempo entre 40.0 y 60.0:
> Total de Tracks únicos: 9
> Track 1: ac136b1a3cae741a26ca4eecaf163dcd con Instrumentalness de 0.846 y Tempo de 59.285
> Track 2: d89902d1d1677eeb3985a7a41cf6e63e con Instrumentalness de 0.817 y Tempo de 43.425
> Track 3: 7d323c9286f68ac35b86392b8a329bc2 con Instrumentalness de 0.76 y Tempo de 58.993
> Track 4: 4978ab7dad5dc1d843af6b3b422a8692 con Instrumentalness de 0.755 y Tempo de 56.228
> Track 5: 2633956017d8c9adc52ef9eb32c963f4 con Instrumentalness de 0.757 y Tempo de 56.441
-----------------------------------------------------
> Tiempo y memoria consumidos:
> Tiempo [ms]:  9.197  ||  Memoria [kB]:  15.059
```

Prueba con los archivos -small.csv

```
--------------------------------------------------  Requerimiento #3  ------------------------

Ingrese el valor mínimo para Instrumentalness: 0.6
Ingrese el valor máximo para Instrumentalness: 0.9
Ingrese el valor mínimo para Tempo: 40.0
Ingrese el valor máximo para Tempo: 60.0

Para un Instrumentalness entre 0.6 y 0.9, y un Tempo entre 40.0 y 60.0:
> Total de Tracks únicos: 29
> Track 1: 1422d0c07151e8a53805d9613c68e68c con Instrumentalness de 0.721 y Tempo de 52.38
> Track 2: 4c4ac8eca60fe51d1fe12c908f179e11 con Instrumentalness de 0.879 y Tempo de 54.664
> Track 3: 7d323c9286f68ac35b86392b8a329bc2 con Instrumentalness de 0.76 y Tempo de 58.993
> Track 4: d920f9a82a0d6a23440cbd60142878da con Instrumentalness de 0.871 y Tempo de 56.881
> Track 5: 99eaf211317d158a01f6aa7d2d5604bd con Instrumentalness de 0.725 y Tempo de 52.217
-----------------------------------------------------
> Tiempo y memoria consumidos:
> Tiempo [ms]:  71.400  ||  Memoria [kB]:  31.197
```

Prueba con los archivos -5pct.csv

Requerimiento 4:

Dada la relación entre Tempo y género musical, se desea conocer cuántas canciones y artistas se tienen por cada género.

Entradas:

- Lista de géneros que se desea buscar.
- Nombre único del nuevo género musical, en caso de que se desee crear.
- Valor mínimo del Tempo para dicho género.
- Valor máximo del Tempo para dicho género.

Salidas:

- El total de eventos de escucha o reproducciones.
- El total de eventos de escucha o reproducciones en cada género.
- El número de artistas únicos (sin repeticiones) en cada uno de los géneros, y el ID de los 10 primeros artistas.

Análisis de complejidad:

Se determinó que el análisis de complejidad en notación O es:

$$O(contador \cdot value \cdot event)$$

Donde contador, value y event son variables del siguiente ciclo en el model:

```
while contador <= lt.size(generos):
    artists_list = lt.newList(datastructure='ARRAY_LIST')
    total_events = 0
    genero = lt.getElement(generos, contador)
    tempo_filtrado = om.values(tempo_total, genero[1], genero[2])
    for value in lt.iterator(tempo_filtrado):
        for event in lt.iterator(value['eventos']):
            total_events += 1
            val = 0
            om.put(events_map, event['id'], val)
            elem = event['artist_id']
            contiene = lt.isPresent(artists_list, elem)
            if contiene:
                pass
            else:
                lt.addLast(artists_list, elem)
    total_artists = lt.size(artists_list)
    ten_artists = lt.subList(artists_list, 1, 10)
    nuevo = {}
    nuevo["info"] = genero
    nuevo["artistas"] = total_artists
    nuevo["eventos"] = total_events
    nuevo["10artistas"] = ten_artists
    lt.addLast(nuevos_generos, nuevo)
    contador += 1
total_tot_eventos = om.size(events_map)
```

Pruebas:

A continuación se presentan los resultados de dos pruebas ejecutadas con subconjuntos de datos de distintos tamaños (-small.csv y -5pct.csv respectivamente). Adicionalmente, se incluyen los resultados de consumo de memoria y tiempo en la ejecución.

```
------------------------------------------------  Requerimiento #4  ----------------------

Si desea añadir un género existente, ingrese 0. Si desea crear uno nuevo, ingrese 1: 0
Ingrese el nombre del género: reggae
Si desea añadir más generos, ingrese 1. De lo contrario, ingrese 0 para comenzar a calcular: 1
Si desea añadir un género existente, ingrese 0. Si desea crear uno nuevo, ingrese 1: 0
Ingrese el nombre del género: hip-hop
Si desea añadir más generos, ingrese 1. De lo contrario, ingrese 0 para comenzar a calcular: 1
Si desea añadir un género existente, ingrese 0. Si desea crear uno nuevo, ingrese 1: 0
Ingrese el nombre del género: pop
Si desea añadir más generos, ingrese 1. De lo contrario, ingrese 0 para comenzar a calcular: 0
El total de eventos para los siguientes géneros fue de: 41329
=======  REGGAE  =======
Para reggae con Tempo entre 60.0 y 90.0 BPM:
Se encontraron 7564 eventos, y 2455 artistas únicos. Algunos de ellos son:
> 00bf7b2909ce4b6e52e9615e10743991
> e55e9a3fe7f853005040898e9a91924a
> aea5c185fec72185949a167c06586bf7
> fb9f836f4a485a058e72d40c1a2a765b
> 741419962f8c758bd1ea7b46d11ea347
> 1f49f770adc6c84629f50ce3ca2a2109
> c9f435587275fd49900e35bc3ca37e52
> d7bb728b6e04d4cb502cfda6cd7affe7
> 2d0b46eadfa603abe3ef9ebe2411cc1a
> 31db0d2a59a6cfdc47ad4f405a5136bd
=======  HIP-HOP  =======
Para hip-hop con Tempo entre 85.0 y 115.0 BPM:
Se encontraron 20036 eventos, y 4977 artistas únicos. Algunos de ellos son:
> c08167b71988c7014370b14bb248c2aa
> c4d71121cff3a245ce181a703ad85147
> ea63947017cf6a0648b9e86c2107b7e1
> b99077a40f9595c0e74eebd74fef69f1
> 8e323bbdabda8a309597e4ad49787d69
> 24953964540d417bda93baf897208099
> e52fe5c7599e508e8ea1bc30e1d8205a
> 275dbab66a3ffc9eb7d8706af0de11c8
> 8be5a73514fb56bfa45d94574391401c
> 4e50678324968101b6728f2fa02625ea
=======  POP  =======
Para pop con Tempo entre 100.0 y 130.0 BPM:
Se encontraron 26539 eventos, y 5891 artistas únicos. Algunos de ellos son:
> f9631c413441059f1d2932bebc36db4a
> 2248a074084b12d4f4caffa2149c5888
> d576f896b05ba55bea45bbbcc22f8f62
> 509021d12408d6d09ac3306b656720f3
> 6ba922b2388c76cb30b6510db318ad56
> f83854b623c728117fb2499199adab31
> cce975997b437b6b946fe4ca3e830d0a
> d5d4e17dcb390a1eb16009bd47b3103b
> 6e307e6fea16b81ecf00ab089ccd4fa6
> 0eb86ff5c7d7a833d7c87aaa29e550ce
------------------------------------------------------
> Tiempo y memoria consumidos:
> Tiempo [ms]:  116789.105  ||  Memoria [kB]:  43.206
```

Prueba con los archivos -small.csv

```
-------------------------------------------------  Requerimiento #4  ----------------------------

Si desea añadir un género existente, ingrese 0. Si desea crear uno nuevo, ingrese 1: 0
Ingrese el nombre del género: reggae
Si desea añadir más generos, ingrese 1. De lo contrario, ingrese 0 para comenzar a calcular: 1
Si desea añadir un género existente, ingrese 0. Si desea crear uno nuevo, ingrese 1: 0
Ingrese el nombre del género: hip-hop
Si desea añadir más generos, ingrese 1. De lo contrario, ingrese 0 para comenzar a calcular: 1
Si desea añadir un género existente, ingrese 0. Si desea crear uno nuevo, ingrese 1: 0
Ingrese el nombre del género: pop
Si desea añadir más generos, ingrese 1. De lo contrario, ingrese 0 para comenzar a calcular: 0
El total de eventos para los siguientes géneros fue de: 398913
=======  REGGAE  =======
Para reggae con Tempo entre 60.0 y 90.0 BPM:
Se encontraron 74256 eventos, y 6704 artistas únicos. Algunos de ellos son:
> b30194a9c40c76d5e406f9cfbb8e8a97
> 8ca07ef0cad9c33fe41da33f9dea039e
> 805474443dde52c181166e52081a1710
> 00bf7b2909ce4b6e52e9615e10743991
> e55e9a3fe7f853005040898e9a91924a
> aea5c185fec72185949a167c06586bf7
> 0df85b1c62ac89de45ddfa2f285a2c5b
> fb9f836f4a485a058e72d40c1a2a765b
> 741419962f8c758bd1ea7b46d11ea347
> f7b3d05b2fdae6bf642621f1925478fd
=======  HIP-HOP  =======
Para hip-hop con Tempo entre 85.0 y 115.0 BPM:
Se encontraron 193566 eventos, y 12185 artistas únicos. Algunos de ellos son:
> 0cb77b09ec7fc66dfe61ce34845ff36b
> a94fb1545d9e42280eba595e883f9daf
> 9a5a845a3b9257f30a789bf560803061
> c08167b71988c7014370b14bb248c2aa
> 6e879b76967bc27491e96962c747cba2
> 49f34c755408a89228f78967771c175f
> c71df3dba85c152bdcf47002256b40b4
> d37c550d781bfb93b0088a2470598bcf
> 7eff327d4452faaa0db443643ae29736
> abed0ab82b52c37eb10a31aff80116b6
=======  POP  =======
Para pop con Tempo entre 100.0 y 130.0 BPM:
Se encontraron 254465 eventos, y 14013 artistas únicos. Algunos de ellos son:
> e4b90932c17289b0972caac2fe8fbdd7
> 2248a074084b12d4f4caffa2149c5888
> 0b9c13d3381af6edddda9e00cc9bc565
> 1438d6eb840299e572cca53254754422
> 2e03262783a3d5f3766633ad2894ea1b
> f9631c413441059f1d2932bebc36db4a
> 9ee93f85bfb4262c7c06c2e3d223eeef
> d576f896b05ba55bea45bbbcc22f8f62
> 509021d12408d6d09ac3306b656720f3
> 149cff97c4a806900e938cda6a083346
-------------------------------------------------------
> Tiempo y memoria consumidos:
> Tiempo [ms]:  2440002.515  ||  Memoria [kB]:  34.165
```

Prueba con los archivos -5pct.csv

Requerimiento 5

Dado un rango de horas indicar el género de música más escuchado en dicho rango, teniendo en cuenta todos los días disponibles e informar el promedio para cada uno de los valores de análisis de sentimiento, en las canciones de dicho rango.

Entradas:

- El valor mínimo de la hora del día.
- El valor máximo de la hora del día.

Salida:

- Género más referenciado en el rango de horas.
- Para dicho género, calcular el valor promedio Vader de las pistas que contiene.

Análisis de complejidad:

Se determinó que el análisis de complejidad en notación O es:

$$O(value \cdot event)$$

Donde value y event son variables del siguiente ciclo en el model:

```
for value in lt.iterator(eventos_filtrado):
    for event in lt.iterator(value['eventos']):
        key = str(event['tempo']) + "," + str(event['user_id']) + "," + str(event['track_id']) + "," + str(event['created_at'])
        om.put(map_tempo, key, 0)
        generos_event = determinarGeneros(event, generos)
```

Pruebas:

A continuación se presentan los resultados de dos pruebas ejecutadas con subconjuntos de datos de distintos tamaños (-small.csv y -5pct.csv respectivamente). Adicionalmente, se incluyen los resultados de consumo de memoria y tiempo en la ejecución.

```
----------------------------------------------  Requerimiento #5  ----------------------------

Ingrese el valor mínimo de hora (únicamente en formato hh:mm:ss): 07:15:00
Ingrese el valor máximo de hora (únicamente en formato hh:mm:ss): 09:45:00

Para el rango de horas 07:15:00 a 09:45:00 el género más escuchado fue m con 4125 reproducciones
El promedio Vader de 10 pistas aleatorias en dicho género fue:
> Track 721b1454fd6703249921f1723c713d06 con 1 hashtags, y vader 0.6
> Track 5fbbcec859c70678436141a6e974e6f5 con 1 hashtags, y vader 0.6
> Track 24c5f4b87d30bbce90a3d5c2200e13b3 con 1 hashtags, y vader 0.6
> Track b36847e3dbb686f0b666a63effa61ac3 con 0 hashtags, y vader 0
> Track e46b6124d25e5158a8b76ba00029e173 con 1 hashtags, y vader 0.6
> Track 27cf05e4fe3e88d2e82faed370bca6f1 con 1 hashtags, y vader 0.6
> Track e7fb23b270e851a97fb98677ec908032 con 1 hashtags, y vader 0.6
> Track 0c06bec0af31b75e12db6575f0681213 con 1 hashtags, y vader 0.6
> Track 1827a1398750b0210f5f2394eb56fe47 con 1 hashtags, y vader 0.6
> Track 0e0063f14281532fc06850c46a04f8d4 con 0 hashtags, y vader 0

------------------------------------------------------
> Tiempo y memoria consumidos:
> Tiempo [ms]:  353.288  ||  Memoria [kB]:  25.226
```

Prueba con los archivos -small.csv

```
----------------------------------------------  Requerimiento #5  ----------------------------

Ingrese el valor mínimo de hora (únicamente en formato hh:mm:ss): 07:15:00
Ingrese el valor máximo de hora (únicamente en formato hh:mm:ss): 09:45:00
Para el rango de horas 07:15:00 a 09:45:00 el género más escuchado fue m con 39056 reproducciones
El promedio Vader de 10 pistas aleatorias en dicho género fue:
> Track c7f1cebeb18a2874bfc7529d63f86149 con 1 hashtags, y vader 0.6
> Track f651618ac21f063c78cba1c6c0f2a5bc con 1 hashtags, y vader 0.6
> Track 4b64716bb8b1b82b42f5863f17d8c081 con 0 hashtags, y vader 0
> Track 34940d594586bb6f814b498d2db4b03a con 1 hashtags, y vader 0.8
> Track b6c71104c4f174979be8712bf15b7e84 con 1 hashtags, y vader 0.6
> Track 11ce191f0342e8ae8198eb7c55d956cc con 0 hashtags, y vader 0
> Track 3d93af795319e50b1190342eb926484b con 0 hashtags, y vader 0
> Track 34940d594586bb6f814b498d2db4b03a con 1 hashtags, y vader 0.6
> Track cc54612c9c4956ee0201a2ddcbe71ba4 con 0 hashtags, y vader 0
> Track 655a654487f8c6a97ff3a48230ff1a68 con 1 hashtags, y vader 0.6
------------------------------------------------------
> Tiempo y memoria consumidos:
> Tiempo [ms]:  4140.337  ||  Memoria [kB]:  53.500
```

Prueba con los archivos -5pct.csv