PGMÚSICA
SISTEMA DE RECOMENDACIÓN DE MÚSICA

Tutores

Msc. Guillermo Moncecchi
Msc. Diego Garat

Integrantes

Leticia Betarte
Rodrigo Machado
Valeria Molina

Instituto de Computación
Facultad de Ingeniería
Universidad de la República
2005 - 2006

RESUMEN

Los sistemas de recomendación tratan de ser una alternativa al proceso social de recomendación: ese acto habitual que se practica al recurrir a las opiniones de conocidos o expertos cuando se tiene que tomar una decisión para adquirir algo sin tener la suficiente información para ello. Éstas son generalmente decisiones sencillas en el entorno de la vida cotidiana, como qué libro leer, qué película ver, qué música escuchar o qué lugares visitar.

Una de las técnicas que se utiliza en este tipo de sistemas es el filtrado colaborativo: se recomiendan elementos que han gustado a usuarios con preferencias similares. Por esta razón, se recolectan preferencias sobre los elementos y a partir de ellas se calculan las recomendaciones. En el diseño de estas aplicaciones se deben tener en cuenta algunos aspectos entre los cuales se destacan: cómo se recolectan las preferencias, qué se sugiere a los nuevos usuarios dado que se cuenta con poca información (*cold start*), la privacidad de la información y el volumen de datos.

En este trabajo se realiza un estudio sobre las técnicas que se aplican en este tipo de sistemas, en particular la de filtrado colaborativo, se describen los algoritmos y métricas más utilizados así como las problemáticas que se presentan a la hora de su diseño. En relación a los aspectos que merecen ser considerados, se realiza especial hincapié en el problema del *Cold Start* así como en el de la recolección de preferencias implícitas, generando soluciones aplicadas al dominio de la música.

El resultado es un sistema de recomendación de música basado en Filtrado Colaborativo, denominado PGMúsica. Este sistema consta principalmente de tres partes. Una de ellas es una extensión de la aplicación de audio Winamp donde el usuario obtiene recomendaciones de canciones que le son enviadas mediante *streaming:* el contenido de audio es reproducido durante su transmisión sin que sea descargado en la propia máquina del usuario. Estas canciones son transferidas desde otros clientes, generando a este nivel una arquitectura *Peer-to-Peer*. Otra parte, es un sitio *Web* donde se puede consultar la información relativa a la interacción del usuario con el sistema, así como el detalle de las recomendaciones realizadas. La tercer parte y la más importante, es el servidor donde se calculan las recomendaciones. El algoritmo utilizado es el algoritmo basado en memoria, donde el cálculo de similitud entre usuarios se realiza mediante el coeficiente de correlación de Pearson.

Palabras claves: sistemas de recomendación, filtrado colaborativo, *streaming*, arquitectura *peer-to-peer*.

INDICE

Capítulo 1 Introducción

En los últimos años ha ocurrido un crecimiento explosivo del volumen de información. El número de productos, libros, música, películas, noticias, anuncios e información en general está sobredimensionado para la capacidad de procesamiento humana. La mayoría de estos artículos se filtran simplemente porque son inaccesibles o invisibles al usuario. Esto se observa día a día donde las librerías y disquerías son quienes deciden qué productos vender, los diarios qué noticias publicar y los cines qué películas exhibir.

Nuevas tendencias en la tecnología de la información intentan romper con esas barreras, ofreciendo una mayor variedad de opciones y permitiendo que el filtrado de la información no se realice por medios que tienen intereses propios, sino que sea por parte de las personas o componentes de software independientes. Es por ello que surge la idea de los Sistemas de Recomendación.

Los Sistemas de Recomendación tratan de ser una alternativa al proceso social de recomendación: ese acto habitual que se practica al recurrir a las opiniones de conocidos o expertos cuando se tiene que tomar una decisión para adquirir algo sin tener la suficiente información para ello. Éstas son generalmente decisiones sencillas en el entorno de la vida cotidiana, como qué libro leer, qué película ver, qué música escuchar o qué lugares visitar.

El enfoque más intuitivo que surge a la hora de crear un sistema de recomendación es que éste analice la relación entre el contenido de los elementos de interés del usuario y el contenido de los elementos del sistema, para así poder realizarle sugerencias. Como ejemplo de ello se puede pensar en una aplicación que recomienda libros donde la relevancia de los ítems a sugerir se determina comparando las descripciones de cada libro (autor, género, etc.) y las descripciones de los libros de interés del usuario. A este tipo de aplicaciones se les denomina “sistemas de recomendación basados en contenido”, dado que el filtrado de la información se realiza analizando el contenido de los elementos.

Existen varios sistemas que optan por este enfoque, entre ellos se encuentra Pandora [31] que es un sistema de recomendación de música. Pandora contiene información acerca de características y atributos de cada uno de sus temas musicales. Las personas ofrecen información al sistema sobre sus canciones y artistas favoritos y luego éste les recomienda aquellas canciones cuyas características almacenadas concuerden con sus preferencias.

Los sistemas que recomiendan elementos del gusto de los usuarios, cuando aplican la técnica de filtrado basado en contenido, cuentan con algunas limitaciones. Por un lado, los elementos que sugieren deben tener atributos que permitan describirlos. Con la tecnología actual y medios como música, fotografía, arte, video o elementos físicos, se dificulta esta tarea, ya que generalmente estos ítems no pueden ser analizados automáticamente debido a la dificultad de obtener un conjunto de atributos que permitan describirlos realmente. Las predilecciones de los usuarios pueden deberse a las sensaciones que provocan tales elementos y no a sus atributos. Por ejemplo, el agrado por una canción puede ser ocasionado por la satisfacción de escucharla, no influyendo en ello su género, autor, intérprete o frecuencia musical. Por otro lado, dadas las características de la

técnica de filtrado basado en contenido, estos sistemas no permiten que el usuario descubra ítems de contenido totalmente distinto a los ya conocidos por él aunque puedan ser de su interés. Por ejemplo, en un sistema en que se recomienda películas teniendo en cuenta el género preferido por el usuario, si éste manifestó interés por películas de suspenso, el sistema omitirá realizarle recomendaciones de películas de otros estilos distintos a suspenso, sin sugerirle que experimente otros géneros.

Con el fin de paliar estas limitaciones surgen los Sistemas de Recomendación basados en filtrado colaborativo. En estos sistemas, el filtrado de información no se realiza analizando las características de los elementos a recomendar, sino que para ello se tiene en cuenta únicamente las valoraciones de todos los usuarios sobre los ítems del sistema.

El mecanismo general de esta técnica consiste en calcular un puntaje de predicción, o sea, el puntaje que se estima que un usuario daría a un determinado elemento del sistema no conocido hasta el momento por él, basándose en los puntajes que las personas han ofrecido sobre los distintos elementos del sistema. Con ese valor de predicción se concluye si ese ítem podría ser de interés para el usuario y en cuyo caso se le recomienda.

El primer sistema de recomendación basado en filtrado colaborativo, Tapestry [3], fue desarrollado por XeroX PARC en el año 1992, adoptando la frase “Filtrado Colaborativo” que a partir de ese momento fue el término utilizado. Este sistema permitía a los usuarios realizar anotaciones sobre documentos electrónicos publicados en un grupo de noticias, para que luego fueran utilizadas por los restantes usuarios. Tapestry utilizaba la idea de filtrado colaborativo pero no lo hacía de forma automática. En este caso el sistema no era quien realizaba la sugerencia sino que permitía a los usuarios, a partir de las anotaciones que se realizaban sobre los artículos, concluir si un determinado elemento era de su interés.

En 1997, Resnick y Varian [11] proponen llamarles a estas aplicaciones “Sistemas de Recomendación” por dos razones: en primer lugar porque puede ocurrir que los usuarios no colaboren explícitamente entre ellos y en segundo lugar porque el sistema puede sugerir elementos no conocidos hasta el momento por el usuario y en ese caso se estaría realizando una “recomendación”. Dado que los algoritmos de filtrado colaborativo tienen la particularidad de no analizar el contenido de los ítems en el cálculo de predicción, pueden aplicarse a una gran variedad de tipos de elementos como libros, películas, chistes, música, comidas, etc.

En particular, Upendra Shardanand [14], investigador del MIT, plantea en su trabajo realizado como tesis de maestría que la música es uno de los dominios de aplicación más interesantes para este tipo de sistemas. En primer lugar porque los gustos de las personas sobre música son muy variados, más que sobre otros elementos (por ejemplo películas), y esto hace que la música sea un mejor dominio para probar el potencial del filtrado colaborativo. En segundo lugar, las personas sienten sus gustos musicales realmente propios, pues tienden a ser parte de su identidad. Por ese motivo manifiestan con mayor precisión el agrado o desagrado de la misma, y de esa forma el sistema obtiene una mayor precisión sobre las preferencias del usuario para realizarle mejores recomendaciones.

Por otra parte, los sistemas de recomendación de música actuales permiten a las personas escuchar la música que se les recomienda. Las canciones son enviadas utilizando la técnica de *streaming*: el contenido de audio es reproducido sin que sea descargado en la propia máquina del usuario. En general estos sistemas cuentan con una arquitectura cliente-servidor, donde el servidor es quien

contiene y envía la música recomendada al cliente; un ejemplo de estos sistemas es Last.fm [30].

Un enfoque alternativo que se plantea, como se muestra en la Figura 1.1, es que sean los propios usuarios quienes dispongan de estos recursos y sean quienes realicen el envío de la música. En este caso el servidor va a ser quien conozca qué usuarios tienen las canciones que desea enviar como recomendación, indicándole de forma transparente a quien solicita la recomendación, con quién debe conectarse para recibir los temas mediante *streaming*.

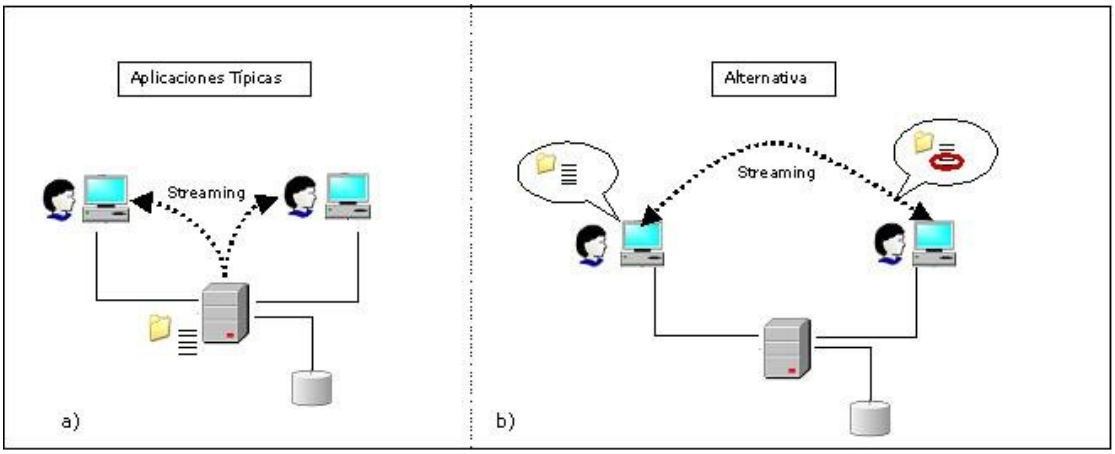

Figura 1.1: a) Envío de *streaming* Cliente-Servidor. b) Envío de *streaming Peer-to-Peer.*

Cuando la arquitectura de un sistema permite la comunicación cliente-cliente, se le denomina arquitectura *Peer-to-Peer* (P2P). En este caso existen una serie de nodos (clientes) que componen una red, donde cada uno cumple funciones de cliente y servidor al mismo tiempo.

Existen varias aplicaciones que utilizan este enfoque; una de las primeras surgió en el año 1999 llamada Napster[1]. Esta aplicación era utilizada para el intercambio de archivos entre usuarios conectados a Internet, especialmente archivos de tipo mp3. En el caso de Napster, además de los usuarios que oficiaban de servidor para el intercambio de archivos, existía un servidor central que almacenaba la información de todos los usuarios conectados y de los recursos con que contaban. Otras aplicaciones más modernas, como por ejemplo Kazaa[2], también para el intercambio de archivos, no cuentan con un servidor central sino que consisten en redes descentralizadas donde no se realiza registro de los usuarios conectados ni de los archivos intercambiados. Este tipo de aplicaciones sufren varios problemas debido a la distribución de música de forma ilegal.

1.1 Objetivos y Resultados esperados

Relacionando los conceptos mencionados anteriormente, el trabajo que se presenta en este proyecto de grado tiene como objetivos el estudio e implementación de técnicas de filtrado colaborativo, aplicándolas a la resolución del problema de recomendación de música, así como el estudio e implementación

[1] www.napster.com
[2] www.kazaa.com

de *streaming* directo entre dos usuarios del sistema, en una arquitectura *Peer-to-Peer*.

Como principal resultado se espera la implementación de un *Sistema de Recomendación* de música basado en *Filtrado Colaborativo.* El sistema debe constar de un servidor central encargado de registrar toda la información necesaria para poder realizar las recomendaciones. Además, se debe implementar un cliente el cual debe ser una extensión (*plugin*) de programas de uso masivo para la escucha de música, que permita recolectar la información para enviarle al servidor y también de comunicarse con los demás clientes del sistema para enviar y recibir canciones mediante *streaming*, determinándose a este nivel una arquitectura *Peer-to-Peer*.

1.2 Organización del Documento

Este documento se conforma de 6 capítulos. En el capítulo 2 se realiza una descripción sobre los Sistemas de Recomendación basados en filtrado colaborativo, así como los algoritmos que utilizan y las problemáticas a las que se enfrentan. Las decisiones principales sobre el diseño del sistema se presentan en el capítulo 3, y en el capítulo 4 se realiza la descripción a nivel de implementación, mostrando los distintos componentes del sistema y su interacción. En el capítulo 5 se detallan las pruebas realizadas sobre el algoritmo utilizado. Y por último, en el capítulo 6, se mencionan las conclusiones alcanzadas en la culminación del proyecto así como las propuestas de mejoras y trabajo a futuro para el sistema PGMúsica.

Capítulo 2 Sistemas de Recomendación basados en Filtrado Colaborativo

Con el objetivo de crear sistemas que realizan recomendaciones personalizadas, el filtrado colaborativo se plantea como técnica alternativa y eventualmente complementaria al filtrado basado en contenido. En el filtrado colaborativo se intentan resolver algunas de las limitaciones que presenta el filtrado basado en contenido, mencionadas en el capítulo anterior. El uso típico de esta técnica es en sistemas de recomendación de libros, música y películas.

La idea principal de estos sistemas es automatizar el proceso "boca a boca" donde personas recomiendan productos y servicios a otras. Si se tiene que elegir entre una variedad de opciones con las cuales no se ha tenido ninguna experiencia, a menudo se confía en las opiniones de otros que tengan tal experiencia, en particular se buscan personas con gustos similares al que desea obtener la recomendación. Estos sistemas actúan como guías que orientan en la toma de decisiones relacionadas con los gustos personales y han tenido gran aceptación debido a que las personas están acostumbradas a recibir recomendaciones de amigos y colegas.

Hasta ahora, las aplicaciones principales que se han realizado de estos sistemas han sido en sitios dedicados al comercio electrónico, proporcionando al usuario una forma de encontrar productos que le podrían interesar adquirir. Un ejemplo de ello es Amazon[3] [4], una tienda *online* de libros y música, que fue la primer librería virtual de la Web.

El filtrado colaborativo consiste en recomendar elementos que han gustado a usuarios con preferencias similares, basándose únicamente en el puntaje que éstos asignan a los ítems del sistema. Para ello las personas puntúan los elementos de acuerdo a sus intereses (a medida que esto sucede se va generando su perfil) y a partir de ellos se calculan las recomendaciones.

Algunos enfoques se basan en calcular la similitud entre usuarios, y a partir de sus preferencias se obtienen los elementos a recomendar. Para cada usuario se crea un conjunto de "vecinos cercanos": personas cuyas evaluaciones tienen grandes semejanzas a las del usuario que solicita la recomendación. Los resultados para los elementos no calificados por él, se predicen en base a la combinación de puntajes conocidos de los vecinos cercanos.

Entonces el procedimiento general aplicado en estos sistemas se resume de la siguiente forma:

1) Los usuarios expresan sus valoraciones sobre elementos del sistema, generalmente mediante una escala numérica.

2) A partir de esa información, se intenta predecir el puntaje que daría el usuario que solicita la recomendación (usuario activo), a los elementos del sistema no conocidos hasta el momento por él.

[3] www.amazon.com

3) De las predicciones calculadas se seleccionan los elementos con valores más altos para realizar la recomendación.

Existen muchas maneras de generar las predicciones mencionadas, desde diferentes enfoques y basando los cálculos en diferentes algoritmos.

En la sección 2.1 de este capítulo, se mencionan los algoritmos más conocidos y estudiados del área, con los cuales se han obtenido los mejores resultados hasta el momento. En la siguiente sección se presentan los procedimientos y métricas más utilizados para medir la efectividad de estos algoritmos. En la sección 2.3 se estudian algunos de los aspectos más importantes a ser considerados en el diseño de una aplicación de este tipo. Y por último, en la sección 2.4, se mencionan algunos sistemas de recomendación que son de referencia en el área y en este proyecto.

2.1 Algoritmos

Como se menciona anteriormente, el objetivo de los algoritmos de filtrado colaborativo es predecir el puntaje que un usuario dará a un ítem en particular, basándose en una base de datos que contiene los puntajes que otros usuarios han ingresado para los ítems del sistema, ignorando por completo los atributos o características de los usuarios y elementos.

Se pueden distinguir dos tipos generales de algoritmos de Filtrado Colaborativo:

- Algoritmos basados en Memoria: el cálculo de una predicción se realiza considerando la similitud entre el usuario activo y los demás usuarios del sistema. A estos algoritmos también se les denomina *"basados en vecindario"*.

- Algoritmos basados en Modelo: se utilizan los puntajes brindados por los usuarios para entrenar un modelo probabilístico, el cual se utiliza para generar las predicciones. En este caso no se tiene en cuenta de forma explícita la similitud entre el usuario activo y los demás usuarios del sistema.

La mayoría de los sistemas de filtrado colaborativo automático utilizan algoritmos basados en memoria para el cálculo de predicción. A su vez, para computar la medida de similitud entre dos usuarios, se utilizan varios métodos, donde el coeficiente de correlación de Pearson y el vector de similitud son los más utilizados.

Dentro de los algoritmos basados en modelo se plantea el modelo de redes bayesianas y el modelo de clusters como los más estudiados.

A continuación se realiza una descripción de los dos tipos generales de algoritmos así como los métodos más usados para cada uno de ellos.

2.1.1 Algoritmos basados en memoria

Cuando el usuario activo solicita una recomendación, la idea general en este tipo de algoritmos consiste en vincularlo con otros usuarios del sistema. El criterio para vincular dos usuarios es que ambos hayan puntuado elementos en común. Al conjunto de usuarios con que se logra vincular al usuario activo lo llamamos

conjunto de vecinos o vecindario.

Son vecinos del usuario activo aquellos que tienen un conjunto de ítems puntuados en común con él, sin importar que las preferencias sobre éstos sean muy variadas. El concepto de similitud se expresa mediante un valor, que se calcula considerando los elementos puntuados en común. Si ambos puntúan de manera similar estos elementos, se obtiene un valor alto de similitud, mientras que si expresan valoraciones opuestas se obtiene un valor de similitud bajo.

La idea en un algoritmo basado en memoria es calcular las predicciones sobre aquellos elementos que los vecinos conocen, pero que son desconocidos para el usuario activo. A este conjunto de elementos le llamaremos conjunto de ítems recomendables, dado que son potencialmente los elementos que pueden llegar a ser recomendados.

Para realizar la recomendación se calcula la predicción del puntaje para cada elemento del conjunto de ítems recomendables. Una vez calculados todos estos puntajes se toma una cantidad arbitraria de elementos, los cuales tienen asociados los mayores valores de puntaje de predicción.

La fórmula general utilizada para el cálculo de la predicción de puntajes es la presentada en la Formula 2.1. Se define un voto $V_{i,j}$ como el puntaje brindado por el usuario *i* para el ítem *j*. Se considera además $\overline{v}_j$ como el puntaje promedio de los votos del usuario *j*, y $w(a,i)$ como el valor de similitud entre el usuario *a* y el usuario *i*.

$$p_{a,j} = \overline{v}_a + k\sum_{i=1}^{n} w(a,i)(v_{i,j} - \overline{v}_i)$$

Fórmula 2.1: Método basado en memoria

En esta fórmula, *n* representa la cantidad de vecinos en el sistema y *k* es un factor de normalización de la ecuación.

Analizando la fórmula, se puede observar que el primer término es el valor promedio de votación del usuario activo y el segundo representa la variación del puntaje a predecir con respecto a ese promedio. Esta variación se calcula considerando todos los usuarios que tienen elementos en común con el usuario activo y que además han puntuado el elemento *j.*

Con respecto a la sumatoria presente en a Fórmula 2.1 se desglosa para su explicación en mayor detalle. En primer lugar es necesario determinar si los votos de los vecinos son positivos o negativos con respecto a el ítem j. Una posibilidad es comparar ese voto con el valor medio de la escala de votación. En tal caso no se consideran las distintas personalidades de los usuarios. Si en vez de utilizar el valor medio se utiliza el promedio de las votaciones registradas del usuario como valor neutro, se asume que los votos superiores al promedio son positivos y los menores negativos (Fórmula 2.2). De esta forma se tiene en cuenta la tendencia o personalidad del usuario al votar, dado que la escala de puntajes de los usuarios es variada y un voto aceptable para un usuario puede estar contenido en un rango de valores diferente al de otro usuario. Por ejemplo, considerando una escala de puntajes de 1 a 5, el promedio de votaciones del usuario u1 igual a 2 y el promedio del usuario u2 igual a 4. Se quiere saber el agrado o desagrado de una canción que tanto el usuario u1 como el u2 le asocia un puntaje igual a 3. Si

comparamos contra el promedio de votaciones de cada usuario se concluye que al usuario u1 le agrada más la canción que al usuario u2, dado que para u1 el valor de 3 supera su promedio de votación (2) mientras que para u2 no alcanza su promedio (4). Sin embargo en caso de considerar el promedio de la escala de votaciones (3) en vez del promedio de cada usuario, se concluiría que a los dos usuarios le agrada de la misma forma, lo cual no sería real.

Este término se multiplica en la sumatoria por el valor de similitud entre el usuario activo *a*, y el vecino *i*, $\boldsymbol{w}(a,i)$, el cual es positivo en caso de ser similares, y negativo si son disímiles.

La multiplicación de estos dos miembros de la sumatoria provoca que cada término de ella sea positivo si el usuario *i* votó positivamente al elemento *j* y además es similar al usuario activo, o bien, si ambos usuarios son disímiles y el elemento *j* fue valorado negativamente por el usuario *i*. Análogamente se obtendrá en la sumatoria un término negativo cuando ambos usuarios son similares y el usuario *i* valoró negativamente el elemento en cuestión. Esto sucede también cuando el usuario *i* valora positivamente el elemento pero ambos usuarios son disímiles.

$$(v_{i,j} - \overline{v_i})$$

Fórmula 2.2: Valoración del usuario i sobre el elemento j

Normalizando el valor obtenido por la sumatoria y sumándolo al promedio de votos del usuario activo, se obtiene la predicción de voto del usuario activo sobre el elemento *j*.

Similitud entre usuarios

Para determinar la similitud entre dos usuarios, $\boldsymbol{w}(a,i)$, uno de los mecanismos más utilizado es el coeficiente de correlación de Pearson, donde la correlación entre los usuarios *a* e *i* se calcula como se muestra en la Fórmula 2.3. En esta fórmula, el índice j corresponde a los ítems que fueron evaluados por los usuarios *a* e *i*.

$$w(a,i) = \frac{\sum_j (v_{a,j} - \overline{v_a})(v_{i,j} - \overline{v_i})}{\sqrt{\sum_j (v_{a,j} - \overline{v_a})^2 \sum_j (v_{i,j} - \overline{v_i})^2}}$$

Fórmula 2.3: Correlación de Pearson

En el numerador podemos ver que, para cada elemento *j* de la sumatoria, se multiplican dos términos. El primer término establece si el usuario activo (*a*) evalúa el elemento *j* de manera positiva o negativa, tomando su promedio de votos como valor neutro. De la misma manera en el segundo término se evalúa el voto del usuario *i* con respecto al ítem *j*. Al multiplicar estos dos términos se obtendrá un valor negativo, solo si uno de los usuarios lo puntuó por debajo del promedio de votación y el otro usuario por encima. Este valor negativo provoca una disminución del valor total de similitud *w*, lo cual tiene sentido si consideramos el hecho de que los usuarios valoraron de manera opuesta el elemento *j*. Análogamente obtendremos un valor positivo en la multiplicación, cuando ambos usuarios valoraron de igual manera el elemento *j* en función de sus valores promedio de votación. El denominador simplemente normaliza el

valor de similitud.

Es preciso observar que, si un usuario del sistema puntúa a todos sus ítems con el mismo valor, no será posible realizarle una recomendación: no se puede aplicar la fórmula de coeficiente de correlación de Pearson pues se realiza una división entre cero. Esto tiene sentido dado que si el usuario asigna el mismo puntaje a todos los ítems, no se tiene información sobre sus preferencias y por lo tanto no se le puede realizar ninguna recomendación.

Si bien se han obtenidos buenos resultados con este algoritmo, como debilidad importante se destaca la gran cantidad de cálculos que se realizan al computar una predicción, lo cual ocasiona una desmejora en la performance de estos sistemas. Cuanto mayor es la cantidad de usuarios y puntajes considerados en el cálculo de predicción, mayor es la exactitud en las predicciones y menor es la performance del sistema. En definitiva estos algoritmos suelen presentar graves problemas de escala.

Vecinos más cercanos

Si bien en el cálculo de predicción presentado en la Fórmula 2.1 se consideran todos los usuarios que tienen votos en común con el usuario activo, una de las ideas más utilizadas en filtrado colaborativo es la de los vecinos más cercanos (K *nearest neighbors*, KNN). Esto implica la selección de un subconjunto de usuarios que cumplan una condición respecto a la similitud con el usuario activo para ser considerados en el cálculo de la predicción. La razón de considerar un conjunto reducido de vecinos es aumentar la eficiencia del algoritmo, disminuyendo el tiempo de cómputo en el cálculo de predicción. Por ejemplo se consideran, o se toman los k vecinos con menor distancia.

En el caso de decidir utilizar un subconjunto de los vecinos posibles, queda por determinar cuál es la cantidad máxima de vecinos que debería seleccionarse. Se han realizado estudios que demuestran que al variar el valor del k, varía la precisión de los resultados. Además ese valor depende de tamaño del conjunto de datos, por lo tanto se estudia en cada caso el k más adecuado para el conjunto de datos considerado.

2.1.2 Algoritmos Basados en el Modelo

Desde la perspectiva probabilística, la tarea del filtrado colaborativo puede ser vista como el cálculo del valor esperado de un voto, dada la información de los puntajes brindados por los usuarios. Para el usuario activo, se intenta predecir el voto para un ítem no evaluado por él. Si se toma una escala de puntajes de 0 a m, se puede ver en la Fórmula 2.4 cómo se plantea la predicción del puntaje considerando el valor esperado del voto del usuario *a* sobre el ítem *j*.

$$p_{a,j} = E(v_{a,j}) = \sum_{i=0}^{m} \Pr(v_{a,j} = i / v_{a,k}, k \in I_a) i$$

Fórmula 2.4: Método basado en modelo

Dos de las alternativas de implementación más usadas para este método son el modelo de *clusters* y el modelo de redes bayesianas. Según el estudio realizado

por Breese, Heckermann y Kadie [5], el cual es de referencia en la mayoría de los trabajos realizados sobre este tema, el modelo de redes bayesianas resulta más efectivo que el modelo de *clusters*. Este modelo consiste en formar redes, donde los nodos se corresponden con cada ítem del dominio de aplicación, y los estados de los nodos, con los posibles valores de votos sobre los ítems. Se aplica el algoritmo sobre el conjunto de datos de entrenamiento para encontrar dependencias entre los elementos y construir las redes. A partir del modelo construido se calculan las predicciones de votos. Cada tabla de probabilidad condicional puede ser representada por un árbol de decisión que codifica las probabilidades condicionales para ese nodo.

Por ejemplo, en un dominio de series de televisión se quiere predecir la probabilidad de que al usuario activo le interese mirar la serie “Melrose Place”. Luego de entrenar el modelo se encuentra que “Melrose Place” está relacionado con los ítems “Beverly Hills” y “Friends” y el árbol de decisión para ese ítem es el que se muestra en la Figura 2.1. Las barras que están en la parte más baja de la figura indican la probabilidad de mirar “Melrose Place” condicionado a que vio los programas padres. Para determinar la preferencia del usuario activo se navega en la red y se calcula la probabilidad de preferencia del usuario activo sobre el ítem “Melrose Place”.

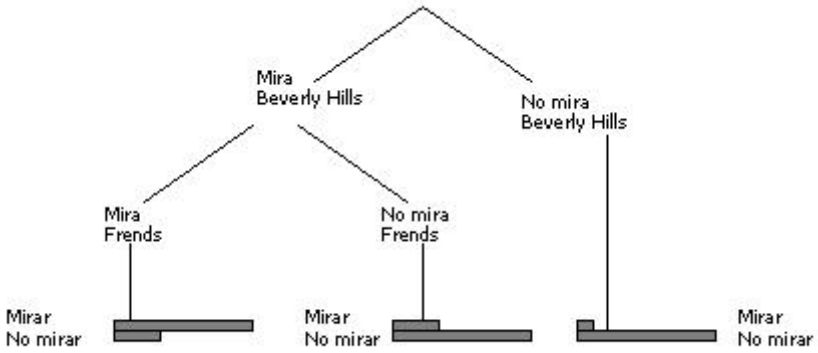

Figura 2.1: Árbol de decisión del nodo “Melrose Place”

Es preciso observar que este modelo cuenta con algunas desventajas importantes. En primer lugar, es necesario disponer de una gran cantidad de datos de entrenamiento para poder lograr un modelo de gran tamaño que permita realizar buenas predicciones. Sin embargo, contar con un conjunto de datos importante, puede ocasionar que el modelo sea extremadamente grande, requiriendo mucho tiempo de entrenamiento. Esto se ve agravado cuando se aplica este modelo en sistemas de filtrado colaborativo donde la escala de votación puede tener muchos valores, a diferencia del ejemplo donde se considera una escala binaria (mirar o no mirar). Esto se debe a que los estados posibles de los nodos son precisamente los valores posibles de puntajes de usuarios sobre ítems. Por esta razón se recomienda utilizar este modelo en sistemas donde la valoración del usuario sobre el ítem es binaria o con un tamaño reducido de valores. Si bien tiene estas desventajas, como ventaja importante se menciona la rapidez en el cálculo de una predicción luego de tener un modelo entrenado.

2.2 Procedimientos y Métricas de evaluación

Cuando se presenta un nuevo algoritmo de filtrado colaborativo, es necesario

medirlo en base a alguna métrica para saber qué tan exactas resultan sus predicciones.

Para ello, dado un conjunto de datos con puntuaciones reales sobre los ítems, se lo divide en dos subconjuntos a los que se los denomina conjunto de entrenamiento (*training set*) y conjunto de prueba (*data test*). La idea es predecir los datos del conjunto de prueba, considerando únicamente al conjunto de entrenamiento como datos de entrada del algoritmo de filtrado colaborativo. Luego, se comparan las predicciones obtenidas con los datos reales del conjunto de prueba para saber qué tan preciso resulta.

Otro mecanismo utilizado, llamado validación cruzada, consiste en dividir el conjunto en n particiones, tomando n-1 particiones como conjunto de entrenamiento y validando contra la partición restante. Este procedimiento se repite con todas las combinaciones de particiones.

Las primeras evaluaciones realizadas sobre sistemas de filtrado colaborativo automático que han sido publicadas son del año 1994. Desde entonces, se han realizado diversos sistemas y distintas métricas han sido utilizadas para su evaluación. Si bien no se ha encontrado una métrica estándar para medir estos sistemas, las más utilizadas son, como se concluye en el trabajo de Herlocker [7], la cobertura (*coverage*) y el error medio absoluto (MAE).

El *Coverage* (cobertura) es simplemente el porcentaje de los elementos para los cuales el sistema podría generar una predicción. En ciertos casos, un sistema de filtrado colaborativo automático puede no generar predicciones para determinados ítems, debido a la falta de información que se considera al momento de realizar los cálculos por parte del algoritmo. Por ejemplo, los elementos que no tienen asignados ningún puntaje, el sistema no los podrá recomendar. La pregunta que se hace entonces es: ¿para qué porcentajes de ítems, el sistema puede generar predicciones?. La manera más fácil de medir la cobertura es seleccionar un conjunto de pares usuarios-elementos elegidos al azar, calcular la predicción para cada par, y medir el porcentaje para los cuales fue proporcionada.

Por otro lado, la métrica más utilizada y aceptada en el área para evaluar la precisión de los algoritmos de filtrado colaborativo es MAE (*Mean Absolute Error*), que mide la desviación media entre el valor de predicción de un ítem y el valor de puntuación (*rating*) real que asigna el usuario al ítem.

$$\overline{|E|} = \frac{\sum_{i=1}^{N} |p_i - r_i|}{N}$$

Fórmula 2.5: Error medio absoluto

La fórmula para calcular el MAE se puede observar en la Fórmula 2.5, donde:

$\overline{|E|}$ = valor absoluto promedio del error

p_i = puntaje de predicción del algoritmo sobre el ítem *i*

r_i = puntaje real del usuario sobre el ítem *i*

N = total de elementos considerados para el cálculo de MAE.

Utilizando esta métrica, cada elemento del conjunto de prueba es tratado de la misma manera. Esto significa que se le da el mismo peso al error cometido en calcular la predicción de un elemento donde su valor real está por encima o por debajo del promedio de la escala de valores.

Por ejemplo, si la escala es de 1 a 5, el error en un ítem que tiene un puntaje verdadero de un punto, tiene el mismo valor que el error que se comete en un elemento con un puntaje de 3 o 5 puntos. Esencialmente, el error medio absoluto proporciona el mismo peso a los errores sea cual sea el elemento considerado. Esto hace que el error medio absoluto sea el más apropiado para los sistemas que no hacen consideraciones o separaciones de los ítems basados en algo que no sea las puntuaciones disponibles.

Existen algunas variaciones sobre esta métrica que consisten en hacer la raíz de la resta o el cuadrado de la resta. Estas variaciones apuntan a dar mayor importancia a los errores más grandes. Por ejemplo, si se utiliza el cuadrado de la resta, se estaría amplificando el efecto de los mayores errores, ya que un error de valor 1, suma 1 al total, mientras que un error de 2 sumaría 4.

Esta métrica tiene la ventaja de ser muy simple de calcular y los resultados son fácilmente interpretables.

2.3 Consideraciones de diseño

Los sistemas de recomendación basados en filtrado colaborativo traen consigo algunos problemas y aspectos a ser considerados a la hora de su diseño. A continuación se detallan algunos: la forma de recolectar las preferencias de los usuarios sobre los elementos del sistema, el problema de *Cold Start*, la confiabilidad y privacidad de los usuarios, y por último, el problema del volumen de información que manejan.

2.3.1 Recolección de Preferencias

Un aspecto fundamental que se plantea a la hora de diseñar un sistema que utiliza la técnica de filtrado colaborativo es la recolección de preferencias de las personas sobre elementos del sistema. De esto depende su funcionamiento, dado que el sistema llega a ser útil luego de haber recogido una “masa crítica” de opiniones.

Las preferencias de las personas sobre los elementos se obtienen de forma explícita o implícita. Las recolección explícita ocurre cuando el usuario manifiesta su preferencia sobre un ítem, generalmente en una escala numérica discreta. Un ejemplo de ello se puede observar en “Amazon”, donde las personas pueden asignar a los artículos una cantidad de estrellas que reflejan su nivel de preferencia.

En cambio, la recolección implícita se basa en obtener las preferencias del usuario analizando las acciones que realiza al utilizar el sistema. Por ejemplo, un sistema de recomendación de páginas *Web* puede recoger información implícita sobre las preferencias registrando la visita de las personas a los diferentes sitios *Web*. También se puede ver como forma de recolección implícita, en un sistema de recomendación de música, la cantidad de veces que un usuario escucha determinada canción.

La recolección de preferencias de forma explícita es simple de realizar pues consiste únicamente en ofrecer la posibilidad de puntuar los ítems del sistema. Sin embargo, para lograr esto se requiere un esfuerzo importante por parte de los usuarios, dado que les puede llevar mucho tiempo realizar estas acciones.

Para recoger las preferencias de forma implícita el sistema debe ser capaz de estudiar el comportamiento del usuario en su utilización, lo cual es claramente una tarea difícil. Además, es preciso considerar que las preferencias que se pueden obtener de esta forma son menos exactas, dado que son estimadas por el sistema y no proporcionadas directamente por el usuario. Sin embargo, esta forma de recolección resulta más cómoda para las personas, dado que no deben puntuar cada uno de los elementos.

Si bien la información explícita tiene una mayor exactitud y es más fácil de analizar por el sistema, la recolección implícita no deja de ser importante ya que facilita la tarea de las personas que utilizan la aplicación. Por lo tanto, muchos sistemas intentan recoger las preferencias de ambas formas, logrando complementar los esfuerzos de las dos partes (usuario y sistema).

2.3.2 Cold Start

Cuando un usuario llega al sistema, no es posible hacerle recomendaciones hasta que su perfil esté lo suficientemente determinado como para encontrarle su grupo de vecinos cercanos. A este problema se lo conoce como *Cold Start* [2]. Además, si los gustos del usuario son poco comunes, podría no ser fácil encontrarle un conjunto de vecinos. Esto hace notar que las recomendaciones dependen directamente del número y variedad de usuarios en el sistema. También es considerado un problema de *Cold Start* cuando la llegada no es por parte de un usuario, sino de un nuevo ítem.

En la mayoría de los casos se ataca este problema exigiéndole a los nuevos usuarios que tomen postura sobre una serie de elementos, de manera de poder determinar un perfil inicial como para poder realizarle una primer recomendación. Por ejemplo, el sistema de recomendación de música Ringo [14] exigía ranquear un conjunto de artistas seleccionado por el sistema, donde la mitad de esos artistas eran los más populares y la otra mitad se seleccionaba aleatoriamente. De esta forma, se le construía un perfil al nuevo usuario, con el cual se le realizaban las primeras recomendaciones. Otros sistemas de recomendación de música le exigen escuchar un gran número de temas antes de realizarle la primer recomendación; un ejemplo de ello es Audioscrobbler [29], el cual exige la escucha de 300 canciones.

Andrew I. Schein [2] y sus colaboradores plantean la utilización de métodos menos triviales, como considerar la información de contenido de los elementos. Para esto, el usuario al momento de registrarse explicita algunas preferencias sobre características de los ítems disponibles. De esta manera, el sistema puede generar recomendaciones iniciales hasta que el usuario puntúe suficientes elementos para construir un perfil que permita vincularlo con otros usuarios del sistema. Un ejemplo de ello es en un sistema de recomendación de música que el usuario indique sus bandas favoritas. Estos enfoques tienen la ventaja de no obligar al usuario a puntuar un conjunto importante de elementos.

Al igual que con la llegada de un nuevo usuario, el ingreso de un ítem también resulta un problema. Sin embargo no se ha encontrado en la bibliografía consultada sugerencias sobre cómo resolver el problema del *cold start* cuando la llegada al sistema es por parte de un elemento. En este caso no se tiene ninguna

valoración por parte de los usuarios sobre ese ítem, lo cual imposibilita su recomendación.

El problema del *cold start* por parte del usuario ocurre tanto en los sistemas de recomendación basados en contenido como en los de filtrado colaborativo. Esto se debe a que en los dos casos es preciso obtener un perfil del usuario para realizar recomendaciones. Sin embargo el *cold start* de un ítem es un problema propio del filtrado colaborativo. Esto ocurre porque se recomiendan elementos que han puntuado usuarios con intereses similares al usuario activo. En este caso el nuevo ítem no tiene ningún puntaje asignado por lo tanto no puede ser recomendado. Sin embargo los sistemas basados en contenido analizan el contenido de los elementos para relacionarlos con otros del sistema y de esa forma poder recomendarlos. Por esta razón la llegada de un nuevo ítem, en la mayoría de los casos, se resuelve de forma casi inmediata.

2.3.3 Usuarios mal intencionados

Uno de los inconvenientes de estos sistemas es que al considerar el puntaje de los usuarios sobre los elementos para realizar recomendaciones puede ocurrir que algunas personas asignen puntajes muy altos a ítems que son de su interés con el objetivo que sean recomendados. Por ejemplo, en Amazon puede ocurrir que el autor de un libro asigne puntajes altos a sus libros para que sean recomendados por el sistema. A estas personas se les denomina “usuarios mal intencionados” ya que pueden sesgar las recomendaciones, por lo que la idea es poder detectarlos y eliminarlos del sistema.

Los investigadores Paolo Massa y Paolo Avesani [12] sugieren solucionar este problema estableciendo redes de confianza entre usuarios. La idea es que un usuario, no solo puntúe los ítems del sistema, sino que también exprese su confianza en las puntuaciones que han hecho otros usuarios. Esto permite basar las recomendaciones únicamente con los puntajes brindados por usuarios en los que se confía directamente e indirectamente con los usuarios confiables según éstos. Los valores bajos de confianza (o valores de desconfianza) pueden jugar un papel muy importante en la detección de los usuarios mal intencionados.

2.3.4 Privacidad

En los sistemas de recomendación basados en filtrado colaborativo es usual mostrarle al usuario información acerca de las recomendaciones que se le realizan. Esto incluye datos sobre los usuarios que influyeron en la generación de la recomendación, así como preferencias de los mismos sobre determinados ítems: cuanto mayor es la información que los usuarios tienen acerca de sus recomendaciones, mejor podrán evaluarlas.

Sin embargo, uno de los problemas que se plantea es la privacidad de los participantes, puesto que existen personas que no les gusta exhibir sus preferencias o ser reconocidas por su aporte.

Algunos sistemas plantean la utilización de seudónimos, como GroupLens [13], y otros utilizan el anonimato. No obstante, en los sistemas de recomendación que se utilizan en el comercio electrónico no es posible aplicar ni el anonimato ni el seudónimo debido a que los usuarios están totalmente identificados. Algunos investigadores, como John Canny [6], han realizado análisis acerca del tema haciendo especial hincapié en su importancia.

2.3.5 Volumen de datos

Las aplicaciones que utilizan la técnica de filtrado colaborativo en su mayoría consideran cantidades de ítems cada vez más grandes. Esto genera un impacto negativo sobre las recomendaciones.

Por un lado, las personas en general puntúan pocos elementos sobre el conjunto total de elementos del sistema. Pero si a su vez el conjunto de ítems aumenta en cantidad y diversidad, la densidad de elementos puntuados disminuye. Esto hace que sea menos probable que un número significativo de vecinos cercanos tengan puntuados los mismos elementos, por lo que decrece la cantidad de recomendaciones posibles.

Además, si el número de elementos en el sistema aumenta, se ve incrementado el tiempo de cómputo al calcular el conjunto de vecinos cercanos. Por lo tanto, se observa que este tipo de sistemas tienen problemas de escala ya que al aumentar el número de ítems en el sistema, se hace difícil mantener una performance razonable en las recomendaciones.

En la mayoría de los sistemas actuales se intenta paliar este inconveniente realizando el cálculo de las recomendaciones, o parte del mismo, periódicamente, aunque el usuario no solicite la recomendación, pues no es deseable que el usuario deba esperar varios minutos para obtener la recomendación. Esta solución tiene como dificultad que las recomendaciones podrían no estar actualizadas en el momento que el usuario las solicita.

Otra forma que se ha propuesto para atacar este problema es particionar el conjunto de elementos. El particionamiento reduce el espacio de dimensiones de los ítems en espacios más pequeños con pocos ítems, menos puntuaciones y muchos menos usuarios. Una vez que este espacio ha sido dividido se aplica el algoritmo de filtrado colaborativo en cada parte, independientemente de las otras. El tiempo para computar una predicción decrece, dado que hay menos datos a considerar. Por otro lado, la densidad de las puntuaciones se incrementa ya que los usuarios que pertenecen a una determinada partición probablemente puntúen ítems que pertenezcan a esa partición. Además como cada partición es independiente, esto hace que la computación de las recomendaciones se pueda hacer en paralelo, incrementando la cantidad de predicciones en un tiempo menor.

El sistema de recomendación para el grupo de noticias GroupLens fue el primer sistema de filtrado colaborativo automático que manejó un conjunto de ítems considerablemente grande. GroupLens creaba una partición en el conjunto de artículos según los grupos de discusión. Dado que la división de temas de discusión en el grupo de noticias era clara, se podía hacer perfectamente la división del conjunto de elementos en subconjuntos según el tema, basado en el contenido del ítem. La predicción de un usuario sobre un tema de discusión como *humor* era computada usando solamente las puntuaciones sobre los mensajes de *humor.* De esta forma, las predicciones son influenciadas solamente por las opiniones de *humor* y no, por ejemplo, por las de otros tópicos, como *cocina*. De esa forma se lograba que las puntuaciones sobre los ítems fuesen más densas, logrando un mayor solapamiento de puntuaciones.

También se están realizando algunos trabajos de investigación con el objetivo de particionar el conjunto de ítems sin tener en cuenta su contenido. Para esto se

estudian distintas técnicas de particionamiento o clusterización como las presentadas por A.K. Jain y R.C. Dubes [1].

Mark O'Connor y Jon Herlocker [9] proponen particionar el conjunto de elementos teniendo en cuenta únicamente los votos sobre los ítems (sin considerar su contenido). Ellos proponen usar estos datos de puntuaciones para descubrir relaciones entre elementos que podrían permitir particionar el conjunto con el propósito de mejorar la exactitud y performance de las recomendaciones.

El objetivo que se plantea con el particionamiento es:

- reducir el tiempo computacional de la generación de recomendaciones
- hacer más predicciones en menos tiempo paralelizando los cálculos
- incrementar la exactitud de las predicciones.

De esta manera vuelven a ser viables los cálculos de las predicciones mientras el usuario está conectado. Sin embargo, no se han obtenido hasta el momento buenos resultados en la exactitud de las predicciones, comparado con la no partición del conjunto de ítems. Por esta razón se plantea la continuación de la investigación en este sentido.

2.4 Sistemas de Referencia

Aunque hace ya varios años que surgió el primer sistema de recomendación basado en filtrado colaborativo, no son muchos los sistemas que han publicado su implementación desde entonces. Sin embargo, algunos de los que han tenido éxito han sido publicados, aportando mejoras de la investigación en el área.

A continuación se detallan algunos sistemas de referencia en el área así como de interés para el desarrollo de este proyecto.

Tapestry [3] fue el primer sistema de recomendación basado en filtrado colaborativo, creado en 1992, el cual permitía que los usuarios hiciesen anotaciones sobre documentos electrónicos pertenecientes a un grupo de noticias. Mediante texto libre se podían realizar anotaciones de agrado o desagrado sobre los documentos, permitiendo de una forma fácil que los usuarios indicaran su valoración al respecto. De esta manera, otros integrantes del sistema podían obtener los documentos junto con las anotaciones realizadas. El enfoque utilizado en este sistema era una mezcla de la técnica de filtrado basado en contenido y colaborativo. Basado en contenido porque permitía que los usuarios creasen filtros a través de sus ítems de interés, y colaborativo pues los usuarios ingresaban las anotaciones con sus opiniones sobre los documentos y de esa forma colaboraban entre ellos para filtrar la información. Tapestry utilizaba la idea de filtrado colaborativo pero no lo hacía de forma automática. En este caso el sistema no era quien realizaba la sugerencia sino que permitía a los usuarios, a partir de las anotaciones que se realizaban sobre los artículos, concluir si un determinado elemento era de su interés.

Dos años más tarde, surge GroupLens [13], quien fue el primer sistema automático de recomendación basado en filtrado colaborativo. Este sistema provee predicciones personalizadas para el grupo de noticias Usenet, ayudando a los usuarios a encontrar artículos de su agrado, dentro de la gran cantidad de artículos disponibles. GroupLens también fue el primer sistema en usar el algoritmo basado en memoria para realizar el cálculo de predicciones y el coeficiente de correlación de Pearson para calcular la similitud entre usuarios, considerando todos los usuarios con coeficiente mayor a cero como vecinos del

usuario activo. Respecto a la privacidad, los usuarios eligen seudónimos para identificarse dentro del mismo, los que serán asociados a sus puntuaciones. Estos seudónimos pueden ser diferentes del nombre que utilizan para anunciar los artículos de noticias.

MovieLens [28] es un sistema *Web* de recomendación de películas que fue desarrollado como proyecto de investigación del grupo GroupLens de la Universidad de Minnesota: en la actualidad MovieLens no es sólo un servicio de recomendación de películas sino que es también una fuente de datos experimental y un marco para investigar interfaces de usuarios referente a sistemas de recomendación.

En lo que respecta a la recomendación de música, se encuentra Ringo [14,15], que fue un sistema Web de recomendación de álbumes y artistas basado en filtrado colaborativo, publicado en Internet el 1° de julio de 1994. Ringo utilizaba el algoritmo general basado en memoria para calcular la predicción. Para el cálculo de la correlación utilizó una variante de la fórmula del coeficiente de correlación de Pearson, donde en vez de los *ratings* promedio de cada usuario, utilizaba una constante (el valor medio de la escala de puntajes). De esta forma, obtenía una mejor performance ya que no se calculan el promedio de votos de todos los usuarios del sistema. Además, los miembros de un vecindario estaban limitados a aquellos vecinos cuya correlación fuera mayor a cierto valor predeterminado. Aumentando este valor se obtenían resultados más precisos pero el sistema era capaz de generar predicciones para un conjunto menor de ítems.

Este sistema en sus comienzos tuvo un gran éxito. Al tiempo, los creadores formaron una empresa y le cambiaron el nombre por Firefly y luego fue comprado por Microsoft. Desde entonces el sistema ya no era sólo de recomendación de música sino que también recomendaba libros y películas. Como no tuvo el éxito esperado, al tiempo fue dado de baja.

Como sistema de recomendación de música más actual, se encuentra la fusión de Audioscrobbler [29] y Last.fm [30] que, unidos en agosto de 2005, conforman el sistema de recomendación Last.fm. Audioscrobbler es un *plugin* de los reproductores de audio más conocidos como Winamp, XMMS, iTunes, el cual recolecta la información de los temas musicales escuchados por los usuarios. Por otro lado se encuentra la aplicación “Last.fm” que permite la escucha de radios personalizadas por Internet. Un usuario puede construir su perfil musical mediante estas dos vías. Las canciones escuchadas son acumuladas para generar la recomendaciones, las cuales son calculadas con un algoritmo de filtrado colaborativo, teniendo en cuenta sólo la información de música escuchada por los usuarios.

Al igual que Last.fm, Pandora [31] es también un sistema de recomendación de música aunque difieren en la forma de recomendación. En Pandora el usuario le brinda al sistema un artista o canción y éste le retorna canciones con un estilo similar. Su recomendación se realiza analizando el contenido de la música, utilizando una serie de *“atributos”* que caracterizan la música almacenada en sus bases, siendo a través de ellos que relaciona cada uno de sus recursos musicales.

Capítulo 3 Diseño General de la Solución

El sistema que se presenta como resultado de este proyecto es un Sistema de Recomendación de música basado en Filtrado Colaborativo denominado PGMúsica, donde a partir de las preferencias de los usuarios registrados sobre las canciones del sistema, se brindan recomendaciones. Además el sistema permite que los usuarios puedan escuchar las canciones recomendadas, obteniéndolas desde otros usuarios conectados al sistema.

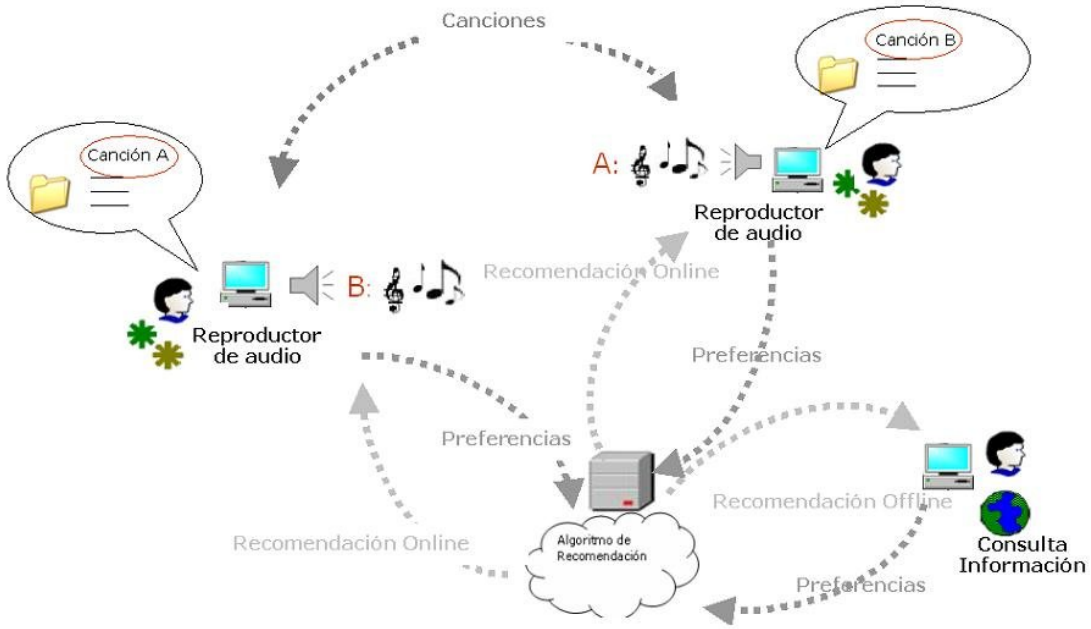

Figura 3.1: Diagrama del Sistema

PGMúsica consta de un servidor encargado de recibir y procesar la información que le es enviada desde cada usuario.

Dado que al realizar las recomendaciones, los algoritmos de filtrado colaborativo vinculan los usuarios tomando en cuenta sus preferencias, la recolección de esta información es un aspecto fundamental a considerar. En ese sentido, una vez registrados en el sistema, las personas colaboran asociándole a cada canción un puntaje dentro de una escala numérica. Además, con el objetivo de maximizar las fuentes de información, la aplicación registra las veces que un usuario escucha cada canción, dato que es tenido en cuenta en la generación de recomendaciones.

Las personas pueden solicitar recomendaciones de canciones desde la aplicación donde escuchan música. El sistema le permite escuchar los temas recomendados utilizando la técnica de *streaming*, los cuales se envían al usuario desde otros que estén conectados al sistema en el momento de la recomendación. A este tipo de recomendación se le denomina recomendación *"Online"*. Aunque la mayoría de

estos sistemas cuentan con un servidor centralizado que contiene los recursos a recomendar, en este caso son los propios usuarios los que tienen las canciones con las que el sistema cuenta para realizar el envío de música sugerida. Por eso se hace necesario que cada usuario defina cuál es el conjunto de canciones que desea compartir, para que el sistema las considere y pueda recomendarlas. Sólo estas canciones son tenidas en cuenta a la hora de realizar recomendaciones *Online*.

Un aspecto diferenciador de este sistema es que los elementos que se consideran son los que proporcionan los usuarios. Cuando una persona puntúa o escucha una canción, si esa canción no existe en el sistema, se registra su información. De la misma forma, cuando un usuario especifica qué canciones desea compartir, se registran las que son nuevas para el sistema. Todos los ingresos de elementos al sistema ocurren por alguna de estas vías.

Teniendo en cuenta que las recomendaciones *Online* dependen de la disponibilidad de las canciones compartidas por los usuarios conectados en el momento de generar la recomendación, se entiende importante que el usuario también pueda obtener recomendaciones que consideren todas las canciones registradas. En este caso se le muestra al usuario una lista de temas considerando todas las canciones, sin tener en cuenta cuáles son las que están disponibles en ese momento. También se muestra la información correspondiente al algoritmo de recomendación utilizado, para que el usuario sepa qué factores influyeron en la recomendación y tenga argumentos para poder decidir si le interesa o no. A este tipo de recomendación se le denomina recomendación "*Offline"*, ya que consta únicamente de la lista de temas sugeridos pero no se le ofrece la posibilidad de escucharlos.

Dado que los elementos a ser considerados son aportados por los usuarios, se toman decisiones con respecto a la obtención e identificación de las canciones del sistema, las cuales se describen en la sección 3.1. Por otro lado, en el diseño de los sistemas de recomendación basados en filtrado colaborativo se deben considerar aspectos como los descriptos en la sección 2.3. Para este proyecto se evalúa como prioridad el problema del *Cold Start* así como el de la recolección de preferencias implícitas. En ese sentido se realiza especial hincapié, encontrando soluciones particulares para el dominio de la música. En la sección 3.2 se describe el mecanismo utilizado para la recolección de preferencias explícitas e implícitas. Luego, en la sección 3.3, se detalla el mecanismo general para el cálculo de las recomendaciones, incluyendo el algoritmo de filtrado colaborativo seleccionado así como la solución planteada para el problema del *Cold Start*.

3.1 Obtención e identificación de canciones del sistema

Una vez que se establece el tipo de elemento que utiliza un sistema de recomendación es necesario definir qué características y consideraciones se debe tener sobre ellos. Los ítems con los que trabaja el sistema, en este caso las canciones, son de gran importancia, ya que son el recurso a recomendar. Teniendo en cuenta que son los propios usuarios los que aportan los elementos nuevos al sistema, es importante definir la obtención y la calidad de la información requerida para cada uno de estos elementos. Un gran volumen de canciones permite diversidad en las recomendaciones brindadas. Por otro lado la calidad de la información obtenida acerca de cada ítem permite ofrecer recomendaciones más precisas.

En PGMúsica la obtención de canciones se realiza por parte de los usuarios. Cada vez que un usuario registrado ingresa al sistema, define su música compartida,

donde cada canción, en el caso de no estar registrada, es ingresada. También se registran nuevas canciones cuando algún usuario escucha un tema que aún no fue registrado. El sistema asume que cada uno de sus usuarios pondrá a disposición un volumen significativo de temas musicales, obteniéndose así una cantidad considerable de canciones a recomendar.

Para un correcto funcionamiento es importante establecer cuál es el criterio para identificar las canciones, pues la información de las mismas depende de las canciones disponibles en cada usuario, y si esta información no es correcta, se podrán ver perjudicadas las recomendaciones que realiza. Una misma canción debe identificarse de forma única; en caso contrario, pueden recomendarse temas que el usuario ya conoce, repetirse un mismo tema musical en una misma recomendación, o distintas canciones de diversos estilos pueden ser consideradas la misma.

Para la identificación se utilizan los atributos *nombre de canción* y *artista*. Estos atributos son importantes para especificar una canción, además de ser los más habituales de encontrar en los archivos que ponen a disposición los usuarios. El resto de la información (como género, año, nombre del disco, etc.) ayudaría en la exactitud de identificación de un tema, pero no siempre está presente. Por lo tanto, las canciones con un mismo título y un mismo artista son consideradas iguales. Notar que al no tenerse en cuenta otra información no se esta totalmente exento de incurrir en los mencionados errores, por ejemplo una misma canción interpretada por el mismo cantante pero en distinto disco, por tratarse de diferentes versiones (versión acústica, versión sinfónica, etc.), es considerada la misma. Como en la mayoría de los casos ésta información esta ausente o es errónea, de igual forma no se podría determinar estas situaciones. Además, con la información que sí es tenida en cuenta, de ocurrir ruido en ella el sistema ve afectada la calidad de sus recomendaciones.

Sin embargo, es habitual que los atributos *nombre de canción* y *artista* contengan información incorrecta ingresada automáticamente por diversas aplicaciones. Por ejemplo, *track01* como título de canción, o incluso la falta de información en esos atributos. En tales casos estas canciones no son consideradas por PGMúsica, para lograr disminuir el ruido que ocasionarían en caso de considerarlas.

Otro problema se plantea cuando los atributos de las canciones tienen la información correcta pero el valor del atributo no es exactamente el mismo en diferentes usuarios. Por ejemplo, la canción “Adagio a mi país” e intérprete “Alfredo Zitarrosa” pertenece a dos usuarios. Uno de ellos tiene los valores correctos mientras que el otro tiene como atributos “Adagio a mi país” e intérprete “A. Zitarrosa”. PGMúsica en este caso considera que esas dos canciones son diferentes, lo cual ocasiona un error en el dominio de la aplicación, siendo esta una de las debilidades del sistema. Este problema lo tienen la mayoría de las aplicaciones de este tipo. Algunos sugieren alternativas, como el caso de Audioscrobbler que recomienda el uso del sistema MusicBrainz[4], que es un sitio que dispone de la información de cientos de temas musicales, ofreciendo servicios para que las personas puedan obtenerlos. Sin embargo, hasta el momento no se han encontrado alternativas donde el propio sistema resuelva el problema.

[4] http://musicbrainz.org/

3.2 Recolección de Preferencias

Luego de definido el tipo de ítem y de planteadas las consideraciones acerca de ellos, se define la forma de recolectar las preferencias o valoraciones de los usuarios sobre los objetos que maneja el sistema.

El agrado o desagrado de las personas sobre los elementos es el insumo principal de este tipo de algoritmos. Además, esta información, para ser útil, debe ser de un volumen significativo para que el algoritmo de recomendación retorne resultados satisfactorios. La obtención de esa información no es una tarea trivial, ya que no es usual que los usuarios manifiesten concientemente sus preferencias por un conjunto significativo de ítems. Por ello, algunos sistemas tratan de absorber toda la información de preferencias que puedan obtener de los usuarios, siendo estas manifestadas concientemente por él o asumiéndolas en base a su comportamiento.

Se distinguen dos formas posibles de recolección de esta información: información explicita que es la manifestación directa del usuario acerca de un objeto (por medio de un puntaje), e información implícita que implica hacer una valoración sobre ciertas acciones del usuario en el sistema. Los sistemas de recomendación basados en filtrado colaborativo no siempre consideran la recolección de información implícita. Dependiendo del tipo de aplicación puede resultar poco intuitivo, además de poco confiable, hacer alguna valoración sobre las acciones de los usuarios, pero eso depende claramente de cada sistema y dominio. Es importante recordar que estas aplicaciones sufren muchas veces la falta de información, y considerar algún enfoque implícito, como apoyo a la recolección explícita, puede ayudar a atacar este problema.

En PGMúsica se manejan estas dos formas de recolección, obteniendo información implícita y explícita. El sistema permite que cada usuario registre explícitamente sus preferencias sobre las canciones en una escala numérica donde a mayor puntaje se entiende como una preferencia mayor.

La recolección de información implícita se realiza registrando la cantidad de veces que un usuario escucha una canción. Es bastante intuitivo pensar que cuantas más veces un usuario escucha una canción, más es de su agrado. Como criterio se considera que un usuario escuchó una canción, cuando lo hizo en un tiempo superior al 80% del tiempo total de su duración.

El algoritmo de recomendación utilizado recibe información de las preferencias de los usuarios en una escala de puntajes. Sin embargo, la información implícita se expresa en una escala distinta a la utilizada en la información explícita. Por ejemplo, considerando una escala numérica de puntajes de 1 a 5, la preferencia de una canción puntuada con 5 por un usuario, puede no significar lo mismo que una canción escuchada 5 veces. Por lo tanto, es necesario realizar algún tipo de procesamiento para las preferencias implícitas que transformen el numero de veces que se escucha cada canción en un valor de la escala numérica, de forma que la información que recibe el algoritmo le sea indistinta.

Los puntajes registrados de forma explícita son considerados como información totalmente fidedigna, la cual no podrá ser modificada por el sistema, sin importar la información implícita de la que se disponga. Cuando no se cuenta con información explícita, el sistema se basa en la información que recolecta implícitamente para generar un valor estimado del puntaje que el usuario le daría a esa canción.

```
Dado el usuario u
Para cada ítem i donde el usuario u manifestó implícita o
explícitamente su preferencia:
      - si el ítem tiene puntaje explícito, será el que se
      considerará en el algoritmo.
      - sino, se realiza el cálculo de predicción de puntaje.
```

Cuadro 3.1: Algoritmo general sobre puntajes

En el Cuadro 3.1 se explica en qué casos se calculan los puntajes de predicción a partir de la información implícita.

Cálculo de puntaje

El objetivo es estudiar el comportamiento general del usuario teniendo en cuenta cómo escucha música y cómo puntúa las canciones, para poder predecir el puntaje que le daría a las canciones aún no puntuadas pero sí escuchadas por él.

Para el proceso de transformación de preferencias, se calculan las siguientes variables por usuario:

1) máximo, mínimo y promedio de veces que escucha canciones. (Ver cuadro 3.2)

escuchaMax (E_{max})	= la cantidad máxima de veces que el usuario escuchó una canción
escuchaMin (E_{min})	= la cantidad mínima de veces que el usuario escuchó una canción
escuchaProm (E_{prom})	= suma de la cantidad de escuchas sobre el total de canciones escuchadas.

Cuadro 3.2: Variables definidas sobre el número de veces que se escucha una canción

Estos valores aportan la forma en que un usuario escucha música, pudiendo a partir de ella establecer su comportamiento en relación a como escucha música.

2) máximo, mínimo y promedio de puntajes brindados explícitamente. (Ver cuadro 3.3)

puntajeMax (P_{max})	= el puntaje máximo asignado por el usuario a una canción.
puntajeMin (P_{min})	= el puntaje mínimo asignado por el usuario a una canción.
puntajeProm (P_{prom})	= suma de todos los puntajes sobre el total de canciones puntuadas.

Cuadro 3.3: Variables definidas según las puntuaciones del usuario

Con estos valores se obtiene información acerca de cómo un usuario determinado puntúa canciones en general.

A partir de estos valores, se calculan los puntajes que predice el sistema, para las

canciones que el usuario ha escuchado, pero que aún no ha puntuado.

La idea entonces es determinar qué puntaje le daría el usuario a la canción *i*, P_i, la cual ha sido escuchada una cantidad de veces igual a E_i.

Con los puntajes máximo, mínimo y promedio, así como con la escucha máxima, mínima y promedio, podemos observar que el usuario puede estar en 3 estados diferentes, que se mencionan a continuación:

a) Se tiene información general de cómo el usuario puntúa y escucha canciones.

Se determina una relación entre los puntajes que el usuario asigna a las canciones y la cantidad de veces que las escucha. Así, a partir de la información de escucha (E_i), se estima el puntaje que sería dado a la canción i.

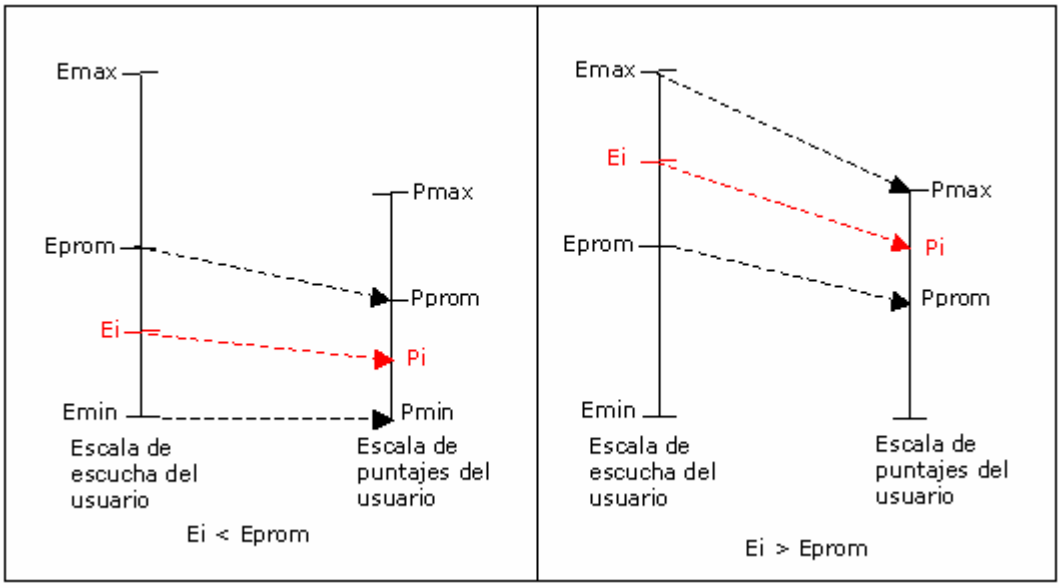

Figura 3.2: Diagrama de transformación del número escuchas en un valor de la escala

Para esto se consideran las variables de referencia (E_{mim}, E_{prom}, $E_{\max}$, $P_{\min}$, P_{prom}, $P_{\max}$) y se hace una correspondencia entre (E_{mim}, $P_{\min}$), ($E_{\max}$, $P_{\max}$) y (E_{prom}, P_{prom}) como se muestra en la Figura 3.2.

Para el caso donde E_i es menor al promedio de escucha, se realizan las siguientes correspondencias:

- El intervalo (P_{prom}, P_i) debe corresponderse con el intervalo (E_{prom}, E_i).
- De igual forma, el intervalo (P_{prom}, $P_{\min}$) debe corresponderse con el intervalo (E_{prom}, E_{mim}).

$$\frac{(P_{prom} - P_i)}{(P_{prom} - P_{min})} = \frac{(E_{prom} - E_i)}{(E_{prom} - E_{min})}$$

Despejando Pi

$$P_i = P_{prom} - ((E_{prom} - E_i) * (P_{prom} - P_{min})) / (E_{prom} - E_{min})$$

Cuadro 3.4: Análisis sobre las fórmulas de cálculo de puntajes

Análogamente, se realiza el mismo razonamiento para el caso donde E_i es mayor que el promedio de escucha.

La correspondencia entre los puntajes dados por el usuario y las veces que escucha canciones permiten predecir el puntaje para un ítem que aún no tiene un puntaje registrado. Se busca que, si E_i es mayor a la escucha promedio, se obtendría un puntaje por encima de la media. En caso contrario, se obtiene un puntaje menor al puntaje promedio. El análisis de las fórmulas que se presentan en el cuadro 3.5, se realiza en el cuadro 3.4

E_i = número de veces que el usuario escuchó el ítem i

1 - Si $E_i < E_{prom} \rightarrow P_i = P_{prom} - ((E_{prom} - E_i) * (P_{prom} - P_{min})) / (E_{prom} - E_{min})$

2 - Si $E_i > E_{prom} \rightarrow P_i = P_{prom} + ((E_i - E_{prom}) * (P_{max} - P_{prom})) / (E_{max} - E_{prom})$

Cuadro 3.5: Cálculo de puntaje de predicción.

b) No se tiene información de cómo el usuario puntúa canciones
En este caso no se tienen valores de referencia para establecer los intervalos de correspondencia. Se considera que los puntajes máximo, mínimo y promedio son el máximo, mínimo y promedio de la escala. Esto ocurre, o bien cuando no tenemos puntajes registrados, o bien cuando el usuario ha asignado el mismo puntaje a todas sus canciones, debido a que en este último caso no se cuenta con información comparativa.

c) No se tiene información de cómo el usuario escucha canciones
Este sería el caso en que el usuario escuchó aproximadamente la misma cantidad de veces todas las canciones que están asociadas a él en el sistema. Entonces se calcula el puntaje P_i como se indica en el cuadro 3.6.

$$E_i \rightarrow P_i = P_{prom}$$

Cuadro 3.6: Cálculo de puntaje cuando no se tiene información de escucha.

Si no se tiene un puntaje promedio debido a que el usuario no ha puntuado canciones, entonces Pi toma el valor medio de la escala (VME).

$$E_i \rightarrow P_i = \text{VME}$$

Cuadro 3.7: Cálculo de puntaje cuando el usuario no ha puntuado canciones.

A continuación se muestra un ejemplo del cálculo de puntaje que se realiza a un usuario a partir del la música que ha escuchado, considerando una escala de puntajes de 1 a 5.

Ejemplo:

Se considera la información registrada de puntajes y escucha de música como se muestra en el Cuadro 3.8. Las canciones 1 y 2, no están puntuadas pero si están escuchadas por él. A continuación se calculan sus puntajes estimados.

Nro.Canción	Puntaje	#Escuchas
4	2	3
2	0	6
7	3	2
8	4	1
1	0	1
3	5	7
5	4	3
9	3	2
6	3	3
17	2	1
24	1	0
23	5	3

Cuadro 3.8: Ejemplo - Información relacionada con canciones del usuario

Por lo tanto se calcula:

a) las variables que refieren a los puntajes explícitos

P_{max} = 5
P_{min} = 1
P_{prom} = (2+3+4+5+4+3+3+2+1+5) / 10 = 3.2

b) las variables que refieren las cantidades de escucha

E_{max} = 7
E_{min} = 1
E_{prom} = (6+3+2+1+1+7+3+2+3+1+3+2) / 12 = 2.83

c) el valor de los puntajes de las canciones 1 y 2

Canción 1:

Como la cantidad de escucha de la canción 1 es igual a uno, menor al promedio (2.83) entonces se aplica la primer fórmula presentada en el Cuadro 3.4.

E_1 = 1
Como $E_1 < E_{prom} \rightarrow P_1 = P_{prom} - ((E_{prom} - E_i)*(P_{prom} - P_{min})) / (E_{prom} - E_{min})$

P_1 = 3.2 – ((2.83 - 1) * (3.2 - 1)) / (2.83 - 1) = **1**

El puntaje calculado para la canción 1 es igual a 1, por lo tanto ese valor será el dado al algoritmo de recomendación.

Canción 2:

La cantidad de escucha de la canción 2 es 6 y la escucha promedio es igual a 2.83, por lo tanto la escucha de la canción 2 es mayor que la escucha promedio, por ello será utilizada la segunda fórmula presentada en el Cuadro 3.4.

$E_2 = 6$
Como $E_2 > E_{prom} \rightarrow P_i = P_{prom} + ((E_i - E_{prom})*(P_{max} - P_{prom}))/(E_{max} - E_{prom})$

$P_2 = 3.2 + ((6 - 2.83) * (5 - 3.2)) / (7 - 2.83) = 4.568 \sim= 5$

El algoritmo de recomendación recibe puntajes enteros, por lo tanto el resultado obtenido para el puntaje de la canción 2, es redondeado a 5.

3.3 Recomendaciones

Un aspecto a considerar en los sistemas de recomendación basados en filtrado colaborativo, es determinar el momento en que se realizan los cálculos de la recomendación.

Como se ha mencionado, este tipo de sistemas debe contener una gran cantidad de información de preferencias de sus usuarios, para brindar recomendaciones de calidad. Por otro lado, la mayoría de los algoritmos que se utilizan en estos sistemas realizan una gran cantidad de cálculos por lo que el costo de procesamiento puede ser muy alto si se cuenta con un volumen significativo de preferencias.

Por tales razones, algunos sistemas optan por realizar los cálculos de las recomendaciones mientras el usuario no esta conectado al sistema, de forma que cuando éste se conecte tenga una recomendación calculada y no tenga que esperar por ella.

Algunos trabajos realizados sobre este problema proponen realizar el cálculo del vecindario de cada uno de los usuarios periódicamente, y al momento del pedido de recomendación utilizar el último vecindario calculado. Si bien esto descongestiona los cálculos para generar la recomendación, tiene como aspecto negativo el hecho de que el cálculo del vecindario no contempla la información más reciente que se haya ingresado.

En PGMúsica se decide realizar las recomendaciones al momento de recibir el pedido de recomendación. Esto implica realizar los cálculos del algoritmos en el instante que el usuario la solicita. Claramente existe un problema de escala asociado a esta decisión: en tanto aumente la cantidad de usuarios y los ítems en el sistema, la carga que implican los cálculos podrían provocar una demora excesiva. Mientras no suceda esto, las recomendaciones que brinda el sistema son vigentes y concuerdan con la las preferencias actuales del usuario.

La idea general de PGMúsica es realizar recomendaciones de canciones y permitirle a los usuarios escucharlas. Sin embargo no todas las canciones registradas en el sistema son canciones disponibles para ser enviadas mediante la técnica de *streaming*. Esto limita la cantidad y diversidad de temas musicales a ser recomendados en un momento dado. Por ello PGMúsica realiza dos tipos de recomendaciones.

Si el usuario solicita la recomendación desde la aplicación de música, solo se le recomendarán canciones que el usuario no haya escuchado y que estén disponibles a través de otros usuarios conectados al momento de solicitar la recomendación. Este tipo de recomendación la denominamos "*Recomendación online"*. Es importante aclarar que también se consideran las canciones compartidas por el usuario que solicita la recomendación, dado que no necesariamente conoce todas las canciones disponibles allí.

Por otro lado el usuario también tiene la posibilidad de obtener una recomendación sobre todas las canciones registradas en el sistema, sin importar cuáles están disponibles para ser escuchadas. Esta recomendación la denominamos "*Recomendación offline",* porque las canciones contenidas en la recomendación no pueden ser escuchadas, pero sí se muestra detalles informativos de las mismas.

La recomendación *online* cumple con el objetivo inicial del proyecto de poder escuchar las canciones sugeridas. En cambio, la recomendación *offline* no fue un requerimiento inicial, pero se considera útil que el usuario pueda obtener mejores resultados, pues no se restringe el conjunto de canciones potenciales a ser recomendadas, sino que se consideran todos los temas musicales del sistema.

El procedimiento general para realizar las recomendaciones ofrecidas se detalla en el Cuadro 3.9, independientemente del tipo de recomendación (*online*, *offline*).

```
Dado el usuario u que solicita una recomendación

Para todos los usuarios se calculan los puntajes de información
explícita

Si para el usuario u se puede calcular al menos un vecino
   -  Se utiliza el algoritmo de filtrado colaborativo
Sino, si se pueden generar vecinos según procedimiento Cold Start
   -  Se utiliza procedimiento Cold Start
Si ninguna de estas opciones genera vecinos, se recomiendan las
10 canciones más escuchadas del sistema
```

Cuadro 3.9: Procedimiento general de generación de recomendaciones

De las predicciones generadas, se seleccionan los diez puntajes más altos, y esa es la recomendación que se presenta al usuario.

3.3.1 Algoritmo

Como se menciona en el capítulo 2, existen varios algoritmos para realizar las recomendaciones en un sistema de este tipo. En el análisis realizado por Breese, Hecherman y Kadie [5] se concluye que los algoritmos con los que se obtienen mejores resultados para sistemas de recomendación basados en filtrado colaborativo son Correlación de Pearson y Redes Bayesianas. Sin embargo para los tipos de sistemas donde el puntaje sobre los ítems no es binario, son mejores los resultados utilizando la fórmula general de los algoritmos basados en memoria.

En PGMúsica se cuenta con un rango no binario de puntajes que pueden ser adjudicados a cada uno de los ítems del sistema. Además, el algoritmo basado en

memoria es simple de calcular y requiere menos cantidad de información para calcular recomendaciones que el basado en modelo. Entonces se selecciona para PGMúsica el algoritmo basado en memoria, donde la fórmula utilizada es la que se presenta en la sección 2.1.1.

$$p_{a,j} = \overline{v_a} + k\sum_{i=1}^{n} w(a,i)(v_{i,j} - \overline{v_i})$$

Fórmula 3.1: Fórmula del algoritmo basado en memoria

donde:

$p_{a,j}$ = predicción del puntaje del usuario *a* sobre el ítem *j*

$\overline{v_a}$ = puntaje promedio de los votos del usuario *a*

$w(a,i)$ = valor de similitud entre el usuario a y el usuario *i*

$V_{i,j}$ = puntaje brindado por el usuario *i* para el ítem *j*.

$\overline{v_i}$ = puntaje promedio de los votos del usuario *i*

En esta fórmula, el k es un factor de normalización para que el valor que se suma al promedio de votos del usuario activo, sea coherente con la escala de puntajes del sistema.

Para lograr ese objetivo se plantea k tal cual se presenta en la Fórmula 3.2.

$$\frac{1}{\sum_{i=1}^{n} w(a,i)}$$

Fórmula 3.2: Fórmula de factor de normalización

Utilizando este k, puede ocurrir que el resultado del puntaje de predicción sea un valor por encima o por debajo del máximo y mínimo puntaje permitido. En este caso, se asignan el máximo y el mínimo puntaje de la escala respectivamente.

Similitud entre Usuarios

En los algoritmos basados en memoria, se calcula la similitud entre usuarios. En PGMúsica para realizar este cálculo se elige el coeficiente de correlación de Pearson, ya que es uno de los más estudiados y utilizados. Además, es en general el algoritmo más aceptado para este tipo de sistemas, debido a los buenos resultados obtenidos.

A la hora de implementar el coeficiente de correlación de Pearson para calcular el peso, *w(a,i)*, no se utiliza la ecuación planteada en la sección 2.1.1. sino que en su lugar se emplea una regresión lineal, que se presenta en la Fórmula 3.3. Material de referencia de sistemas ya implementados indican que esta regresión

ayuda a disminuir el error en el cálculo debido a que se omite la raíz cuadrada en la fórmula.

$$w(a,i) = \frac{\sum_j (v_{a,j} * v_{i,j}) - h * \overline{v_a} * \overline{v_i}}{\left(\sum_j (v_{a,j}^2) - h * \overline{v_a^2}\right)\left(\sum_j (v_{i,j}^2) - h * \overline{v_i^2}\right)}$$

Fórmula 3.3: Regresión lineal de la correlación de Pearson

donde:

$v_{a,j}$ = voto (puntaje) del usuario activo sobre el ítem j

$v_{i,j}$ = voto del usuario i sobre el ítem j

h = cantidad de vecinos del usuario activo

$\overline{v_a}$ = promedio de votos del usuario activo

$\overline{v_i}$ = promedio de votos del usuario i

Análisis acerca del vecindario

Una de las decisiones fundamentales a la hora de utilizar el algoritmo de filtrado colaborativo es evaluar qué criterio se utiliza para indicar que dos usuarios son vecinos. Por otro lado también se evalúa el tamaño mínimo y máximo de vecinos necesarios para generar mejores predicciones.

Para calcular el vecindario del usuario activo, se tienen en cuenta únicamente los usuarios que tienen ranqueado por lo menos dos ítems en común con él. Esta consideración se debe a que si tienen únicamente un ítem en común, no se podría aplicar la fórmula del coeficiente de correlación de Pearson, debido a que el resultado sería siempre uno independientemente de los valores asignados a los elementos. Por lo tanto, el cálculo del coeficiente de correlación de Pearson no se realiza sobre todo el conjunto de usuarios del sistema, sino sobre ese subconjunto.

Luego de calcular todas las correlaciones se determina el vecindario, como se analizó en la sección 2.1.1. Para la selección del vecindario es necesario establecer una cantidad máxima y mínima de vecinos, determinado de esta forma el conjunto de usuarios donde se trabajará para realizar las predicciones.

En PGMúsica, se elige como mínimo y máximo tamaño del conjunto de vecinos, uno y once respectivamente. La elección de la cantidad mínima, se debe a que en estos sistemas es difícil encontrar un vecindario considerablemente grande. Por lo tanto, se decide realizar recomendaciones aún cuando el usuario tiene únicamente un vecino, aunque la recomendación no sea precisa. Esto se debe a que se prefiere utilizar el algoritmo de recomendación con un usuario, a no disponer de una recomendación. Por otro lado, la elección de la máxima cantidad de vecinos, se elige luego de realizar un conjunto de pruebas (descriptas en el capítulo 5).

3.3.2 Cold Start

Como se describe en el capítulo 2, uno de los problemas más importantes a resolver en este tipo de sistemas es el denominado *Cold Start*. Este problema se presenta cuando un usuario es nuevo en el sistema y por tanto no se cuenta con la suficiente información sobre sus preferencias como para poder recomendar utilizando el algoritmo de filtrado colaborativo. Incluso puede suceder que el sistema conozca un conjunto de preferencias del usuario sobre determinados elementos pero que ningún otro usuario del sistema haya valorado esos mismos elementos. En ese caso el sistema no es capaz de relacionarlo con otros usuarios (quedando en el mismo estado que un usuario nuevo), no pudiendo generarle recomendaciones. Esto hace notar que la posibilidad de generar recomendaciones depende directamente del número y variedad de usuarios en el sistema.

En PGMúsica, se considera que un usuario tiene el problema de *Cold Start* cuando no se pueden encontrar vecinos para él. Esto se debe a que el algoritmo de recomendación necesita al menos un vecino para realizar los cálculos.

Cuando el usuario está en esa situación, el sistema podría reaccionar de distintas formas: desde no realizar recomendaciones, hasta exigirle que tome determinadas acciones, como por ejemplo puntuar una lista de canciones arbitraria, para entonces sí poder encontrarle un vecindario y realizarle una recomendación.

En la solución propuesta se intenta evitarle al usuario realizar tareas que puedan resultarle tediosas o pesadas, como podría ser puntuar una lista de canciones. Pero también es un objetivo para el funcionamiento de PGMúsica que el usuario siempre reciba alguna recomendación. Para lograr esto se debe tomar alguna medida para atacar el problema del *Cold Start*.

La decisión que se tomó para este problema es aplicar un enfoque basado en contenido considerando además los datos personales del usuario. Para ello en el registro de usuario se requiere ingresar los siguientes datos que se consideran relevantes:

- Estilo de música de preferencia
- Fecha de nacimiento
- País de residencia
- Sexo

Cuando se solicita al usuario que elija su estilo de música de preferencia, es para poder vincularlo con otros que compartan la preferencia.

La elección de la fecha de nacimiento se utiliza para vincular al usuario con otros que sean como máximo 5 años mayores o hasta 5 años menores. Es de esperar que aspectos generacionales influyan en los gustos musicales. Esta información es útil para generar una recomendación inicial un poco más precisa, sin conocer aún las preferencias sobre las canciones. De la misma manera, tanto el país de residencia, como el sexo del usuario pueden ayudar a vincular al usuario activo a otros usuarios con gustos similares.

A partir de esta información se realiza el procedimiento que se describe a continuación para poder realizar una recomendación al usuario, mientras está en la etapa de *Cold Start*.

Procedimiento COLD START

En primer lugar se realiza la construcción de un vecindario “ficticio” para el usuario activo. Para ello, del conjunto de usuarios registrados en el sistema, se toman aquellos que sean del mismo país, del mismo sexo, que hayan elegido el mismo estilo de música de preferencia y que estén en un entorno de más menos cinco años de edad.

Luego se seleccionan las canciones puntuadas explícitamente por ese conjunto de usuarios, tales que el puntaje sea mayor a tres. Si no hay suficientes canciones en esas condiciones, se consideran las canciones puntuadas implícitamente. El conjunto de diez canciones que se recomienda se selecciona tomando cinco canciones del estilo de preferencia del usuario activo y cinco de otros estilos. Brindándole recomendaciones de canciones variadas se intenta que tenga acceso a otros estilos de música y de esta forma permitirle escuchar y eventualmente puntuar distintos estilos, lo cual ayudaría a generar otros vecinos.

Este procedimiento se practica hasta que el perfil del usuario comience a determinarse, de modo que se establezca un conjunto de vecinos reales. El perfil del usuario será definido en base a las puntuaciones que realice sobre los temas musicales o sobre las canciones que escuche en su reproductor de audio.

Capítulo 4 Descripción de la Implementación

En este capítulo se describe la implementación de la solución del sistema PGMúsica considerando el diseño propuesto en el capítulo 3. Se puntualizan las decisiones que se toman con respecto a la arquitectura e implementación. También se describen las herramientas y librerías utilizadas así como las características consideradas para seleccionarlas. Finalmente se detallan las funcionalidades de cada uno de los componentes de la arquitectura.

4.1 Descripción general del sistema

PGMúsica es un sistema de recomendación de música basado en filtrado colaborativo, donde a partir de las preferencias de los usuarios sobre las canciones, se ofrecen recomendaciones. Además el sistema permite que los usuarios puedan escuchar las canciones recomendadas.

El sistema consta por un lado de usuarios que brindan sus preferencias sobre las canciones así como el número de veces que las escuchan. Por otro lado, se cuenta con un motor de recomendaciones donde a partir de esa información realiza los cálculos mediante el algoritmo de filtrado colaborativo, para luego enviarle la lista de canciones recomendadas. Por lo tanto se define como diseño general del sistema, una arquitectura cliente-servidor.

A nivel de cliente la idea es recolectar la mayor cantidad de información posible sobre las preferencias de los usuarios sobre las canciones. Mientras que a nivel de servidor la idea es enviar recomendaciones *online* que podrán ser escuchadas por el usuario, así como recomendaciones *offline* y el detalle de éstas.

Se entiende importante para el usuario que tanto el puntaje que asigna a las canciones como la recepción de audio de aquellas recomendadas, se realice desde la misma aplicación. Teniendo en cuenta además que se plantea registrar cuáles canciones escucha el usuario y cuántas veces lo hace, se decide que el cliente sea una extensión de una aplicación de música. Esto facilita la tarea del usuario, ya que no tiene que utilizar aplicaciones separadas. En este caso se selecciona Winamp [21] como reproductor de audio dado que permite extender sus funcionalidades ofreciendo la documentación adecuada. Winamp es un reproductor de audio y video de libre acceso, que trabaja sobre la plataforma Windows, soportando múltiples formatos de archivos, entre ellos MP3.

Además, el usuario puede obtener recomendaciones que consideren todas las canciones del sistema y no únicamente aquellas que están disponibles para escuchar en el momento de la recomendación. Estas recomendaciones (offline) serán ofrecidas al usuario a través de un sitio *Web*.

Por ello se define que la interacción del usuario con el sistema, a nivel de cliente, se realice mediante dos vías. Por un lado una extensión de la aplicación Winamp (*plugin*) y por otro lado un sitio *Web*.

4.1.1 Plugin de Winamp

Algunas aplicaciones dejan abierta la posibilidad de extender sus funcionalidades mediante la construcción de aplicaciones que interactúan, a través de una interfaz definida, con la aplicación principal sin afectar las funcionalidades ya existentes. Un tipo particular de estas extensiones son los *plugin (o plug-in)*. La diferencia principal entre los *plugins* y las extensiones clásicas de las aplicaciones, es que ellos generalmente trabajan sobre la interfaz de usuario del programa principal y tienen bien definido su posible conjunto de acciones.

En PGMúsica se construye un *plugin* del reproductor de audio Winamp. Este *plugin* es el encargado de monitorear la música escuchada y puntuada por los usuarios. También permite acceder a las recomendaciones *online* realizadas por el sistema.

A través de la funcionalidad de ranqueo que provee Winamp, se permite a los usuarios asignarle entre una y cinco estrellas a cada canción, según su nivel de preferencia. Este puntaje es recolectado por el *plugin* y enviado al servidor.

El *plugin* también monitorea la música que escucha el usuario. Luego de haber escuchado el 80% de la duración de un tema musical, el *plugin* se encarga de enviar la información de este tema al servidor indicando además que ha sido escuchado.

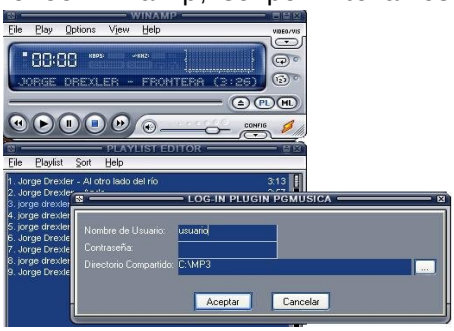

Figura 4.1: Winamp y plugin PGMúsica

Por otro lado el sistema realiza recomendaciones *online* permitiendo a los usuarios escuchar las canciones recomendadas. En este sentido los *plugins* se comunican entre sí para enviar y recibir canciones. El servidor central es quien conoce qué usuario tiene las canciones que desea enviar como recomendación, indicándole al usuario que desea obtener la recomendación, con qué *plugin* debe conectarse para recibir los temas. Por eso se hace necesario que cada usuario posea un directorio donde ubique las canciones que desea compartir y por lo tanto puedan ser recomendadas.

La música que se comparte con los demás usuarios del sistema consiste en archivos de tipo *MP3* (o *MPEG-3*), siendo los únicos archivos considerados por PGMúsica. Esta decisión se basa en que este formato es uno de los más utilizados y cuenta con etiquetas que describen algunos de los atributos de las canciones como son nombre, artista, álbum, género y año entre otros.

El envío entre usuarios de las canciones sugeridas se realiza mediante *streaming*. La tecnología de *streaming* permite ver y oír contenido de audio y video mientras es transmitido. Los contenidos multimedia puedan empezar a ser reproducidos segundos después de establecer la conexión, sin importar el tamaño total del archivo. Esto marca la principal diferencia con la descarga de archivos, que obliga a recibir la totalidad de la información antes de que pueda ser utilizada.

A diferencia de otras tecnologías que dependen de la transmisión de datos a través de una red, el *streaming* posee restricciones de tiempo, pues debe respetar los tiempos de reproducción del archivo de audio o video. Se requiere, por esta misma razón, una mayor coordinación cliente-servidor, dado que el ritmo de transmisión está marcado por los tiempos de reproducción del contenido.

En estas tecnologías es importante considerar cómo resolver la pérdida de paquetes en la red. La retransmisión no es, en general, una opción viable para este tipo de problemas ya que un paquete que llega tarde al receptor es un paquete inútil. En general se aplican técnicas de reconstrucción o bien técnicas para minimizar el impacto de un paquete perdido. El problema de la retransmisión también influye al momento de elegir un protocolo. Los protocolos implementados sobre TCP (como por ejemplo HTTP) no son eficientes ya que retransmiten la información perdida. Es por esto que en general los protocolos mas utilizados, como RTP y RTSP, trabajan sobre UDP.

En términos de arquitectura para este tipo de tecnología, la más simple es la arquitectura *Unicast*, donde el servidor envía la información a un único destinatario utilizando alguno de los protocolos mencionados con anterioridad. Parece muy poco adecuada en situaciones como la transmisión de un evento en directo por Internet, donde la cantidad de destinatarios puede ser muy grande, y esto implica que el servidor deba repetir la información a cada uno de los destinatarios. Una posible solución a este problema es el *Multicast*, donde el servidor envía el audio a otro *hardware* encargado específicamente de distribuir la información repitiéndola.

El *plugin* de PGMúsica utiliza esta técnica actuando de servidor de *streaming* de la música que es compartida por cada cliente. Por lo tanto, cada usuario al escuchar las canciones de una recomendación *online* no realiza la descarga del archivo *mp3*.

4.1.2 Sitio Web

Si bien los objetivos iniciales del proyecto no planteaban el desarrollo de un sitio *Web*, se decidió realizarlo debido a dos motivaciones principales. En primer lugar, ofrecer más posibilidades al usuario para que ingrese la mayor cantidad de información al sistema sobre sus preferencias. La captación de más información permite mejorar la precisión del sistema, además de ayudar a superar lo antes posible la etapa de *Cold Start*. En segundo lugar, se quiere mostrar al usuario mayor información sobre los detalles de las recomendaciones. Como plantean Herlocker, Constan y Ried [8], los usuarios prefieren sistemas que justifiquen o muestren información sobre el proceso de generación de la recomendación.

Las funcionalidades del sitio *Web* son:

- Registro al sistema
- Obtener recomendación *offline*
- Ver detalle de recomendaciones recibidas
- Ver y modificar puntajes de canciones conocidas
- Buscar y puntuar canciones

El registro permite al usuario crear un seudónimo para ingresar a la aplicación y completar los datos que permitan generar recomendaciones durante la etapa de *Cold Start*. Como se menciona en la sección 3.3.2, los datos obligatorios que se deben ingresar son: estilo de música de preferencia, fecha de nacimiento, país de residencia y sexo, además de usuario y contraseña. Los estilos musicales sobre los que se basa el sistema son los provistos por Winamp. Esta lista contiene más de 130 estilos entre los cuales hay elementos que no presentan diferencias significativas, por ejemplo "*Jazz*" y "*Acid Jazz*". Una lista más corta ayuda a vincular más fácilmente a los usuarios en la etapa de *Cold Start*. Por esta razón se decide condensar la lista a 33 estilos musicales.

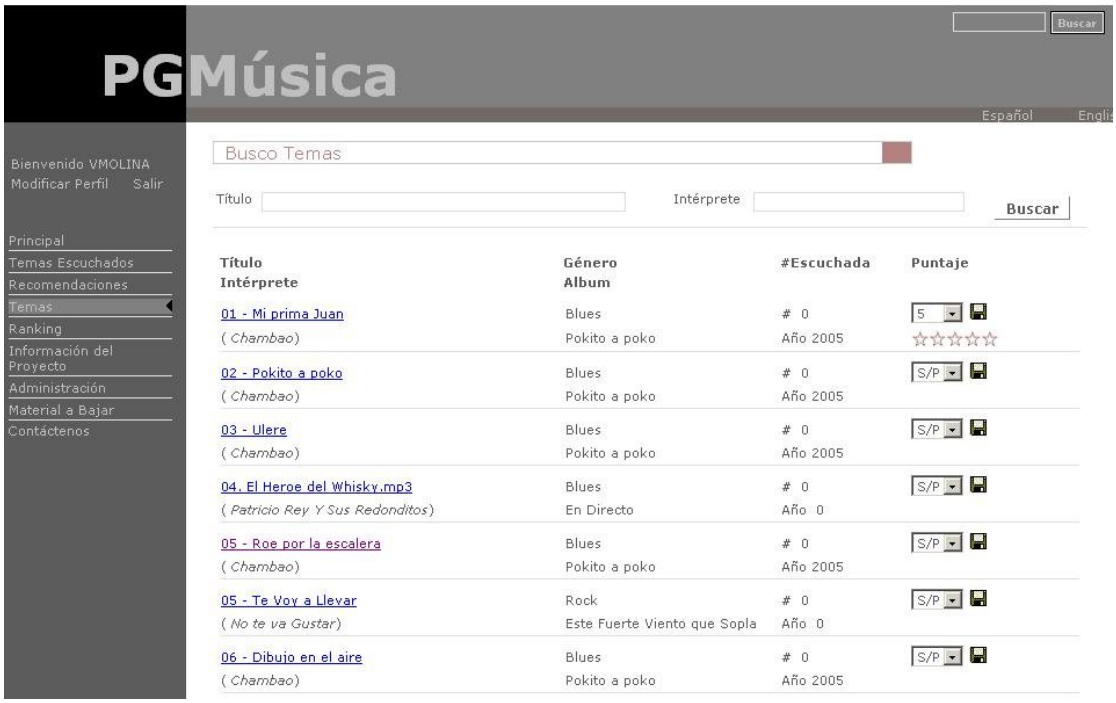

Figura 4.2: Búsqueda y asignación de puntajes a canciones en el sitio *Web*

Con respecto a las recomendaciones realizadas, el sitio *Web* permite al usuario ver la justificación, informándole cómo se generaron. Si la recomendación se realiza utilizando un vecindario, se le informa cuáles son los vecinos seleccionados para el cálculo y cuál es el valor de similitud con los mismos. Si la recomendación no es generada utilizando un vecindario porque, por ejemplo, el usuario es nuevo en el sistema y no se pude generar un conjunto de vecinos apropiado, se le informa que se le recomiendan las diez canciones más escuchadas. El usuario puede también puntuar las canciones recomendadas asociándole un valor entre uno y cinco según su nivel de preferencia. Se elige esta escala de puntaje para PGMúsica por contener un rango razonable de valores, en los cuales el usuario puede manifestar y diferenciar sus preferencias.

A través del sitio *Web* también se le ofrece la posibilidad de ver la información que es registrada de sus preferencias, tanto la cantidad de veces que escucha cada canción como los puntajes que asigna (Figura 4.2). A su vez permite modificar los puntajes registrados.

Los usuarios en general no tienen disponibles en archivos MP3, para escuchar o puntuar a través del *plugin* de Winamp, todas las canciones que conocen. Por ello también se permite buscar y puntuar canciones sobre todas las registradas en el sistema.

El sitio *Web* puede ser utilizado como sistema de recomendación sin siquiera instalar el *plugin*. El usuario puede registrarse, puntuar canciones y recibir recomendaciones sin haber escuchado una sola canción en Winamp. Si bien esto no fue un objetivo planteado al momento de decidir implementar el sitio, es una posibilidad más de interacción entre los usuarios y el sistema.

4.2 Arquitectura del sistema PGMúsica

Como se describe en la sección 4.1, el sistema consta de un servidor central encargado de recibir y procesar la información que le es enviada, a través del *plugin* desarrollado para Winamp o a través del sitio *Web* del sistema. Como la comunicación para el envío y recibo de *streaming* es cliente-cliente, se determina a este nivel una arquitectura *Peer-to-Peer*. El diseño general de la arquitectura del sistema se puede observar en la Figura 4.3.

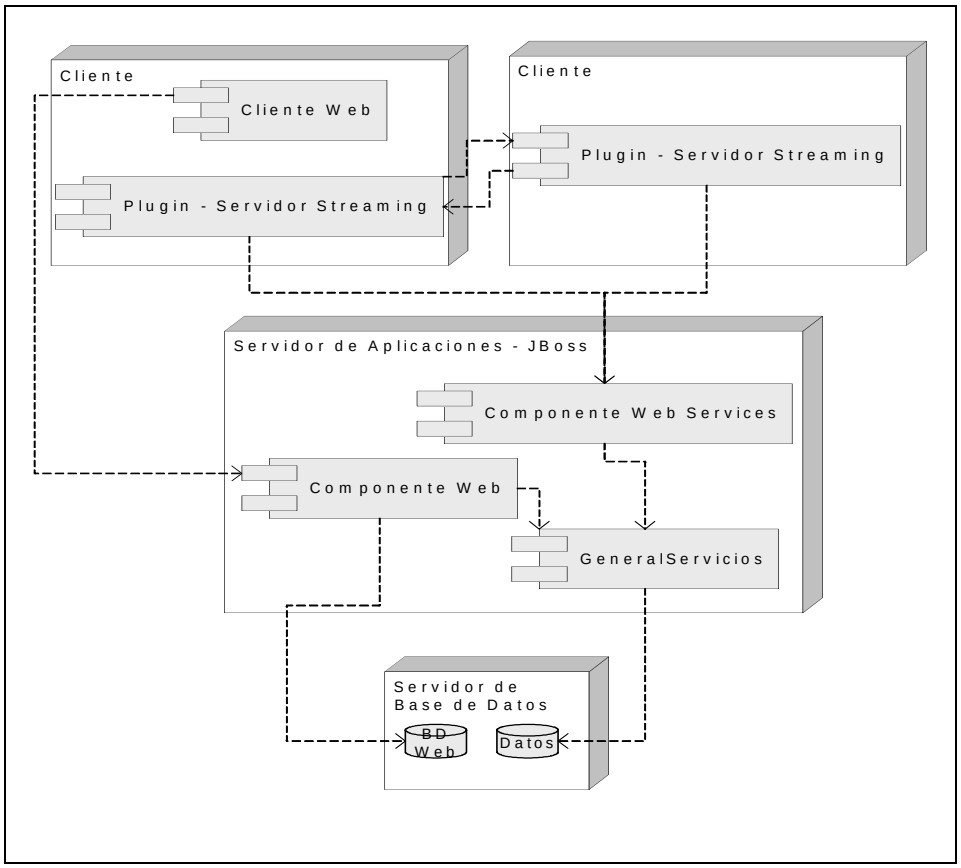

Figura 4.3: Diagrama de componentes y sus comunicaciones

La arquitectura se encuentra fragmentada en dos porciones bien diferenciadas. Una de ellas es la que contienen la lógica del nodo Cliente y la restante es la que considera la lógica del nodo Servidor. A su vez, cada una de las particiones está dividida en componentes encargados de un conjunto determinado de tareas. Se describe a continuación cada uno de los nodos y sus componentes.

4.2.1 Nodo Cliente

La partición cliente se divide en dos componentes denominados cliente *Web* y cliente *Plugin*.

El cliente *Web* es cualquier navegador de Internet (*Internet Explorer*, *Mozilla Firefox*, etc.) con el cual el usuario puede acceder al sitio *Web* del sistema.

El cliente *plugin* es una extensión de la aplicación Winamp, mediante la cual el sistema monitorea y envía la música escuchada y puntuada por el usuario al servidor de aplicaciones. Por otra parte, cada usuario comparte su música con los demás usuarios del sistema, siendo el cliente *plugin* el encargado de publicar la música que el usuario brinda a los restantes usuarios del sistema así como de recibir la música que le es enviada al usuario. Por lo tanto, este componente

contiene un servidor de *streaming,* el cual es accedido por los demás *plugins* residentes en otros clientes del sistema. De esta manera se determina una arquitectura *Peer-to-Peer* (P2P) [10] en lo que respecta a la comunicación directamente entre *plugins*, ya que cada cliente *plugin* es cliente y servidor de *streaming* a la vez.

Se puede observar que este enfoque es diferente al habitual en este tipo de sistemas. Por ejemplo, el sistema de recomendación de Last.fm, utiliza un enfoque cliente-servidor, dado que el *streaming* es enviado por el servidor central (que es quien contiene todos los recursos) a los clientes.

4.2.2 Nodo Servidor

En el nodo servidor reside la lógica principal de la aplicación, la cual se encuentra dividida en tres componentes:

- Componente *Webservices*, donde se publican los servicios que permiten la comunicación del *plugin* con el servidor central.
- Componente *Web,* donde se encuentra el sitio *Web* del sistema.
- Componente *GeneralServicios*, donde se encapsula la lógica principal del sistema. Esto implica las funcionalidades relacionadas a la comunicación con la base de datos del sistema, así como las relacionadas con la lógica de generación de recomendaciones.

Tanto el componente *Web* como el componente *Webservices*, toman las funcionalidades brindadas por el componente *GeneralServicios*, e implementan únicamente la lógica de comunicación y presentación al cliente. La razón de este diseño es que no estén acopladas la lógica principal del sistema con la de los diferentes clientes (*Web* y *plugin)*. De esta manera se hace muy simple crear otros tipos de clientes que utilicen las funcionalidades de *GeneralServicios* definiendo e implementando la lógica de comunicación cliente-servidor.

Al igual que las primeras aplicaciones que utilizaban arquitectura P2P, como fue el caso de la aplicación Napster, en este sistema se cuenta con un nodo servidor donde se centraliza toda la información sobre qué usuarios se encuentran conectados y qué recursos disponen cada uno de ellos. En este caso la arquitectura planteada no es una arquitectura *Peer-to-Peer* pura dado que para serlo no debería existir un servidor central, sino que varios de los nodos clientes deben actuar de servidor.

4.2.3 Comunicaciones

El *plugin* es una extensión de Winamp y como tal debe ceñirse a las herramientas y funcionalidades que ofrece este programa de escucha de música. Winamp permite crear *plugins* que extiendan sus funcionalidades y se recomienda el desarrollo de los mismos en el lenguaje C o C++. Como el servidor se decidió desarrollarlo en Java, se evalúan varias alternativas para resolver la comunicación, considerando que el cliente y servidor estarían implementados en distintos lenguajes. Entre otras posibilidades se evaluaron JNI, CORBA, un protocolo propio sobre HTTP o la utilización de w*eb services*. Dado que se entiende que el flujo de información entre el *plugin* y el servidor es escaso, se decide utilizar *web services* ya que es una alternativa que se ajusta mejor a los objetivos planteados anteriormente. Los *web services* resuelven la comunicación

cliente-servidor sin tener en cuenta el lenguaje de implementación de ambas partes. Además, se deja abierta la posibilidad de implementar otros clientes (para la recolección de preferencias de usuarios sobre canciones) en otros lenguajes sin tener que modificar en absoluto la lógica del servidor.

Para la comunicación que mantiene el *plugin* con sus pares debido al intercambio de *streaming*, se utiliza el protocolo RTSP (*Real Time Streaming Protocol*) sobre UDP. La arquitectura de comunicación de *streaming* que se realiza es *unicast* (punto a punto) dado que en PGMúsica en la mayoría de los casos, el servidor de *streaming* (*plugin*) envía el *stream* a un solo cliente en un momento determinado, siendo innecesario la implementación de una arquitectura *multicast*.

4.3 Lenguajes y Herramientas utilizadas

En esta sección se describen los lenguajes y herramientas utilizados en el proyecto, explicando las razones para su utilización.

4.3.1 Servidor

Para el desarrollo del servidor se decide utilizar el lenguaje Java por sus conocidas propiedades además de la posibilidad de usar herramientas y servidores de aplicaciones de libre acceso, útiles para el proyecto.

Como servidor de aplicaciones se evalúa utilizar *Jboss* [16] o *Tomcat* [17], decidiendo por el primero por ser un servidor de aplicaciones muy robusto que cumple con las características necesarias para aplicar el diseño comentado en la sección 4.1. JBoss, además de un contenedor de aplicaciones web y aplicaciones *J2EE*, también provee soporte para definir *web services*. Esta elección se define en una etapa inicial pero posteriormente para disponer de la aplicación en un ambiente público fue necesario, por asuntos administrativos del Instituto de Computación de la Facultad de Ingeniería, utilizar Tomcat como servidor. Por lo tanto, ser realiza una adaptación para que funcione sobre Tomcat. En consecuencia existen dos versiones de PGMúsica, una que funciona sobre Jboss y la otra sobre Tomcat.

Se utiliza como entorno de desarrollo *Eclipse* [18], el cual es gratuito y además posee extensiones para integrar el desarrollo en Java con el servidor JBoss, facilitando el trabajo con ambas herramientas.

El sitio *web* es desarrollado mediante la herramienta *Genexus* [19], la cual se utiliza para la generación de código en los lenguajes tradicionales, como lo es java. Genexus no es una herramienta gratuita, pero la empresa que la comercializa otorga licencias con fines académicos. La elección de esta herramienta se debe a que el sitio web no es un componente medular del sistema, pero su interfaz y funcionamiento debían ser los adecuados, en términos de usabilidad y amigabilidad. Se evalúa que estas características pueden lograrse mediante esta herramienta de forma sencilla, acortando notoriamente el tiempo de implementación, por lo cual es seleccionada y utilizada para su desarrollo.

4.3.2 Bases de Datos

Como manejador de bases de datos se elige *MySQL* [20], el cual puede ser utilizado libremente. Además dispone de un conjunto de herramientas que

también son de libre acceso y que facilitan su uso. En particular para este proyecto se utilizan dos de esas herramientas:

- *MySQL Administrator*: permite administrar la base de datos en lo que respecta a creación de usuarios. También dispone de herramientas gráficas para la creación de bases de datos y sus tablas.

- *MySQL Browser*: permite ejecutar consultas SQL sobre la base de datos así como editar los valores de manera sencilla.

4.3.3 Plugin

Como se comentó anteriormente, la aplicación de música que se selecciona es Winamp, la cual permite el desarrollo de *plugins*. El *plugin* se implementa en C++ ya que es éste el lenguaje en el que se sugieren desarrollar las extensiones de este programa.

Para el desarrollo de los *web services* se utiliza la herramienta *gSOAP* [22], la cual a partir de la definición del *web service* (archivo WSDL que publica el servidor) genera código C/C++ tanto para una aplicación cliente de esa definición como para una aplicación servidor. En este caso se utiliza para la generación de una aplicación cliente. Esta herramienta es también de libre acceso y uso.

Para obtener la información de las canciones del directorio compartido del usuario, se tiene en cuenta que los archivos MP3 cuentan con etiquetas que permiten guardar información de las canciones. Estas etiquetas contienen datos como el nombre de la canción, artista y género entre otros. Este formato de etiquetado se llama *ID3* [23], el cual consta de dos versiones ID3v1 e ID3v2. Se considera importante que el *plugin* pueda obtener la información de ambas versiones ya que los usuarios pueden tener sus archivos en cualquiera de los dos formatos, y se considera muy importante maximizar la obtención de información. Para esto se utiliza una librería llamada *id3lib* [24], la cual permite obtener la información de ambas versiones del etiquetado *ID3*.

Con respecto a la implementación del *streaming* de audio se utiliza la librería *live* [25], la cual tiene desarrollados en C++ varios protocolos de *streaming* de audio y video como son RTP y RTSP. Mediante esta librería se implementa el servidor de *streaming*, embebido en el *plugin*, utilizando el protocolo de comunicación RTSP. Para que Winamp pueda recibir *streaming* de audio debe utilizar una librería que le permita actuar como cliente del protocolo RTSP. Los desarrolladores de la librería *live* también proveen una librería dinámica (DLL) para que Winamp cumpla con el requisito de aceptar *streaming* a través de RTSP.

Como entorno de desarrollo para C++ se utiliza *DEV-C++* [26], el cual es de libre acceso. También se evalúa utilizar una extensión de eclipse que permite utilizar código C++, pero incluye menos funcionalidades que las brindadas por *DEV-C++*.

4.4 Implementación del sistema PGMúsica

A continuación se detallan las características más relevantes de los componentes del sistema, describiendo las decisiones de diseño e implementación tomadas en cada uno de ellos.

4.4.1 Componente Plugin

Este componente consiste de un *plugin* de la aplicación Winamp, el cual realiza varias tareas. Una de ellas es el envío al servidor de la información sobre la música que escucha el usuario así como los puntajes que el usuario asigna a cada canción usando la funcionalidad de ranqueo de Winamp. Cuando el usuario se conecta desde Winamp, el *plugin* es quien recolecta la información de las canciones del directorio compartido del usuario para enviarlas al servidor. Luego publica ese conjunto de canciones en su servidor de *streaming*.

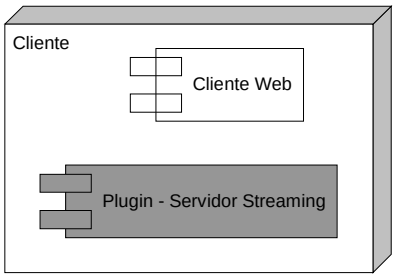

Figura 4.4: Plugin en el nodo cliente

Para implementar estos requerimientos, se debe lograr independencia de ejecución entre algunas de las funcionalidades. Esto ocasiona que el *plugin* requiera de varios hilos de ejecución que se encarguen de las diferentes tareas. Tanto el monitoreo de Winamp como el servidor de *streaming* se ejecutan en hilos separados del hilo principal del *plugin*, que es el encargado de procesar los eventos de Winamp y en función de ellos controlar la ejecución de los demás.

La estructura del *plugin* esta implementada alrededor de cuatro clases principales, las cuales se muestran en la Figura 4.5.

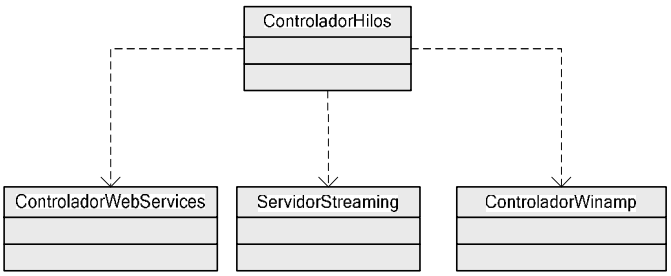

Figura 4.5: Estructura clases principales del *plugin*

El *ControladorHilos* es el encargado de resolver la lógica de creación y manejo de hilos de ejecución del *plugin*, encapsulando las llamadas a las funciones del sistema que permiten su manejo.

El *ControladorWebServices* encapsula las operaciones que resuelven la comunicación con el servidor de aplicaciones, ocultando la lógica de comunicación debido al uso de *web services*.

La clase *ServidorStreaming* es quien oficia de servidor de *streaming*, el cual permite publicar un conjunto de recursos (archivos MP3) para que sean accedidos a través del protocolo RTSP por otros clientes del sistema logrando una comunicación punto a punto.

El *ControladorWinamp* define todas las operaciones que se utilizan para comunicarse con la aplicación Winamp, ya sea para solicitar como para enviar información a la aplicación. Winamp define un conjunto de mensajes para solicitarle información así como para indicarle acciones a realizar. Esta clase se encarga de encapsular esta lógica, y declarar funciones que sean las que se vayan a utilizar desde el *plugin*, permitiendo la comunicación con la aplicación anfitriona.

Una limitación que presenta el *plugin* implementado con respecto al *streaming*, es que los usuarios solo pueden recibir audio de los directorios compartidos de otros usuarios que estén en la misma red LAN. Esto se debe a que el *plugin* envía al servidor de aplicaciones la información sobre la IP y puerto locales en la que publica los recursos a compartir, sin considerar a través de qué máquina se realiza la conexión a Internet.

Otra limitación que ocurre con el *plugin* es que no siempre el audio del *streaming* logra escucharse de manera fluida. En ciertas ocasiones el audio se escucha con cortes, debido presumiblemente a una falla en la librería utilizada para el *streaming*. Se contactó a quienes la implementaron, comentando éstos que el error no se encontraba en la librería sino que era un problema de Winamp. No se pudo determinar en dónde realmente se genera el error y es un punto a mejorar en caso de realizar trabajos futuros sobre este sistema.

4.4.2 Componente Web Services

Este componente se encarga de publicar los servicios implementados en el componente *GeneralServicios* que son utilizados por el componente *plugin*. Los *web services* deben, por razones de seguridad evidentes, validar usuario y contraseña en cada operación que se invoca desde el cliente. El componente *Web Services* se encarga de esta validación para poder invocar la funcionalidad requerida.

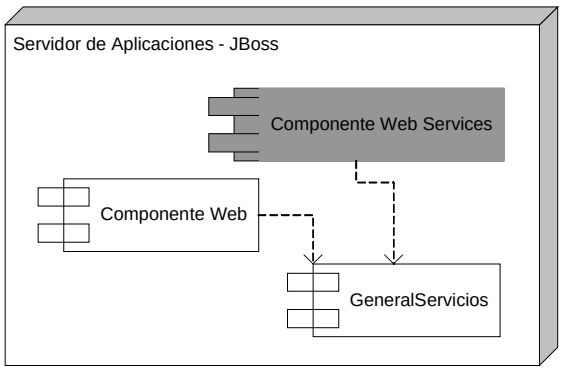

Figura 4.6: Componente Web Services dentro del nodo servidor

Las operaciones de *GeneralServicios* que el componente *Web Services* publica son:

- *Login*: recibe del cliente la dirección de su servidor de *streaming* (IP y puerto) junto con la lista de canciones a publicar. Esta operación marca en el sistema al usuario como conectado y establece las canciones compartidas como canciones disponibles para otros usuarios.

- *Desconectar:* marca al usuario como desconectado, y desmarca las canciones de su directorio compartido como disponibles a través de él.

- *Escucha canción*: envía al servidor la información de la canción que escucha el usuario.

- *Actualizar Puntaje:* actualiza el puntaje del usuario para una determinada canción.

- *Obtener Recomendación:* solicita una recomendación al servidor considerando las canciones disponibles en los directorios compartidos de los usuarios conectados. Retorna una lista de canciones recomendadas, con la información de las direcciones IP y puerto al que deben comunicarse para recibir el *streaming*.

Estas funcionalidades son las que se pueden utilizar si se quiere implementar otra aplicación cliente de recolección de información, sin necesidad de modificar la

lógica residente en el servidor de aplicaciones.

Analizando con mayor nivel de detalle encontramos dos clases en este subsistema que son *GeneralServiciosWrapper* y la interfaz *GeneralServiciosWS*. Para utilizar *web services* en JBoss, se debe definir una interfaz donde se declaran las operaciones que estarán disponibles a través de ellos. La interfaz GeneralServiciosWS es quien declara estas operaciones y GeneralServiciosWrapper es quien las implementa. GeneralServiciosWrapper es el mediador entre los servicios provistos por la lógica principal del sistema y el componente *Plugin*.

4.4.3 Componente Web

Este componente es el encargado de implementar la lógica del sitio web obteniendo la información del sistema (usuarios, recomendaciones, etc.) a través del componente *GeneralServicios*. De esta manera se logra que el componente *Web* sea independiente de la lógica principal del sistema, pudiendo ser reemplazado o eliminado en cualquier momento sin afectar los demás componentes.

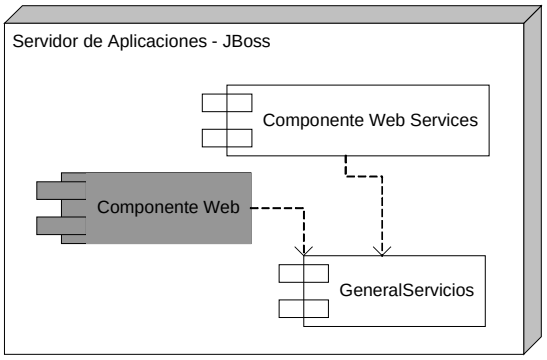

Figura 4.7: Componente Web dentro del nodo servidor

También administra su propia base de datos que es independiente de la del sistema de recomendación. Ella contiene toda la información de configuración del sitio. Se utiliza una base de datos separada de la base principal por la misma razón de diseño por la cual se implementa la lógica del sitio independiente de la lógica central del sistema.

A continuación se describen los módulos con los que cuenta este componente.

- *Módulo de Usuarios:* Este módulo contiene las interfaces referentes al ingreso y modificación de datos personales del usuario. Es quien se encarga de recolectar y desplegar la información que será enviada o recibida del componente *GeneralServicios.*

- *Módulo de Administradores del Sitio:* El sitio no administra los usuarios generales del sistema pero sí registra, a través de este módulo, usuarios administradores del sitio que son aquellos con privilegios sobre configuración.

- *Módulo de Menú:* El sitio contiene un menú de selección el cual es configurable en su contenido, no siendo en sus niveles ya que únicamente permite la existencia de hasta tres niveles desplegables. Este módulo se encarga de las altas, bajas y modificaciones de los ítems del menú. El usuario administrador es el encargado de configurar estos componentes. El sitio también permite que sea visualizado en dos idiomas, español e inglés, por lo que se podrán configurar los textos de los ítems del menú en ambos idiomas.

- *Módulo de Idioma:* Como se menciona anteriormente, el sitio permite su visualización en dos idiomas, por lo que sus textos deben ser desplegados en el idioma que haya indicado el usuario. El módulo de idioma se encarga de las altas, bajas y modificaciones de los textos por idioma que se

visualizan en el sitio.

- *Módulo de Páginas de Contenido:* El sitio, además de mostrar la información que es obtenida desde el componente *GeneralServicios*, también permite desplegar información estática que será configurada por los usuarios administradores. Esta información es mostrada en base a las páginas de contenido, que son básicamente contenidos html. Este módulo se encarga del mantenimiento de estas páginas.

- *Módulo de consultas:* En el módulo de consultas se encuentran las referencias a las consultas que son hechas a través de la comunicación con el componente *GeneralServicios*, en lo que concierne a las preferencias del usuario en cuanto a canciones, la información que se tenga de las mismas y sobre las recomendaciones.

4.4.4 Componente GeneralServicios

En este componente se implementa la lógica principal del sistema. Se divide en módulos que se detallan a continuación, los cuales se encargan entre otras cosas del acceso a la base de datos y la generación de las recomendaciones.

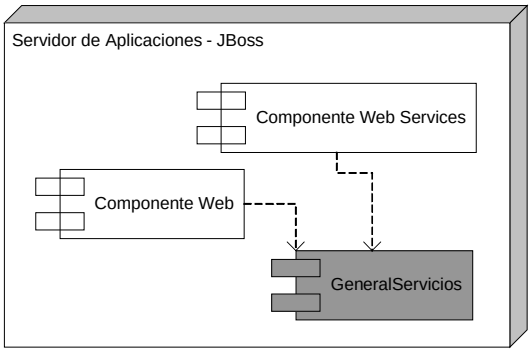

Figura 4.8: Componente GeneralServicios dentro del nodo Servidor

4.4.4.1 Servicios del Sistema

En este módulo, radica la lógica central del sistema y contiene el punto de entrada para los demás componentes.

Sus principales clases son las siguientes:

- *GeneralServicios:* GeneralServicios es la interfaz a través de la cual se acceden a todos los servicios implementados en el servidor de aplicaciones. Cada funcionalidad provista por esta interfaz pertenece a un caso de uso.

- *ServiciosWebLogic*: ServiciosWebLogic es la clase que implementa los métodos que se proveen al componente *Web* desde la interfaz GeneralServicios.

- *ServiciosPluginLogic*: ServiciosPluginLogic es la clase que implementa los métodos que se proveen al componente *Plugin* desde la interfaz GeneralServicios.

- *ConsultaRecomendacion*: Esta clase es la interfaz por la cual se accede a la obtención de una recomendación. Contiene solo un método al que se le pasa por parámetro el nombre de usuario y un indicativo de si la recomendación se solicita por el subsistema Interfaz *Web* o por el subsistema *Plugin*.

4.4.4.2 Algoritmo de Recomendación

El algoritmo de recomendación del sistema se encuentra encapsulado en este

módulo, pudiendo ser modificado por otro, sin afectar los demás subsistemas ni componentes.

La clase implementada para el algoritmo cumple la interfaz requerida por el motor de recomendación CoFE, que es la herramienta que se utiliza para realizar las pruebas que se describen en el capítulo 5. De esta manera, si en algún momento se desea cambiar el algoritmo, en caso de seguir respetando dicha interfaz, el nuevo algoritmo puede ser probado utilizando esta herramienta.

Para la implementación general del algoritmo se utilizan una serie de funciones. Una de ellas es la función que determina la similitud entre dos usuarios mediante el coeficiente de correlación de Pearson, que es usada por la función que calcula la predicción de un voto de un usuario para un determinado ítem. En base a esta función se obtendrán las recomendaciones que serán enviadas al usuario.

4.4.4.3 Acceso a datos

Este módulo está conformado por un conjunto de clases que proveen las funcionalidades necesarias para la lectura y guardado de datos en la base.

Esta dividido en las clases ABMUsuario, ABMCancion y ABMRecomendacion. El cometido de estas clases es realizar las altas, bajas y modificaciones en las tablas de Usuarios, Canciones y Recomendaciones respectivamente.

4.4.5 Base de Datos

En el sistema PGMúsica se cuenta con dos bases de datos residentes en el mismo servidor de BD:

- *pgmusica*: Es la base de datos principal del sistema donde se incluyen todos los datos de los usuarios, canciones, recomendaciones y puntajes.

- *sitiopgmusica*: Esta base de datos contiene solamente datos administrativos y de configuración del sitio Web. Se crea separada de la base principal ya que se entiende que los datos referentes al sitio *web* deben estar separados de los del sistema. De esta manera, si se cambia o elimina el sitio *web*, no genera ningún impacto sobre la base *pgmusica*.

El modelo entidad relación considerado para diseñar la base *pgmusica* se muestra en la Figura 4.9.

El sistema contiene un conjunto de usuarios de los cuales un subconjunto de ellos son usuarios que están conectados mediante el *plugin*. Por otro lado se cuenta con un conjunto de canciones, donde algunas son canciones que están disponibles. Esto último significa que pertenecen al directorio compartido de alguno de los usuarios conectados. Además se tiene un conjunto de recomendaciones, que son las realizadas a cada usuario, donde cada una tiene asociado un conjunto de canciones. Por último se cuenta con información acerca de la relación entre usuarios y canciones.

Es importante destacar que el diseño de esta base se realizó considerando que no debía tener en cuenta aspectos particulares del algoritmo. De hecho, la base se diseñó independientemente del algoritmo de recomendación. Pero al momento de plantearse ese objetivo surge el problema de que no se obtenía información

acerca de la recomendación para mostrarle al usuario. Por lo tanto, para guardar esa información relevante, se incluye un campo de texto en la entidad Recomendación, para que al momento de generar una recomendación, se pueda registrar una descripción para ser mostrada al usuario.

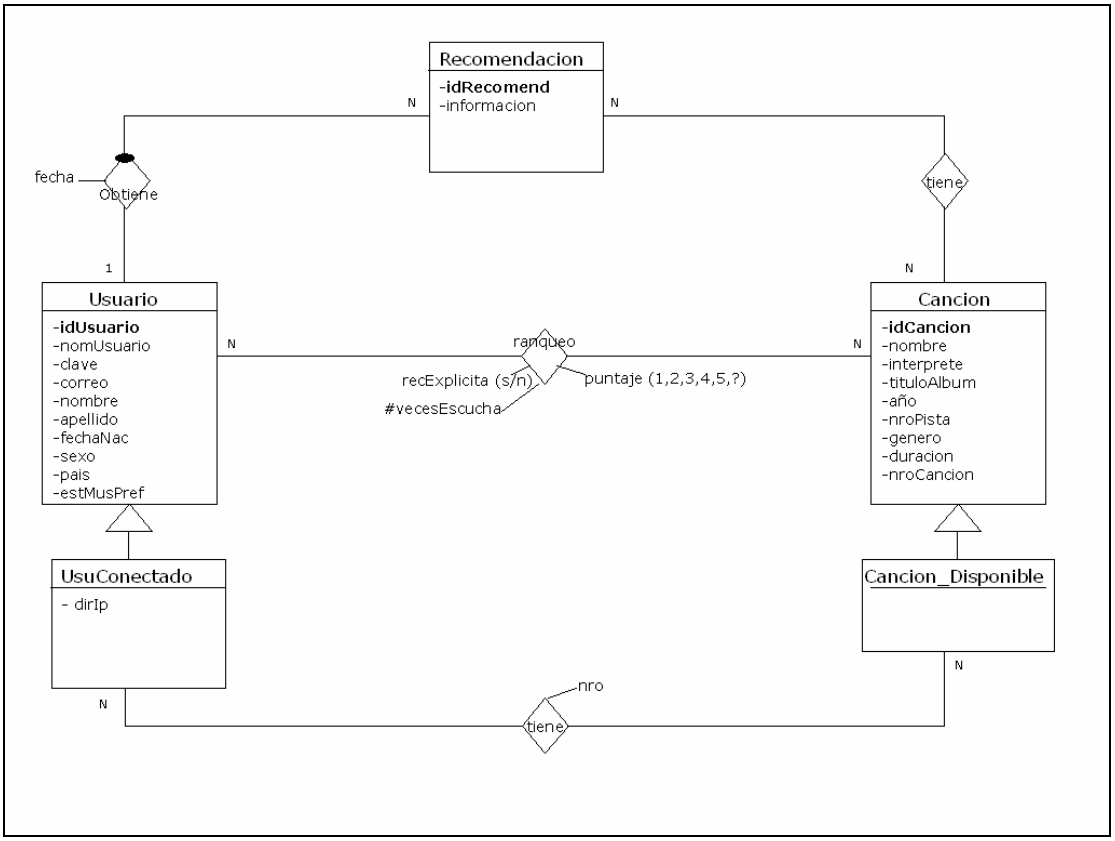

Figura 4.9: Modelo Relacional de la base de datos *pgmusica*

A partir de este modelo entidad-relación, se crearon las tablas de la base de datos. Por mayor detalle consultar el documento anexo de diseño de la base de datos PGMúsica.

Capítulo 5 Pruebas del Sistema

Uno de los aspectos fundamentales a la hora de analizar un sistema de recomendación, es medir la calidad de las recomendaciones que realiza. Para ello es necesario realizar pruebas del sistema y del algoritmo utilizado.

Dentro de las dificultades que poseen las aplicaciones que utilizan la técnica de filtrado colaborativo se encuentra la gran cantidad de datos confiables que se necesitan para poder realizar pruebas. Para recolectar esta información es preciso conocer las preferencias de una gran cantidad de usuarios sobre distintos elementos, y la única forma de obtener esa información es que los propios usuarios la proporcionen. Esto hace que sea muy difícil obtener una cantidad suficiente de información, dado que en general se requiere mucho esfuerzo, y son pocos los usuarios que están dispuestos a utilizar un sistema de recomendación en una etapa inicial donde en general no realizan buenas recomendaciones.

Debido a la dificultad y falta de tiempo para la recolección de un conjunto de datos importante, no se pudieron realizar pruebas con datos recolectados por PGMúsica. Realizar estas pruebas hubiese sido interesante ya que la calidad de las recomendaciones no depende únicamente del cálculo del algoritmo de recomendación sino que también depende de los datos de entrada que toma el algoritmo. La importancia se debe a que el puntaje de los usuarios sobre las canciones que considera el algoritmo de recomendación, no es solamente el puntaje que recolecta explícitamente el sistema, sino que también existe un puntaje implícito que toma el algoritmo, que afecta la exactitud de la recomendación.

Con respecto a las pruebas del algoritmo de recomendación, es necesario encontrar el valor óptimo de los parámetros que utiliza, para poder garantizar que el error que se comete es mínimo.

Es preciso observar que el cálculo de predicción de los algoritmos de filtrado colaborativo tiene la particularidad de ser independiente del tipo de ítem a recomendar. Esto se debe a que el cálculo se realiza únicamente en base a los puntajes de los usuarios sobre los elementos del sistema de forma independiente a cómo se originan. Por esta razón, estos algoritmos se pueden validar sobre cualquier conjunto de usuarios, ítems y preferencias, independientemente si los elementos a recomendar son canciones, películas u otros.

El grupo de investigación GropuLens de la universidad de Minnesota tiene disponibles en Internet dos bases de datos de referencia en el área con información de usuarios, películas y preferencias de los usuarios sobre las películas[5]. La primera consiste en 100.000 puntuaciones para 1682 películas por 943 usuarios y la segunda consiste de aproximadamente 1 millón de votos anónimos para 3900 películas por 6040 usuarios. El puntaje que se le da a las películas es de uno a cinco. Se decide utilizar los datos de una de estas bases para poder realizar las pruebas.

[5] Disponible en http://www.grouplens.org/data/

Por otro lado se utiliza un motor que realiza pruebas en forma automática sobre algoritmos de filtrado colaborativo, llamado CoFE (*Collaborative Filtering Engine*). Este motor fue desarrollado por el grupo de investigación de Sistemas de Información Inteligente de la Universidad de Oregon [27]. CoFE tiene como objetivos fomentar y facilitar la investigación de nuevos algoritmos de filtrado colaborativo, definiendo un entorno para realizar pruebas de forma muy sencilla. Está desarrollado en java, es gratuito y de código abierto.

Este motor permite variar los valores de los parámetros de entrada del algoritmo, y como resultado se obtienen los valores de las métricas más usadas para evaluar algoritmos de filtrado colaborativo, como son el *Coverage*, el MAE y el porcentaje de ítems bien clasificados. Para ello utiliza el mecanismo de validación cruzada con una cantidad de 5 particiones por defecto.

En base a estas herramientas se realizan las pruebas que se describen a continuación. Estas pruebas se realizan en dos partes: la primera es para verificar la implementación del algoritmo, y la segunda es sobre el algoritmo y sus parámetros.

5.1 Verificación de la implementación del algoritmo

Luego de implementado el algoritmo basado en memoria en el sistema PGMúsica, se verifica su implementación. Para ello se utiliza la herramienta CoFE.

El motor CoFE, además de permitir realizar pruebas sobre algoritmos de filtrado colaborativo, también permite obtener recomendaciones sobre la base de datos MovieLens, utilizando el algoritmo basado en memoria con el coeficiente de correlación de Pearson, al igual que en PGMúsica. Por lo tanto se utiliza esa faceta del motor CoFE para verificar el algoritmo implementado en PGMúsica.

El mecanismo que se sigue es el siguiente:

a) Se realiza la adaptación de la base de datos de películas, a la base de datos de canciones de PGMúsica, cargando los mismos en la base de datos PGMúsica.

   La transformación que se realiza es la siguiente:

MovieLens	PGMúsica
identificador de película	identificador de canción
identificador de usuario	identificador de usuario
Puntaje sobre película de 1 a 5	Puntaje sobre canción de 1 a 5

b) Luego se elige un usuario al azar y se solicita una recomendación para ese usuario en CoFE y en PGMúsica, dando el mismo resultado en los 2 casos. Se repite el procedimiento con una cantidad de aproximadamente 20 usuarios dando exactamente los mismos resultados.

Tomando como base el algoritmo implementado en CoFE, se verifica de esta forma el algoritmo implementado en PGMúsica.

5.2 Pruebas del algoritmo

El parámetro más importante a definir del algoritmo basado en memoria, implementado en PGMúsica, es la cantidad máxima de vecinos que se considera en el cálculo de predicción de un ítem. Por esta razón, el objetivo de las pruebas realizadas fue encontrar el valor de ese parámetro que minimice el error en las predicciones.

Para realizar las pruebas, se consideran los conjuntos de datos disponibles por el grupo de investigación GroupLens, recolectados con su sistema de recomendación de películas llamado MovieLens. De las 2 bases de datos disponibles, se utiliza únicamente la que contiene 100.000 puntuaciones, 1682 películas y 943 usuarios debido a que la herramienta CoFE no soporta la cantidad de datos contenida en la otra base de datos.

Las métricas que se utilizan para el estudio de los resultados obtenidos son el MAE (error medio absoluto), descrito en la sección 2.2, y el porcentaje de ítems bien clasificados.

$$\left|\overline{E}\right| = \frac{\sum_{i=1}^{N} \left|p_i - r_i\right|}{N}$$

Fórmula 5.1: Error Medio Absoluto

La fórmula para calcular el MAE se puede observar en la Fórmula 5.1, donde:

$|\overline{E}|$ = valor absoluto promedio del error

p_i = puntaje de predicción del algoritmo sobre el ítem *i*

r_i = puntaje real del usuario sobre el ítem *i*

N = total de ítems considerados para el cálculo de MAE.

Cantidad máxima de vecinos	MAE	Clasificados Correctamente (%)	Cantidad máxima de vecinos	MAE	Clasificados Correctamente (%)
1	0.8200	39.0	13	0.6469	47.9
2	0.7302	42.8	14	0.6465	47.9
3	0.6944	45.0	15	0.6463	48.0
4	0.6774	45.9	20	0.6468	47.9
5	0.6667	46.5	30	0.6494	47.9
6	0.6602	46.9	40	0.6524	47.6
7	0.6560	47.2	50	0.6548	47.4
8	0.6532	47.4	60	0.6570	47.1
9	0.6509	47.6	70	0.6587	47.0
10	0.6492	47.8	80	0.6601	46.9
11	0.6482	47.8	90	0.6613	46.8
12	0.6473	47.9	100	0.6623	46.7

Tabla 5.1: Resultados sobre los datos de MovieLens

En primer lugar se realizan pruebas del algoritmo implementado en PGMúsica sobre los datos MovieLens variando la cantidad máxima de vecinos. En la Tabla 5.1 se muestran los resultados obtenidos con la herramienta CoFE. En la Gráfica 5.1 se observa la relación entre la cantidad máxima de vecinos y el error medio

absoluto (MAE). Estas pruebas se realizan mediante validación cruzada con una cantidad de particiones igual a 5.

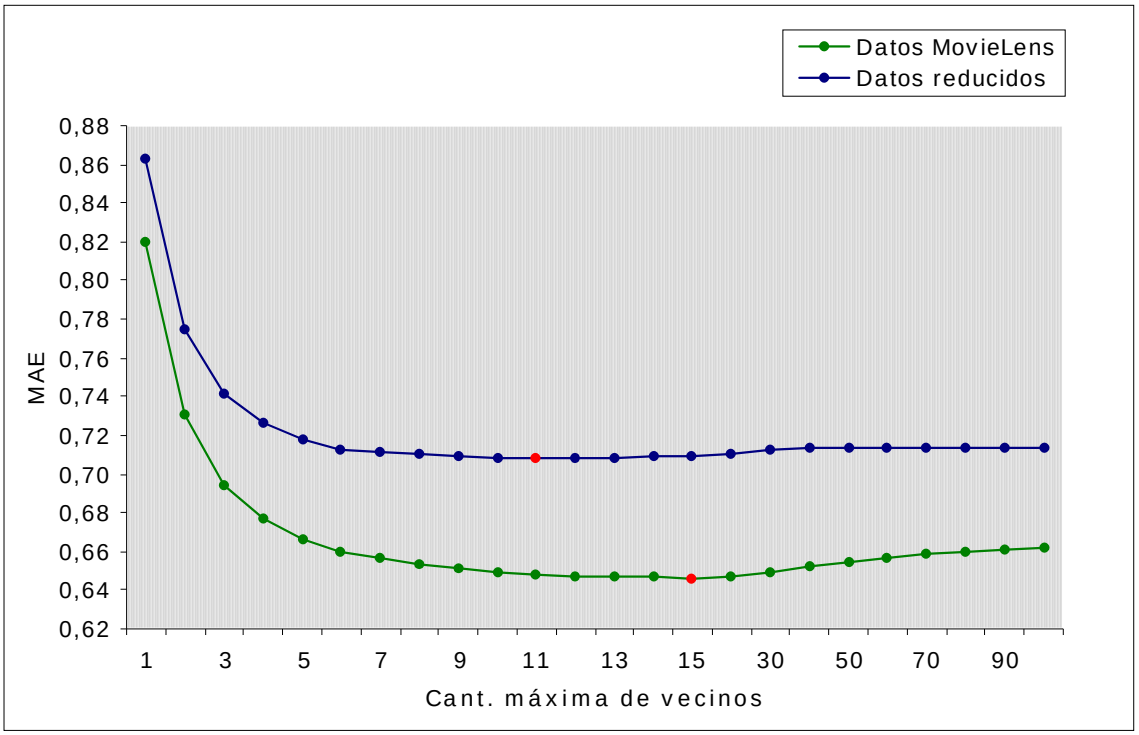

Gráfico 5.1: Gráfico del error para ambos conjuntos de datos

De la Gráfica 5.1, considerando los datos MovieLens, se puede inferir que la cantidad máxima de vecinos óptima es 15, ya que es el valor donde el error medio absoluto es mínimo y la cantidad de ítems clasificados correctamente es máxima (48%). Sin embargo, es previsible que este valor dependa de la cantidad de información contenida en la base de datos.

Observar el comportamiento del error en un conjunto de datos menor es una prueba que se evalúa como interesante, dado que la cantidad de datos en los sistemas de recomendación varía en el tiempo. Por esta razón se reduce la base de datos MovieLens para estudiar el comportamiento del error y analizar el valor más adecuado para el parámetro en cada situación.

En este sentido se analiza el conjunto de usuarios de MovieLens, encontrando que del total:

a) 61.7% (582 usuarios) puntuaron menos de 100 ítems
b) 32.6% (307 usuarios) puntuaron entre 100 y 300 ítems
c) 5.7% (54 usuarios) puntuaron más de 300 ítems.

Para reducir el conjunto de usuarios, manteniendo las características de la base de datos, se selecciona un conjunto de 200 usuarios en total, elegidos aleatoriamente dentro de cada franja, respetando la relación presentada anteriormente. Luego se filtra el conjunto de puntajes sobre ítems, seleccionando aquellos puntajes que fueron dados por los usuarios seleccionados. De esta forma se obtiene un conjunto de datos de 200 usuarios, 1682 películas y 19747 ratings sobre películas.

Con este conjunto de datos, “datos reducidos”, se realizan las mismas pruebas que con los datos de MovieLens: los resultados se muestran en la Tabla 5.2. Además se puede ver en el Gráfico 5.1 la relación entre la cantidad máxima de vecinos y el error medio absoluto cometido variando ese parámetro.

Cantidad máxima de vecinos	MAE	Clasificados Correctamente (%)	Cantidad máxima de vecinos	MAE	Clasificados Correctamente (%)
1	0.8631	37.7	13	0.7085	44.7
2	0.7752	40.8	14	0.7087	44.7
3	0.7412	42.9	15	0.7090	44.6
4	0.7259	43.5	20	0.7103	44.5
5	0.7179	44.1	30	0.7123	44.2
6	0.7129	44.6	40	0.7133	44.1
7	0.7112	44.6	50	0.7137	44.1
8	0.7100	44.7	60	0.7138	44.1
9	0.7090	44.7	70	0.7139	44.1
10	0.7083	44.7	80	0.7139	44.1
11	0.7081	44.8	90	0.7139	44.1
12	0.7082	44.7	100	0.7139	44.1

Tabla 5.2: Resultados sobre los datos reducidos

Del análisis realizado se concluye que independientemente del volumen de datos, al variar la cantidad máxima de vecinos se cumple un patrón de comportamiento sobre el error medio absoluto cometido en el cálculo de predicción.

El comportamiento que se observa es que en cualquier caso, si se considera una cantidad reducida de vecinos, el error que se comete es muy grande. Esto se debe a que se dejan de considerar vecinos que pueden tener alta similitud con el usuario activo, perdiendo información para el cálculo de predicción.

También se observa que teniendo en cuenta cantidades mayores de vecinos, el error desciende hasta un óptimo y luego comienza a incrementarse muy lentamente. Esto ocurre porque al considerar un número grande de vecinos, se está considerando aquellos con mayor similitud al usuario activo, que son quienes tienen mayor influencia en el cálculo de la recomendación, y además se consideran vecinos no tan similares. Al considerar una gran cantidad de vecinos no similares, si bien cada uno de ellos por separado tienen poca incidencia en el cálculo de predicción, la suma de todos termina afectando el resultado de forma negativa.

Además, se percibe que luego de cierto valor, el error se estabiliza. Esto se debe a que en determinado momento, por más que se incremente la cantidad máxima de vecinos, los datos disponibles no permiten generar un número de vecinos que alcance ese máximo. Por ejemplo, como se observa en la tabla 5.2 luego de los 70 vecinos el valor del error se estabiliza. Esto sucede porque no hay más de 70 vecinos por lo cual aunque se aumente el valor del parámetro se obtiene la misma cantidad de vecinos.

Asimismo se observa en ambas pruebas que la diferencia del error entre el valor óptimo y un valor mayor de vecinos, por ejemplo cien, no es significativa. Pero se debe tener en cuenta que considerando menos vecinos el algoritmo tiene una mejor performance. Por lo tanto, se intenta elegir el menor valor para este parámetro cometiendo el menor error posible.

Para una etapa inicial del sistema PGMúsica, asumiendo pocos usuarios registrados, se decide elegir una cantidad máxima de once vecinos. Según el análisis realizado, es de esperar que el error provocado por la elección de este valor no tenga una diferencia significativa con el valor óptimo variando la cantidad de usuarios del sistema. De todas formas, a medida que aumenta la cantidad de usuarios es necesario ajustar este parámetro para lograr mayor exactitud en las predicciones.

Capítulo 6 Conclusiones

El trabajo realizado en este proyecto se enfocó en el estudio de los sistemas de recomendación analizando particularmente la técnica de filtrado colaborativo. En ese sentido, se analizaron los algoritmos más usados y con los que se han obtenido mejores resultados, así como las métricas más utilizadas para medir su efectividad. Se evaluaron las problemáticas asociadas a este tipo de sistemas y algunas soluciones que han sido planteadas. También se analizaron algunos sistemas de referencia en el área, en particular de recomendación de música. Por otro lado se estudió la técnica de *streaming* así como el diseño de una arquitectura *Peer-to-Peer*.

Como principal resultado del trabajo se obtuvo el diseño e implementación de un sistema de recomendación, denominado PGMúsica, el cual utiliza la técnica de filtrado colaborativo aplicado al dominio de la música. PGMúsica consta de un servidor central encargado de generar recomendaciones y de un cliente (*plugin*) de la aplicación Winamp el cual recolecta la información que se envía al servidor. Además, el *plugin* se comunica con los demás clientes del sistema para enviar y recibir canciones mediante *streaming*, determinando entre ellos una arquitectura *Peer-to-Peer*. Por otro lado, se construyó un sitio web como vía alternativa de ingreso y visualización de información de interés del usuario.

El algoritmo de filtrado colaborativo implementado es basado en memoria; para el cálculo de similitud entre usuarios se utilizó el coeficiente de correlación de Pearson. Si bien este algoritmo produce buenos resultados cuando es aplicado sobre grandes cantidades de información, tiene en la performance su principal limitación.

El parámetro más importante a definir del algoritmo implementado es la cantidad máxima de vecinos que se considera en el cálculo de predicción de un ítem. Por esta razón, se realizaron pruebas que permitieron comprobar la existencia de un patrón de comportamiento del error medio absoluto en función de la cantidad máxima de vecinos, lo que permitió definir un valor adecuado para el parámetro. Por otra parte, debido a la dificultad en la recolección de un conjunto de datos considerable, no se pudieron realizar pruebas sobre el cálculo de puntajes a partir de información implícita, el cual afecta la calidad de las recomendaciones.

En relación a los aspectos que merecen ser considerados en el diseño de la solución, se realizó especial hincapié en el problema del *Cold Start* así como en el de la recolección de preferencias implícitas, generando soluciones aplicadas al dominio de la música. Para el problema del *Cold Start* se aplica un enfoque basado en contenido (en función del estilo musical de las canciones) considerando además algunos datos personales del usuario para poder vincularlo con otros usuarios del sistema. Con respecto a la recolección de preferencias implícitas se estudia el comportamiento del usuario a partir de la cantidad de veces que escucha canciones.

Con respecto al diseño se logra por un lado independencia entre la implementación del algoritmo y el resto de la aplicación, permitiendo que sea reemplazado fácilmente. De esta manera se ofrece la posibilidad de utilizar el sistema para comparar la performance y correctitud de distintos algoritmos de

filtrado colaborativo. Por otra parte, debido a la utilización de *Web Services* para la comunicación cliente-servidor, se permite la creación de otros clientes para la recolección de información en distintos lenguajes.

En resumen, se realizó un estudio de la técnica de filtrado colaborativo aplicándola a la resolución de la recomendación de música, así como el estudio de la técnica de *streaming* en una arquitectura *Peer-to-Peer*. De esta manera se logró la implementación un sistema de recomendación de música basado en filtrado colaborativo, con un diseño flexible, permitiendo de esta forma su utilización en trabajos futuros. En base a los objetivos planteados al comienzo de este proyecto y realizando un análisis general del trabajo realizado, se puede concluir que éstos han sido alcanzados.

6.1 Mejoras y Trabajos Futuros

A lo largo del proyecto se identificaron algunas mejoras y trabajos que se podrían realizar en un futuro. A continuación se detallan los considerados más relevantes.

Obteniendo datos reales a través del sistema PGMúsica se podrían realizar pruebas con el objetivo de medir la precisión de las recomendaciones que realiza. Además se podrían evaluar variantes sobre el algoritmo de recomendación así como la implementación de otros algoritmos.

En PGMúsica las canciones que recomienda el sistema son archivos mp3 que se identifican mediante el etiquetado ID3. Estas etiquetas pueden contener errores o falta de información afectando el reconocimiento por parte del sistema, de las canciones que describen. Esto ocasiona un problema en el dominio de la aplicación afectando la calidad del sistema, como se plantea en la sección 3.1 en el ejemplo de la canción “Adagio a mi país”. Un trabajo interesante a realizar en este sentido es el estudio de técnicas para establecer criterios que permitan reconocer de manera más precisa, cuándo dos etiquetas corresponden a la identificación de la misma canción.

Una posible mejora al sistema, en relación al *streaming,* sería evaluar alternativas para lograr que el envío de música pueda realizarse por parte de usuarios conectados entre sí a través de Internet, y no solamente a través de una red LAN. Queda también por resolver los cortes que sufre en ocasiones el envío de audio. Para ello se podría probar solucionar este problema reemplazando la librería (*live*) utilizada.

Un trabajo interesante que se plantea en los sistemas de recomendación es combinar la técnica de filtrado colaborativo con la basada en contenido de forma de obtener los beneficios de cada una. En este sentido se podría agregar análisis de contenido de los elementos del sistema de forma de poder mejorar las recomendaciones así como también asistir en la etapa de *Cold Start* del usuario. Para ello, en particular se podría utilizar la ontología de música Musicmoz[6]: nuclea información acerca de grupos y estilos musicales.

Como Winamp es una aplicación que funciona bajo la plataforma Windows, sería interesante estudiar la forma de adaptar el *plugin* desarrollado específicamente para la aplicación Winamp a otras aplicaciones, como por ejemplo XMMS. De esta forma funcionaría el sistema PGMúsica sobre las plataformas Mac OC X, Linux, Unix, BSD.

[6] http://musicmoz.org/

Como se ha mencionado reiteradamente en este documento, PGMúsica recomienda elementos que el usuario no conoce. La música es un elemento particular, debido a que no solo interesa conocer una determinada canción, sino que a veces es atractivo escucharla en determinado momento del tiempo. Se plantea la posibilidad de extender el sistema para que recomiende canciones conocidas por el usuario sabiendo que son de su agrado además de las no conocidas por él realizando el cálculo de predicción y seleccionando aquellas que se asume serán de su agrado. En consecuencia se le recomendaría música de su gusto sin importar si es conocida o no por él. De esta forma se podría lograr una “radio” personalizada para cada usuario, teniendo en cuenta únicamente sus preferencias, como lo hace Last.fm.

Referencias

[1] A.K. Jain, R.C. Dubes. *Algorithms for Clustering Data.* Prentice Hall, ISBN:0-13-022278-X, 1988.

[2] Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, David M. Pennock. *Methods and Metrics for Cold-Start Recommendations,* Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2002),pages 253-260, Agosto 2002.

[3] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. *Using Collaborative Filtering to Weave an Information Tapestry*. Communications of the ACM, 35(12):61-70, Diciembre 1992.

[4] Greg Linden, Brent Smith, Jeremy York. *Amazon.com Recommendation: Ítem-to-Ítem Collaborative Filtering,* IEEE Internet Computing, 7(1):76-80, 2003.

[5] John S. Breese, David Heckerman, Carl Kadie. *Empirical Analysis of Predictive Algorithms for Collaborative Filtering*, Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, pages 43-52, Julio 1998.

[6] John Canny. *Collaborative Filtering with Privacy via factor analysis*, Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 238-245, Agosto 2002

[7] Jonathan Lee Herlocker. *Understanding and Improving Automated Collaborative Filtering Systems*, Thesis submitted to the Faculty of the Graduate School of the University of Minnesota, http://web.engr.oregonstate.edu/~herlock/papers/thesis.pdf, Setiembre 2000.

[8] Jonathan L. Herlocker, Joseph A. Konstan, and John Ried. *Explaining Collaborative Filtering Recommendations*, Proceedings Computer Supported Cooperative Work (CSCW '00), pages 241-250, 2000.

[9] M. O'Conner, J. Herlocker. *Clustering Ítems for Collaborative Filtering,* Proceedings of the ACM SIGIR Workshop on Recommender Systems, 1999.

[10] Matei Ripeanu. *Peer-to-Peer Architecture Case Study: Gnutella Network*, Technical report, University of Chicago, 2001.

[11] Paul Resnick and Hal R. Varian. *Recomender Systems.* Communications of the ACM, 40(3):56-58, Marzo 1997.

[12] Paolo Massa, Paolo Avesani. *Trust-aware Collaborative Filtering for Recommender Systems*, Proceedings of the International Conference on Cooperative Information Systems (CoopIS'04), pages 492-508, Octubre 2004.

[13] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, John Riedl. *Grouplens: an open architecture for collaborative filtering of netnews,* Proceedings of ACM, 35(12):80-81, 1994

[14] Upendra Shardanand. *Social Information Filtering for Music*

Recommendation, MIT EECS M. Eng. thesis, Technical report, Learning and Common Sense Group, MIT Media Laboratory, 1994.

[15] Upendra Shardanand, Pattie Maes. *Social Information Filtering: Algorithms for Automating 'Word of Mouth'",* Proceedings of CHI'95 Conference on Human Factors in Computing Systems, ACM Press 1:210-217, 1995.

[16] Sitio oficial de Jboss. http://www.jboss.org.
Última consulta:15 de marzo de 2006.

[17] Sitio oficial de Tomcat. http://tomcat.apache.org.
Última consulta: 15 de marzo de 2006.

[18] Sitio oficial de Eclipse. http://www.eclipse.org
Última consulta: 15 de marzo de 2006.

[19] Sitio oficial de Genexus. http://www.genexus.com
Última consulta: 15 de marzo de 2006.

[20] Sitio oficial de MySQL. http://www.mysql.com
Última consulta: 15 de marzo de 2006.

[21] Sitio oficial de Winamp. http://www.winamp.com
Última consulta: 15 de marzo de 2006.

[22] Sitio de la herramienta gSoap. http://www.cs.fsu.edu/~engelen/soap.html
Última consulta: 15 de marzo de 2006.

[23] Sitio dedicado al estándar ID3. http://www.id3.org
Última consulta: 15 de marzo de 2006.

[24] Sitio de la librería id3lib. http://id3lib.sourceforge.net
Última consulta: 15 de marzo de 2006.

[25] Sitio de la librería live. http://www.live555.com/liveMedia/
Última consulta: 15 de marzo de 2006.

[26] Sitio dedicado al entorno de desarrollo DevC++.
http://www.bloodshed.net/devcpp.html
Última consulta: 15 de marzo de 2006.

[27] Página herramienta CoFE. Sitio Universidad de Oregon.
http://eecs.oregonstate.edu/iis/CoFE/
Última consulta: 15 de marzo de 2006

[28] Página del sistema de recomendación MovieLens.
http://movielens.umn.edu/login
Última consulta: 12 de diciembre de 2006

[29] Sitio técnico del plugin audioscrobbler. http://www.audioscrobbler.net/
Última consulta: 30 de marzo de 2006.

[30] Sitio del sistema Last.fm. http://www.last.fm/
Última consulta: 30 de marzo de 2006.

[31] Sitio del sistema de recomendación Pandora. http://www.pandora.com/
Última consulta: 30 de marzo de 2006.

Glosario

CORBA (*Common Object Request Broker Architecture*): estándar que establece una plataforma de desarrollo de sistemas distribuidos facilitando la invocación de métodos remotos bajo un paradigma orientado a objetos, permitiendo la comunicación entre distintos lenguajes de programación.

HTTP *(HyperText Transfer Protocol)*: protocolo de transferencia de hipertexto el cual se usa en transacciones Web. El hipertexto es el contenido de las páginas web, y el protocolo de transferencia es el sistema mediante el cual se envían las peticiones de acceso a una página.

Ítem: objeto que es evaluado por el usuario, el cual podrá ser recomendado.

J2EE (*Java 2 Enterprise Edition*): edición empresarial del paquete Java creada y distribuida por Sun Microsystems. Comprenden un conjunto de especificaciones y funcionalidades orientadas al desarrollo de aplicaciones empresariales.

JNI (Java Native Interface): interfaz de programación estándar para definir métodos nativos java y embeber la máquina virtual java en aplicaciones nativas.

LAN *(Local Area Network):* red informática de varios computadores y periféricos. Su extensión esta limitada físicamente a un edificio o a un entorno de unos pocos kilómetros. Su aplicación más extendida es la interconexión de computadores personales y estaciones de trabajo en oficinas, fábricas, etc; para compartir recursos e intercambiar datos y aplicaciones.

Peer-to-peer: red informática entre iguales (en inglés *Peer-to-Peer,* más conocida como P2P) se refiere a una red que no tiene clientes y servidores fijos, sino que una serie de nodos que se comportan como clientes y servidores a la vez.

Plugin (o Plug-in): tipo particular de extensión de una aplicación que interactúa con la aplicación principal, a través de una interfaz definida, sin afectar las funcionalidades ya existentes.

Puntuación: Evaluación que realizan los usuarios sobre un determinado ítem.

Rating (o Puntaje): Evaluación que realiza un usuario sobre un ítem

RTP (*Real Time Transport Protocol*): protocolo de transmisión de audio o video en tiempo real.

RTSP (*Real Time Streaming Protocol*): protocolo de transmisión de audio o video en tiempo real que agrega el control del flujo de información por parte del usuario, ya que tiene mensajes para controlar la reproducción (Play, Stop, etc.).

Streaming: tecnología que permite a un usuario ver y oír contenido de audio y video mientras es transmitido. Permite que variados contenidos multimedia puedan empezar a ser reproducidos segundos después de establecer la conexión, sin importar el tamaño total del archivo.

TCP (*Transmission Control Protocol*): es uno de los protocolos de comunicación fundamentales en Internet que garantiza que los datos serán entregados en su destino sin errores y en el mismo orden en que se transmitieron.

UDP (*User Datagram Protocol*): protocolo de comunicación que permite el envío de datagramas a través de una red sin que se haya establecido previamente una conexión, ya que el propio datagrama incorpora suficiente información de direccionamiento en su cabecera.

Usenet: Es un servicio por el que se accede vía Internet, donde los usuarios pueden leer o enviar mensajes a diversos grupos de noticias.

Usuario: Persona que hace uso del sistema.

Usuario activo: Persona que utiliza el sistema, a la cual se le esta definiendo su perfil para hacerle una recomendación.

Vecino: usuario que tiene puntuados ítems en común con el usuario activo.

Vecindario: conjunto de vecinos.

Web services: colección de protocolos y estándares que sirve para intercambiar datos entre aplicaciones. Distintas aplicaciones de *software* desarrolladas en lenguajes de programación diferentes y ejecutadas sobre cualquier plataforma pueden utilizar los *web services* para intercambiar datos.

WSDL (*Web Services Description Language*): formato XML que se utiliza para describir la interfaz pública de los web services. Describe la forma de comunicación, es decir, los requisitos del protocolo y los formatos de los mensajes necesarios para interactuar con los servicios listados en su catálogo.

XML(*Extensible Markup Language*): es una especificación que define una forma estándar de agregar marcas a un documento, es en definitiva un lenguaje para documentos que contienen información estructurada.