Universidad Nacional Autónoma de México

Facultad de Ciencias

Proyecto 3: Sistema de Recomendaciones

Galeana Araujo Emiliano 314032324
Fernández Del Valle Diego Marcial 314198039
Reyes Granados Naomi Itzel 314141341
Vázquez Rodríguez Jésica 720034837

Inteligencia Artificial

Gustavo De la Cruz Martínez
Estefanía Prieto Larios
Osvaldo Baruch Acevedo Badillo
Rodrigo Eduardo Colín Rivera

November 4, 2019

Contents

1 Introducción

¿Cómo razonamos los humanos? De manera simple podemos decir que nos dan datos que de alguna manera procesamos y nos hacen generar nuevo conocimiento, cuando hablamos de procesar nos referimos a aprender y entender, actividad que cada ser humano tiene de una manera particular, hay quienes entre más estímulos visuales tengan mejor aprendizaje realiza y otros que prefieren los estímulos auditivos. Pues bien, ¿Cómo podemos simular esta actividad en un agente? ¿Cómo hacemos para que el agente genere nuevo conocimiento?

1.1 Esquemas de Representación

Como primer objetivo para lograr esto es diferenciar información de conocimiento.

- **Información**: Conjunto de datos básicos, sin interpretar, que se obtienen como entrada del sistema.
- **Conocimiento**: Conjunto de datos de primer orden, que modelan de forma estructurada la experiencia que se tiene sobre un cierto dominio o que surgen de interpretar los datos básicos.

Así podemos ver que la información es un montón de datos, mientras que el conocimiento son datos interpretados, o bien "procesados". Entonces de manera intuitiva podemos decir que: Conocimiento = Información + Interpretación.
Ahora nos tocaría pensar cuál sería la manera correcta de guardar la información, que es lo que nos interesa, o bien cómo procesarla, qué va a hacer el agente con toda la información que recopiló; seguramente esto suene bastante alejado de algo que podamos hacer ya que estamos limitados por razones biológicas y/o neurofisiológicas, por lo que buscamos modelos que simulen la adquisición, estructuración y manipulación del conocimiento y que nos permitan crear sistemas artificiales inteligentes.
Comencemos con la representación, queremos un instrumento para codificar la realidad en un ordenador, o llamaremos *Esquema de Representación*, el cual tiene que contener información y ser capaz de convertirlo y obtener nuevo conocimiento, para esto vamos a separar dos partes de nuestro esquema:

- **Parte estática**: Va a estar conformada por la información del problema, desde la estructura de datos que codifica al problema con operaciones de crear, modificar, eliminar y consultar elementos, así como una semántica para definir la relación entre la realidad y la representación escogida.
- **Parte dinámica**: Está conformada de una estructura de datos con el conocimiento acerca del problema, así como procedimientos que permiten interpretar, controlar y adquirir nuevo conocimiento.

Es claro ver que nunca podremos representar en su totalidad al entorno ya sea por la cantidad de detalles que tiene o bien por lo cambiante que es.
¿Qué será lo importante a tomar en cuenta cuando hablamos de sistemas de representación? La habilidad para representar todas las clases de conocimiento que son necesarias, lo denominaremos *Adecuación Representacional*. Por otro lado podemos pensar en la habilidad de manipular estructuras de representación para generar nuevas estructuras que correspondan a nuevos conocimientos inferidos de los anteriores, a esto le llamaremos *Adecuación Inferencial*. Estas características están ligadas directamente con la representación de los datos (información y conocimiento), entonces podemos preguntarnos por lo necesario en la manera de ocupar dichos datos, o bien del uso del sistema de representación. Empecemos con lo mas directo, ¿Qué

necesitamos que haga nuestro sistema de referencia con la información/conocimiento que haya obtenido? Inferir y crear nuevo conocimiento, así nuestro primer punto importante a tomar en cuenta será la capacidad para incorporar información adicional a la estructura de representación, a lo que llamaremos *Eficiencia Inferencial*; de igual manera necesitamos que tenga la capacidad de incorporar fácilmente nueva información, a lo que llamaremos *Eficiencia en la Adquisición*.

1.2 Representación e inferencia de Conocimiento

Al inicio de la introducción hablamos de la diversidad de los humanos en adquirir conocimiento, justo esta parte habla de la diversidad pero en sistemas computarizados.

Existen dos tipos de representar e inferir conocimiento, el declarativo y procedimental. Empecemos con el más simple y con menos ramificaciones, el procedimental. En este se necesita la especificación del uso que va a tener el conocimiento que se está almacenando. Podemos pensarlo como un sistema estructurado que usa funciones para la creación de nuevo conocimiento y las reglas de producción son de la forma: SI condición ENTONCES acción.
Ahora hablemos del conocimiento declarativo, el conocimiento se representa de forma independiente a su uso posterior, es decir un conocimiento que se almacena sin saber la utilidad por la cual se esta almacenando. De aquí tenemos varios tipos que desglosaremos de una manera breve a continuación.

- **Relacional**:La información se representa como un conjunto de relaciones expresables mediante tablas (se podría ver parecido a una base de datos relacional). Su motor de inferencia se basa en generar conocimiento a partir de información.
- **Heredable**:Se trata de construir un grafo tal que, los nodos son la información y las aristas/arcos son las relaciones. El mecanismo de inferencia es la herencia de propiedades y valores.
- **Inferible**:Conocimiento descrito mediante lógica, utilizando su semántica y operadores. Se usará Modus-Ponens para la inferencia.

1.3 Agentes basados en conocimiento

Ya tenemos cómo representar a nuestro entorno, pero como en cualquier problema de inteligencia artificial necesitamos un agente que interactué con el. Para esto utilizaremos agentes basados en conocimiento cuya primera característica es que el entorno se tiene que representar con esquemas de representación, es decir que tiene que ser capaz de inferir nuevo conocimiento.
Podemos generalizar el comportamiento de estos agentes de la siguiente manera:
Cada vez que el programa del agente sea invocado realiza dos acciones, primero DICE al esquema de representación lo que ha percibido; segundo, PREGUNTA al esquema de representación qué acción debe ejecutar.
Dicha acción será tomada a partir de las reglas de inferencia y conocimiento hasta el momento del mundo en el esquema de representación.
De manera directa podemos ver a nuestro agente basado en conocimiento como un agente basado en objetivos, donde no importa tanto el procesamiento de los datos sino llegar al objetivo deseado y por tanto podemos hablar de esquemas de representación en general y no tanto en cómo es su implementación (declarativo o procedimental).

1.4 Sistemas de recomendación

Nuestra finalidad es construir un sistema de recomendación, es decir un sistema capaz de poder recomendar algo dependiendo de la información que el agente pueda obtener; usaremos esquemas de representación y agentes basados en conocimiento.
Para poder hacer estas recomendaciones podemos pensar en dos maneras de hacerlo, la primera es de manera explícita, que el usuario nos de una valoración de las cosas y con esa información podamos recomendar cosas con características parecidas a la de mayor puntuación; la otra seria sacar esta evaluación de manera implícita viendo el comportamiento del usuario y nuevamente recomendar cosas con características semejantes. Claro que para la segunda se necesita que el usuario interaccione mucho con el sistema para poder hacer "deducciones" pertinentes.
Entonces nuestro esquema de representación tiene que regresar información tratando de maximizar el gusto del usuario por dicha recomendación. Dado que quien actúa es el agente podemos pensar que el mismo tendrá que ser basado en utilidades donde dicha utilidad será el agrado del usuario.
A continuación hablaremos de algunos algoritmos de recomendación de una manera breve.

- **Algoritmos de vecinos cercanos**: Utilizan toda la base de datos de elementos y usuarios para generar predicciones. Primero emplean técnicas estadísticas para encontrar a vecinos, es decir usuarios con un historial de valoraciones sobre los elementos similar al usuario actual. Una vez que se ha construido una lista de vecinos se combinan sus preferencias para generar una lista con los n elementos más recomendables para el usuario actual.

- **Algoritmos basados en elementos**: En lugar de buscar similitudes entre usuarios buscan cercanías entre elementos. El procedimiento consiste en seleccionar los elementos que un usuario determinado ha votado y después comprobar qué tan similar es cada uno del resto de los elementos del sistema, para terminar recomendando los más parecidos. Existen diversas formas de calcular las similitudes entre ellos, algunas que podemos mencionar son Basadas en Coseno, en Correlación, en Coseno Ajustado, etc.

- **Predictores "Slope-one"**: A la hora del cálculo de la predicción para un usuario U se tiene en cuenta tanto la información de usuarios que tienen en común la votación de algún elemento U como la información del resto de los elementos votados.

1.5 Métodos de evaluación de rendimiento

Evaluar los algoritmos resulta un poco ambiguo ya que las recomendaciones no siempre son las mismas y lo que se tiene que tomar en cuenta es bastante ambiguo. Sin embargo existen dos tipos de pruebas para evaluar el rendimiento: [2]

- **Métodos estadísticos**: se utiliza el error medio absoluto que mide la desviación de las recomendaciones predichas y los valores reales. Mientras más chico sea el error medio absoluto, mejor predicciones realiza el sistema.

- **Métricas de decisión**: se asume que el proceso de predicción tiene dos opciones posibles: o bien al usuario le gustará la recomendación o no. Se evalúa la efectividad del sistema de predicción teniendo en cuenta las respuestas de los usuarios. Los resultados dependen fuertemente del porcentaje de elementos relevantes que el usuario haya calificado.

2 Descripción del problema y justificación de utilización de sistema de recomendación

El problema planteado es que una persona desea escuchar música nueva dados sus géneros favoritos, y sus canciones favoritas de dichos géneros desea conocer canciones afines. El objetivo de nuestro sistema es que dados 3 géneros musicales y 5 canciones de dichos géneros un usuario pueda obtener nuevas canciones que sean afines a los géneros y canciones previamente mencionados. El problema es resuelto por medio de un sistema de recomendaciones ya que justamente el usuario desea escuchar nuevas canciones dadas otras que ya conoce, si parafraseamos esto podemos decir que el usuario quiere que le *recomienden* nuevas canciones a partir de las canciones que el ha escuchado y le gustan.

Dicho lo anterior nuestro perfil de usuario será una representación muy sencilla y modular, por cada ejecución el perfil estará compuesto por los 3 géneros y 5 canciones que el usuario desee seleccionar y con base en eso y nuestro conjunto de datos es que decidiremos que canciones recomendarles.

Ahora la composición de nuestro sistema es la siguiente: Primero tenemos nuestro conjunto de canciones, es con base en este que tomaremos todas nuestras decisiones de lo que vamos a recomendar, nuestro conjunto de datos está compuesto por los siguientes campos:

1. genre-El género al que está asociada la canción
2. artist-name- El nombre del artista que interpreta la canción
3. track-name- El nombre de la canción
4. track-id- El id con el que se almacena la canción en el dataset
5. popularity- El índice de poularidad de la canción en Spotify
6. acousticness- El índice que calcula la aguzdeza acústica de la canción
7. danceability- El índice que calcula que tan bailable es una canción
8. duration-ms- La duración de la canción en milisegundos
9. energy- Índice que calcula la energía que provee la canción
10. instrumentalness- Índice que calcula la cantidad de instrumental en la canción
11. key- Una llave para acceder a la canción
12. liveness- Índice que mide la viveza de la canción
13. loudness- Índice que mide la potencia en sonido de la canción
14. speechiness- Índice que representa un estimado del número de palabras en una canción
15. tempo- Índice que representa el tempo de la canción
16. time-signature- Índice que representa como se cuenta la canción
17. valence- Índice que representa la positividad musical de la canción

3 Modelado del problema

Para nuestro problema la modelación utilizada consiste en lo siguiente. Primero, dado que contamos con un dataset de un gran tamaño decidimos acotar por contenido nuestra búsqueda de tal forma que el primer paso es acotar la búsqueda a las canciones que cumplen con pertenecer a al menos uno de los 3 géneros, posteriormente si nos fijamos en el conjunto de datos los campos acousticness, danceability, energy, instrumentalness, liveness, speechiness, tempo y valence representan aspectos que describen a la canción de una forma incríblemente detallada desde un punto técnico así que decidimos hacer nuestra recomendación como un filtrado colaborativo sobre estos campos, otra propiedad de suma importancia es que todos estos campos se encuentran "estandarizados" ya que todos estos campos están acotados a un rango entre 0 y 1, así decidimos trabajar sobre el polítopo en el espacio generado por dichos campos y para completar nuestro filtrado colaborativo, nuestro vector de búsqueda será contemplado como el promedio de las 5 canciones provistas, por último calculamos la diferencia del resto de canciones a nuestro vector promedio y las que se encuentren mas cercanas, serán por razones obvias las de condiciones y características mas cercanas a las que consideramos en el perfil del usuario, para terminar devolvemos las 5 canciones que se encuentren a menor distancia de nuestro vector.

Esa es la descripción general de nuestro producto, mientras tanto nuestro modelado del perfil de usuario es, como lo habíamos planteado en el punto anterior es muy modular y sólo consiste en los géneros seleccionados, con los cuales hacemos nuestra primera selección de información y las canciones seleccionadas con las cuales continuamos nuestra búsqueda, finalizando nuestra búsqueda y la forma en que finalmente respondemos con las 5 opciones de recomendación que devolvemos al usuario.

4 Esquema de representación

Como explicamos en la introducción para un esquema de representación necesitamos tener claras dos partes, la estática y la dinámica. La estática es la que se encarga de la descripción del problema mientras que la dinámica de la forma de operar.

4.1 Descripción o Parte estática

Nuestra estructura de datos que codifica al problema es el data set, ahí tendremos la información necesaria para poder recomendar canciones a nuestro usuario. La semántica a ocupar serán vectores que codificaran las siguientes características de las canciones con valores numéricos: agudeza acústica, bailabilidad, energía, cantidad instrumental, viveza, potencia en sonido, numero de palabras, tiempo y positividad. Así si c es una canción, entonces $c = [acu, dan, ene, ins, liv, lou, spe, tem, val]$ respectivamente por sus siglas en inglés.

4.2 Forma de operar o Parte dinámica

Nuestras estructura para guardar conocimiento es un conjunto de vectores que representa a cada canción del data set; y un vector que represente el promedio de las características de las canciones que el usuario nos dio como base, este vector será el que nos ayude a saber que canción hay que recomendar (para facilidad de explicación le llamaremos vector inicial).
Nuestra forma de adquirir nuevo conocimiento será a base de un filtrado colaborativo donde vamos a recomendar una canción si y solo si la diferencia total del vector inicial con dicha está entre las 5 mas pequeñas. La diferencia total se obtendrá restando las características del vector inicial con las de la canción a evaluar, y sumando todas estas diferencias, es decir si c es una canción del data set e i es el vector inicial, su diferencia total la definimos como:
$difTotal = (c[0] - i[0]) + (c[1] - i[1]) + (c[2] - i[2]) + (c[3] - i[3]) + (c[4] - i[4]) + (c[5] - i[5]) + (c[6] - i[6]) + (c[7] - i[7]) + (c[8] - i[0])$

Dado que el conocimiento que estamos recolectando sabemos su propósito, nuestra forma de representación de este es procedimental.

4.3 Propiedades

Falta ver que las propiedades de un buen esquema de representaciones se cumplen.

- **Adecuación Representacional.**: Dado que nuestro sistema es procedimental, todo el conocimiento necesario se representa.
- **Adecuación Inferencial**: Nuestro sistema de inferencia nos asegura generar nuevo conocimiento basado en el que ya teníamos previamente.
- **Eficiencia Inferencial**: Si se desea agregar mas características a los vectores, fácilmente se puede agregar una nueva columna al dataset y agregar análogamente a los anteriores en la diferencia total.
- **Eficiencia en la adquisición**: Dado que nuestro sistema solo se ocupa una vez no se necesita ir incorporando nueva información así por vacuidad se cumple esta propiedad.

5 Agente

"Los agentes constituyen el próximo avance más significativo en el desarrollo de sistemas y pueden ser considerados como la nueva revolución en el software"[1].
Recordemos que podemos definir a un agente como una entidad que percibe y actúa sobre su entorno. En este caso, se trata sobre un agente basado en utilidad cuya función de utilidad es la que determina los puntajes de las recomendaciones a realizar. Aquellas canciones que tengan la menor diferencia con respecto a las elegidas por el usuario serán las recomendadas[3].

5.1 REAS

Rendimiento	Entorno	Actuadores	Sensores
Adecuación de la recomendación con respecto a elección del usuario	Dataset	Pedir recomendación	Lector de información que ingresa el usuario

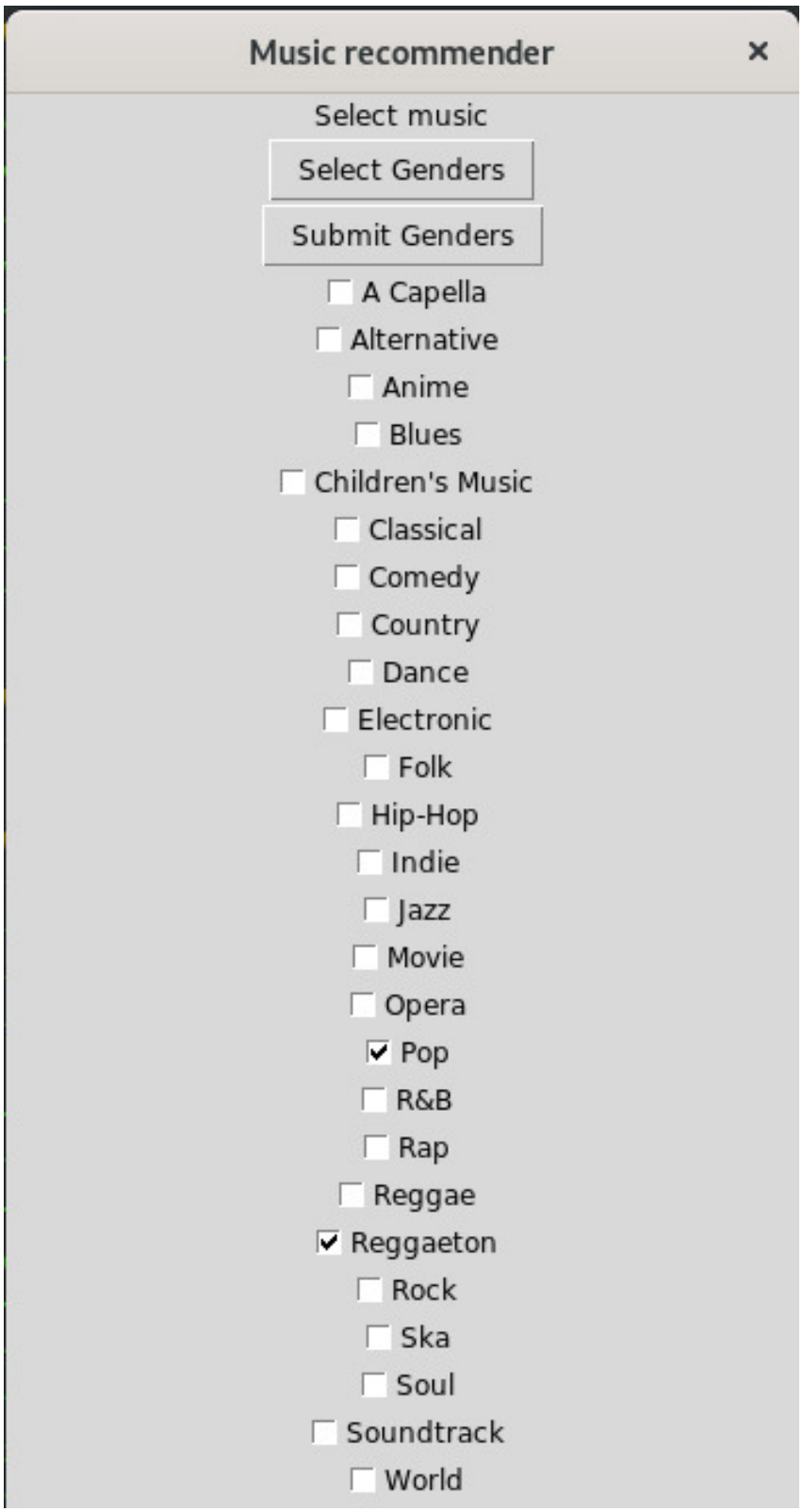
5.2 Propiedades del entorno

Descripción del entorno en el que estará el agente[4] :

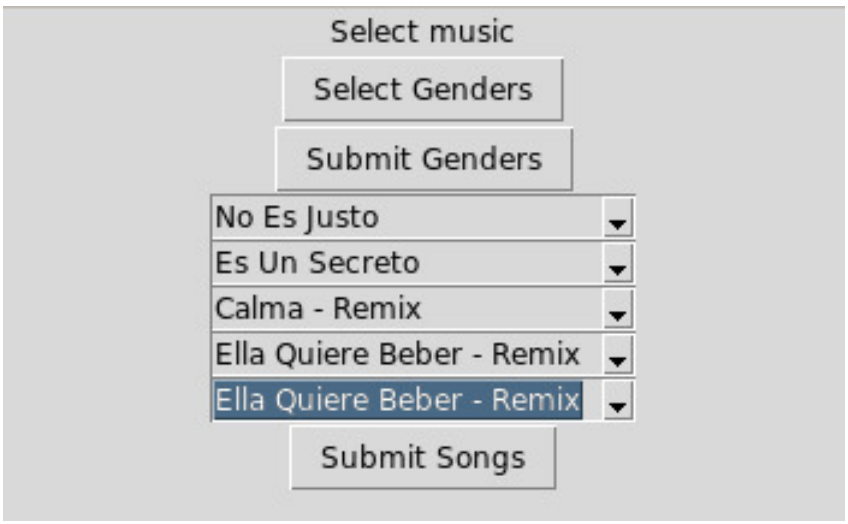
- **Totalmente observable**: se conocen todas las canciones del dataset en todo momento.
- **Agente simple**: dado que opera solo (que no coopera ni trabaja con otros agentes).
- **Determinista**: la recomendación se determina teniendo en cuenta las preferencias del usuario y el dataset. Dadas las mismas referencias y utilizando el mismo conjunto de datos, la recomendación realizada es la misma.
- **Estático**: El entorno (es decir el dataset y las preferencias obtenidas) no cambia mientras el agente decide qué recomendación realizar.
- **Discreto**: La cantidad de recomendaciones posibles a partir de dataset y de la entrada ingresada por el usuario son finitas.
- **Episódico**: El sistema de recomendaciones no está determinado por las recomendaciones realizadas anteriormente ya que pide al usuario determinados atributos y en base a eso realiza la recomendación.

5.3 Comportamiento mediante un ejemplo

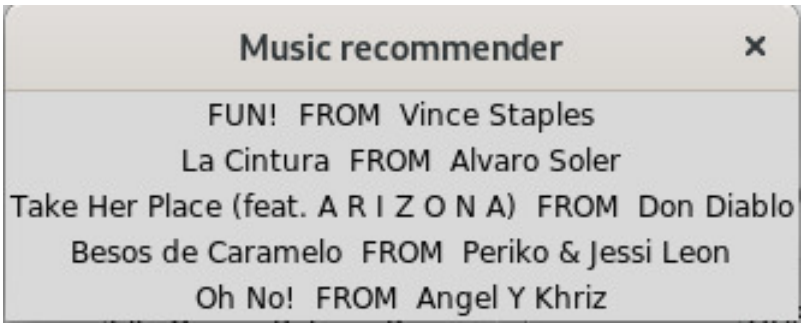
Procederemos a mostrar el comportamiento del agente con un ejemplo.
Primero elegimos los géneros, son los que en el equipo más se escuchan, por lo que se podría decir que son de los que más conocemos.



Continuamos y elegimos cinco canciones, son muy sonadas y en la aplicación de Spotify son fáciles de encontrar.



Y finalmente vemos los resultados de la recomendación. Los cuales son canciones que aunque no son de los mismos artistas, podríamos decir que tienen algún parecido.



6 Complejidad

6.1 Complejidad teórica

La complejidad teórica de nuestro algoritmo la podemos analizar de la siguiente manera, primero veamos que hacer la selección de los géneros nos toma tiempo $O(n)$ ya que en una pasada del dataset completo podemos definir los 3 géneros sobre los que vamos a hacer la búsqueda, después en tiempo lineal también podemos encontrar las 5 canciones que solicitamos para hacer nuestra recomendación, después obtener nuestro vector de búsqueda nos toma un número constante de operaciones (sólo aplicamos fórmulas) y después asignar la distancia entre nuestro vector resultante y las opciones que tenemos para recomendar igualmente está acotado por $O(n)$, por último el ordenar con respecto a la distancia obtenida lo hacemos en tiempo $O(nlog(n))$ por lo que la complejidad teórica de toda la ejecución de nuestro sistema de recomendaciones es $O(nlog(n))$.

Ahora para la complejidad en espacio tomando en cuenta que nuestra entrada es el dataset y que solo almacenamos una fracción de ese dataset nuestra memoria está acotada por $O(n)$.

6.2 Complejidad práctica

Para analizar el peor caso posible para nuestro sistema decidimos practicar una búsqueda donde no descartáramos ningún género y les dimos 5 canciones de géneros distintos, para este caso que repetimos 5 veces y vimos que en la ejecución que mas tiempo nos tomó dos minutos con 45 segundos, pero este caso resulta bastante irreal ya que al sujetar nuestra búsqueda preferentemente a 3 géneros el caso más extremo que pudimos generar fue menor a un minuto.

7 Conclusiones

1. El análisis de nuestro dataset, el trabajo y planeación hizo que encontraremos cómo aprovechar las características numéricas de nuestro conjunto de datos, sin dejar a un lado la característica más relevante a nuestros ojos que es el género de la música, la cual fue contemplada de una manera muy diferente pero que cobra la relevancia pertinente.

2. Tomando en consideración que nuestro sistema vive y toma decisiones sobre un conjunto de datos de aproximadamente 250,000 elementos nuestro sistema contesta de manera bastante eficiente y a pesar de que la complejidad es perfectible el tiempo de espera es bastante corto.

7.1 Adecuación de las recomendaciones

Algunas adecuaciones que pudieran explorarse sería el analizar si refinar los campos con los que hacemos el filtrado o incluso tomar otros distintos, ya que podría representar una mejoría para algunos casos, pero de manera general podemos decir que el análisis a pesar de ser perfectible es sumamente competente ya que los campos actuales son bastante completos desde el punto de vista técnico del caso de estudio actual que son las canciones.
Con respecto a las recomendaciones de canciones que realiza el sistema se adecúa a las canciones elegidas por el usuario en estilo de la música.

7.2 Ventajas

1. El sistema resuelve un problema real y que todos al menos alguna vez hemos pensado o en mayor o menor medida hemos convivido con el

2. La complejidad tanto teórica como práctica del sistema tomando en consideración la cantidad de datos con los que trabajamos.

3. Lo intuitivo tanto del desarrollo del problema, de la solución y de la versión final del sistema.

7.3 Desventajas

Una de las desventajas del sistema de recomendación planteado es que no tiene en cuenta el historial de elecciones de canciones que los usuarios fueron realizando. De esta forma, podría sumarse al cálculo de recomendaciones qué canciones le gustaron, y así poder realizar una recomendación que sea específica del usuario.

Otra desventaja es que la interfaz con la que entra en contacto el usuario, a pesar de que es completamente funcional, podría ser mas estética.

En una siguiente iteración podrían almacenarse las canciones que fueron recomendadas así como si al usuario le gustó o no la recomendación. De este modo, las recomendaciones serían cada vez más personalizadas.

References

[1] Nicholas Jennings.

[2] Sergio Manuel Galán Nieto. Filtrado colaborativo y sistemas de recomendación, 2007.

[3] P. Russell, S. Norman. Artificial intelligence a modern approach, 2009.

[4] P. Russell, S. Norman. Artificial intelligence a modern approach - section 2.4, 2009.