Trabajo de Fin de Máster
DPTOIA-IT
julio, 2016

Generación automática de listas de reproducción de música mediante técnicas de minería de datos

Creada por María del Pilar Arista Flores
Revisada por María Moreno García

VNiVERSiDAD
D SALAMANCA

Departamento de Informática y Automática
Universidad de Salamanca

Dra. Dña María Moreno García, profesora del Departamento de Informática y Automática de la Universidad de Salamanca,

HACE CONSTAR:

que Dña. María del Pilar Arista Flores con Pasaporte Nro. 6316413, ha realizado bajo mi tutela el Trabajo Fin de Máster que lleva por título "Generación automática de listas de reproducción de música mediante técnicas de minería de datos", con el fin de optar al título de Máster en Sistemas Inteligentes. Y para que surta los efectos oportunos, se firma el presente documento en Salamanca, a 15 de Julio de 2016.

María Moreno García

Declaración de Autoría para el Trabajo de Fin de Máster

Declaro que he redactado el Trabajo de Fin de Máster (TFM) titulado **Generación automátoca de listas de reproducción de música mediante técnicas de minería de datos** del Máster Universitario en Sistemas Inteligentes de la Universidad de Salamanca en el segundo semestre del curso académico **2015-2016** de forma autónoma, con la ayuda de las fuentes y la literatura citadas en la bibliografía, y que he identificado como tales todas las partes tomadas de las fuentes y de la literatura indicada, textualmente o conforme a su sentido.

Además, soy conocedor de que el citado TFM forma parte de los trabajos de investigación que lleva a cabo mi directora **Dra. Dña María Moreno García** dentro del grupo de investigación **MIDA (Grupo de Minería de Datos)** de la Universidad de Salamanca y, en consecuencia, comparto con ella la propiedad intelectual de los resultados alcanzados.

En Salamanca, 15 de julio de 2016.
Fdo.: María del Pilar Arista Flores

Revisado por:
Dr. - María Moreno García -

Aprobado en el Consejo de Departamento de

Resumen

Los sistemas de recomendación representan un medio eficaz de filtrado cuando existe mucha información disponible para el usuario. Los métodos de recomendación más conocidos y ampliamente desarrollados en los últimos años son los de filtrado colaborativo, en los que se trabaja en base a las preferencias de los usuarios, es decir, las relaciones del pasado usuarios-ítems se utilizan para encontrar usuarios con preferencias similares y hacer uso de dicha similitud para predecir los ítems en los que un usuario podría estar interesado.

Este trabajo se enfoca en los sistemas de recomendación de música, ya que debido al aumento de consumo musical, estos sistemas pueden resultar útiles para la creación de listas de reproducción de música relevante y novedosa en función de los gustos musicales personales del usuario. El gusto por la música es de carácter altamente subjetivo, por lo que esta tarea de investigación resulta desafiante.

En este sentido, se ha desarrollado un método híbrido de recomendación, que integra las técnicas de filtrado colaborativo y las basadas en contenido. Para ello, se ha utilizado como fuente, datos reales del consumo de música obtenidos mediante la API de *Last.fm*, para su posterior tratamiento y análisis. Asimismo, se han aplicado los algoritmos tradicionales y el propuesto, y se ha realizado un estudio comparativo entre éstos para evaluar su comportamiento en este contexto.

Los resultados revelaron que al aplicar conjuntamente los métodos de filtrado colaborativo y basado en contenido, la fiabilidad de las recomendaciones es mayor, que al utilizarlos de forma independiente.

Abstract

Recommender systems represent an effective filtering method when there is much information available to the user. The most widely known and developed method, in recent years, is the recommendation of collaborative filtering, which works based on users preferences, in other words, past users-items relations, are used to find users with similar preferences and make use of that similarity and predict the items in which a user might be interested.

This work focuses on music recommendation systems, due to the increased music consumption; these systems can be useful for creating playlists of relevant and innovative music depending on personal musical tastes of the user. The taste for music is highly subjective, so this research task is challenging.

In this sense, a hybrid recommendation method has been developed which integrates collaborative filtering techniques and content-based. To this end, real music data from *Last.fm* API have been used as the source for further processing and analysis. Moreover, the traditional algorithms and the proposed algorithm have been applied, and a comparative study between them has been conducted to assess their performance in this context.

The results showed that the joint by applying methods based on collaborative filtering and content-based, the reliability of the recommendations is greater than when used independently.

Índice

Índice de figuras

Indice de tablas

1. Introducción

1.1. Motivación

En los últimos años, la web se ha convertido en la principal fuente de títulos de música en formato digital. Los millones de pistas disponibles y la variedad de sitios web hace que la búsqueda de las canciones constituya un problema para los usuarios. Ésta es la razón del gran interés en el desarrollo de algoritmos de recomendación que permitan la generación automática de listas personalizadas de reproducción. De esa forma se puede ayudar a los consumidores de música a descubrir y filtrar la enorme cantidad de contenido musical disponible en el espacio digital.

Algunas compañías que ofrecen música como Apple y Pandora han desarrollado algoritmos de generación de listas de reproducción basados en el análisis de contenido realizado por expertos humanos y en filtrado colaborativo, respectivamente [15]. Sin embargo, no se conoce cómo funcionan estos algoritmos, cómo ordenan las canciones, ni la fiabilidad de las recomendaciones. Los métodos utilizados para la generación automática de listas de reproducción son muy variados, desde los que se basan simplemente en la popularidad o el género de las canciones, hasta los que se basan en el cálculo de alguna medida de similitud musical entre pares de canciones o de usuarios.

1.2. Objetivos

El propósito de este trabajo se centra en la generación de listas de reproducción personalizadas mediante la aplicación de técnicas de filtrado colaborativo. Para ello, se tendrá en cuenta el perfil de escucha de los usuarios y la similitud de preferencias con otros usuarios del sistema (cuyos datos se han recopilado desde la API de *Last.fm* [1]). Los métodos de filtrado colaborativo requieren las valoraciones del usuario sobre los productos a recomendar, en este caso las canciones de las listas de reproducción. Sin embargo, en las bases de datos sobre música no se dispone de valoraciones explícitas de los usuarios por lo que ha sido necesario recurrir a un tratamiento de los datos para obtener de forma implícita las preferencias de los usuarios.

1.3. Organización del Contenido

En el segundo capítulo de este trabajo, estado del arte, se describen los trabajos relacionados con la recomendación de música.

En el tercer capítulo, desarrollo del sistema de recomendación, se muestran las tecnologías usadas para la recolección de los datos. Asimismo, se presentan los diferentes pasos de preprocesamiento que se han seguido para lograr un conjunto de

[1] Last.fm: `http://www.last.fm/`

datos idóneo para aplicar las técnicas de recomendación. Adicionalmente, se incluye la implementación de los procesos de recomendación en *RapidMiner*.

En el cuarto capítulo, experimentación, cubre la aplicación de los procesos ya implementados en el capítulo anterior y la validación de sus resultados.

El quinto capítulo, conclusiones y trabajos futuros, muestra las conclusiones relativas a los resultados del estudio y el análisis de los retos y problemas que surgieron a lo largo del trabajo. En este capítulo se presentan sugerencias para el trabajo futuro.

2. Estado del Arte

En este apartado se hará una revisión de trabajos relacionados a la recomendación de música.

2.1. Sistemas de Recomendación

La razón por la que las personas podrían estar interesadas en usar un sistema de recomendación es porque está relacionada con la dificultad de evaluar todas las posibles opciones cuando se dispone de muchos ítems para escoger, en un tiempo corto y limitado. Un sistema de recomendación facilita esta tarea a los usuarios debido a que realiza un filtrado personalizado de la información.

Los sistemas de recomendación son un tipo específico de filtro de información cuyo objetivo es mostrar ítems al usuario que le sean relevantes o de interés. Se entiende por filtro de información, a un sistema que elimina información no deseada de un flujo de información de forma automática o semiautomática para ser presentada a los usuarios. Si bien los sistemas de recomendación buscan proporcionar información relevante al usuario, no se debe confudir con el concepto de la búsqueda de la información. Mientras que el proceso de filtrado en la búsqueda de información es la de encontrar información, en la recomendación, este proceso se relaciona con la acción de eliminar información irrelevante en el conjunto de datos que se recolecta de acuerdo al perfil del usuario [2].

Según Xiao y Benbasat [23], los sistemas de recomendación son agentes de software que obtienen los intereses y preferencias de los consumidores individuales y hacen recomendaciones al respecto. Tienen el potencial para apoyar y mejorar la calidad de las decisiones que los consumidores hacen mientras buscan y seleccionan productos en línea.

Adicionalmente, Celma [6] menciona que el problema de recomendación se puede dividir en dos fases, una es la predicción de las calificaciones de los ítems y la segunda, la recomendación de una lista de N ítems. El propósito principal de la primera es predecir la valoración de los ítems que le puedan gustar a un usuario, considerando que estos ítems no están incluidos a la lista de ítems que posee el usuario. En cambio, el objetivo de la segunda, es proporcionar una lista clasificada (*ranking*) de ítems que le gustarán al usuario. Para ello se consideran las calificaciones predichas de los ítems no consumidos y así se deriva la lista clasificada de dichos ítems, de los cuales se recomiendan los N primeros al usuario activo.

A continuación se mencionan las principales categorías de sistemas de recomendación [12]:

- **Recomendación Colaborativa**: Se puede resumir en la frase *"Enséñame lo que es popular entre mis vecinos"*. También llamado Filtrado colaborativo (*Collaborative filtering*), mediante el cual se recomiendan los ítems que les gustan a los usuarios similares ("vecinos") al usuario activo. Un perfil de usuario se

construye a partir de los ítems que han sido valorados por el usuario, por tanto, la similitud de gustos de los usuarios se deduce de sus valoraciones anteriores. Es decir, se asume que si los usuarios han compartido algunos de sus intereses en el pasado, tendrán gustos similares en el futuro. La entrada que recibe este sistema es una matriz de usuarios-ítems, y el resultado va a depender del objetivo buscado. Puede devolver la predicción del *rating* (valoración) de cada ítem o puede devolver una lista de N ítems recomendados que no han sido vistos aún por el usuario activo. Aunque es la técnica más empleada en aplicaciones comerciales, todavía tiene que superar los problemas de escalabilidad y arranque en frío que limitan su rendimiento.

- **Recomendación basada en el contenido (*Content based*)**: Se puede describir mediante la frase *"Muéstrame más de lo que ya me ha gustado"*. En estos sistemas los perfiles de usuario se construyen a partir de las características de los ítems que un usuario ha valorado muy positivamente. Se emparejan aquellos ítems que mejor cumplan las preferencias del usuario y que aún no han sido probados por él. La tarea de esta técnica es la de aprender las preferencias del usuario y recomendar aquellos elemenos que son similares a sus gustos.

- **Recomendación basada en el conocimiento (*Knowledge based*)**: *"Enséñame lo que se adapta a mis necesidades"* es la frase que se ajusta a esta categorías de métodos. Esta técnica se usa cuando no se tiene historial en el caso de los usuarios nuevos o cuando los ítems tienen un bajo número de valoraciones por ser productos que se han introducido recientemente en el sistema. Son recomendadores que están más centrados en el dominio de la aplicación.

- **Recomendación híbrida**: Esta categoría combina dos o más tipos de las técnicas anteriormente mencionadas. Su propósito es aprovechar las ventajas y evitar los incovenientes de dichos métodos para obtener "mejores" recomendaciones.

2.2. Métodos de Filtrado Colaborativo

Los métodos de filtrado colaborativo surgieron con el objetivo de aportar mayor fiabilidad a las recomendaciones utilizando información relativa a las preferencias de cada usuario. Se basa en la premisa de que si un usuario ha tenido un comportamiento similar (en cuanto a las valoraciones de ítems) a otro, entonces ambos deben tener similares preferencias con respecto a otros ítems [7]. Por lo tanto, en estos métodos el rol que juegan los usuarios es importante, pues son ellos quienes valoran la calidad de los ítems y determinan la preferencia sobre éstos. De manera que la solución no se efectúa analizando los ítems, sino estudiando las valoraciones que los usuarios han realizado.

Desde el punto de vista de Sarwar et al. [19], el filtrado colaborativo considera un conjunto de m usuarios $\mathrm{U} = \{u_1, u_2, ..., u_m\}$, y otro de n ítems $\mathrm{I} = \{i_1, i_2, ..., i_n\}$. Además, cada usuario u_i posee una lista de k valoraciones de un conjunto de ítems I_{u_i} que ha realizado, donde $I_{u_i} \subseteq \mathrm{I}$. En ese sentido, una recomendación realizada,

al usuario activo $u_a \in$ U, consiste en un conjunto de N ítems $I_r \subset$ I a los cuales se predijo que el usuario activo tendría interés. Cabe resaltar que $I_r \cap I_{u_a} = \emptyset$, pues se desea que se recomienden ítems que el usuario no haya consumido aún. Las valoraciones se almacenan en una matriz de cuya representación se muestra en la figura 1Matriz de valoracionesfigure.1 [19] donde se verifica si el usuario u_a podrá tener interés por el ítem i_j.

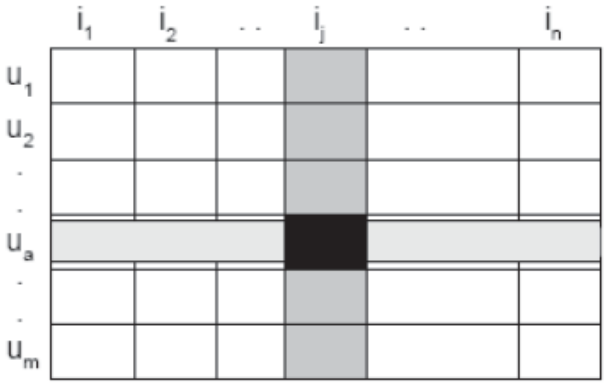

Figura 1: Matriz de valoraciones

Los algoritmos de recomendación colaborativos pueden ser clasificadas en dos tipos según la forma en que procesen la información [20]:

2.2.1. Algoritmos basados en memoria

En estos algoritmos las predicciones son concebidas utilizando todas las valoraciones disponibles en el sistema. Es decir, procesa la matriz de valoraciones cada vez que calcula una predicción. Generalmente, usan medidas de similitud para encontrar usuarios o ítems con un patrón de valoraciones similar, y los utilizan para llevar a cabo la recomendación.

Para proporcionar recomendaciones a un usuario, la primera tarea a desempeñar por los algoritmos de esta categoría es la computación de la similitud entre los usuarios del sistema. Para ello, según Sarwar et al. [19], usualmente se utiliza el coeficiente de correlación de Pearson. La ecuación siguiente aclara como tal coeficiente puede ser calculado para dos usuarios, donde $v_{a,j}$ expresa una valoración realizada por el usuario activo acerca de un producto j, $\bar{v}_i$ el promedio de las valoraciones del usuario i y $\bar{v}_a$ el promedio de las valoraciones del usuario activo.

$$w(a,i) = \frac{\sum_j (v_{a,j} - \bar{v}_a)(v_{i,j} - \bar{v}_i)}{\sqrt{\sum_j (v_{a,j} - \bar{v}_a)^2 \sum_j (v_{i,j} - \bar{v}_i)^2}} \quad (1)$$

En tal sentido, el promedio de las valoraciones de un usuario representa un resumen de las preferencias del mismo. Tal promedio puede ser calculado a través de la ecuación siguiente:

$$\bar{v}_i = \frac{1}{\mid I_i \mid} \sum_{j \in I_i} v_{i,j} \tag{2}$$

Después de disponer de los coeficientes de similitud, el algoritmo obtiene una lista de los n usuarios U $= \{u_1, u_2, ..., u_n\}$ más similares al usuario activo u_a, donde u_1 es el usuario más próximo a u_a, u_2 es el segundo más próximo y así sucesivamente.

Una vez obtenidos los vecinos más cercanos a u_a, ya se puede realizar la predicción referente a un determinado ítem j, la cual define si tal ítem va a ser recomendado o no. Una manera de obtener dicha predicción es a través de la suma ponderada de las valoraciones de otros usuarios. Breese et al. [4] sugieren que sea calculada por medio de la ecuación que se muestra a continuación, donde k es un factor de normalización:

$$p_{v,j} = \bar{v}_i + k \sum_{i=j}^{n} w(a,i)(v_{i,j} - \bar{v}_i) \tag{3}$$

La principal ventaja de utilizar métodos basados en memoria es la rápida incorporación de información reciente [20]. Por lo contrario, un grave problema inherente a estos métodos se refiere a la necesidad de utilizar toda la base de datos de valoraciones, pues sabiéndose que los sistemas de recomendación actuales poseen una inmensa cantidad de ítems, difícilmente se tiene disponible una cantidad suficiente de valoraciones. Además, la escalabilidad puede ser comprometida debido al volumen de datos a procesar. Este problema se refiere al incremento exponencial del tiempo de computación con el aumento del número de usuarios e ítems en el sistema.

2.2.2. Algoritmos basados en modelo

En estos algoritmos se utiliza parte de las preferencias para construir un modelo de estimación de valoraciones. Es decir, hacen uso de la información presente en la matriz de valoraciones para entrenar un modelo previamente especificado. Las predicciones se hacen directamente a partir de dicho modelo, sin necesidad de volver a procesar la matriz.

Los métodos basados en modelos también son conocidos como métodos "basados en ítems", pues según Sarwar et al. [19], a diferencia de los métodos basados en memoria, dichos métodos también consideran, para predecir la valoración acerca de un determinado ítem, la similitud de tal ítem con aquellos ítems que el usuario ya haya valorado. Para ello, generalmente se utilizan los mismos tipos de métricas que se utilizan para calcular similitudes. Tratándose de ítems, la correlación de Pearson, por ejemplo, puede ser obtenida, en relación a dos ítems p y q, como a continuación, donde $\bar{v}_p$ y $\bar{v}_q$ expresan, respectivamente, los promedios de valoraciones en relación a los ítems p y q:

$$sim(p,q) = \frac{\sum_{u \in U}(v_{u,p} - \bar{v}_p)(v_{u,q} - \bar{v}_q)}{\sqrt{\sum_{u \in U}(v_{u,p} - \bar{v}_p)^2}\sqrt{\sum_{u \in U}(v_{u,q} - \bar{v}_q)^2}} \tag{4}$$

Otro método para calcular la similitud entre ítems es la medida del coseno [19], donde se considera a cada ítem como un vector dentro de un espacio vectorial de m dimensiones.La similitud entre ellos sería el coseno del ángulo que forman. En otras palabras, en la matriz de valoraciones $m \times n$, la similitud entre los ítems i y j viene dada por:

$$sim(p, q) = \cos(\vec{i}, \vec{j}) = \frac{\vec{i} \cdot \vec{j}}{\|\vec{i}\|_2 * \|\vec{j}\|_2} \tag{5}$$

donde "$\cdot$" es el producto de los dos vectores.

Después de disponer de los datos relativos a la similitud de los ítems presentes en un sistema, se construye un modelo de estimación de valoraciones, el cual se crea *off-line*, es decir, con anterioridad a la entrada del usuario activo en el sistema. Por lo tanto, se realiza el aprendizaje de un modelo de comportamiento de usuario con datos almacenados por el sistema y de modo *off-line* para que posteriormente tal modelo sea aplicado en tiempo real [14]. Para ello, se suelen utilizar, en general, técnicas aprendizaje automático.

La principal ventaja de esta categoría de métodos se refiere a la simplicidad y velocidad con la que se realizan las recomendaciones, pues la mayor parte del procesamiento de datos se realiza *off-line*, por lo que el tiempo utilizado en la construcción del modelo no repercute en el tiempo de respuesta al usuario. Tienen, en cambio, la desventaja de que la información nueva no se incorpora de manera inmediata al modelo, sino sólo cuando se genera un nuevo modelo incluyendo la nueva información disponible. Siendo así, de acuerdo con Shaffer et al. [20], dichos métodos son más indicados para sistemas en los cuales las preferencias de sus usuarios cambian de forma lenta en relación al tiempo necesario para construir el modelo.

2.3. Trabajos Relacionados

Actualmente muchos investigadores están trabajando en el desarrollo y mejora de métodos de recomendación de música. A continuación, se revisan algunos trabajos en esta área de investigación.

Bogdanov et al. [3] proponen tres enfoques basados en el contenido. Estos autores infieren descriptores semánticos de alto nivel a partir de un conjunto explícito de canciones proporcionadas por el usuario, como prueba de sus preferencias musicales. Estos descriptores abarcan diferentes facetas musicales, tales como el género, la cultura, los estados de ánimo, los instrumentos, el ritmo y el tempo. Sobre esta base, dos de los enfoques propuestos emplean una medida de similitud semántica de la música para generar recomendaciones; y el tercer enfoque crea un modelo probabilístico de las preferencias del usuario en el dominio semántico. La propuesta fue evaluada, no pudiendo llegar a un rendimiento comparable a un sistema de filtrado colaborativo.

Park et al. [17] proponen un método de recomendación llamado *Session-based*

Collaborative Filtering (SSCF) y observan los diferentes parámetros que afectan la precisión en la recomendación. Este método incluye los ítems seleccionados actualmente en el perfil de la sesión y encuentra las sesiones más similares para generar la recomendación. Se compararon los resultados con la técnica de recomendación de filtrado colaborativo básica y se encontró que el método propuesto la supera.

Dias et al. [8] han hecho uso del contexto temporal y la diversidad de sesiones en las técnicas de filtrado colaborativo basado en la sesión para la recomendación de música. Además, compararon dos algoritmos que utilizan características temporales: uno que extrae de forma explícita las propiedades temporales y la diversidad de sesiones y otro que es capaz de modelar implícitamente patrones temporales. Los resultados revelaron que ambos algoritmos mejoran la precisión en comparación con el típico filtrado colaborativo basado en la sesión.

Xing et al. [24] introducen el concepto de exploración en el filtrado colaborativo y tratan de compensarlo con la explotación que estos sistemas suelen utilizar. Para conocer los gustos musicales de los usuarios, han usado un modelo bayesiano que tiene en cuenta tanto los factores latentes del filtrado colaborativo como la novedad en la recomendación. Además, diseñaron un algoritmo de inferencia bayesiana para estimar de manera eficiente las distribuciones de probabilidad a posteriori del *rating*. Sus resultados demuestran que mejora la fiabilidad de la recomendación.

Su et al. [21] proponen un sistema de recomendación que utiliza etiquetas de medios sociales para calcular la similitud entre las canciones. Al igual que en el presente trabajo, se usa el número de *plays* de las canciones para calcular las valoraciones, pero de una manera distinta. Además, se aprovecha la información de las etiquetas de las canciones para capturar las preferencias del usuario. Los resultados son buenos y revelan que con el uso de etiquetas sociales se pueden predecir preferencias de música del usuario.

Hornung et al. [10] presentan un sistema híbrido de recomendación que fusiona tres técnicas de recomendación que son el uso de la similitud de la canción, de la etiqueta y la similitud del tiempo. La nueva música se descubre con el uso de una nueva métrica llamada *serendipia*. Tras la evaluación de este sistema observaron que la calidad predictiva mejora a lo largo del número de listas de recomendación proporcionadas a cada usuario. Adicionalmente, ellos hicieron un estudio para saber cuál de las tres técnicas devolvía mejores resultados y llegaron a la conclusión que la recomendación basada en la canción fue la mejor.

Domingues et al. [9] proponen un sistema de recomendación de música híbrido, que combina datos de uso y contenido. Lo compararon con dos sistemas de recomendación independientes, el primero basado en el uso y el segundo basado en el contenido (es decir, etiquetas de audio y de texto). Los resultados mostraron que el recomendador híbrido propuesto por ellos presenta ventajas con respecto a los sistemas basados en el uso y contenido.

Van den Oord et al. [22] utilizan una matriz de factorización modificada dirigida a conjuntos de datos de valoración implícita, propuesto por Hu et al. [11]. En primer lugar, ellos obtuvieron vectores de variables ocultas de las canciones que utilizaron

posteriormente para el entrenamiento de un modelo de regresión. Utilizaron redes neuronales profundas convolucionales para predecir variables ocultas de audio de la música. Los resultados de este trabajo produjeron recomendaciones razonables que demuestran que los avances en el aprendizaje profundo pueden ayudar en las recomendaciones de la música.

Reafee et al. [18] proponen la inclusión de las relaciones sociales implícitas para la recomendación, para ello proponen un modelo *EISR (explicit and implicit social relation)* y también, un algoritmo *PFNF (Possibility of Friendship Between Non-Friends)* para extraer la relación implícita (relación social escondida) en los grafos no dirigidos de redes sociales mediante la explotación de las técnicas de predicción de enlace. Tanto las amistades explícitas como implícitas se incorporan en el modelo propuesto. La evaluación de este modelo con un conjunto de datos de *Last.fm* obtiene resultados de recomendaciones más precisos que los métodos de recomendación mencionados en su estado del arte.

3. Desarrollo del Sistema de Recomendación

En este apartado se describen las tecnologías usadas en el trabajo y se detallan los pasos que se han realizado para la implementación del Sistema de Recomendación.

3.1. Scrobbling y Last.fm

Scrobbling [5] es un concepto que nació gracias a un proyecto de ciencias de computación de Richard Jones llamado *Audioscrobbler*, y se trataba de un *plugin* para *Winamp* [2] capaz de recoger lo que el usuario está escuchando para darle todos los datos y recomendaciones de música y artistas que se ajustan a sus gustos. En agosto de 2005, *Audioscrobbler* se fusionó con *Last.fm*, una comunidad web de radio por Internet y música. La página web resultante mantiene el nombre de *Last.fm*, mientras que el nombre *Audioscrobbler* se utiliza para referirse al programa de fondo que apoya la construcción de los perfiles de los usuarios y otras características como el servicio de recomendación. El 30 de mayo de 2007, el equipo de *Last.fm* anunció que había sido adquirido por la cadena de televisión CBS.

Por lo tanto, el concepto *Scrobbling* ahora es usado para referirse al proceso de presentación de información a *Last.fm* sobre cada canción que el usuario escucha. Esta información se almacena en los servidores de *Last.fm* y se utiliza para construir perfiles de los usuarios sobre la base de lo que escuchan. El proceso de envío se logra a través de un software específico que se puede instalar como un *plugin* en el reproductor musical del usuario. Cada envío a *Last.fm* contiene detalles sobre la canción escuchada, como el nombre de la canción y la duración, el artista y el álbum. También se envía una marca de tiempo (*timestamp*), que es el momento en el cual la canción fue escuchada. Toda esta información se almacena en la base de datos de *Last.fm*.

Un usuario de *Last.fm* puede construir un perfil musical usando dos métodos: escuchando su colección musical personal en una aplicación de música con un *plugin* de *Audioscrobbler*, o escuchando el servicio de radio a través de Internet de *Last.fm*, normalmente con su propio reproductor. Esta información le sirve a *Last.fm* para recomendarle nuevas canciones. Se trata de un servicio que permite al usuario escuchar música a través de *streaming*, leer información sobre artistas, descubrir artistas que no conoce, conocer personas con sus mismos gustos musicales, etc.

Las recomendaciones son obtenidas usando un algoritmo de filtrado colaborativo. Así los usuarios pueden explorar una lista de artistas no listados en su propio perfil pero que sí aparecen en los perfiles de otros usuarios con gustos similares. *Last.fm* también permite a los usuarios recomendar manualmente discos específicos a otros usuarios (siempre que el disco esté incluido en la base de datos).

[2] Winamp: `http://www.winamp.com/`

3.2. API de Last.fm

Los servicios web de *Last.fm* proporcionan una API pública que permite a cualquiera acceder a los datos de Last.fm. Esta API permite a los desarrolladores llamar a métodos cuyos resultados se presentan en formato XML o JSON estilo REST. Por ejemplo, una petición HTTP GET a la siguiente URL: *getfriends* [3]

```
<lfm status="ok">
 <friends for="soolyme" page="1" perPage="50" totalPages="1" total="3">
  <user>
   <name>mocephus</name>
   <realname>Austin Moody</realname>
   <image size="small">
     http://img2-ak.lst.fm/i/u/34s/cd8afc372cf243c1c36fd880bd4a9939.png
   </image>
   <image size="medium">
     http://img2-ak.lst.fm/i/u/64s/cd8afc372cf243c1c36fd880bd4a9939.png
   </image>
   <image size="large">
     http://img2-ak.lst.fm/i/u/174s/cd8afc372cf243c1c36fd880bd4a9939.png
   </image>
   <image size="extralarge">
     http://img2-ak.lst.fm/i/u/300x300/cd8afc372cf243c1c36fd880bd4a9939.png
   </image>
   <url>http://www.last.fm/user/mocephus</url>
   <country>United States</country>
   <age>0</age>
   <gender>n</gender>
   <subscriber>FIXME</subscriber>
   <playcount>73895</playcount>
   <playlists>0</playlists>
   <bootstrap>0</bootstrap>
   <registered unixtime="1045664815">2003-02-19 14:26:55</registered>
   <type>FIXME</type>
   <scrobblesource>FIXME</scrobblesource>
  </user>
  <user>
   <name>IPayTheCalliope</name>
   <realname>Yoni Berk</realname>
   <image size="small">
     http://img2-ak.lst.fm/i/u/34s/9f2f30d23c384c4fcc99f68ae23e67c1.png
   </image>
   <image size="medium">
     http://img2-ak.lst.fm/i/u/64s/9f2f30d23c384c4fcc99f68ae23e67c1.png
   </image>
   <image size="large">
     http://img2-ak.lst.fm/i/u/174s/9f2f30d23c384c4fcc99f68ae23e67c1.png
   </image>
   <image size="extralarge">
     http://img2-ak.lst.fm/i/u/300x300/9f2f30d23c384c4fcc99f68ae23e67c1.png
   </image>
   <url>http://www.last.fm/user/IPayTheCalliope</url>
   <country>United States</country>
   <age>0</age>
   <gender>n</gender>
   <subscriber>FIXME</subscriber>
   <playcount>27916</playcount>
   <playlists>0</playlists>
   <bootstrap>0</bootstrap>
   <registered unixtime="1103922666">2004-12-24 21:11:06</registered>
   <type>FIXME</type>
   <scrobblesource>FIXME</scrobblesource>
  </user>
```

[3] getfriends: http://ws.audioscrobbler.com/2.0?method=user.getfriends&user=soolyme&api_key=d0c2f36e8e87e8ba8af78cad302f8c92

```
 </friends>
</lfm>
```

En este trabajo se ha hecho uso de la API de *Last.fm* para la generación del conjunto de datos. Para ello se han usado como base los siguientes programas Python: *LastExport.py* [4] y *ScrobblesToday.py* [5]. He aquí un resumen de todos los métodos y sus respectivos parámetros que se han utilizado en este trabajo:

`User.getFriends`	
Descripción	Devuelve los nombres de los usuarios que son amigos del usuario dado. Este método fue muy útil para descubrir nuevos nombres de usuarios, de esa manera se pudo alimentar el conjunto de datos con la información de más usuarios.
Parámetros	
`username`	Un nombre de usuario de *Last.fm*

Tabla 1: Método: User.getFriends

`User.get_country`	
Descripción	Devuelve el nombre del país del usuario dado. Este método fue útil para descubrir de dónde proviene el usuario.
Parámetros	
`username`	Un nombre de usuario de *Last.fm*

Tabla 2: Método: User.get_country

`User.getRecentTracks`	
Descripción	Devuelve una lista de las últimas pistas escuchadas por este usuario. "Recientes" no es un nombre apropiado, puesto que las pistas que devuelve este método son las escuchadas en su totalidad. Este método fue útil para rastrear las pistas escuchadas por los usuarios.
Parámetros	
`username`	Un nombre de usuario de *Last.fm*
`limit`	Número de pistas a recuperar (en función de cada página)
page	Número de la página a recuperar.
`from`	Registro de tiempo. Sólo se devuelven pistas escuchadas después de este punto en el tiempo.
`to`	Registro de tiempo. Sólo se devuelven pistas escuchadas antes de este punto en el tiempo.

Tabla 3: Método: User.getRecentTracks

[4] LastExport.py: `https://gist.github.com/bitmorse/5201491`

[5] ScrobblesToday.py: `https://github.com/dufferzafar/Python-Scripts/blob/master/Last.fm\%20Plays/ScrobblesToday.py`

3.3. Tecnologías Usadas

A continuación se describen las tecnologías utilizadas en este trabajo:

3.3.1. Python

Python es un lenguaje de programación de alto nivel, interpretado, multipropósito y multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma. Su filosofía hace hincapié en la legibilidad del código. Puede ser utilizado en diversas plataformas y sistemas operativos, entre los que podemos destacar los más populares, como Windows, Mac OS X y Linux. Pero, además, Python también puede funcionar en smartphones. Python es open source, cualquiera puede contribuir a su desarrollo y divulgación. Además, no es necesario pagar ninguna licencia para distribuir software desarrollado con este lenguaje. Hasta su intérprete se distribuye de forma gratuita para diferentes plataformas. En los últimos años Python se ha hecho muy conocido y en la actualidad es uno de los lenguajes de programación más empleados para el desarrollo de software, gracias a varias razones como: La cantidad de librerías que contiene, tipos de datos y funciones incorporadas en el propio lenguaje, que ayudan a realizar muchas tareas habituales sin necesidad de tener que programarlas desde cero. La sencillez y velocidad con la que se crean los programas. Un programa en Python puede tener de 3 a 5 líneas de código menos que su equivalente en Java o C. La cantidad de plataformas en las que podemos desarrollar, como Unix, Windows, OS/2, Mac, Amiga y otros. Además, Python es gratuito, incluso para propósitos empresariales. Por otra parte, la popularidad de Python ha hecho que varias bibliotecas de terceros con soporte para las interfaces necesarias para este trabajo: base de datos Postgresql y la API de *Last.fm*.

3.3.2. PostgreSQL

PostgreSQL es un sistema de base de datos relacional orientado a objetos de código abierto. PostgreSQL es gratuito y libre, además de que hoy nos ofrece una gran cantidad de opciones avanzadas. Se ejecuta en los principales sistemas operativos que existen en la actualidad como: Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) y Windows. Es altamente escalable, tanto en la enorme cantidad de datos que puede manejar y en el número de usuarios concurrentes que puede administrar. PostgreSQL cuenta con características avanzadas tales como Multi-Version Control de concurrencia (MVCC), puntos en tiempo de recuperación, tablespaces, replicación asincrónica, transacciones anidadas (*savepoints*), respaldos online/hot, un sofisticado query planner/optimizer. Soporta el conjunto de caracteres internacional, codificaciones de caracteres multibyte, Unicode, mayúsculas y minúsculas.

3.3.3. Librería: Psycopg2

Psycopg2 es el adaptador de base de datos PostgreSQL más popular para el lenguaje de programación Python. Está escrito en C y permite llevar a cabo de manera eficiente toda la gama de operaciones de SQL en bases de datos PostgreSQL.

3.3.4. Librería: PyLast

PyLast es una interfaz Python para Last.fm de código abierto que brinda acceso a todos los datos expuestos por los servicios web de *Last.fm*.

3.3.5. RapidMiner

RapidMiner es un programa informático que sirve para realizar análisis y minería de datos. Está escrito en el lenguaje de programación Java y permite el desarrollo de procesos de análisis de datos mediante el encadenamiento de operadores a través de un entorno gráfico. Entre los procedimientos de aprendizaje automático que ofrece, están: la carga y transformación de datos (ETL), pre-procesamiento de datos, visualización, modelado, evaluación y despliegue [1].

Se puede usar RapidMiner para construir flujos de trabajo de filtrado de información. Sin embargo, estas tareas no vienen incluidas por defecto en el programa. Gracias a que RapidMiner es un programa extensible, se puede hacer uso de la extensión *"RapidMiner Recommender Extension"* con el fin de utilizar algunas de las técnicas de recomendación para el trabajo.

Como se observa en la figura 2RapidMiner - Extensión de recomendaciónfigure.2, esta extensión contiene 3 Categorías: *Ir*, *Irp* y *Rper*. Cada una de éstas contiene operadores que sirven para diferentes propósitos. A continuación, se presenta la explicación de cada uno:

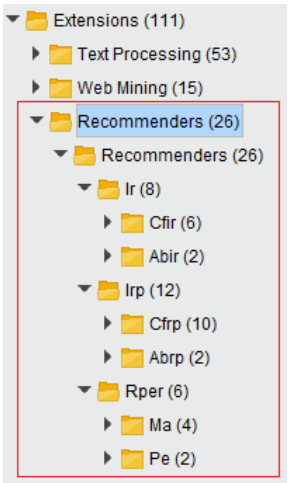

Figura 2: RapidMiner - Extensión de recomendación

- ***Ir (Item Recommendation)***: Tiene 8 operadores que contienen algoritmos de recomendación de ítems. Donde se busca que para un usuario determinado se prediga los ítems que le va a gustar mediante un *ranking*.
- ***Irp (Item Rating Prediction)***: Tiene 12 operadores que contienen algoritmos de recomendación basada en *rating*. Donde se busca que para un usuario determinado y un ítem determinado se prediga el valor del *rating*.
- ***Rper (Recommender Performance)***: Tiene 6 operadores que contienen 4 para la aplicación del modelo y 2 para la medición de las métricas de calidad.

De acuerdo a lo descrito por Su et al. [13], las 2 primeras categorías (*Ir* y *Irp*) se dividen en 2 tipos de operadores: los basados en filtrado colaborativo (***Cfir***-*Collaborative Filtering Item Recommendation*- y ***Cfrp***-*Collaborative Filtering Rating Prediction*-) y los basados en el atributo (***Abir***-*Attribute Based Item Recommendation*- y ***Abrp***-*Attribute Based Rating Prediction*-). Mientras que los operadores basados en filtrado colaborativo hacen uso sólo de los *ratings*, los operadores basados en el atributo, hacen uso también de los atributos del usuario como edad, país, etc.; o del ítem, como el artista. Estos operadores devuelven un modelo entrenado. Cada operador tiene sus propios parámetros que pueden ser ajustados, como el número k de vecinos más próximos y la medida de similitud, que puede ser Coseno o Pearson. Estos ajustes pueden lograr una mejora en el tiempo de ejecución y en la precisión.

Para cada una de estas 2 primeras categorías, existen operadores que evalúan el comportamiento de los algoritmos, y éstos son: ***Ma*** *(Model Application)* y ***Pe*** *(Performance Evaluation)*, que se encuentran en la tercera categoría *Rper*. Dentro de *Ma*, se encuentra el operador de ***Apply Model (Rating Prediction)*** que recibe como entradas un modelo obtenido a partir de los datos de entrenamiento y un conjunto de datos de prueba. Este operador aplica el modelo sobre los datos de prueba y devuelve un conjunto de datos que contiene, además de los atributos iniciales, el atributo predicho, es decir, el *rating* por cada par de usuario-ítem del conjunto de datos de prueba. La salida de este operador, se utiliza para calcular las métricas de calidad usando el operador ***Perfomance (Rating Prediction)*** que se encuentra de *Pe*. Este operador calcula el valor de las medidas de error de predicción del *rating* que son: Error cuadrático medio (*RMSE*), Error absoluto medio (*MAE*) y el Error Absoluto medio normalizado (*NMAE*). Estas medidas de error son rertornadas como un vector de desempeño y un conjunto de datos ejemplo.

Otro operador que se encuentra dentro de *Ma* es el ***Apply Model (Item Recommendation)***. Este operador también recibe un modelo y un conjunto de datos de prueba como entradas. Además, aplica el modelo sobre los datos de prueba y devuelve una lista clasificada de los N primeros ítems para cada usuario del conjunto de datos de prueba, donde N es un parámetro que puede ser definido por el usuario. La salida de este operador no se puede utilizar para el cálculo de las métricas de calidad. Dentro de *Pe*, también se encuentra el operador ***Perfomance (Item Recommendation)***. A diferencia del anterior operador, éste recibe como entradas: un conjunto de datos de entrenamiento, un conjunto de datos de pruebas y un modelo. Este operador devuelve medidas de error de recomendación de ítems que son:

el área bajo la curva (AUC), la precisión en k ($prec@k$), Ganancia acumulada con descuento normalizado ($NDCG$) y la media de precisión promedio (MAP).

Adicionalmente a los operadores mencionados, existe un operador llamado ***Model Combiner*** para cada una de las 2 primeras categorías. Este operador recibe como entradas a múltiples modelos de entrenamiento y devuelve un modelo agrupado ponderado. Este operador permite crear modelos híbridos, combinando operadores basados en filtrado colaborativo y basados en atributo. [13]

Más adelante se mostrarán los procesos construidos con esta herramienta que han hecho posible el desarrollo de este trabajo.

3.4. Generación del Conjunto de Datos

El proceso de generación del conjunto de datos se ha realizado con el programa desarrollado en *Python* llamado: *generDatosLastFM.py*. Para el almacenamiento de los datos recolectados se ha utilizado la base de datos *Postgresql*. La comunicación entre estas dos tecnologías ha sido posible gracias al uso de la librería *psycopg2* y finalmente, para la comunicación de Python con el API de *Last.fm* se ha usado la librería *PyLast*.

Para este fin, el proceso se inicia con el descubrimiento de nombres de usuario de *Last.fm* utilizando el método *User.getFriends* (Tabla 1Método: User.getFriendstable.1). Para cada uno de ellos, se recupera el país de dónde es (Tabla 2Método: User.get_countrytable.2). Cabe resaltar que se recuperaron también la edad y género, sin embargo no tenían valores, es por esa razón que en este trabajo el único atributo del usuario que se considera es el *país*. Por cada usuario, un proceso iterativo recupera la totalidad de su perfil de escucha haciendo uso del método *User.getRecentTracks* (Tabla 3Método: User.getRecentTrackstable.3). Se ha colocado como límites de tiempo recuperar información de 2 meses hacia adelante.

Estos procesos se han repetido hasta que el número de perfiles de usuario alcance la cantidad deseada. La figura 3Diagrama de flujo de la generación del conjunto de datosfigure.3 ilustra este proceso.

3.4.1. Preparación de la Base de Datos

La fase de preparación de base de datos consiste en la creación de las tablas, de acuerdo con la especificación que se ilustra en la figura 4Diagrama de base de datosfigure.4 . La tabla *musiclistened* va a contener la información detallada de los perfiles de escucha de los usuarios; es decir, que cada registro representa el momento en que el usuario escuchó una canción. En esta tabla también se incluye la información del usuario como son: nombre, país y género. La tabla *musiclist_ agr* es una tabla agregada de la anterior; y almacenará la información preprocesada.

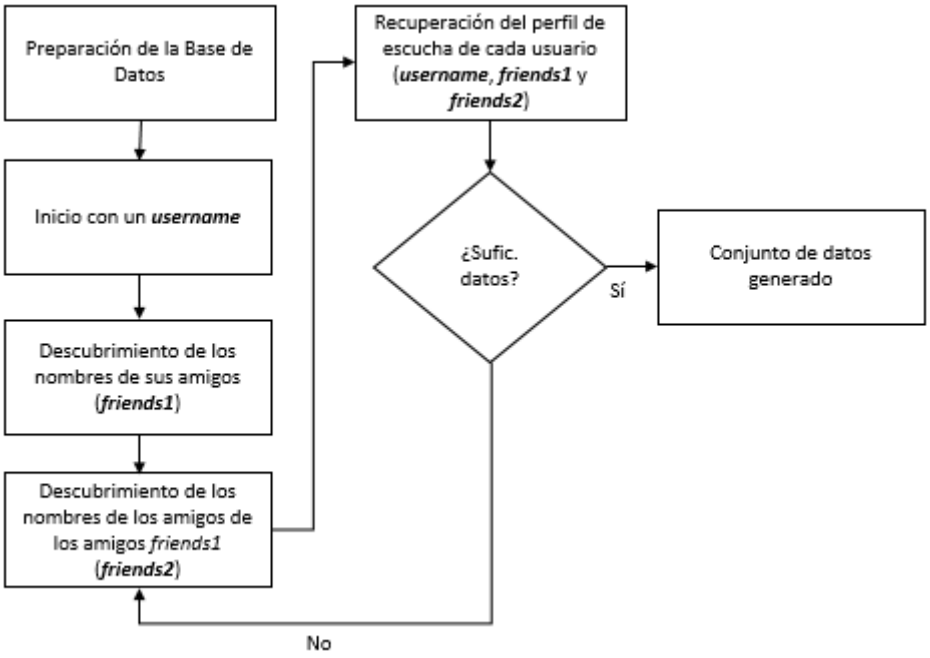

Figura 3: Diagrama de flujo de la generación del conjunto de datos

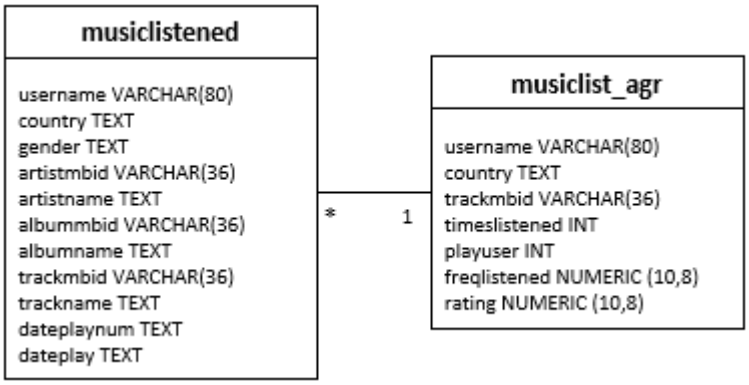

Figura 4: Diagrama de base de datos

Los campos de la tabla *musiclistened* se describen en la tabla 4Campos de la tabla musiclistenedtable.4.

musiclistened	
username	Nombre del usuario
country	País del usuario
gender	Género del usuario
artistmbid	Identificador del artista
artistname	Nombre del artista
albummbid	Identificador del álbum
albumname	Nombre del álbum
trackmbid	Identificador de la canción
trackname	Nombre de la canción
dateplaynum	Fecha en que el usuario escuchó la canción (formato Tiempo Unix [6]
dateplay	Fecha en que el usuario escuchó la canción (Formato dd mon aaaa, hh:mm)

Tabla 4: Campos de la tabla musiclistened

Los campos de la tabla *musiclist_agr* se describen en la tabla 5Campos de la tabla musiclist_agrtable.5.

musiclist_agr	
username	Nombre del usuario
country	País del usuario
artistmbid	Identificador del artista
trackmbid	Identificador de la canción
timeslistened	Número de *plays* que el usuario le dio a la canción
playsuser	Número de *plays* totales del usuario (todas las canciones que escuchó)
freqlistened	Frecuencia de *plays* de la canción (timeslistened/playsuser)
rating	*Rating* de la canción para ese usuario

Tabla 5: Campos de la tabla musiclist_agr

3.4.2. Descubrimiento de usuarios

El descubrimiento de usuarios se logra mediante el uso del método *User.getFriends* de la API de *Last.fm*. Este método devuelve la lista de los usuarios que son amigos de un usuario determinado. Por lo tanto, se puede hacer la llamada al mismo método para recuperar los nombres de los usuarios que son amigos de los amigos previamente recuperados.

Este proceso se lleva a cabo como se muestra en la tabla 6Algoritmo: Descubri-

[6]Tiempo Unix es un sistema para describir momentos, definido como el número de segundos transcurridos desde la medianoche del Tiempo Universal Coordinado (UTC) 1 de enero de 1970

miento de usuariostable.6. Como se puede observar, la recuperación se realizó de forma iterativa y con cada usuario, se recuperó su género y país (aunque como ya se mencionó anteriormente, el género no contiene valor). Además, por cada usuario se recuperó su perfil de escucha. Este proceso será explicado en el siguiente apartado 3.4.3Recuperación del perfil de escucha de cada usuariosubsubsection.3.4.3. Para este trabajo se recopilaron 1537 nombres de usuarios.

```
function recuperacionEscucha(usuario,país,género): inserta la información del perfil de es

user = "Nombre de cualquier usuario"
friends1andme <-- [user] //se inicializa la lista con el usuario inicializado
friends1 <-- user.get_friends() //se obtienen los amigos de user
friends1andme <-- friends1andme + friend1 // tener la lista completa

username_list <-- [ ] // se inicializa la lista vacía

for friend in friends1andme do

  if friend not in username_list then
    country <-- friend.get_country()
    gender <-- friend.get_gender()
    recuperacionEscucha(friend,country,gender)
    username_list <-- username_list + friend
  end if

  friends2 <-- friend.get_friends()

  for friend2 in friends2 do
          if friend2 not in username_list then
                country2 <-- friend2.get_country()
                gender2 <-- friend2.get_gender()
                recuperacionEscucha(friend2,country2,gender2)
                username_list <-- username_list + friend2
          end if
  end for
end for
```

Tabla 6: Algoritmo: Descubrimiento de usuarios

3.4.3. Recuperación del perfil de escucha de cada usuario

La recuperación de la historia de escucha de un usuario se logró mediante el uso del método *User.getRecentTracks*. Esta recuperación por usuario cuesta mucho tiempo, razón por la cual, se limitó el tiempo a la escucha de los últimos 2 meses. Este proceso que registra los datos en la tabla *musiclistened* se llevó a cabo con el

programa Python *generDatosLastFM.py*. Se recogió la historia de escucha de una muestra de 1537 usuarios.

3.4.4. Características del Conjunto de Datos Inicial

Después de la recuperación de la información de escucha de cada usuario, la tabla *musiclistened* contiene registros tal y como se muestra en la figura 5Conjunto de datos inicialfigure.5.

username character varying(80)	country text	gender text	artistmbid character varying(36)	artistname text	albummbid character varying(36)	albumname text	trackmbid character varying(36)	trackname text	dateplaynum text	dateplay text
GhostHaunted	Russian Federa	None	01809552-4f87-45b0-	Elvis Presle	9b03d39d-50d2-4ebc-	CMon Everybody	ea417d0c-e0a3-474d-	A Whistling Tune	1465939788	14 Jun 2016, 21:29
GianlucaVilla	Italy	None	fcba6ab6-4a51-4688-	Jack Garratt		Chemical		Chemical	1465132754	05 Jun 2016, 13:19
gilamor	None	None		Hip Hop Jaz				The Oldscool Vib	1464073689	24 May 2016, 07:08
Givi234	Brazil	None	b9472588-93f3-4922-	Panic! at th	90efb495-16f8-4b8d-	Pretty. Odd.	e5a2f5bd-7a26-4597-	Nine in the Afte	1462564275	06 May 2016, 19:51
gizzurl	Brazil	None	650e7db6-b795-4eb5-	Lady Gaga		The Fame Monster (	c50edf48-ac2b-442f-	Poker Face	1467559903	03 Jul 2016, 15:31
gnhnm	Turkey	None	ac69016f-42cc-4322-	Sébastien Te	a560f66f-3398-48fc-	Confection		LAmour Naissant	1463923275	22 May 2016, 13:21
gokcellywood	Turkey	None	2f548675-008d-4332-	Sia		Cheap Thrills		Cheap Thrills	1464118025	24 May 2016, 19:27
gokceye	Turkey	None		Oh Wonder		Heart Hope		Heart Hope	1463028753	12 May 2016, 04:52
goksun01	Turkey	None	b77a873a-9faf-4f59-	REID		Fractures	322e1fee-0c59-45ab-	Singapore	1464604216	30 May 2016, 10:30
Gon	Argentina	None	bdacc37b-8633-4bf8-	Slayer	eadd8a4d-7336-3caa-	World Painted Bloo	6708235b-b4aa-42b7-	Hate Worldwide	1462495193	06 May 2016, 00:39
good excuse	Turkey	None	65f4f0c5-ef9e-490c-	Metallica	d08247fc-25b6-3570-	Load	6826f1ed-119b-4ad7-	Until It Sleeps	1465727083	12 Jun 2016, 10:24
gordi01	Switzerland	None	f0602f55-1770-483d-	Jennifer Lop		Aint Your Mama		Aint Your Mama	1463872000	21 May 2016, 23:06
Gothamcool	India	None	cc197bad-dc9c-440d-	Coldplay		A Head Full Of Dre		Amazing day	1464374383	27 May 2016, 18:39
gottahaveareaso	Poland	None	98af45d8-fff7-48e4-	Tyketto	aae349ef-6ae6-40e4-	Dont Come Easy	d162b438-38f5-476d-	Forever Young	1465598391	10 Jun 2016, 22:39
gouldritual	France	None		Jonas Blue		Perfect Strangers		Perfect Stranger	1467461221	02 Jul 2016, 12:07
Gouofleta	None	None	ad996aef-cc1c-42ac-	Emerson, Lak	239764e2-043d-4ee9-	Tarkus	c441cfd6-4129-4aca-	Jeremy Bender	1465411364	08 Jun 2016, 18:42

Figura 5: Conjunto de datos inicial

La figura 6Número inicial de usuarios por país (20 primeros)figure.6 muestra el número de usuarios por país ordenados de mayor a menor. Como se puede observar, sólo se muestran los 20 primeros países. Era de esperar que la mayoría de usuarios sean de Brasil, pues el usuario inicial que se escogió para recopilar el conjunto de datos es de ahí.

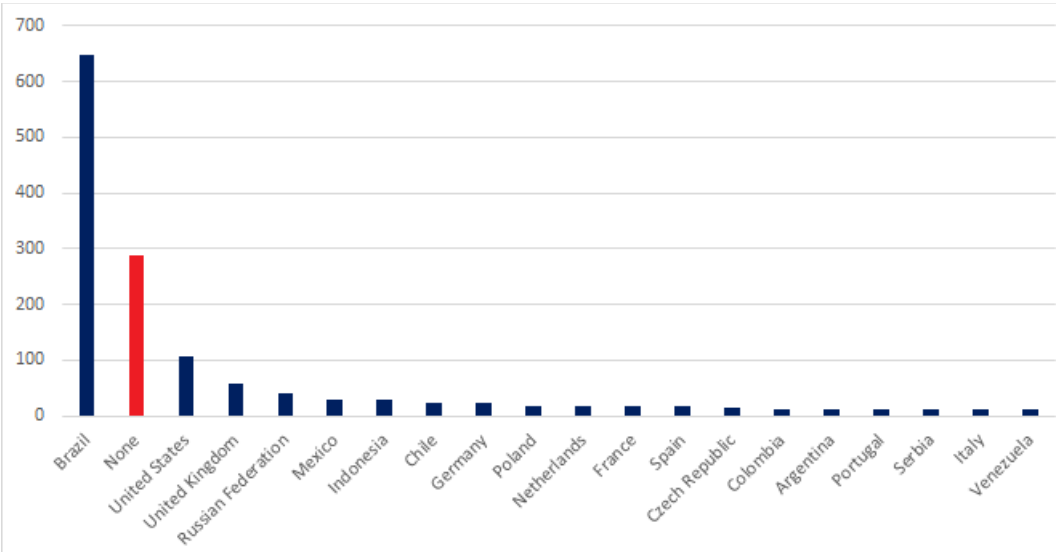

Figura 6: Número inicial de usuarios por país (20 primeros)

No se puede hacer una distinción de canciones por su nombre puesto que una

misma canción figura con dos nombres diferentes en la base de datos, tal y como se muestra en la figura 7Muestra de canciones repetidasfigure.7.

artistmbid character varying(38)	artistname text	trackmbid character varying(36)	trackname text	count bigint
27e2997f-f7a1-4353-bcc4-57b9274fa9a4	BABYMETAL	135309f0-7fb1-4eae-9321-3919c0c52aad	Iine!	3
27e2997f-f7a1-4353-bcc4-57b9274fa9a4	BABYMETAL	135309f0-7fb1-4eae-9321-3919c0c52aad	いいね!	23
45a663b5-b1cb-4a91-bff6-2bef7bbfdd76	Britney Spears	44abd7d3-c593-4587-a109-6d9582f13f36	Oops!...I Did It Again (album version)	1
45a663b5-b1cb-4a91-bff6-2bef7bbfdd76	Britney Spears	44abd7d3-c593-4587-a109-6d9582f13f36	Oops!...I Did It Again	1047
45a663b5-b1cb-4a91-bff6-2bef7bbfdd76	Britney Spears	44abd7d3-c593-4587-a109-6d9582f13f36	Oops!... I Did It Again	123
bc471fd0-4e12-49f3-951b-2371b274590d	Perfume	100692c7-d5a7-4554-a131-62ef05691713	Daijyobanai	1
bc471fd0-4e12-49f3-951b-2371b274590d	Perfume	100692c7-d5a7-4554-a131-62ef05691713	だいじょばない	2
bc471fd0-4e12-49f3-951b-2371b274590d	Perfume	100692c7-d5a7-4554-a131-62ef05691713	Daijobanai	1
fbe054ec-a143-4101-9e9e-64abc5ff5ac9	Nat King Cole	005cf760-815b-4652-b8f8-7546e60fa9a0	Those Lazy-Hazy-Crazy Days of Summer	2
fbe054ec-a143-4101-9e9e-64abc5ff5ac9	Nat King Cole	005cf760-815b-4652-b8f8-7546e60fa9a0	Those Lazy, Hazy, Crazy Days Of Summer	1

Figura 7: Muestra de canciones repetidas

Asimismo, como se puede observar en la figura 8Muestra de canciones sin identificadorfigure.8, no se puede especificar el número de canciones sin id, porque muchas son las mismas canciones pero se diferencian por el nombre.

trackmbid character varying(36)	trackname text
	Never Say Good Bye
	Never Say Goodbye
	Never Say Goodbye - Radio Edit
	Never Say Goodbye (feat. Bright Lights) [Radio Edit]
	Never Say Goodbye [Mix Cut]
	Never Say Never
	NEVER SAY NEVER
	Never Say Never - Non-Lp Version
	Never Say Never - Single Version
	Never Say Never (Acoustic Version)
	Never Say Never (Extended Mix)
	Never Say Never (feat. Jaden Smith)
	Never Say Never (Feat. Jaden Smith) [Acoustic Version]
	Never Say Never (Live in Mexico)

Figura 8: Muestra de canciones sin identificador

De la misma forma, no se puede hacer una distinción de los artistas por su nombre, porque un mismo artista figura con diferentes nombres como se puede observar en la figura 9Muestra de artistas repetidosfigure.9.

artistmbid character varying(38)	artistname text	trackmbid character varying(36)	plays bigint
0d79fe8e-ba27-4859-bb8c-2f255f346853	BTS	34ad7c0f-6f8f-429a-b7bd-b35a3625ce5c	15
0d79fe8e-ba27-4859-bb8c-2f255f346853	방탄소년단	34ad7c0f-6f8f-429a-b7bd-b35a3625ce5c	98
10e2d18c-7a8c-4d26-bb4a-450bb6851cb2	kyary pamyu pamyu	dd1fa6ea-56a9-4c62-badb-a0c0981df910	10
10e2d18c-7a8c-4d26-bb4a-450bb6851cb2	きゃりーぱみゅぱみゅ	dd1fa6ea-56a9-4c62-badb-a0c0981df910	12
b7442f18-d9be-4185-8b51-482510046156	少女時代	133e538c-bc51-408c-a98e-183cb72682f6	9
b7442f18-d9be-4185-8b51-482510046156	Girls Generation	133e538c-bc51-408c-a98e-183cb72682f6	13
d5be5333-4171-427e-8e12-732087c6b78e	The Black Eyed Peas	b793d36c-e61e-4154-8096-e891f1e0ad37	15
d5be5333-4171-427e-8e12-732087c6b78e	Black Eyed Peas	b793d36c-e61e-4154-8096-e891f1e0ad37	32

Figura 9: Muestra de artistas repetidos

Teniendo en cuenta todas las características anteriormente mencionadas, se muestra un resumen de la cantidad de registros que figuran en este conjunto de datos inicial (Tabla 7Número de registros del conjunto de datos inicialtable.7).

Número de Usuarios	1537
Número de Usuarios con País	1248
Número de Artistas con Id	44429
Número de Canciones con Id	260094
Número de Plays Totales	5373399

Tabla 7: Número de registros del conjunto de datos inicial

3.4.5. Filtrado

Como se pudo observar en la figura 5Conjunto de datos inicialfigure.5 del apartado anterior, existen usuarios registrados cuyo atributo *country* figura con el valor *"None"*. Adicionalmente, se observa que todos los usuarios tienen el atributo gender con el valor *"None"*. Finalmente, se puede notar que algunos registros figuran con el atributo *trackmbid* vacío. Por estas razones, es preciso realizar un preprocesamiento en la información.

En esta etapa de filtrado lo que se busca es obviar los registros del conjunto de datos que no generen valor. Por lo tanto, se decidió considerar la siguiente secuencia de pasos para la limpieza de la información:

1. Tener en cuenta sólo a los usuarios que especifiquen su país de origen, pues más adelante este atributo será usado para aplicar algoritmos de filtrado colaborativo basado en atributos. Excluyendo así a 289 usuarios.

2. No considerar el atributo género para el análisis del trabajo, pues no genera valor al figurar *"None"* en todos los registros.

3. Tener en cuenta sólo los registros donde el atributo *trackmbid* tenga un valor válido; es decir, que no sea nulo. Excluyendo así a 2849908 registros.

4. Sólo se van a considerar las canciones que fueron escuchadas por lo menos 10 veces por el mismo usuario. Pues hay registros que no generan valor para la recomendación, ya que hay pistas de canciones que han sido escuchadas una o muy pocas veces por un mismo usuario.

Para almacenar este nuevo conjunto de datos, se hará uso de la tabla agregada *musiclist_agr* mencionada en el apartado 3.4.1Preparación de la Base de Datossubsubsection.3.4.1. En esta tabla se registrará la información de forma agregada, por lo tanto no tendrá el mismo detalle que la tabla *musiclistened*, donde cada registro mostraba el momento en el que un usuario escuchaba una canción en particular. En este caso, esta tabla agregada servirá para mostrar cuántas veces ha escuchado el usuario una misma canción en estos 2 meses de análisis de información.

En la tabla 5Campos de la tabla musiclist_agrtable.5 se observa que además de los campos conocidos: *username*, *country* y *trackmbid*, figuran otros campos como son: *timeslistened*, *playsuser*, *freqlistened* y *rating*. Los primeros 3 campos son calculados en el momento que se ejecuta la función *insertMusiclist_agr* descrita en la tabla 8Función: insertMusiclist_agrtable.8. El campo *rating* queda vacío porque requiere que los campos descritos anteriormente, estén llenos.

```
--Insert de los campos: username, country, trackmbid, timeslistened, playsuser, freqlistened.

insert into musiclist_agr2
    select b.username,b.country,b.trackmbid, b.playsnum,a.totalplays,b.playsnum/a.totalplays :: float
    from (select username, sum(playsnum) totalplays
          from (select  username, country, trackmbid, count(1) playsnum
                from musiclistened2
                where trackmbid <> ''
                and country <> 'None'
                group by 1,2,3
                having count(1)>5) q
          group by 1) a,
          (select  username, country, trackmbid, count(1) playsnum
          from musiclistened2
          where trackmbid <> ''
          and country <> 'None'
          group by 1,2,3
          having count(1)>5)b
    where a.username=b.username
    order by 1;
```

Tabla 8: Función: insertMusiclist_agr

Como se observa en la función *insertMusiclist_agr*, el campo *timeslistened* es calculado con el número de veces que un usuario escucha una canción; el campo *playsuser* es calculado con el número de *plays* que un usuario ha realizado en estos 2 meses de análisis; el campo *freqlistened* es calculado con la división de los 2 campos anteriores. Por lo tanto, el campo *freqlistened* contiene la frecuencia de *plays* para una determinada canción y usuario.

Una vez ejecutada la función *insertMusiclist_agr*, la tabla *musiclist_agr* contiene la información que aparece en la figura 10Muestra de tabla agregada sin ratingfigure.10.

Como se puede observar, se confirma que el único campo vacío es el *rating* cuya función de actualización será detallado en el siguiente apartado.

3.4.6. Cálculo del Rating

El conjunto de datos no contiene ninguna información sobre las preferencias del usuario por las canciones que escucha, por lo tanto, será necesario estimar esta medida, que será llamada rating, en base a la cantidad de plays que el usuario da a una canción, pues es la única información disponible que encierra de forma implícita las preferencias del usuario. Se ha seguido el método Pacula [16] que se basa en la

username character varying(80)	country text	trackmbid character varying(36)	timeslistened integer	playuser integer	freqlistened numeric(10,8)	rating numeric(10,8)
DouglasPaixao	Brazil	da74c3a8-7915-4df8-9b2c-7e24ae899878	9	67	0.13432836	
DouglasPaixao	Brazil	9a5d99e2-9e6f-4acd-a1d6-1bccaff82914	6	67	0.08955224	
DouglasPaixao	Brazil	8ff03177-33b4-4e61-bb05-9cc6b9f1553b	11	67	0.16417910	
DouglasPaixao	Brazil	7bb4d31f-8470-43b3-91a9-adb8be434b0a	8	67	0.11940299	
DouglasPaixao	Brazil	11a32de6-bce4-4c8a-aac7-849996f3ecd6	8	67	0.11940299	
DouglasPaixao	Brazil	bb76dcf3-9b37-4b75-98e5-5689e24d528f	8	67	0.11940299	
DouglasPaixao	Brazil	08d07438-9b9c-4c41-a1d5-7211a32cc9ad	9	67	0.13432836	
DouglasPaixao	Brazil	c9017c7b-0d67-4cac-8626-35e4b73e5285	8	67	0.11940299	
ElintonLima	Brazil	1153e59b-615c-459c-ad88-1ba8a64340fa	51	672	0.07589286	
ElintonLima	Brazil	fb2c6e4e-4fcb-49a5-a49e-a7c3018ae220	8	672	0.01190476	
ElintonLima	Brazil	dd7b009d-8890-4ab6-b6f7-cde626c33ed9	34	672	0.05059524	

Figura 10: Muestra de tabla agregada sin rating

frecuencia de *plays* donde claramente existe una distribución de ley de potencias, pues hay pocas canciones escuchadas muy frecuentemente y la mayoría de ellas tienen pocos *plays*. La figura 11Distribución de ley de potencia de las frecuencias de playsfigure.11 muestra esta distribución de frecuencias en el conjunto de datos utilizados en este trabajo.

Figura 11: Distribución de ley de potencia de las frecuencias de plays

La frecuencia de *plays* de una determinada canción i y un usuario j se define de la siguiente manera:

$$Freq_{i,j} = \frac{p_{i,j}}{\sum_{i'} p_{i',j}} \tag{6}$$

Donde $p_{i,j}$ es el número de veces que un usuario j escucha una canción i. Por otro lado, $Freq_k(j)$ denota la k-ésima canción más escuchada por el usuario j. Entonces, una calificación para una canción con rango k se calcula como una función lineal de la frecuencia percentil:

$$r_{i,j} = 4(1 - \sum_{k'=1}^{k-1} Freq_{k'}(j)) \tag{7}$$

Una vez que el *rating* es calculado, los métodos de filtrado colaborativo pueden ser aplicados tal y como se aplican en un conjunto de datos que contiene las preferencias del usuario de forma explícita.

Como ya se mencionó, la tabla *musiclist_ agr* contiene todos los campos excepto el campo *rating*. En este apartado se describe la manera en que este campo es calculado.

Para hallar el valor del *rating* por usuario y canción, es necesario el uso del campo *freqlistened*. Para este fin, se ejecutó la función *updateRating* descrita en la tabla 9Función: updateRatingtable.9. Esta función ha sido desarrollada en base a la fórmula 7Cálculo del Ratingequation.3.7.

```
--Creacion de funcion que Actualiza el Rating que esta en Null

create or replace function updateRating()
returns void as
$BODY$
declare
  cur cursor for select DISTINCT username from musiclist_agr where rating is null;
  rec RECORD;
Begin
  open cur;
  loop
  fetch cur into rec;
    exit when not found;

        update musiclist_agr
        set rating = sq.rating2
        from
        (Select a.username, a.trackmbid, a.rating,
        4.0 * (1-(CASE WHEN sum(b.freqlistened) is NULL THEN 0 ELSE sum(b.freqlistened) END)) rating2
        from (select username, trackmbid, freqlistened, rating
              from musiclist_agr
              where username=rec.username
              and rating is null
              ) a left outer join (select username, trackmbid, freqlistened
                                   from musiclist_agr
                                   where username=rec.username
                                   ) b on a.username=b.username
             and a.freqlistened < b.freqlistened
             group by 1,2,3) sq
        where musiclist_agr.username=sq.username
        and musiclist_agr.trackmbid=sq.trackmbid;
  end loop;
  close cur;
End;
$BODY$
Language PLPGSQL;
```

Tabla 9: Función: updateRating

3.4.7. Características del Conjunto de Datos Final

En esta sección se presenta las características del conjunto de datos final despúes de la etapa de preprocesamiento.

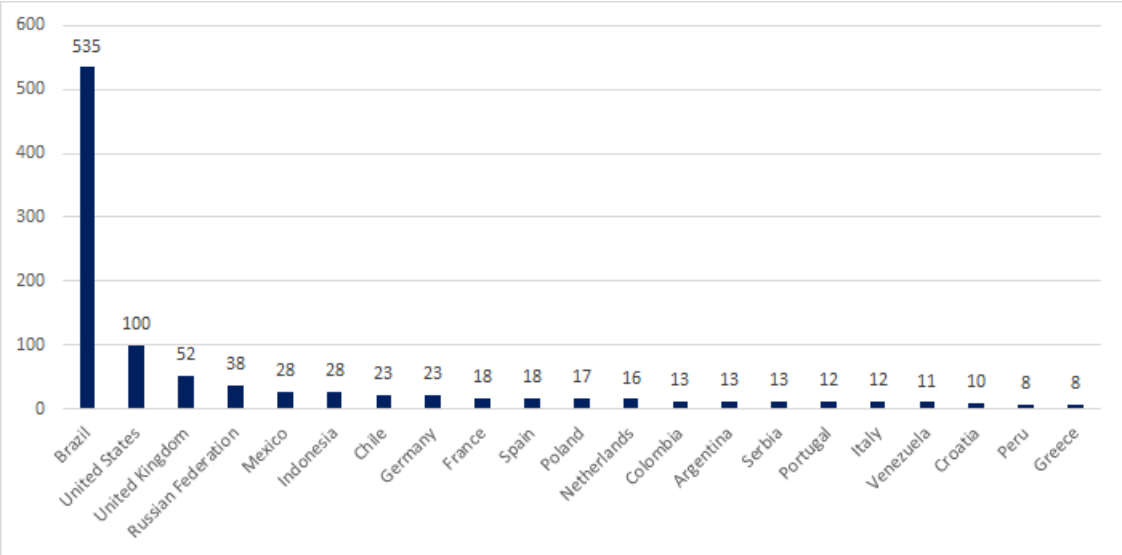

Figura 12: Número final de usuarios por país (20 primeros)

Cantidad de Usuarios	820
Cantidad de Artistas con Id	2544
Cantidad de Canciones con Id	13888
Cantidad de Registros Totales	22501

Tabla 10: Número de registros del conjunto de datos final

3.5. Implementación del Sistema de Recomendación de Música

El conjunto de datos final disponible constituye la entrada a los procesos de recomendación que se han implementado. Los procesos se dividen en dos: Recomendación basada en Rating y Recomendación de ítems. A continuación, se explicarán los procesos implementados en el trabajo.

3.5.1. Recomendación basada en Rating

Este tipo de recomendación no produce una lista de canciones sino que sólo proporciona la predicción del *rating* de las mismas para un usuario determinado. Sin embargo, se pueden utilizar estas predicciones para elaborar listas de reproducción personalizadas para los usuarios que incluyan aquellas canciones que tengan los mejores valores de los *ratings* predichos para cada uno de ellos.

La figura 13Recomendación basada en Ratingfigure.13 representa el proceso seguido para la recomendación basada en el *rating*. El operador *"Read CSV"* toma los datos de entrada del archivo excel generado con el conjunto de datos final. El operador *"Attributes"* sólo selecciona los campos de interés para aplicar el algoritmo de recomendación: *username*, *trackmbid* y *rating*. El operador *"Set Role"* sirve para definir los roles de los atributos; en este caso se definen los atributos *username*, *trackmbid* y *rating* con los roles *user identification*, *item identification* y *label*, respectivamente. Por último, el operador *"Validation"* realiza la validación cruzada con el fin de estimar las métricas de calidad del algoritmo de recomendación. Es un operador anidado. Tiene dos subprocesos: uno de entrenamiento y otro de pruebas.

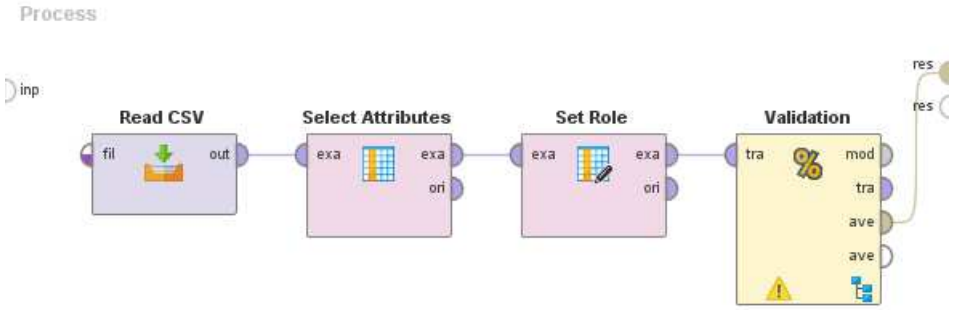

Figura 13: Recomendación basada en Rating

El sub-proceso de entrenamiento es utilizado para inducir el modelo, el cual se aplica posteriormente en el sub-proceso de pruebas. Durante esta fase de prueba se miden también las métricas de calidad del modelo. Para el trabajo se han desarrollado varios procesos usando tres algoritmos de recomendación: *User k-NN*, *Item k-NN* y *User Attribute k-NN*.

- **Algoritmo *User k-NN***: Con este algoritmo se aplica la técnica de filtrado colaborativo de los vecinos más cercanos basado en usuario. Es decir, calcula

la similitud mediante la búsqueda de un grupo de k usuarios que han valorado las canciones de una manera muy similar al usuario en cuestión. Para esto se deben establecer dos opciones: el número de vecinos más próximos, k (por defecto es 80) y la medida de similitud. La similitud Coseno es muy popular, sin embargo el coeficiente de correlación de Pearson tiende a ser un poco más precisa. La figura 14Recomendación basada en Rating: User k-NNfigure.14 muestra cómo fue implementada la validación de este algoritmo.

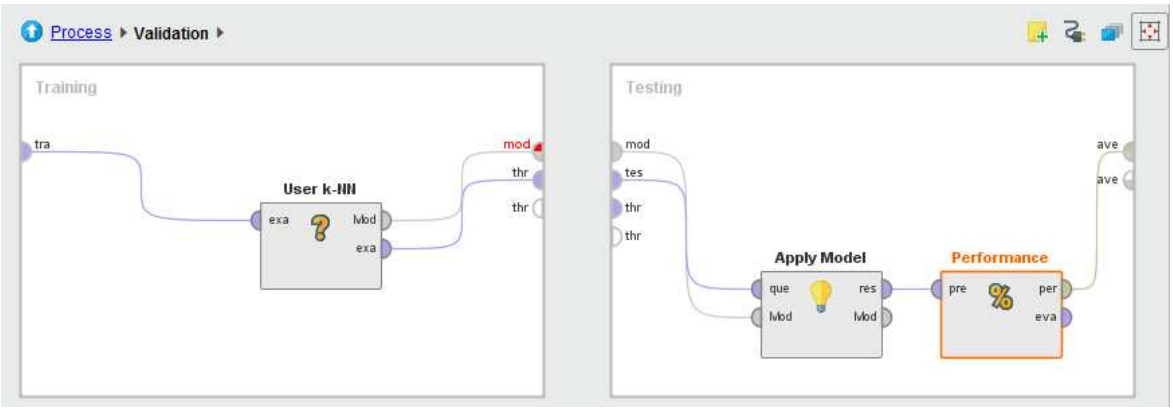

Figura 14: Recomendación basada en Rating: User k-NN

- **Algoritmo *Item k-NN***: Con este algoritmo se aplica la técnica de filtrado colaborativo de los vecinos más cercanos basado en ítems. Es decir que es similar al algoritmo *User k-NN* pero en este caso, la métrica de similitud es aplicada entre los ítems. La manera en cómo se implementó esta validación se muestra en la figura 15Recomendación basada en Rating: Item k-NNfigure.15.

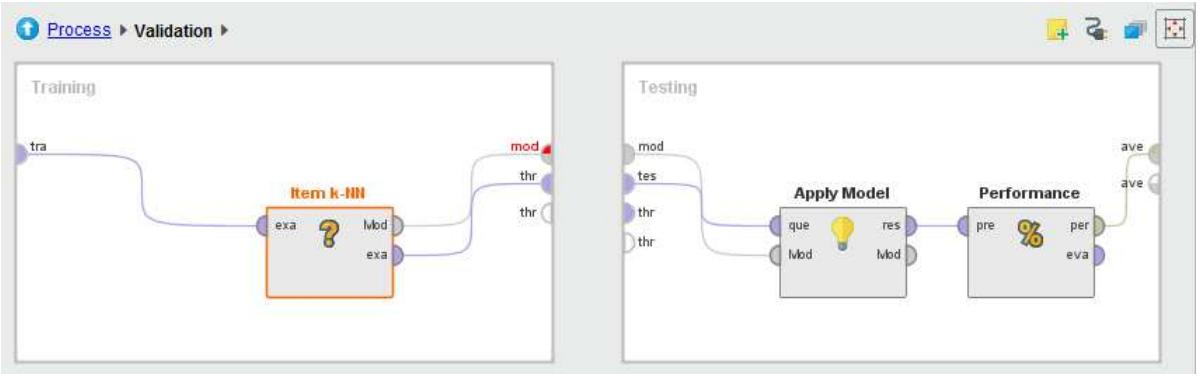

Figura 15: Recomendación basada en Rating: Item k-NN

- **Algoritmo *User Attribute k-NN***: Con este algoritmo se aplica la técnica de recomendación basada en características del usuario, en este caso se basa en el atributo *país* del usuario. Hay que señalar que no es posible utilizar otro atributo del usuario, pues el *país* es el único atributo disponible en el conjunto

de datos. Los operadores usados para construir esta validación se muestran en la figura 16Recomendación basada en Rating: User Attribute k-NNfigure.16.

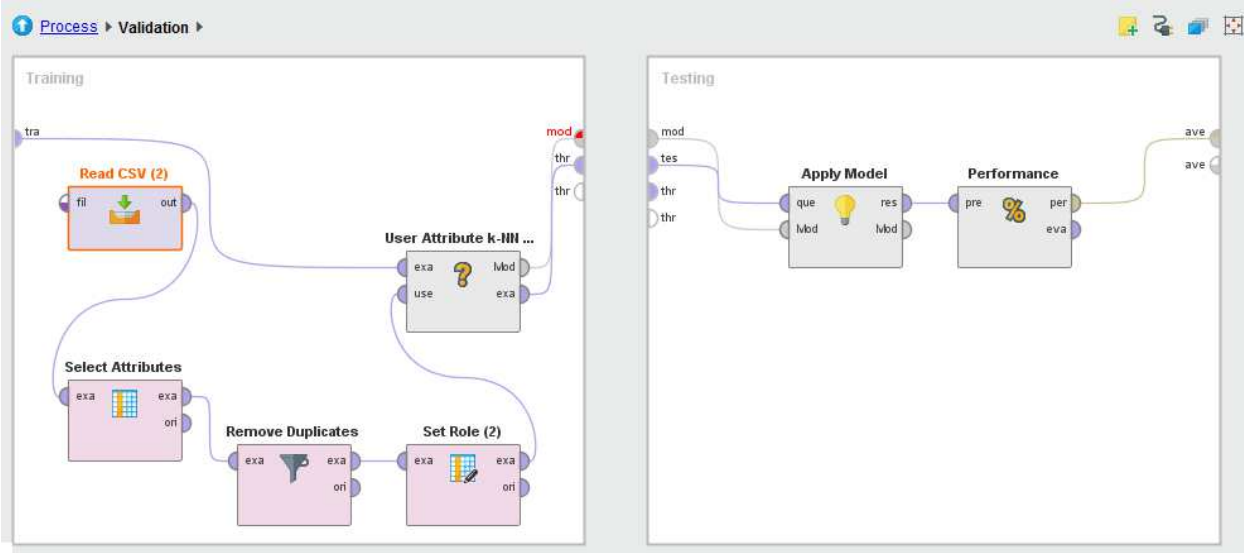

Figura 16: Recomendación basada en Rating: User Attribute k-NN

- **Algoritmo *Item Attribute k-NN***: Con este algoritmo se aplica la técnica de recomendación basada en características del ítem, en este caso se basa en el atributo *artista* de la canción. La validación de este algoritmo fue implementada como se muestra en la figura 17Recomendación basada en Rating: Item Attribute k-NNfigure.17.

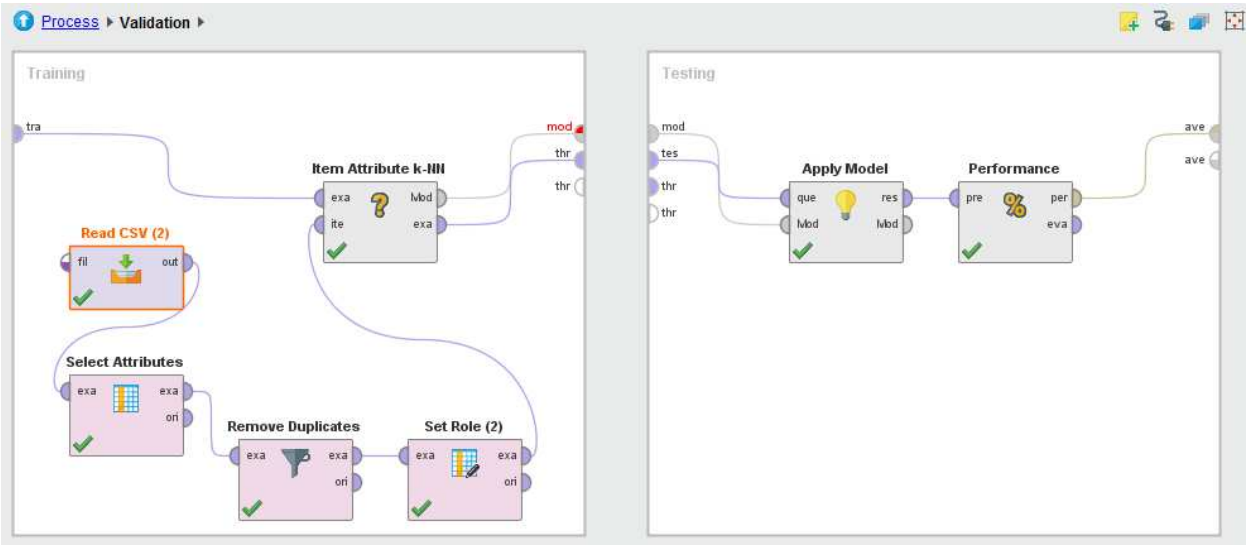

Figura 17: Recomendación basada en Rating: Item Attribute k-NN

3.5.2. Recomendación de Items

La figura 18Recomendación de Itemsfigure.18 representa el proceso que se sigue en la recomendación de ítems. El operador *"Read CSV"* toma como datos de entrada al archivo excel generado con el conjunto de datos final. El operador *" Attributes"* sólo selecciona los campos de interés para aplicar el algoritmo de recomendación: *username*, *trackmbid* y *rating*. El operador *"Set Role"* sirve para definir los roles de los atributos; en este caso se definen los atributos *username*, *trackmbid* y *rating* con los roles *user identification*, *item identification* y *label*, respectivamente. Por último, el operador *"Validation"* realiza la validación cruzada con el fin de estimar las métricas de calidad del algoritmo de recomendación. Es un operador anidado. Tiene dos subprocesos: uno de entrenamiento y otro de pruebas.

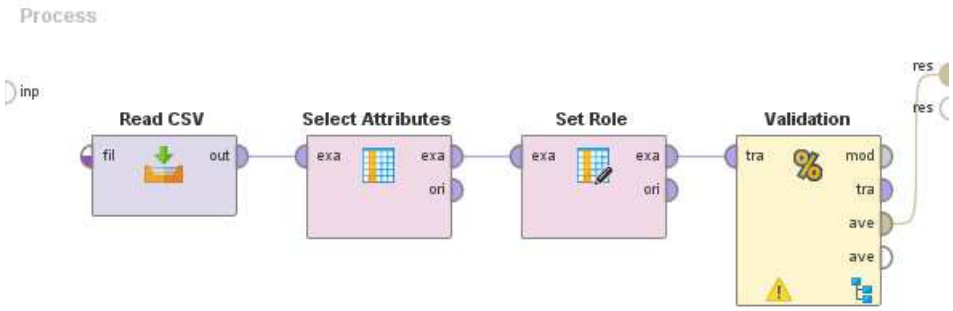

Figura 18: Recomendación de Items

- **Algoritmo *User k-NN***: De la misma forma que el algoritmo *User k-NN* aplicado en la recomendación basada en *rating*, con este algoritmo se aplica la técnica de filtrado colaborativo de los vecinos más cercanos basado en el usuario. La validación fue implementada como se muestra en la figura 19Recomendación de Items: User k-NNfigure.19.

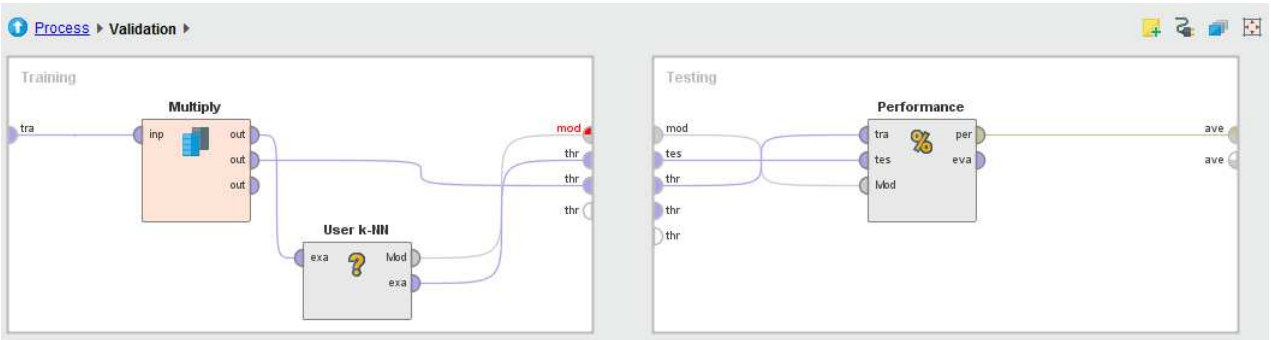

Figura 19: Recomendación de Items: User k-NN

- **Algoritmo *Item k-NN***: Al igual que el algoritmo Item k-NN aplicado en la recomendación basada en *rating*, con este algoritmo se aplica la técnica de filtrado colaborativo de los vecinos más cercanos basado en ítems. La figura 20Recomendación de Items: Item k-NNfigure.20 presenta cómo fue implementado este algoritmo.

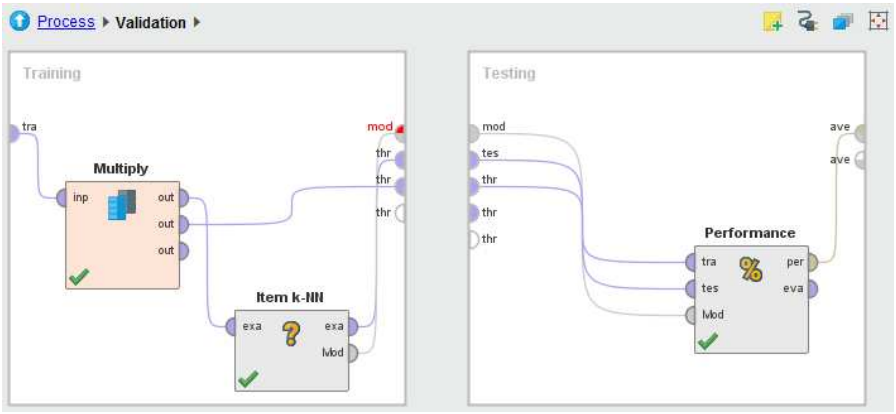

Figura 20: Recomendación de Items: Item k-NN

- **Algoritmo *User Attribute k-NN***: Al igual que el algoritmo *User Attribute k-NN* aplicado en la recomendación basada en Rating, con este algoritmo se aplica la técnica de recomendación basado en el *país* del usuario. La validación fue implementada como se muestra en la figura 21Recomendación de Items: User Attribute k-NNfigure.21.

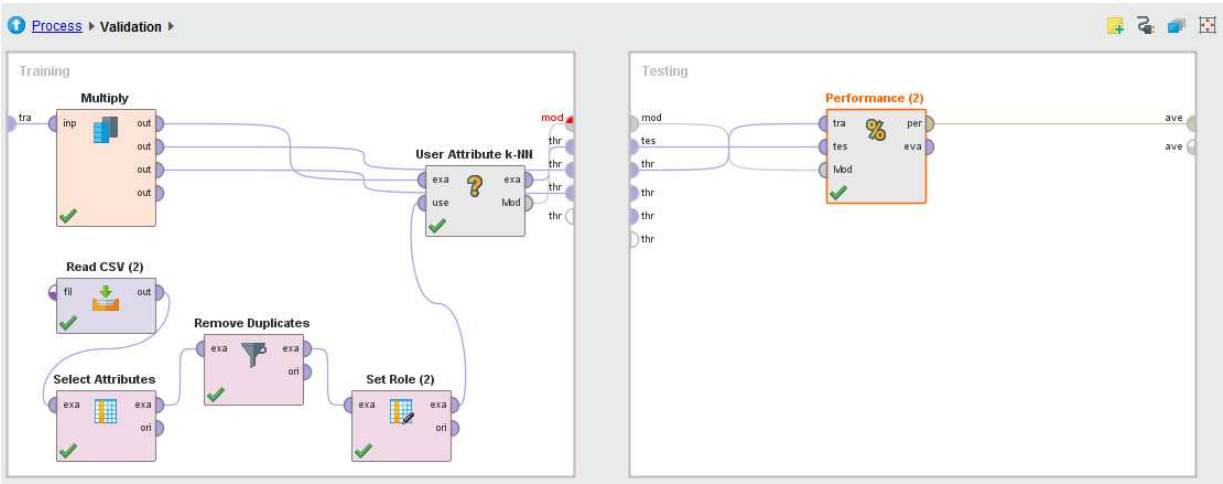

Figura 21: Recomendación de Items: User Attribute k-NN

- **Algoritmo *Item Attribute k-NN***: Al igual que el algoritmo *Item Attribute k-NN* aplicado en la recomendación basada en *rating*, con este algoritmo se aplica la técnica de recomendación basada en contenido, en este caso en el *artista* de la canción. La figura 22Recomendación de Items: Item Attribute k-NNfigure.22 muestra los operadores que se usaron para implementar esta validación.

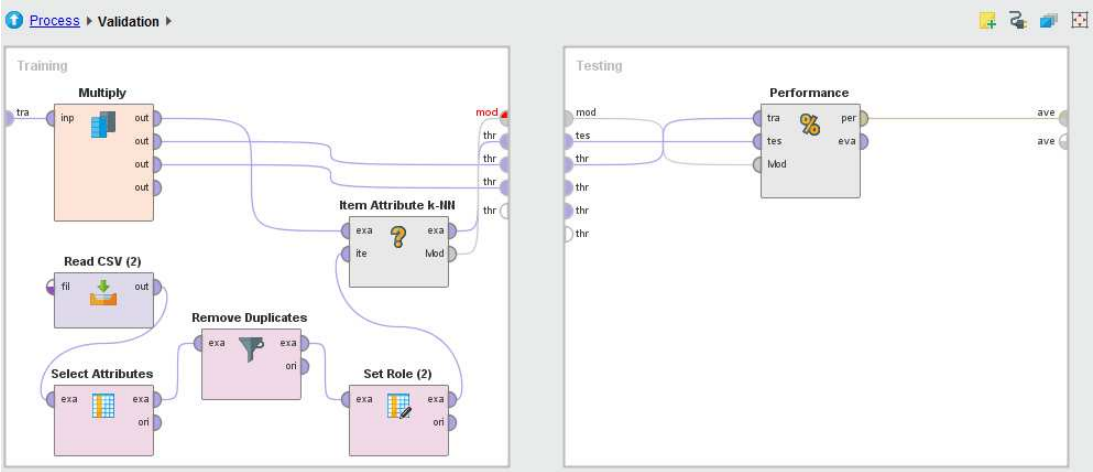

Figura 22: Recomendación de Items: Item Attribute k-NN

- **Híbrido (*User k-NN* + *Item Attribute k-NN)***: Como ya se ha comentado en el apartado 2.1Sistemas de Recomendaciónsubsection.2.1, el enfoque híbrido de los sistemas de recomendación consiste en la combinación de diferentes técnicas de recomendación con el fin de lograr un mejor comportamiento de los algoritmos y evitar limitaciones de las técnicas individuales. En este caso se han combinado los siguientes algoritmos aplicados de forma independiente anteriormente: *User k-NN* e *Item Attribute k-NN*. Los operadores usados para construir esta validación se muestran en la figura 23Recomendación de Items: User k-NN e Item Attribute k-NNfigure.23.

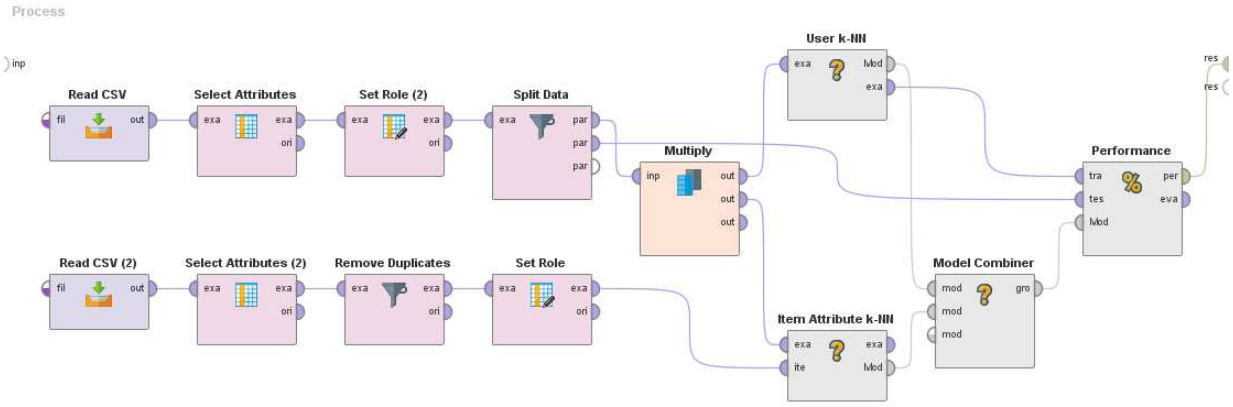

Figura 23: Recomendación de Items: User k-NN e Item Attribute k-NN

4. Experimentación

En esta sección se detallan los resultados de la ejecución de los procesos implementados en el apartado 3.5Implementación del Sistema de Recomendación de Músicasubsection.3.5.

4.1. Recomendación basada en Rating

La manera más habitual de validar los resultados de la predicción de las calificaciones (*ratings*) es calcular la desviación entre las calificaciones reales y predichas. Las métricas más conocidas en la literatura de los sistemas de recomendación son: MAE (*Mean Absolute Error*), $NMAE$ (*Normalized Mean Absolute Error*) y $RMSE$ (*Root-Mean-Square Error*).

La métrica MAE se define como el promedio del error absoluto o diferencia entre la calificación real (r) y la calificación predicha (pr).

$$MAE = \frac{1}{c}\sum_{i=1}^{c} \mid r_{i,j} - pr_{i,j} \mid \tag{8}$$

Donde c es el número de ítems que el usuario j ha evaluado, $r(i,j)$ es la calificación que el usuario j ha dado al ítem i y $pr_{(i,j)}$ es la calificación predicha para el usuario j y el ítem i.

La métrica $NMAE$ se define como la precisión independiente del rango de evaluación.

$$NMAE = \frac{MAE}{r_{max} - r_{min}} \tag{9}$$

La métrica $RMSE$ representa la desviación estándar de las diferencias entre los

valores predichos y los valores observados. Es decir, cuanto menor es el $RMSE$, menor es el error, mayor será la precisión, y por lo tanto mejor es la recomendación. $RMSE$ es una buena medida para comparar los errores de los diferentes modelos de predicción.

$$RMSE = \sqrt{\frac{1}{c}\sum_{i=1}^{c}(r_{i,j} - pr_{i,j})^2} \tag{10}$$

Seguidamente se muestran los resultados de la ejecución de los procesos implementados en el apartado 3.5.1Recomendación basada en Ratingsubsubsection.3.5.1. Cada algoritmo presenta las métricas anteriormente expuestas.

4.1.1. Algoritmo *User k-NN*

Se aplicó este algoritmo de recomendación de dos formas. Una usando la similitud Coseno y la otra, con el coeficiente de correlación de Pearson. Ambas aplicaciones se hicieron con el valor de k=80. Las medidas de calidad de las predicciones fueron:

Algoritmo	**RMSE**	**MAE**	**NMAE**
User k-NN (Coseno)	1.178	0.96	0.24
User k-NN (Pearson)	1.15	0.938	0.235

Tabla 11: Medidas de User k-NN (Pred. de rating)

4.1.2. Algoritmo *Item k-NN*

Del mismo modo que el anterior algoritmo, se usaron las dos medidas de similitud con el valor de k=80. Las medidas resultantes fueron:

Algoritmo	**RMSE**	**MAE**	**NMAE**
Item k-NN (Coseno)	1.077	0.875	0.219
Item k-NN (Pearson)	1.081	0.882	0.221

Tabla 12: Medidas de Item k-NN (Pred. de rating)

4.1.3. Algoritmo *User Attribute k-NN*

Para este algoritmo sólo se puede aplicar la medida de similitud del coseno. Las medidas fueron:

Algoritmo	**RMSE**	**MAE**	**NMAE**
User Attribute k-NN	1.159	0.945	0.236

Tabla 13: Medidas de User Attribute k-NN (Pred. de rating)

4.1.4. Algoritmo *Item Attribute k-NN*

Igual que en el caso anterior, en este algoritmo sólo se puede emplear la medida de similitud del coseno. Los resultados fueron:

Algoritmo	**RMSE**	**MAE**	**NMAE**
Item Attribute k-NN	1.041	0.839	0.21

Tabla 14: Medidas de Item Attribute k-NN (Pred. de rating)

4.2. Recomendación de Items

Por otro lado, la evaluación de las N principales recomendaciones requiere diferentes métricas para aplicar a la lista clasificada proporcionada al usuario. Dos métricas habituales son MAP (*Mean average Precision*) y $NDCG$ (*Normalized Discounted Cumulative Gain*).

MAP es el promedio del valor de precisión obtenida para cada ítem de la lista de los N principales.

$NDCG$ es una medida basada en la suposición de que cuanto menor sea la posición clasificada de un ítem relevante, menos útil es para el usuario. Al aplicar esta medida, la ganancia se acumula a partir del primer lugar de la lista. El valor de relevancia graduada de ítems en posiciones más bajas se reduce en una cantidad logarítmicamente proporcional a la posición del resultado. DCG es el resultado acumulado descontado en una clasificación k particular.

A continuación, se muestran los resultados de los procesos descritos en el apartado 3.5.2Recomendación de Itemssubsubsection.3.5.2. Cabe mencionar que se hicieron varias pruebas modificando el valor de k, y con el valor de k=30 se logró mejores medidas de precisión.

4.2.1. Algoritmo *User k-NN*

Se aplicó este algoritmo de recomendación. Las medidas de calidad de las predicciones fueron:

Algoritmo	**AUC**	**NDCG**	**MAP**
User k-NN	0.754	0.238	0.083

Tabla 15: Medidas de User k-NN (Recom. de ítems)

4.2.2. Algoritmo *Item k-NN*

Se aplicó este algoritmo de recomendación y generó las siguiente medidas de calidad:

Algoritmo	**AUC**	**NDCG**	**MAP**
Item k-NN	0.59	0.196	0.055

Tabla 16: Medidas de Item k-NN (Recom. de ítems)

4.2.3. Algoritmo *User Attribute k-NN*

Al aplicar este algoritmo de recomendación se generaron las siguientes medidas de calidad:

Algoritmo	AUC	NDCG	MAP
User Attribute k-NN	0.439	0.116	0.005

Tabla 17: Medidas de User Attribute k-NN (Recom. de ítems)

4.2.4. Algoritmo *Item Attribute k-NN*

Se aplicó este algoritmo de recomendación y las medidas de calidad fueron:

Algoritmo	AUC	NDCG	MAP
Item Attribute k-NN	0.773	0.256	0.091

Tabla 18: Medidas de Item Attribute k-NN (Recom. de ítems)

4.2.5. Algoritmos: *User k-NN + Item Attribute k-NN*

Se aplicó este algoritmo de recomendación con diferentes valores de k (30,40,50,80). Las medidas de calidad para cada una de ellas fueron:

Algoritmo	AUC	NDCG	MAP
User+Item Attr (k=80)	0.837	0.333	0.145
User+Item Attr (k=50)	0.806	0.335	0.146
User+Item Attr (k=40)	0.805	0.337	0.148
User+Item Attr (k=30)	0.793	0.337	0.149

Tabla 19: Medidas de User k-NN + Item Attribute k-NN (Recom. de ítems)

Se puede observar que al variar el valor de k, la medida AUC disminuye ligeramente, sin embargo, este valor se mantuvo alto, en torno al 80 %.

5. Evaluación de Resultados

En este apartado se evalúan las medidas de calidad calculadas anteriomente.

5.1. Recomendación basada en Rating

La figura 24NMAE de cada algoritmo de recomendación basada en ratingfigure.24 muestra las medidas de error normalizado por cada algoritmo utilizado en la predicción de *ratings*.

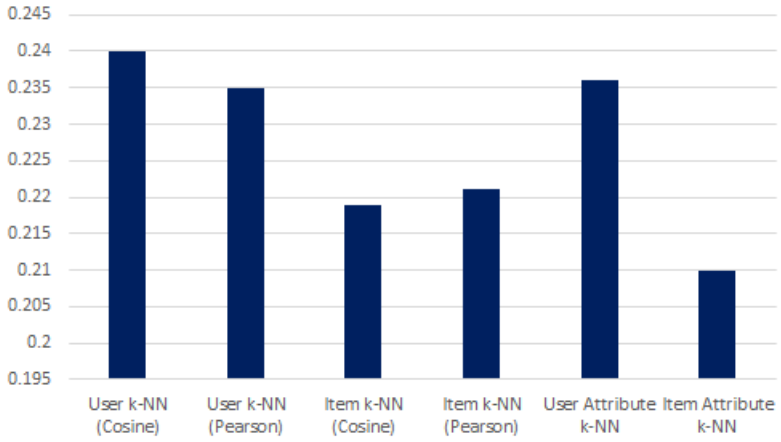

Figura 24: NMAE de cada algoritmo de recomendación basada en rating

Se puede observar que la medida de error es menor al aplicar el algoritmo *Item-Attribute k-NN* con la métrica de similitud coseno. Este algoritmo es un método basado en contenido puesto que hace uso de un atributo del ítem. Para este trabajo se usó el *artista* (atributo de la canción) para realizar la predicción del *rating*.

5.2. Recomendación de Items

En la figura 25AUC y MAP de cada algoritmo con k=30figure.25 se muestran las medidas de rendimiento AUC y MAP que se generaron al aplicar cada uno de los algoritmos para recomendar ítems.

Se puede notar que el modelo híbrido, que combina el método de filtrado colaborativo de los vecinos más cercanos basado en usuario (User k-NN) y el método basado en contenido (Item Attribute k-NN con el atributo artista de la canción), proporcionó mayores valores de las medidas de calidad que el resto de técnicas aplicadas. Asimismo, los valores más bajos de estas medidas se obtienen cuando se aplica la técnica basada en el atributo del usuario (user Attribute k-NN con el atributo pais).

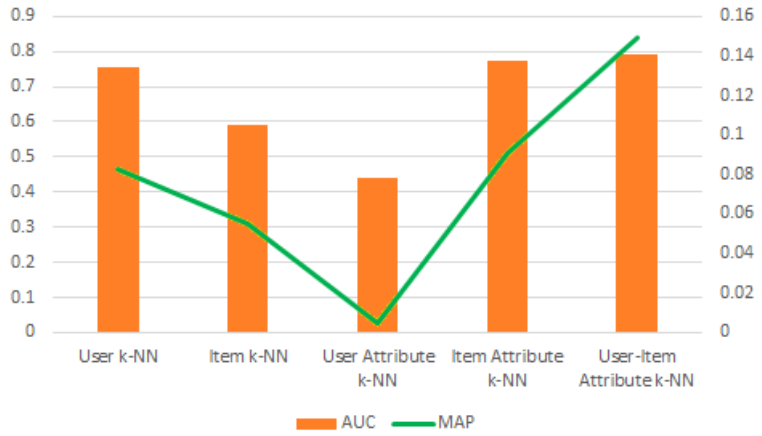

Figura 25: AUC y MAP de cada algoritmo con k=30

El motivo de haber asignado a k el valor de 30 se debe al hecho de que fue el valor que proporcionó mejores resultados con la mayoría de los métodos. Se realizó un estudio comparativo de las medidas de calidad de la recomendación de todos los algoritmos para diferentes valores de k. La figura 26AUC y MAP variando el kfigure.26 muestra la variación de las medidas en el caso del algoritmo híbrido (User k-NN e Item Attribute k-NN).

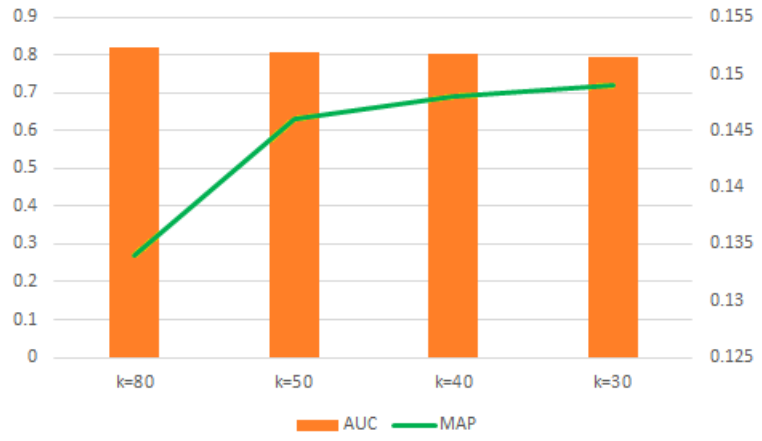

Figura 26: AUC y MAP variando el k

Se puede apreciar que a medida que el valor de k disminuye, la medida MAP aumenta y AUC disminuye ligeramente, manteniendo un valor alto en torno al 80 %.

6. Conclusiones y Trabajos Futuros

En este trabajo se han aplicado diferentes técnicas de recomendación de filtrado colaborativo y de contenido, con el fin de generar listas de reproducción de música a partir de las recomendaciones obtenidas. El estudio experimental se ha realizado con datos obtenidos mediante la API de Last.fm. En primer lugar, se ha comprobado que en la categoría de los algoritmos de predicción de *rating*, el método de la recomendación basado en contenido, que utiliza como atributo el artista de la canción, genera una mejor predicción que las demás técnicas. En segundo lugar, al aplicar los algoritmos de recomendación de ítems, se observó que el modelo híbrido es el que clasifica mejor las piezas musicales, porque además de considerar el comportamiento del usuario, tiene en cuenta al artista de la canción. En base a estos dos enfoques se deduce que el artista es un aspecto importante del contenido de la canción. Por el contrario, el país es un atributo del usuario que no añade valor a la recomendación. Hubiera sido interesante contar con atributos del usuario como la edad o género para observar el comportamiento de los algoritmos basados en el atributo del usuario (User Attribute k-NN), sin embargo estas pruebas no se pudieron llevar a cabo porque estos atributos no estaban disponibles en el API de Last.fm.

Teniendo en cuenta la importancia actual de los sistemas de recomendación en el campo de la música y sus variadas líneas de investigación, se propone como trabajos futuros investigar sobre técnicas de hibridación con otro tipo de contenido que mejoren las pruebas realizadas en el trabajo. Por otra parte, dado que el elevado número de canciones a recomendar no permite alcanzar una alta fiabilidad en las recomendaciones, podría recurrirse a algún procedimiento de filtrado basado en la inducción previa de modelos de recomendación de artistas.

Referencias

[1] Rapidminer. https://es.wikipedia.org/wiki/RapidMiner. Fecha de consulta: 2016-06-15. [*Citado en pág. 14.*]

[2] Nicholas J Belkin and W Bruce Croft. Information filtering and information retrieval: Two sides of the same coin? *Communications of the ACM*, 35(12):29–38, 1992. [*Citado en pág. 3.*]

[3] Dmitry Bogdanov, Martín Haro, Ferdinand Fuhrmann, Emilia Gómez, and Perfecto Herrera. Content-based music recommendation based on user preference examples. *Copyright Information*, page 33, 2010. [*Citado en pág. 7.*]

[4] John S Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 43–52. Morgan Kaufmann Publishers Inc., 1998. [*Citado en pág. 6.*]

[5] Mário João Teixeira Carneiro et al. Towards the discovery of temporal patterns in music listening using last. fm profiles. 2012. [*Citado en pág. 10.*]

[6] Òscar Celma Herrada. Music recommendation and discovery in the long tail. 2009. [*Citado en pág. 3.*]

[7] Mark Claypool, Anuja Gokhale, Tim Miranda, Pavel Murnikov, Dmitry Netes, and Matthew Sartin. Combining content-based and collaborative filters in an online newspaper. In *Proceedings of ACM SIGIR workshop on recommender systems*, volume 60. Citeseer, 1999. [*Citado en pág. 4.*]

[8] Ricardo Dias and Manuel J Fonseca. Improving music recommendation in session-based collaborative filtering by using temporal context. In *2013 IEEE 25th International Conference on Tools with Artificial Intelligence*, pages 783–788. IEEE, 2013. [*Citado en pág. 8.*]

[9] Marcos Aurélio Domingues, Fabien Gouyon, Alípio Mário Jorge, José Paulo Leal, João Vinagre, Luís Lemos, and Mohamed Sordo. Combining usage and content in an online recommendation system for music in the long tail. *International Journal of Multimedia Information Retrieval*, 2(1):3–13, 2013. [*Citado en pág. 8.*]

[10] Thomas Hornung, Cai-Nicolas Ziegler, Simon Franz, Martin Przyjaciel-Zablocki, Alexander Schätzle, and Georg Lausen. Evaluating hybrid music recommender systems. In *Proceedings of the 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)-Volume 01*, pages 57–64. IEEE Computer Society, 2013. [*Citado en pág. 8.*]

[11] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272. Ieee, 2008. [*Citado en pág. 8.*]

[12] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. *Recommender systems: an introduction*. Cambridge University Press, 2010. [*Citado en pág. 3.*]

[13] Matej Mihelčić, Nino Antulov-Fantulin, Matko Bošnjak, and Tomislav Šmuc. Extending rapidminer with recommender systems algorithms. In *RapidMiner Community Meeting and Conference (RCOMM 2012)*, 2012. [*Citado en págs. 15 y 16.*]

[14] Miquel Montaner Rigall et al. *Collaborative recommender agents based on case-based reasoning and trust*. Universitat de Girona, 2003. [*Citado en pág. 7.*]

[15] Joshua L Moore, Shuo Chen, Thorsten Joachims, and Douglas Turnbull. Learning to embed songs and tags for playlist prediction. In *ISMIR*, pages 349–354, 2012. [*Citado en pág. 1.*]

[16] Maciej Pacula. A matrix factorization algorithm for music recommendation using implicit user feedback. 2009. [*Citado en pág. 24.*]

[17] Sung Eun Park, Sangkeun Lee, and Sang-goo Lee. Session-based collaborative filtering for predicting the next song. In *Computers, Networks, Systems and Industrial Engineering (CNSI), 2011 First ACIS/JNU International Conference on*, pages 353–358. IEEE, 2011. [*Citado en pág. 7.*]

[18] Waleed Reafee, Naomie Salim, and Atif Khan. The power of implicit social relation in rating prediction of social recommender systems. *PloS one*, 11(5):e0154848, 2016. [*Citado en pág. 9.*]

[19] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001. [*Citado en págs. 4, 5, 6 y 7.*]

[20] J Ben Schafer, Joseph A Konstan, and John Riedl. E-commerce recommendation applications. In *Applications of Data Mining to Electronic Commerce*, pages 115–153. Springer, 2001. [*Citado en págs. 5, 6 y 7.*]

[21] Ja-Hwung Su, Wei-Yi Chang, and Vincent S Tseng. Personalized music recommendation by mining social media tags. *Procedia Computer Science*, 22:303–312, 2013. [*Citado en pág. 8.*]

[22] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems*, pages 2643–2651, 2013. [*Citado en pág. 8.*]

[23] Bo Xiao and Izak Benbasat. E-commerce product recommendation agents: Use, characteristics, and impact. *Mis Quarterly*, 31(1):137–209, 2007. [*Citado en pág. 3.*]

[24] Zhe Xing, Xinxi Wang, and Ye Wang. Enhancing collaborative filtering music recommendation by balancing exploration and exploitation. In *ISMIR*, pages 445–450, 2014. [*Citado en pág. 8.*]