Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Implantación y evaluación de un recomendador social de canciones

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Carlos Javier Martínez Pedrón

Tutor: Vicente Javier Julián Inglada

2014/2015

Resumen

Los sistemas de recomendación están presentes cada vez más en diversos sitios web o aplicaciones, para ayudar a los usuarios a elegir los productos que desean. En este trabajo diseñamos, implementamos y evaluamos un sistema de recomendación social de canciones. Este se basa en la idea de que los amigos tienen mayor poder de persuasión que los usuarios desconocidos. Además diseñamos e implementamos otros recomendadores para compararlos con este, como son el basado en ítem, el basado en usuario y el híbrido, que mezcla las técnicas del recomendador social con las del recomendador basado en usuario. Por último, comparamos los recomendadores utilizando diversas métricas.

Palabras clave: sistemas de recomendación, recomendación social, recomendación de música.

Abstract

Recommender systems are increasingly present in various websites or applications to help users to select the products they want. In this paper we design, implement and evaluate a musical social recommender. This is based on the idea that friends have greater power of persuasion than unknown users. In addition, we design and implement other recommenders to compare with this, such as the item based, the user based and the hybrid, which mixes social recommender and user based recommender techniques. Finally, we compare the recommenders using different metrics.

Keywords: Recommender systems, social recommendation, music recommendations.

Tabla de contenidos

1. Introducción

En la actualidad, gracias a los avances tecnológicos, que permiten a los usuarios estar conectados a Internet casi en cualquier momento, se han ido creando diferentes tipos de sitios web o aplicaciones que permiten a los usuarios adquirir productos o relacionarse. En estos ámbitos, con la utilización de los algoritmos adecuados, se puede encontrar mucha información acerca de los usuarios y esta puede ser explotada por las empresas para obtener beneficios y satisfacer los gustos de los usuarios.

Este proyecto se centra en el análisis de una red social de música, en la que los usuarios interaccionan entre ellos y eligen la música que desean escuchar. En muchas ocasiones los usuarios no saben que música elegir, en ese momento es cuando entran los sistemas de recomendación, que ofrecen una lista con un número determinado de canciones, que probablemente puedan gustarle al usuario. Así se consigue ayudar a los usuarios a encontrar nuevas canciones que puedan ser de su gusto y gracias a ello, la red social puede promocionar canciones que no son tan famosas como otras.

Los sistemas de recomendación tratan de recopilar información sobre las canciones, usuarios y la interacción entre estos para poder obtener unos buenos resultados. Por ello, para el buen funcionamiento de estos, es necesario que la red social esté en uso y se guarde toda la información sobre las acciones que los usuarios realizan, para que los algoritmos de los sistemas de recomendación la utilicen de forma adecuada y puedan sacar buenos resultados, ya que así se pueden conocer los gustos de cada usuario y utilizarlos posteriormente para realizar las recomendaciones.

Existen diferentes tipos de sistemas de recomendación según la información que se utiliza durante el proceso de recomendación. En este proyecto se crearán cuatro recomendadores diferentes: el primero, se centrará únicamente en las canciones, basándose en los valores que las identifican – *content-based filtering* –; el segundo, utilizará la similitud entre usuarios para escoger de estos las canciones más similares – *colaborative filtering* – ; el tercero se basará en la interacción entre los usuarios para establecer el grado de amistad que tienen entre sí y posteriormente obtener de ellos las canciones que más les gusten; el cuarto y último, será una combinación de los dos anteriores para observar si al mezclar las técnicas se obtienen mejores resultados.

La idea principal de partida es que un sistema de recomendación que se basa en la amistad entre los usuarios dará mejores resultados por las siguientes razones: al mostrar las canciones de los usuarios a los que un usuario dado conoce, se considera que estos tienen mayor capacidad de persuasión, ya que no es lo mismo que te recomiende una canción un usuario al que conoces y puedes tener gustos en común que, que te la recomiende un usuario cualquiera; si existe mucha interacción entre dos usuarios, lo más probable es que tengan gustos similares y hablen de la música que escuchan, los grupos que les gustan…, por lo que seguramente en sus playlists contendrán canciones que le puedan gustar al otro usuario.

2. Objetivos

El objetivo principal del proyecto es desarrollar un recomendador de canciones basado en la interacción social o grado de amistad entre los usuarios de una red social y evaluarlo comparando con otros recomendadores como los basados en contenido o filtrado colaborativo. Este objetivo principal se divide en una serie de objetivos secundarios a realizar:

- Estudio de la base de datos de la red social proporcionada, desarrollada en mysql, para obtener los datos que nos puedan resultar útiles para implementar los sistemas de recomendación y descartar aquellos que no muestren información relevante para estos sistemas.

- Elegir un lenguaje de programación que permita trabajar de forma ágil con mysql, ya que la base de datos proporcionada sobre la que se evaluaran los resultados está diseñada en este lenguaje, y nos permita realizar operaciones de forma sencilla.

- Definir e implementar una función de similitud que permita determinar el parecido existente entre dos canciones o entre una canción y el perfil de un usuario.

- Diseño e implementación de un recomendador basado en ítem, que utiliza la información correspondiente a las canciones y al usuario dado, que permita devolver el número de canciones que se desee, introduciendo dicho número como parámetro.

- Diseño e implementación de un recomendador de filtrado colaborativo, que encuentre un número de usuarios similares pasado como parámetro y escoja de estos las canciones que más le puedan gustar al usuario.

- Diseño e implementación de un recomendador basado en la interacción social que devuelva un número de canciones que puedan gustarle al usuario, obtenidas de los usuarios con los que se considera que tiene una amistad.

- Diseño e implementación de un recomendador híbrido que combine el recomendador basado en el usuario y el basado en la interacción social, para comprobar si obtiene mejores resultados que los anteriores.

- Realizar un conjunto de pruebas que permitan su evaluación:

    - Ejecutar los recomendadores con diferentes parámetros para encontrar cuales permiten obtener unos mejores resultados. La variación tendría que ver con el número de usuarios utilizados en el proceso de recomendación y con el número de canciones devuelto.

    - Realizar un estudio sobre cuántos usuarios debe tener la base de datos sobre la que se realizan las recomendaciones para observar el comportamiento del recomendador y ver así el número de usuarios necesario para que los resultados de la evaluación sean óptimos.

- Evaluar los recomendadores devolviendo un diferente número de canciones y utilizar métricas como la precisión, recall y F1 Score para observar cuál o cuáles tienen mayor eficacia.

3. Estado del arte

3.1 Sistemas de recomendación

Los sistemas de recomendación son herramientas software, que utilizan técnicas, que permiten ofrecer ítems de gran interés a los usuarios. “Item” es un término general que se utiliza para denotar qué es lo que el sistema recomienda al usuario. Normalmente un sistema de recomendación se centra en un único tipo específico de ítem, como podrían ser imágenes, videos, libros, canciones… [1]

Una definición formal para los sistemas recomendadores podría ser: "Se trata de aquel sistema que tiene como principal tarea seleccionar ciertos objetos de acuerdo a los requerimientos del usuario"[2].

Otra posible definición formal sería: "Un Sistema de Recomendación, trata de aquel sistema que utiliza las opiniones de los usuarios de una comunidad para ayudar a usuarios de esa comunidad a encontrar contenidos de su gusto entre un conjunto sobrecargado de posibles elecciones" [2].

Los sistemas recomendadores surgen como respuesta a la sobrecarga de información disponible en numerosos sitios, que dificulta a los usuarios a encontrar los productos relevantes para ellos. Estos sistemas ofrecen sugerencias personalizadas, seleccionando entre grandes cantidades de productos aquellos que puedan gustarle al usuario, dependiendo de sus preferencias. Para ello se emplean diferentes estrategias que se basan en la información que recopilan en perfiles personales [3].

Los recomendadores ofrecen una lista ordenada, cuyo primer ítem será el que más pueda gustarle a un usuario dado. Al utilizar este ranking, los sistemas de recomendación tratan de predecir cuales son los ítems que más puedan interesarle al usuario, dependiendo de las características y preferencias de este. Para que esto sea posible, estos sistemas recopilan toda la información posible del usuario, para saber cuáles son las preferencias de este y así poder realizar unas recomendaciones exitosas.

Estos sistemas han ido cobrando cada vez más importancia, y actualmente los podemos encontrar en muchas páginas de Internet, como pueden ser páginas de ventas de productos de cualquier tipo o en los diferentes tipos de redes sociales. Algunos motivos por los que los sistemas de recomendación pueden considerarse importantes para algunas empresas serian los siguientes [1]:

- Aumentan el número de artículos vendidos. Esta sería la razón más importante ya que de ello depende el beneficio que puede obtener una empresa. Cuando tú eliges un producto en una página web, en la mayoría de páginas te ofrecen otros productos similares o complementos que pueden resultarte útiles.

- Vender una mayor variedad de artículos. Normalmente, en páginas para ver películas por ejemplo, la gente accede a las más populares y gracias a los sistemas de recomendación, se consiguen vender aquellas que no son tan populares.

- Aumentar la satisfacción del usuario. Si las recomendaciones que ofrece son satisfactorias para el usuario, este disfrutará de ello y podrá encontrar productos de su gusto.

- Ayudan a entender lo que el usuario quiere. Al recopilar información sobre los usuarios se consigue entender qué tipo de productos buscan, por lo que resulta más factible realizar buenas recomendaciones.

3.1.1 Tipos de recomendaciones

Los sistemas de recomendación pueden clasificarse en diferentes tipos dependiendo de las técnicas utilizadas para realizar sus recomendaciones. Por ello podemos clasificarlos en los siguientes:

- Sistemas de recomendación basados en contenido.

- Sistemas de recomendación colaborativos.

- Sistemas de recomendación basados en conocimiento.

- Sistemas de recomendación demográficos.

- Sistemas de recomendación basados en utilidad.

- Sistemas de recomendación híbridos.

3.1.1.1 Sistemas de recomendación basados en contenido

Un sistema de recomendación basado en contenido es aquel que se basa en un perfil creado a partir del análisis del contenido de los objetos que el mismo usuario ha comprado, utilizado o visitado en algún momento [4].

En otras palabras, los sistemas de recomendación basados en contenido extraen la información de los objetos no conocidos por el usuario y la compararan con la información del perfil del mismo para predecir sus preferencias sobre esos objetos. Lo que pretenden estos sistemas es recomendar objetos similares a los que le gustan al usuario.

3.1.1.2 Sistemas de recomendación colaborativos

Los sistemas de recomendación basados en un filtrado colaborativo son aquellos en los que las recomendaciones se centran en la similitud entre usuarios. Es decir, los sistemas colaborativos recomiendan ítems que les gustan a otros usuarios de intereses parecidos [4].

Estos sistemas filtran la información de forma similar a como lo hacemos nosotros cuando queremos adquirir o ver un producto nuevo, preguntamos a otras personas que lo hayan adquirido y en base a su opinión hacemos una cosa u otra.

Con el paso del tiempo, el uso de estos sistemas se ha ido popularizando y con ello se han descubierto una serie de problemas como son [5]:

- Escasez de datos. Los sistemas de recomendación colaborativos necesitan una gran cantidad de datos. Estos hacen referencia a la puntuación de los usuarios sobre los diferentes objetos disponibles y son utilizados para calcular sus gustos y el grupo de usuarios similares. Esta información resulta necesaria para poder realizar unas recomendaciones adecuadas, ya que cuanta más información haya disponible será más fácil obtener el grupo de usuarios similares.

- Escalabilidad. Estos sistemas utilizan generalmente algoritmos de cálculo de los k vecinos más cercanos para obtener la similitud entre usuarios. Estos algoritmos son costosos computacionalmente y su coste crece de forma lineal en base al número de usuarios e ítems. Por ello, cuando tenemos bases de datos de tamaños muy grandes se obtendrán problemas de escalabilidad.

- Problema del ítem nuevo. En estos sistemas los ítems nuevos, al no tener muchas puntuaciones o ninguna, no van a ser recomendados prácticamente nunca. De igual forma, los usuarios nuevos obtendrán unas predicciones muy pobres, ya que no han realizado muchas puntuaciones sobre los ítems y resulta complejo obtener sus usuarios más similares. Por estos motivos podemos decir que para un correcto funcionamiento de estos sistemas de recomendación se requiere cierto tiempo, en el que los usuarios interaccionen con los ítems, para poder realizar recomendaciones y predicciones ciertamente relevantes y acertadas.

3.1.1.3 Sistemas de recomendación basados en conocimiento

Los sistemas de recomendación basados en conocimiento realizan recomendaciones partiendo del conocimiento que da el usuario sobre sus necesidades, y del conocimiento de los productos a recomendar, buscando los que mejor se adapten a las necesidades del usuario [6].

Aunque los sistemas de recomendación más utilizados y más conocidos son los colaborativos y los basados en contenido, no siempre son los adecuados. Por ejemplo, los sistemas de recomendación colaborativos necesitan partir de una serie de evaluaciones de los usuarios sobre los ítems ofrecidos para poder realizar recomendaciones precisas y acertadas a cualquiera de estos usuarios. Los basados en contenido buscan productos a recomendar basándose en los valorados por el usuario en el pasado. Por tanto, estos sistemas requieren que el usuario haya evaluado un número mínimo de productos para realizar unas recomendaciones adecuadas.

En la realidad nos encontramos con este tipo de situaciones, en las que los sistemas anteriores no se pueden aplicar por problemas de datos o de tiempo. En estas situaciones podemos utilizar los recomendadores basados en conocimiento, que utilizan el conocimiento que proporciona el usuario sobre sus necesidades y el conocimiento que tiene el sistema sobre los productos [7].

3.1.1.4 Sistemas de recomendación demográficos

Los sistemas de recomendación demográficos se basan en la idea de que usuarios con unas características dadas tengan gustos similares a otros con características similares.

Se clasifica al usuario basándose en sus atributos personales y se realizan recomendaciones según el grupo demográfico al que pertenece. La ventaja de estos sistemas de recomendación es que no requieren que el usuario puntúe ítems ni requiere información de otros usuarios. Es la

técnica más simple, pero la menos precisa. Su gran ventaja es que siempre es capaz de producir una recomendación [8].

3.1.1.5 Sistemas de recomendación basados en utilidad

Los recomendadores basados en utilidad recomiendan utilizando el cálculo de la utilidad de cada uno de los servicios para el usuario. Evidentemente, el problema clave a resolver aquí es cómo crear una función que defina la utilidad para cada usuario y que después pueda ser empleada de manera adecuada para la recomendación [9].

El beneficio de las recomendaciones basadas en utilidad viene del hecho de que puede tener en cuenta para el cálculo de la utilidad algunas características que no están estrictamente relacionadas con los servicios ofrecidos, como por ejemplo, la confianza en el vendedor o la disponibilidad del producto, siendo posible llegar a soluciones de compromiso, por ejemplo entre precio y plazo de entrega para un usuario que tiene una necesidad inmediata.

Para los sistemas de recomendación basados en utilidad también existen una serie de inconvenientes tales como que requieren que el usuario defina la función de utilidad que debe ser satisfecha, puesto que requieren que el usuario tenga en cuenta todas las cualidades del dominio.

Podemos añadir que estos son sistemas estáticos y que no pueden aprender o mejorar sus recomendaciones como pueden hacer otros sistemas. Tampoco pueden adaptarse al usuario individual o a los dominios cambiantes.

3.1.1.6 Sistemas de recomendación híbridos

Todos los sistemas de recomendación anteriores tienen sus puntos fuertes y débiles, por lo que es lógico pensar que se pueden mejorar los resultados realizando una mezcla de dos o más técnicas de recomendación.

Un recomendador híbrido es aquel que combina múltiples técnicas en un único sistema consiguiendo una participación activa de todas ellas. Se definen hasta siete métodos diferentes para combinar estrategias de recomendación [10]:

- Método ponderado (Weighted): Se combinan las puntuaciones de los diferentes recomendadores que componen el sistema híbrido.

- Método de conmutación (Switching): El sistema conmuta entre las técnicas de recomendación en función de la situación actual. Se aplica una de las técnicas, si esta no entrega resultados, se prosigue a aplicar la siguiente técnica.

- Método mixto (Mixed): Se presentan las recomendaciones de diferentes sistemas de recomendación al mismo tiempo.

- Método de cascada (Cascade): Se basa en crear un recomendador con jerarquía, de forma que el recomendador más débil no puede revocar las decisiones tomadas por uno más fuerte, pero sí perfeccionarlas. Un recomendador en cascada utiliza un recomendador secundario solo para romper empates en la puntuación del primario.

- Método de combinación de características (Feature combination): Se toman las características de diferentes fuentes de conocimiento y se combinan y se dan a un único algoritmo de recomendación.

- Método de aumento de características (Feature augmentation): Una técnica de recomendación se utiliza para calcular una característica o conjunto de características, que luego es parte de la entrada de la siguiente técnica.

3.2 Trabajos relacionados

Se realizó una búsqueda sobre trabajos relacionados con el uso de datos sociales para realizar una recomendación, a continuación describiremos sus aportes.

Se encontró un sitio web, *receteame.com*, que utiliza un sistema de recomendación social persuasiva que recomienda recetas personalizadas para cada usuario. El sistema recupera recetas de Internet, calcula automáticamente su información nutricional y las restricciones de la dieta y usa esta información para realizar las recomendaciones. El sitio web ejecuta un algoritmo inteligente para aprender los gustos y necesidades de cada usuario y recomendar recetas personalizadas. *receteame.com* está capacitado para aprender las preferencias del usuario de dos fuentes de información: de los votos que el usuario realiza sobre cada receta, y, si el usuario está registrado en Facebook, de la actividad del usuario y sus amigos. [11]

En otro trabajo relacionado con *receteame.com*, [12] describe la forma de utilizar un modelo basado en grafos para extraer y modelar la información para utilizarla para el desarrollo de un sistema de recomendación social. Utilizar toda la información perteneciente a la interacción entre los usuarios y las relaciones de amistad conlleva un gran coste computacional, por ello es muy importante la forma en la que se guarda esta información.

Otro trabajo relacionado, [13] trata de sacar el máximo provecho a los datos sociales. En este trabajo se estudia el efecto que causa incluir datos sociales auxiliares en recomendadores basados en grafos. Se examina la contribución de dos tipos de información social: etiquetas y lazos de amistad. Como conclusión se descubre que al utilizar estos tipos de información social se consigue aumentar la precisión de la recomendación para estos sistemas. El impacto del uso de los datos de amistad es más fuerte que el del uso de las etiquetas, mientras que el mayor impacto viene dado por el uso de ambos tipos de información.

Estos trabajos vienen a refundar la hipótesis inicial de que la información social puede ser útil en el proceso de recomendación a un usuario. En el próximo punto se explicarán las tareas realizadas para cumplir los objetivos propuestos en este proyecto.

4. Desarrollo del recomendador

Como parte más importante del proyecto, se procedió a desarrollar los diferentes recomendadores. Para ello, primero fue necesario investigar la base de datos para encontrar la información útil. Para investigar los campos y tablas de la base de datos proporcionada se ha utilizado la herramienta *MySQL Workbench 6.2* cuya interfaz es muy intuitiva y fácil de manejar, permitiendo crear conexiones a la base de datos de forma sencilla, así como realizar las diferentes consultas.

Luego se implementaron diferentes métodos para poder extraer dicha información. Llegados a este punto, en el que ya teníamos clara la información disponible en la base de datos, sabíamos cuál podía resultarnos útil y teníamos implementados los diferentes métodos para extraer esos datos, se comenzó a implementar los siguientes recomendadores:

- Item based
- User based
- Social based
- Hybrid

Todo el código fue implementado en python 2.7 mediante el editor de textos *PyCharm Educational Edition*, que es una edición gratuita para estudiantes. Este editor hizo que el trabajo fuera más fácil y rápido gracias a su función de autocompletado, chequeo de errores, fácil navegación entre proyectos y archivos, entre otras cosas.

Se eligió este lenguaje por las siguientes ventajas:

- Facilidad de escribir código a partir de un algoritmo, así como la legibilidad del mismo.
- Se trata de un lenguaje bastante poderoso, con el que se pueden diseñar grandes algoritmos sin la necesidad de crear tanto código como en otros lenguajes.
- Se importan librerías de forma rápida y sencilla.
- Dado que es un lenguaje dinámico no es necesario declarar las variables, lo que reduce la probabilidad de errores y hace más ágil la programación.

4.1 Extracción de información

Como punto inicial al desarrollo fue necesaria la observación de la base de datos proporcionada para experimentar con los recomendadores. Cabe decir que esta base de datos contiene mucha información, pero realmente sólo nos centraremos en algunas tablas como son:

- La tabla de canciones: esta se llama ‘song’ y en ella podemos encontrar el campo del identificador de canción llamado ‘id’ y los campos ‘c1’, ‘c2’, ‘c3’, ‘c4’, ‘c5’ en los cuales están los 5 componentes que definen la huella de una canción. Además, podemos

ver a que banda y álbum pertenecen, autor, licencia y muchos otros datos que no vamos a utilizar en los algoritmos porque no los consideramos relevantes.

- La tabla de las listas de canciones: esta se llama 'playlist' y está formada por los campos 'id', 'usuario_id' y 'name'. De ella podemos extraer la información sobre las *playlist* que tiene cada usuario.

- La tabla que relaciona las *playlist* y las canciones: su nombre es 'song_in_playlist', en esta encontraremos las canciones que contienen las *playlist* de los usuarios y sus campos son 'song_id' y 'playlist_id'.

- La tabla de usuarios: esta tabla se llama 'usuario' y como campos a destacar contiene el 'id', que es un numero identificativo del usuario y 'username', que contiene el nombre del usuario. Además de estos, contiene muchos otros datos que no nos resultan útiles para la implementación de los recomendadores como: último login, la localización, fecha de nacimiento, si está bloqueado o no, etc.

- La tabla de *followers*: cuyo nombre es 'usuario_follower', de aquí podremos sacar información sobre la amistad entre dos usuarios, ya que contiene dos campos llamados 'follower_id' y 'followee_id', que nos indican quien sigue a quien.

Además de estas tablas la base de datos contiene otras que han sido descartadas porque se ha considerado que no poseen información valiosa para nuestro desarrollo. Aunque contiene una tabla de mensajes enviados entre usuarios, que podría ser muy útil para evaluar qué grado de amistad hay entre dos usuarios, se ha descubierto que existían muy pocos mensajes, lo cual nos ha llevado a descartarla.

Una vez investigada la base de datos, se ha llegado a las siguientes formas de representación:

- Las canciones se representan mediante sus huellas, que servirán para compararlas entre otras canciones, y mediante su número identificativo en la base de datos.

- Los usuarios serán representados por tantas huellas como *playlists* tengan, y estas huellas serán la media de las huellas de las canciones de dichas *playlist*. Esto nos servirá para obtener usuarios similares.

Después de saber cuáles son los datos necesarios para desarrollar los recomendadores, se crearon una serie de métodos, cuya función es obtener la información necesaria de la base de datos. Estos métodos son los siguientes:

- playlists_usuario(user): Consulta en mysql que devuelve una lista con los identificadores de las playlists que tiene un usuario dado. El valor de entrada es el nombre del usuario del que queremos obtener las playlists.

```
"SELECT playlist.id FROM playlist, usuario WHERE usuario_id= usuario.id and username= '" + user + "'"
```

- canciones_playlist(play): Obtiene los identificadores de las canciones que forman una *playlist*. El valor de entrada es el numero identificativo de una *playlist*.

```
"SELECT song_id FROM song_in_playlist WHERE playlist_id=" + str(play)
```

- canciones(): Método que devuelve una lista con todas las canciones, dichas canciones son representadas con su identificador y los 5 componentes de la huella.

```
"SELECT id,c1,c2,c3,c4,c5 FROM song"
```

- canciones_de_usuario(user): Devuelve los identificadores de las canciones de un usuario. El parámetro de entrada es el nombre del usuario.

```
"SELECT song_id FROM song_in_playlist,playlist,usuario WHERE usuario_id = usuario.id and username= '"+user+"'and playlist.id=song_in_playlist.playlist_id"
```

- huella_de_cancion(id): Obtiene la huella de una canción dada. El parámetro de entrada es el numero identificativo de dicha canción.

```
"SELECT c1,c2,c3,c4,c5 from song WHERE id = "+str(id)
```

- usuarios(): Devuelve una lista con todos los nombres de los usuarios.

```
"SELECT username FROM usuario"
```

- id_usuario(user): Devuelve el número identificador de un usuario dado. El parámetro de entrada es el nombre del usuario.

```
"SELECT id FROM usuario WHERE username = '"+user+"'"
```

- sigue(user): Obtiene la lista de usuarios a los que sigue el usuario dado. El parámetro de entrada es el nombre del usuario. Como recibe el nombre del usuario es necesario utilizar el método id_usuario(user) para obtener el identificador y usarlo en la consulta.

```
"SELECT username FROM usuario_follower,usuario WHERE followee_id=usuario.id and follower_id = "+str(id)+" "
```

- amigos(user): Devuelve la lista con los amigos de un usuario, los amigos son aquellos usuarios que se siguen mutuamente. Únicamente devuelve aquellos amigos que tengan 1 canción o más, ya que los que no tienen canciones, no nos valen para la recomendación. Este algoritmo calcula los usuarios a los que sigue el usuario dado mediante el método 'sigue(user)' y luego para cada uno de ellos comprueba si es recíproco o no.

- huella_usuario(user): Devuelve la huella de un usuario dado. El parámetro de entrada es el nombre del usuario del que queremos obtener la huella. Este método realiza estos cálculos para obtener la huella del usuario:

    - Consulta las *playlists* del usuario mediante una llamada al método 'playlists_usuarios(user)'.

    - Una vez tenemos las *playlist*, obtenemos las canciones de cada una de ellas utilizando el método canciones_playlists(play).

- Cuando ya tenemos las *playlists* con todas sus canciones obtenemos la huella de cada canción mediante el método huella_de_cancion(id) y calculamos la media de las huellas de cada canción perteneciente a una *playlist*.

- De esta forma el usuario queda representado por su huella, formada por las diferentes huellas de las *playlists* de dicho usuario.

4.2 Función de similitud

Además para el desarrollo de estos ha sido necesaria la implementación de una función que devuelve el grado de similitud entre dos canciones, que se representan mediante una lista compuesta por 5 elementos, a la que llamaremos huella. El valor dado será menor cuanto más similares sean las dos huellas. Esta función se llama similitud_simple(c1,c2) y realiza el siguiente cálculo:

$$\sum_{i=0}^{i=4} \frac{|h_i(c1) - h_i(c2)|}{5}$$

Donde c1 y c2 son las canciones a comparar.

Por ejemplo: dadas dos canciones representadas por las siguientes huellas: c1= [9.368, 2.2594, 4.4968, 4.4088, 5.7156] y c2= [7.275, 3.2957, 3.6843, 3.5434, 5.5745], la función de similitud realizaría el siguiente cálculo:

$$\frac{|9.386 - 7.275|}{5} + \frac{|2.2594 - 3.2957|}{5} + \frac{|4.4968 - 3.6843|}{5} + \frac{|4.4088 - 3.5434|}{5} + \frac{|5.7156 - 5.5745|}{5} = 0.98966$$

Esta función además de utilizarse para comparar huellas de canciones nos sirve para calcular similitudes entre usuarios, siendo estos representados por las huellas de sus *playlists*.

4.3 Basado en ítem

Este es un recomendador basado en contenido, el cual compara las huellas de todas las canciones con la huella del usuario, mostrando a este las canciones más similares. El algoritmo desarrollado sería el siguiente:

Algoritmo 1. Item based:

IN: usuario, num_canciones;
OUT: res;

huella_u = huella_usuario(usuario)
canciones_u = canciones_usuario(usuario)
songs1 = canciones()

```
for i=0, ..., i=len(songs1) - 1:
        if songs1[i][0] in canciones_u: continue
        huella_s =  songs1[i][1:6]
        for j=0, ..., j=len(huella_u) - 1:
                sim = similud_simple(huella_u[j], huella_s)
                sim_us.append(sim)
        min_sim = min(sim_us)
        if len(res) < num_canciones:
                res.append([min_sim, songs1[i][0]])
        else :
                max_res = max(res)
                if min_sim < max_res[0]:
                        res[res.index(max_res)] = [min_sim, songs1[i][0]]
return sorted(res)
```

Este algoritmo utiliza los métodos: 'huella_usuario', con el que se obtiene la huella del usuario; 'canciones_usuario', con el que se obtienen las canciones que tiene el usuario en sus *playlists*; 'canciones', con el que se obtienen todas las canciones para guardarlas en la variable songs1. Estos métodos han sido explicados anteriormente.

Se recorren todas las canciones, si la canción está en la lista de canciones del usuario se descarta y se pasa a la siguiente iteración. Si no, se extrae la huella de la canción. Esta huella corresponde a las posiciones 1-6 de la lista que la define. Una vez obtenida la huella, se compara con las diferentes huellas del usuario mediante el método 'similitud_simple'. Con ello, obtenemos una similitud entre la huella de la canción y la huella de la *playlist* que más se parezca.

Una vez hallada esta similitud, se guarda y si la lista de recomendación aún no contiene el número de canciones que indica el parámetro de entrada, se inserta a esta lista. En caso de que la lista ya tenga tantas canciones como indica el parámetro, se comparará con el mayor y se insertará si es menor a este.

Un ejemplo de ejecución para un usuario llamado 'pau' y una recomendación de 5 canciones sería el siguiente:

- Llamamos al algoritmo mediante la instrucción: *item_based('pau',5)*

- El resultado que obtenemos es: [[0.10404159999999996, 3446L], [0.11881999999999994, 1500L], [0.126737142857143, 252L], [0.12797468571428602, 5196L], [0.1315944000000001, 4667L]]

Como podemos ver, el algoritmo devuelve una lista de tantas tuplas como canciones hemos pedido, ordenado de menor a mayor, cuyo contenido es el grado de similitud y el identificador de la canción.

4.4 Basado en usuario

Este recomendador está basado en un filtrado colaborativo, su funcionamiento consta en obtener los usuarios más similares a un usuario dado. Una vez obtenidos estos usuarios, se extraen las canciones que más puedan gustarle al usuario.

Algoritmo 2. User based:

```
IN: usuario, num_usuarios_sim, num_canciones;
OUT: resCanciones;

huella_u = huella_usuario(user)
canciones_u= canciones_usuario(user)
lista_usuarios = []
users = usuarios()
for i= 0, ..., i=len(users) – 1:
   can = canciones(users[i])
   if len(can) >5:
            lista_usuarios.append(users[i])

res = []
for i=0, ..., i = len(lista_usuarios) – 1:
   if lista_usuarios[i] == usuario: continue
   huella_u_s = huella_usuario(lista_usuarios[i])
            sim_us = []
            if len(huella_u_s) == 0: continue
            for j = 0, ..., j = len(huella_u) - 1:
                     for k = 0, ..., k = len(huella_u_s) - 1:
                              sim = similitud_simple(huella_u[j],
                              huella_u_s[k])
                              sim_us.append((sim, j, k, huella_u[j],
                              huella_u_s[k], lista_usuarios[i]))

            min_sim = min(sim_us)

            if len(res) < num_usuarios_sim:
                     res.append([min_sim])
            else:
                     max_res = max(res)
                     if min_sim < max_res[0]:
                              res[res.index(max_res)]=[min_sim]

resCanciones = []
res.sort()

for i = 0, ..., i =  len(res) - 1:

            playlists = playlists_usuario(res[i][0][5])
            playlists2 = []

            for playlist in playlists:
                     if len(canciones_playlist(playlist[0])) > 0:
                              playlists2.append(playlist[0])
```

```
            canciones = canciones_playlist(playlists2[res[i][0][2]])

            for cancion in canciones:
                    if cancion[0] in canciones_u: continue
                    huella = huella_de_cancion(cancion[0])
                    sim = similitud_simple(res[i][0][4],huella[0]) *
                    res[i][0][0]
                    igual = 0
                    for j=0, ..., j=len(resCanciones) -1:
                            if resCanciones[j][1] == cancion[0]:
                                    igual = 1

                    if igual == 1: continue
                    if len(resCanciones)< num_canciones:
                            resCanciones.append([sim,cancion[0]])
                     else:
                            max_res= max(resCanciones)
                            if sim < max_res[0]:
                                    resCanciones[resCanciones.index
                                    (max_res)] = [sim, cancion[0]]

    return sorted(resCanciones)
```

Lo primero que realiza este algoritmo es una filtración de usuarios. Se obtienen todos los usuarios mediante el método 'usuarios()' y se guardan en una lista, luego para cada uno de ellos se comprueba si tienen al menos 5 canciones y si es así se meten en una lista llamada 'lista_usuarios', que es la que se utilizará posteriormente para calcular los usuarios similares. Esto se realiza para descartar aquellos usuarios que consideramos que tienen pocas canciones como para realizar una recomendación.

Una vez realizada la filtración, procedemos al cálculo de los usuarios más similares al usuario dado ('num_usuarios_sim'). Para ello se recorre la lista de usuarios y compara las huellas una a una para saber que *playlists* se parecen más. Una vez obtenidas las playlists más similares, se guardan sus huellas e índices en una lista junto con su grado de similitud. Luego se inserta a la lista de usuarios similares – llamada *res* – cuando el número de usuarios obtenido sea menor que el solicitado o cuando el usuario actual sea más similar que los insertados en dicha lista.

Cuando ya tenemos los usuarios similares procedemos a calcular las 'num_canciones' canciones a recomendar. Para ello recorremos los usuarios similares y calculamos sus *playlists* mediante el método 'playlists_usuario', que nos devuelve todas las *playlists*. Como en algunos casos hay usuarios que han creado *playlists* vacías, este método nos devuelve también el identificador de estas. Por lo tanto, tenemos que hacer un filtrado y quitarlas para poder utilizar el índice almacenado en *res*, ya que el método 'huella_usuario' descarta estas *playlists*.

Una vez realizado ese filtrado obtenemos las canciones de la playlist cuyo índice habíamos guardado anteriormente y para cada una de ellas comprobamos si pertenece a las canciones del usuario, en cuyo caso pasamos a la siguiente iteración. Si no, calculamos la similitud con la huella de la playlist del usuario e insertamos la canción junto a su similitud – esta se corresponde a la similitud entre los usuarios por la similitud entre la canción y la playlist del

usuario – si la canción actual es más similar que las anteriores o si el número de canciones aún no es el solicitado.

Un ejemplo de ejecución para un usuario llamado 'pau' y una recomendación de 5 canciones utilizando 5 usuarios similares sería el siguiente:

- Ejecutamos el algoritmo mediante la instrucción: *user_based('pau', 5, 5)*

- El resultado obtenido es: [[0.036387181876866966, 279L], [0.038278615415590876, 5001L], [0.04042785329999987, 3063L], [0.04063681169999983, 3064L], [0.04554863986086692, 1502L]]

Como en el recomendador anterior se obtiene una lista de 5 canciones con sus respectivas similitudes con la huella del usuario, ordenadas de menor a mayor, indicando que es más probable que le guste la primera canción, que la última.

4.5 Basado en la amistad

Este recomendador da más importancia a la interacción entre los usuarios de la red social, dado que trata de buscar las canciones más similares a un usuario en las playlist de sus amigos. El algoritmo inserta en una lista a los amigos de un usuario y luego realiza las mismas acciones que el recomendador anterior – ordena los usuarios dependiendo de su similitud y después añade a la lista de recomendación las canciones más similares –.

Cabe destacar que se pretendía utilizar más información para calcular el grado de amistad entre dos usuarios, como mensajes enviados entre sí, *likes* u otro tipo de interacciones pero, dado que la base de datos que se nos ha proporcionado no contiene casi mensajes, sólo se ha podido establecer ese grado dependiendo de si se siguen mutuamente o no.

Con este algoritmo se pretende mejorar los resultados de la recomendación basada en usuarios, ya que en este se utiliza mayor información sobre ellos y se piensa que el hecho de que dos usuarios se sigan mutuamente significa que sus gustos de música son similares, además de que al venir la recomendación de un amigo es más probable que el usuario escuche la canción. Se basa en una idea de persuasión por amistad.

Algoritmo 3. Social based:

```
IN: usuario, num_usuarios_sim, num_canciones;
OUT: resCanciones;

huella_u = huella_usuario(usuario)
canciones_u = canciones_usuario(usuario)
lista_usuarios = amigos(usuario)
res = []

for i=0, ..., i = len(lista_usuarios)-1

        huella_u_s = huella_usuario(lista_usuarios[i])
        sim_us = []
        if len(huella_u_s) == 0: continue
```

```
            for j = 0, ..., j = len(huella_u)-1:
                  for k = 0, ..., k = len(huella_u_s)-1
                        sim = similitud_simple(huella_u[j],
                        huella_u_s[k])
                        sim_us.append((sim, j, k, huella_u[j],
                        huella_u_s[k], lista_usuarios[i]))

            min_sim = min(sim_us)

            if len(res) < num_usuarios_sim
                  res.append([min_sim])
            else
                  max_res = max(res)
                  if min_sim < max_res[0]:
                        res[res.index(max_res)]=[min_sim]

resCanciones = []
res.sort()

for i = 0, ..., i =  len(res) - 1:

            playlists = playlists_usuario(res[i][0][5])
            playlists2 = []

            for playlist in playlists:
                  if len(canciones_playlist(playlist[0])) > 0:
                        playlists2.append(playlist[0])

            canciones = canciones_playlist(playlists2[res[i][0][2]])

            for cancion in canciones:
                  if cancion[0] in canciones_u: continue
                  huella = huella_de_cancion(cancion[0])
                  sim = similitud_simple(res[i][0][4],huella[0]) *
                  res[i][0][0]
                  igual = 0
                  for j=0, ..., j=len(resCanciones) -1 :
                        if resCanciones[j][1] == cancion[0]:
                              igual = 1

                  if igual == 1: continue
                  if len(resCanciones)< num_canciones:
                        resCanciones.append([sim,cancion[0]])
                  else:
                        max_res= max(resCanciones)
                  if sim < max_res[0]:
                        resCanciones[resCanciones.index(max_res)]
                        = [sim, cancion[0]]

return sorted(resCanciones)
```

La explicación sobre este algoritmo es la misma que para el anterior, con la única diferencia de que en este, el número de usuarios sobre los que buscar está limitado a los amigos del usuario.

El número de usuarios elegidos para realizar la recomendación será introducido como parámetro, como se hacía en el recomendador anterior.

Una ejecución del algoritmo sería la siguiente:

- Llamamos al método mediante la instrucción: *social_based('pau', 5, 5)*

- El resultadado proporcionado es: [[0.06036866493727805, 510L], [0.06117819750000018, 95L], [0.06701928386272185, 467L], [0.06815244207337273, 471L], [0.06909614631005902, 512L]]

La llamada utilizada utiliza los 5 amigos más parecidos al usuario para recomendar una lista de 5 canciones, ordenada de más a menos similitud entre canción y usuario.

4.6 Híbrido

El recomendador híbrido es una mezcla entre los dos anteriores. Trata de obtener un número de usuarios similares al usuario dado de diferentes formas. Un porcentaje de estos se obtendrá mediante el cálculo de usuarios similares y el resto se obtendrá con los usuarios que sean amigos. Para escoger entre todos los amigos, se utilizará al igual que en el recomendador social, la técnica de similitud entre usuarios. Una vez hallados los usuarios se obtendrán de ellos las canciones a recomendar.

La idea de este recomendador es mantener la idea de la fuerza social, pero añadir la potencialidad de los usuarios similares dado que en determinados casos el número de amigos es reducido.

Algoritmo 4. Híbrido:

```
IN: usuario, nut, p1, p2, num_canciones;
OUT: resCanciones;


ig = nut*p1/100
am = nut-ig
huella_u = huella_usuario(usuario)
lista_amigos = amigos(usuario)
lista_usuarios = usuarios()
res = []
for i=0, ..., i = len(lista_usuarios)-1
  if lista_usuarios[i] == usuario: continue
  huella_u_s = huella_usuario(lista_usuarios[i])
            sim_us = []
            if len(huella_u_s) == 0: continue
            for j = 0, ..., j = len(huella_u)-1:
                        for k = 0, ..., k = len(huella_u_s)-1
                                    sim = similitud_simple(huella_u[j],
                                    huella_u_s[k])
                                    sim_us.append((sim, j, k, huella_u[j],
                                    huella_u_s[k], lista_usuarios[i]))

            min_sim = min(sim_us)

            if len(res) < num_usuarios_sim:
```

```
                    res.append([min_sim])
            else:
                    max_res = max(res)
                    if min_sim < max_res[0]:
                            res[res.index(max_res)]=[min_sim]
res2 = []
for i=0, ..., i = len(lista_amigos) – 1:
  huella_u_s = huella_usuario(lista_amigos[i])
            sim_us = []
            if len(huella_u_s) == 0: continue
            for j = 0, ..., j = len(huella_u)-1:
                    for k = 0, ..., k = len(huella_u_s)-1
                            sim = similitud_simple(huella_u[j],
                            huella_u_s[k])
                            sim_us.append((sim, j, k, huella_u[j],
                            huella_u_s[k], lista_amigos[i]))

            min_sim = min(sim_us)

            if len(res2) < am:
                    if [min_sim] not in res:
                            res2.append([min_sim])
            else:
                    max_res = max(res2)
                    if min_sim < max_res[0] and [min_sim] not in res:
                            res2[res2.index(max_res)]=[min_sim]


res = res+res2
resCanciones = []
res.sort()

for i = 0, ..., i =  len(res) - 1:

            playlists = playlists_usuario(res[i][0][5])
            playlists2 = []

            for playlist in playlists:
                    if len(canciones_playlist(playlist[0])) > 0:
                            playlists2.append(playlist[0])

            canciones = canciones_playlist(playlists2[res[i][0][2]])

  for cancion in canciones:
            if cancion[0] in canciones_u: continue
            huella = huella_de_cancion(cancion[0])
            sim = similitud_simple(res[i][0][4],huella[0]) * res[i][0][0]
            igual = 0
            for j=0, ..., j=len(resCanciones) -1 :
                    if resCanciones[j][1] == cancion[0]:
                            igual = 1

            if igual == 1: continue
            if len(resCanciones)< num_canciones:
                    resCanciones.append([sim,cancion[0]])
            else:
```

```
                    max_res= max(resCanciones)
                    if sim < max_res[0]:
                            resCanciones[resCanciones.index(max_res)]
                            = [sim, cancion[0]]

return sorted(resCanciones)
```

Este algoritmo calcula el número de usuarios similares y el número de amigos que tiene que obtener dependiendo del porcentaje introducido como parámetro. Luego realiza las mismas acciones que los algoritmos anteriores, para calcular primero los usuarios similares y después los amigos, teniendo en cuenta en este último cálculo, que no hayan sido insertados en la lista de usuarios similares.

Cuando ya tenemos las dos listas con todos los usuarios, los mezclamos en una única lista para proceder al cálculo de las canciones a recomendar de la misma manera que se hacía en los recomendadores anteriores.

Un ejemplo de ejecución para este algoritmo sería:

- Para ejecutar el algoritmo se utiliza la llamada: *hibrido('pau', 10,50,50,5)*.

- El resultado de la ejecución es el siguiente: [[0.018928136705227416, 4218L], [0.02443230494186156, 4216L], [0.025510902990356108, 2569L], [0.026565227199151536, 2491L], [0.036387181876866966, 279L]]

Mediante la llamada utilizada indicamos que se calculará la recomendación con 10 usuarios similares, un 50% obtenido de usuarios cualquiera y el otro 50% obtenido de amistades, para obtener una lista de 5 canciones. El resultado, es una lista de 5 elementos, correspondientes al identificador de cada canción y su similitud.

5. Pruebas y evaluación

Una vez implementados todos los recomendadores necesitamos saber cuál de ellos ofrece unas mejores recomendaciones. Pero, ¿cómo definimos exactamente lo que es una buena recomendación? ¿Y cómo sabemos si nuestros recomendadores la producen? Sabemos que el mejor recomendador posible trataría de acertar con todos los ítems recomendados y predecir si al usuario pueden o no gustarle. Para ello son necesarias diferentes herramientas o técnicas que evalúen los resultados proporcionados.

Para poder determinar la calidad de los recomendadores se han utilizado estas tres métricas, que relacionan las canciones recomendadas y las relevantes para los usuarios, que son aquellas que pertenecen a sus *playlists*:

- Precisión: se calcula haciendo la intersección entre las canciones del usuario y las canciones recomendadas. Esto nos calcula el número de canciones acertadas por el recomendador, una vez obtenido, se divide por el número de canciones recomendadas

- Recall: es muy parecida al anterior, se hace igualmente la intersección entre las canciones recomendadas y las canciones que le gustan al usuario, pero se divide entre el número de canciones que le gustan al usuario.

- F1 Score: esta métrica combina los valores de precisión y recall y se calcula de la siguiente manera:

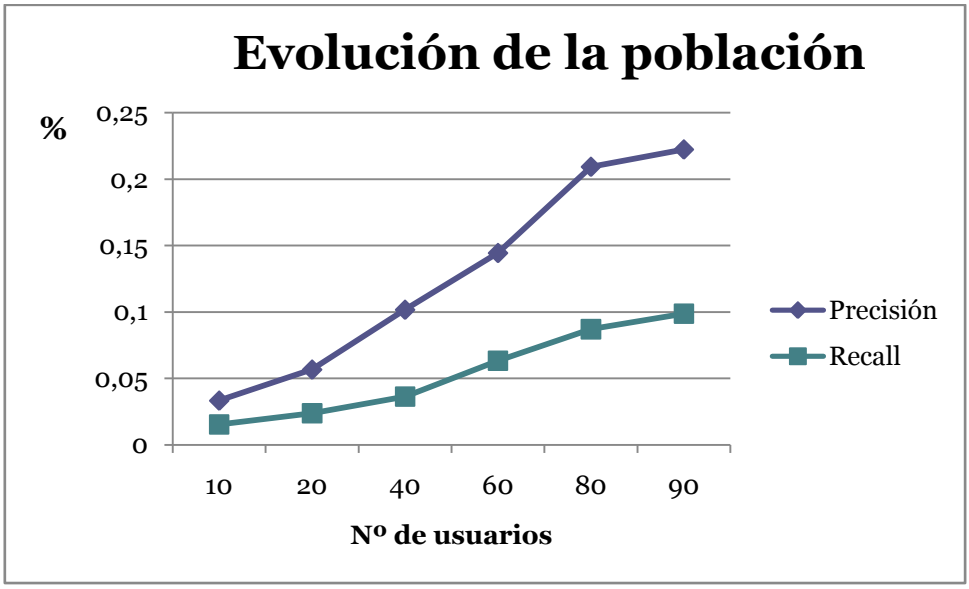
$$2 * \frac{precisión * recall}{precisión + recall}$$

Ahora que ya sabemos que métricas utilizar, lanzaremos diferentes ejecuciones de todos los algoritmos desarrollados, variando los parámetros de entrada. En concreto, se realizarán las siguientes pruebas:

- Primero, se estudiará la evolución del recomendador basado en usuario para determinar cuántos usuarios son necesarios para que las recomendaciones sean buenas.
- Posteriormente, se harán las siguientes pruebas sobre los recomendadores, con recomendaciones de 5 y 10 canciones:
    - Estimación de los mejores parámetros para los recomendadores basado en usuario, social e híbrido.
    - Comparación entre los recomendadores basados en ítem y usuario.
    - Comparación entre los recomendadores social e híbrido.
    - Comparación de las medias obtenidas para los recomendadores basado en usuario, social e híbrido.
- Por último, se compararan los porcentajes de mejora de cada uno de los recomendadores al cambiar de 5 canciones recomendadas a 10.

5.1 Evolución de los recomendadores dependiendo del número de usuarios

Por una parte, se ha realizado un estudio sobre cuál es la cantidad de usuarios necesarios en una red social para que los recomendadores obtengan los mejores resultados posibles. Cabe decir que únicamente se ha podido realizar este estudio con el recomendador basado en usuario. El recomendador basado en ítem no depende de los usuarios que haya, sino de las canciones, por lo que no tendría sentido realizar esta prueba para él. El basado en la interacción social necesita que los usuarios tengan amistades, y en la base de datos proporcionada solo hay 15 usuarios con amigos que tengan canciones en sus *playlists*, por lo que al ser tan pocos usuarios también se ha descartado hacer esta prueba. Para ello se ha ejecutado el recomendador basado en usuario escogiendo un número aleatorio de usuarios cada vez mayor y se han utilizado las métricas recall y precisión, explicadas anteriormente.



Gráfica 1. Evolución de los valores de precisión y recall dependiendo del número de usuarios

Con los resultados obtenidos mostrados en la gráfica 1, podemos decir que no hemos llegado a un límite de usuarios en los que se estanquen los valores de precisión y recall, por lo tanto, nos sería necesario contar con mas usuarios en la base de datos para llegar a unos resultados óptimos. Conforme se aumenta el número de usuarios, los valores de precisión y recall son cada vez mayores, ya que al haber más usuarios existen más canciones sobre las que calcular la similitud y posteriormente recomendar. Además, al haber una mayor cantidad de usuarios los escogidos para realizar la recomendación serán más similares entre sí que cuando el número de usuarios es pequeño.

Con estos resultados podemos concluir que nuestra base de datos limita la calidad de las recomendaciones ofrecidas por los sistemas implementados, ya que en la gráfica observamos claramente que los valores van ascendiendo sin llegar a un punto máximo.

5.2 Pruebas con una recomendación de 5 canciones

En este apartado, se estudiará cuales son los mejores parámetros para cada recomendador. Una vez obtenidos estos valores, se realizarán diferentes comparaciones entre los recomendadores. Para ello, se lanzaran los sistemas de recomendación con el objetivo de que el número de canciones recomendadas para cada uno de ellos sea 5.

Mediante estas pruebas se pretende determinar cuál de ellos funciona mejor, dependiendo de sus valores obtenidos para cada métrica.

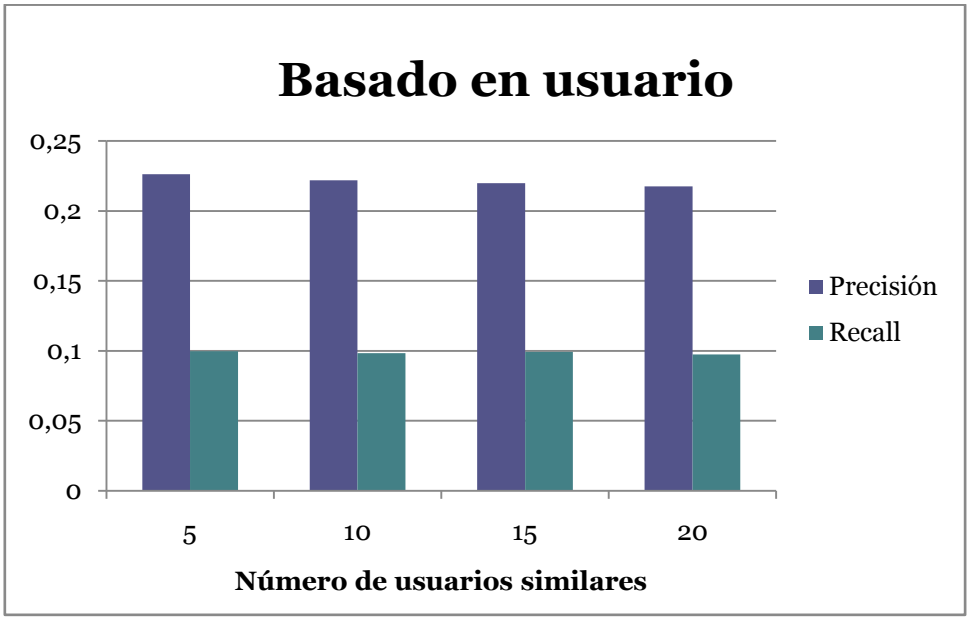
5.2.1 Estimación de los mejores parámetros para los recomendadores basado en usuario, social e híbrido.

Antes de comparar los recomendadores resulta necesario averiguar qué valor para el parámetro de usuarios similares es el que mejor resultados proporciona. Para ello se lanzaran ejecuciones de los tres recomendadores con los valores 5, 10, 15, 20, incluso 25 si se considera oportuno, y se calcularán las métricas de precisión y recall correspondientes para poder saber cuáles dan unos resultados óptimos para cada recomendador.

5.2.1.1 Basado en usuario

	5	10	15	20
Precisión	**0,2263**	**0,2219**	**0,2197**	**0,2175**
Recall	**0,0998**	**0,0983**	**0,0992**	**0,0974**

Tabla1. Precisión y recall para el recomendador basado en usuario con diferentes cantidades de usuarios.



Gráfica 2. Precisión y recall para el recomendador basado en usuario con diferentes cantidades de usuarios.

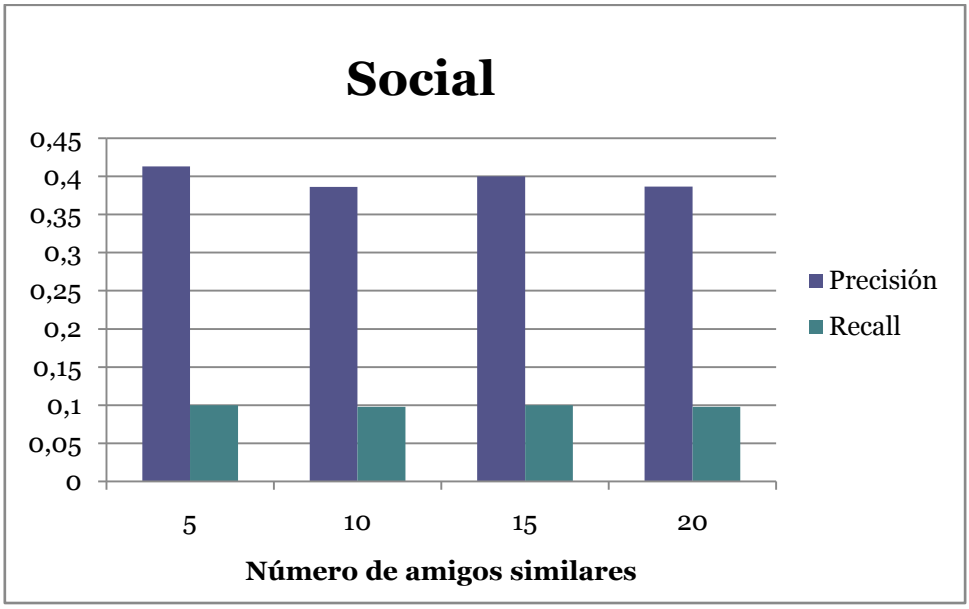
En la gráfica 2 y tabla 1 se muestran las medias de precisión y recall obtenidos para cada uno de los diferentes valores, para el parámetro de usuarios similares, con los que se han probado. Como se puede observar, aunque los resultados son muy parecidos, los valores van decreciendo conforme aumenta el número de usuarios similares, por lo que se ha decidido escoger 5 usuarios similares para realizar las siguientes comparaciones.

5.2.1.2 Social

Para el sistema de recomendación basado en la interacción social se ha decidido lanzar el recomendador variando el número de usuarios de 5 a 20 con incrementos de 5. Al realizar esta prueba se han obtenido los resultados mostrados en la tabla 2 y gráfica 3.

	5	10	15	20
Precisión	**0,413**	**0,386**	**0,4**	**0,3866**
Recall	**0,1**	**0,098**	**0,1**	**0,098**

Tabla 2. Precisión y recall para el recomendador social, utilizando diferentes números de amigos.



Gráfica 3. Precisión y recall para el recomendador social, utilizando diferentes números de amigos.

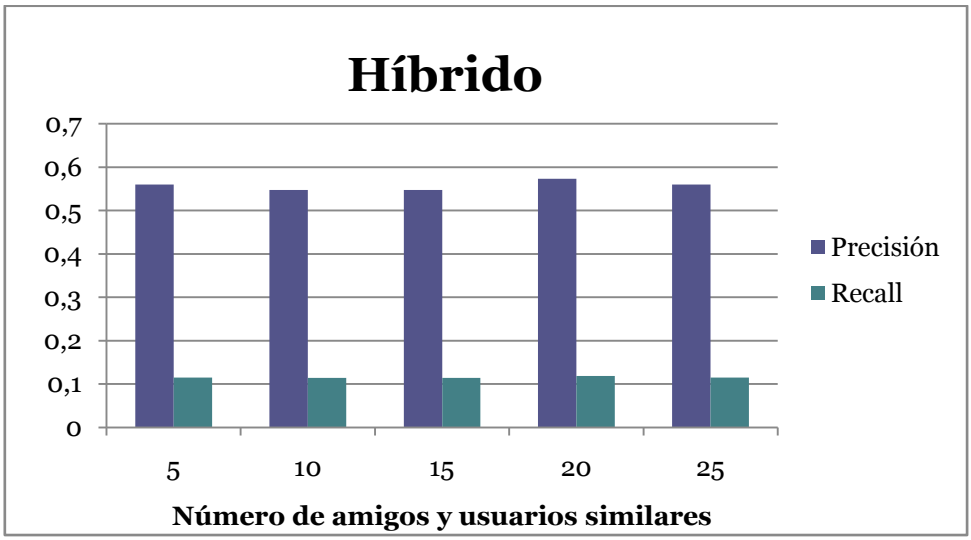
Como podemos ver en la tabla 2 y gráfica 3 los valores son muy similares pero el que mejor resultados proporciona, aunque la diferencia sea muy pequeña, es el recomendador con una lista de amigos de 5 usuarios. Por lo que para realizar las pruebas siguientes utilizaremos este valor para el parámetro.

5.2.1.3 Híbrido

Para el sistema de recomendación híbrido se ha decidido lanzar ejecuciones variando el parámetro de usuarios similares de 5 a 25, con incrementos de 5. Con ello hemos obtenido los resultados de la tabla 3 y gráfica 4.

	5	10	15	20	25
Precisión	**0,56**	**0,547**	**0,547**	**0,573**	**0,56**
Recall	**0,115**	**0,1146**	**0,1146**	**0,119**	**0,115**

Tabla 3. Precisión y recall para el recomendador híbrido con diferentes cantidades de usuarios



Gráfica 4. Precisión y recall para el recomendador híbrido, utilizando diferentes números de amigos.

Observando la tabla 3 y gráfica 4 llegamos a la conclusión de que el valor para el parámetro de amigos y usuarios similares con el que se obtienen mejores resultados para las diferentes métricas, aunque las diferencias no son muy grandes, es el de 20 amigos y usuarios similares, por lo que a partir de ahora, para realizar las diferentes comparaciones se utilizará este valor para el parámetro.

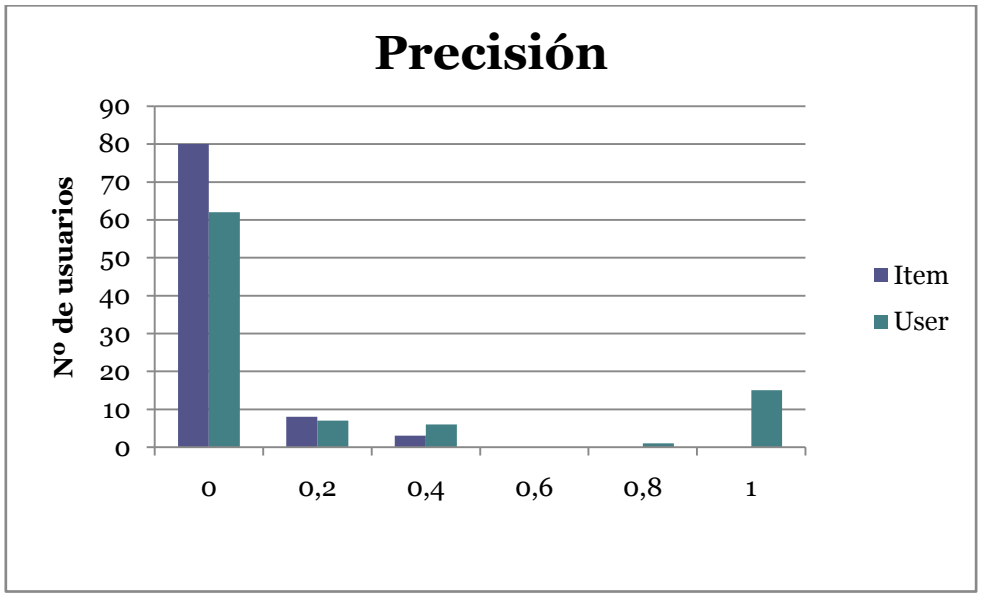
5.2.2 Comparación entre los recomendadores basados en ítem y usuario

Una vez obtenidos los valores adecuados para el parámetro de usuarios similares, se ha procedido a comparar los recomendadores basados en ítem y usuario, con una recomendación de 5 canciones. En esta prueba se ha realizado una recomendación para cada usuario que contenga más de 5 canciones en sus playlist y se ha calculado la precisión, recall y F1 Score para cada uno de ellos. Finalmente, se ha realizado una comparativa entre las medias de las métricas.

5.2.2.1 Precisión

		Precisión					
	%	0	0,2	0,4	0,6	0,8	1
Usuarios	Basado en ítem	**80**	**8**	**3**	**0**	**0**	**0**
	Basado en usuario	**62**	**7**	**6**	**0**	**1**	**15**

Tabla 4. Número de usuarios con los diferentes valores de precisión, para los recomendadores basado en ítem y usuario.



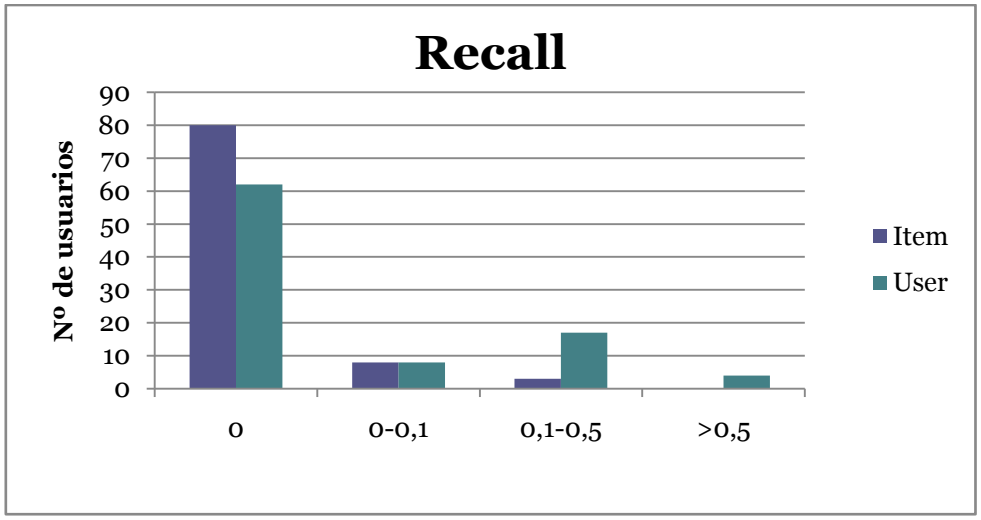
Gráfica 5. Número de usuarios con los diferentes valores de precisión, para los recomendadores basado en ítem y usuario.

La gráfica 5 y tabla 4 muestran el número de usuarios que tienen un valor de 0, 0.2, 0.4, 0.6, 0.8 y 1, lo que significa que de las 5 canciones recomendadas se aciertan 0, 1, 2, 3, 4 o 5 canciones respectivamente. Como podemos observar en esta gráfica ambos algoritmos muestran unos resultados no muy buenos, ya que con un total de 91 usuarios, en el basado en ítem, tenemos a 80 usuarios cuya precisión es del 0% y aunque el basado en usuario proporciona unos mejores resultados tampoco son muy buenos, ya que tenemos a 62 usuarios con una precisión del 0%. Aunque podemos ver que, el basado en usuario acierta con todas las canciones recomendadas a 15 usuarios, lo que le sitúa por encima del basado en ítem, el cual no acierta todas las canciones con ningún usuario.

5.2.2.2 Recall

		Recall			
	%	0	0-0,1	0,1-0,5	>0,5
Usuarios	Basado en ítem	**80**	**8**	**3**	**0**
	Basado en usuario	**62**	**8**	**17**	**4**

Tabla 5. Número de usuarios con los diferentes valores de recall, para los recomendadores basado en ítem y usuario.



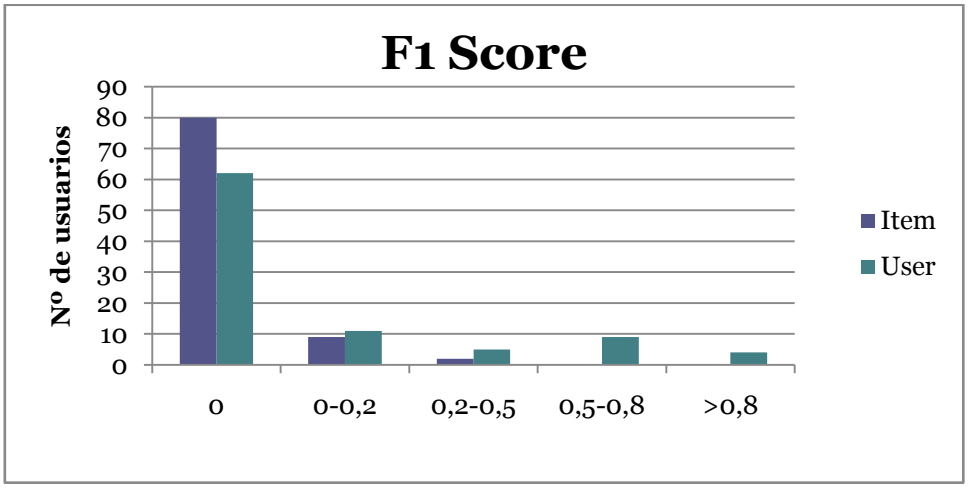
Gráfica 6. Número de usuarios con los diferentes valores de recall, para los recomendadores basado en ítem y usuario.

La gráfica 6 y tabla 5 son del mismo estilo que las anteriores, representan el número de usuarios con los que el recall de la recomendación ha obtenido los valores de 0, 0-0.1, 0.1-0.5, >0.5. En cuanto al recall podemos afirmar lo mismo de antes, el basado en usuario es un recomendador que obtiene mejores resultados que el basado en ítem, es decir recomienda un mayor número de canciones relevantes.

5.2.2.3 F1 Score

		F1 Score				
	%	0	0-0,2	0,2-0,5	0,5-0,8	>0,8
Usuarios	Basado en ítem	**80**	**9**	**2**	**0**	**0**
	Basado en usuario	**62**	**11**	**5**	**9**	**4**

Tabla 6. Número de usuarios con los diferentes valores de F1 Score, para los recomendadores basado en ítem y usuario.



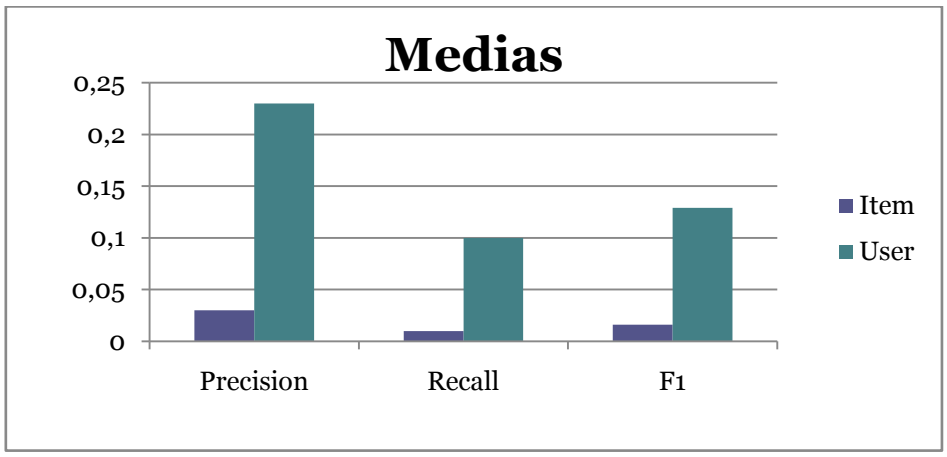
Gráfica 7. Número de usuarios con los diferentes valores de F1 Score, para los recomendadores basado en ítem y usuario.

La gráfica 7 y tabla 6 indican la cantidad de usuarios con los que se han obtenido unos valores de F1 Score de 0, 0-0.2, 0.2-0.5, 0.5-0.5, >0.8. La métrica F1 Score combina las métricas precisión y recall convirtiéndolas en una única métrica. En esta última métrica, podemos ver que también obtiene mejores resultados el sistema basado en usuario, algo que era de esperar ya que si en las métricas recall y precisión era mejor este sistema, en está métrica, que combina las dos, debe de serlo también.

5.2.2.4 Valores medios

	Precisión	Recall	F1
Basado en ítem	**0,03**	**0,01**	**0,016**
Basado en usuario	**0,23**	**0,1**	**0,129**

Tabla 7. Valores medios de las tres métricas, para los recomendadores basado en ítem y usuario.



Gráfica 8. Valores medios de las tres métricas, para los recomendadores basado en ítem y usuario.

Como podemos observar en la tabla 7 y gráfica 8, el algoritmo basado en usuario es bastante mejor que el basado en ítem, comparando las medias de cada una de las métricas podemos ver con exactitud cuanto mejor es dicho recomendador:

- En precisión, el basado en ítem se queda en un 3% mientras que el basado en usuario tiene una precisión del 23%.

- En cuanto al recall, tenemos un 1% en el basado en ítem frente al 10% del basado en usuario.

- Por último el F1 Score del basado en ítem está sobre el 1.6% mientras que en el basado en usuario llega al 12.9%.

5.2.3 Comparación entre el recomendador social e híbrido

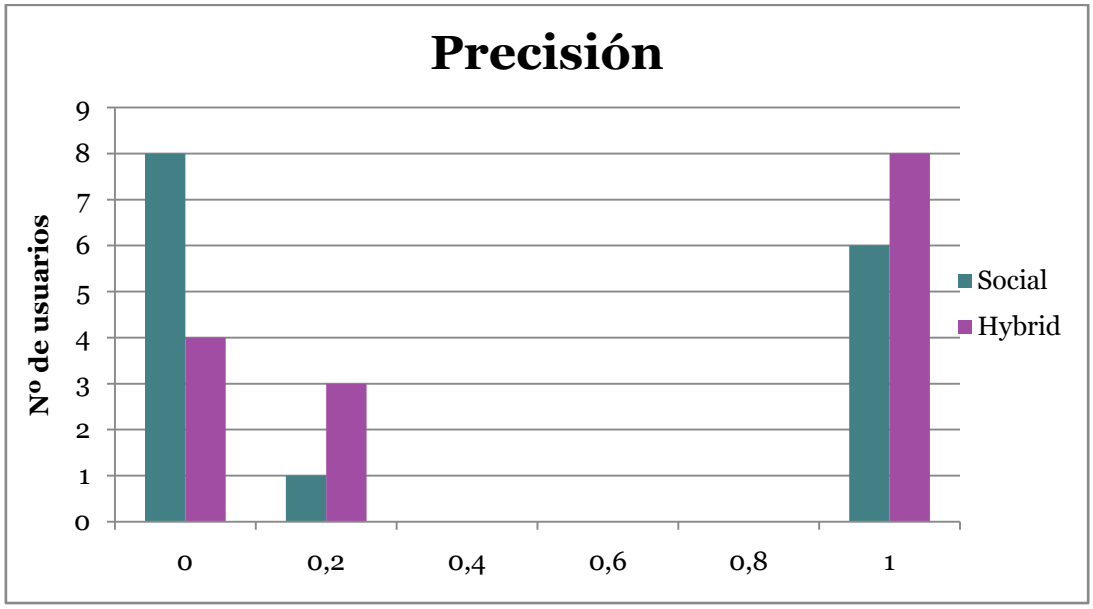
Una vez comparados los recomendadores basados en ítem y usuario, procedemos a comparar el recomendador social y el híbrido. Para ello se ha lanzado una ejecución para cada usuario cuyos

amigos contienen al menos una canción y se han obtenido los rankings de 5 canciones, sobre los que se han calculado las métricas anteriormente explicadas. Para el recomendador social se han utilizado 5 amigos, ya que como hemos visto en la comparación anterior es el número de amigos que mejores resultados proporciona a la hora de realizar una recomendación de 5 canciones. Para el híbrido se han utilizado 20 usuarios totales, calculando el 50% de ellos mediante la técnica del basado en usuario y el otro 50% utilizando la técnica del recomendador social.

5.2.3.1 Precisión

		Precisión					
	%	0	0,2	0,4	0,6	0,8	1
Usuarios	Social	**8**	**1**	**0**	**0**	**0**	**6**
	Híbrido	**4**	**3**	**0**	**0**	**0**	**8**

Tabla 8. Número de usuarios con los diferentes valores de precisión para el recomendador social e híbrido.



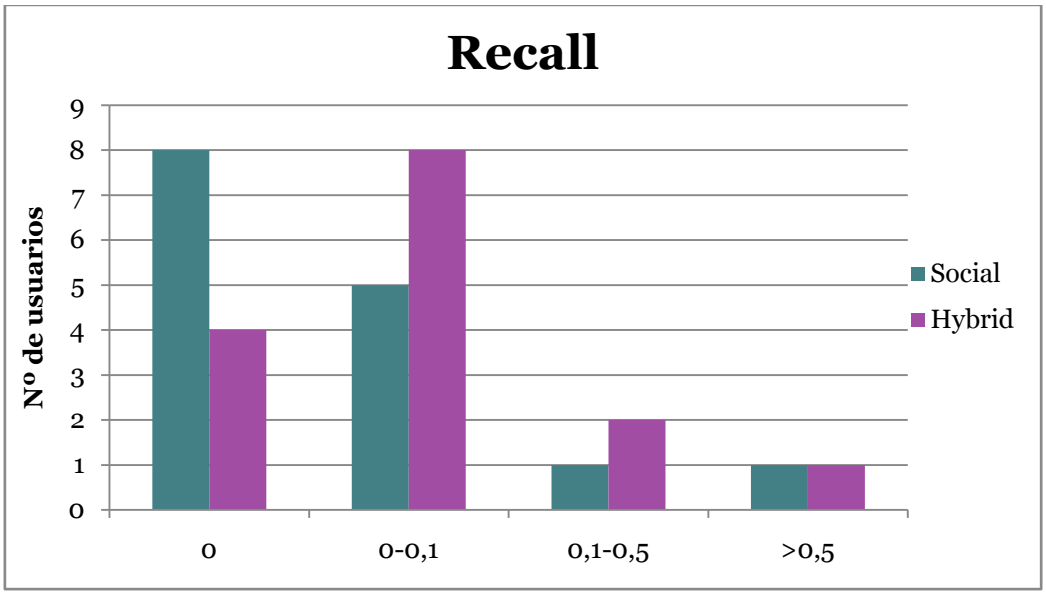
Gráfica 9. Número de usuarios con los diferentes valores de precisión para el recomendador social e híbrido.

En la tabla 8 y gráfica 9 observamos el número de usuarios con los valores de precisión de 0, 0.2, 0.4, 0.6, 0.8 y 1. Se puede apreciar que el recomendador híbrido obtiene mejores resultados de precisión. Además se puede decir que estos recomendadores, con estos usuarios, suelen o acertar todas las canciones recomendadas o no acertar ninguna, ya que no vemos usuarios en los que se hayan acertado 2, 3 o 4 canciones.

5.2.3.2 Recall

		Recall			
	%	0	0-0,1	0,1-0,5	>0,5
Usuarios	Social	**8**	**5**	**1**	**1**
	Hibrido	**4**	**8**	**2**	**1**

Tabla 9. Número de usuarios con los diferentes valores de recall para el recomendador social e híbrido.



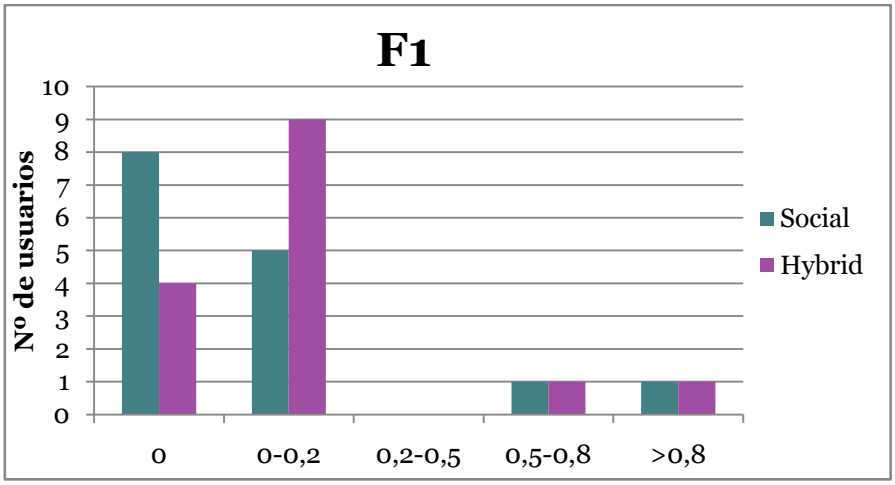
Gráfica 10. Número de usuarios con los diferentes valores de recall para el recomendador social e híbrido.

En cuanto al recall obtenido, como se puede apreciar en la tabla 9 y gráfica 10, los resultados son más similares, aunque el recomendador híbrido sigue funcionando mejor que el basado en la interacción social.

5.2.3.3 F1 Score

		F1 Score				
	%	0	0-0,2	0,2-0,5	0,5-0,8	>0,8
Usuarios	Social	**8**	**5**	**0**	**1**	**1**
	Híbrido	**4**	**9**	**0**	**1**	**1**

Tabla 10. Número de usuarios con los diferentes valores de F1 para el recomendador social e híbrido.



Gráfica 11. Número de usuarios con los diferentes valores de F1 para el recomendador social e híbrido.

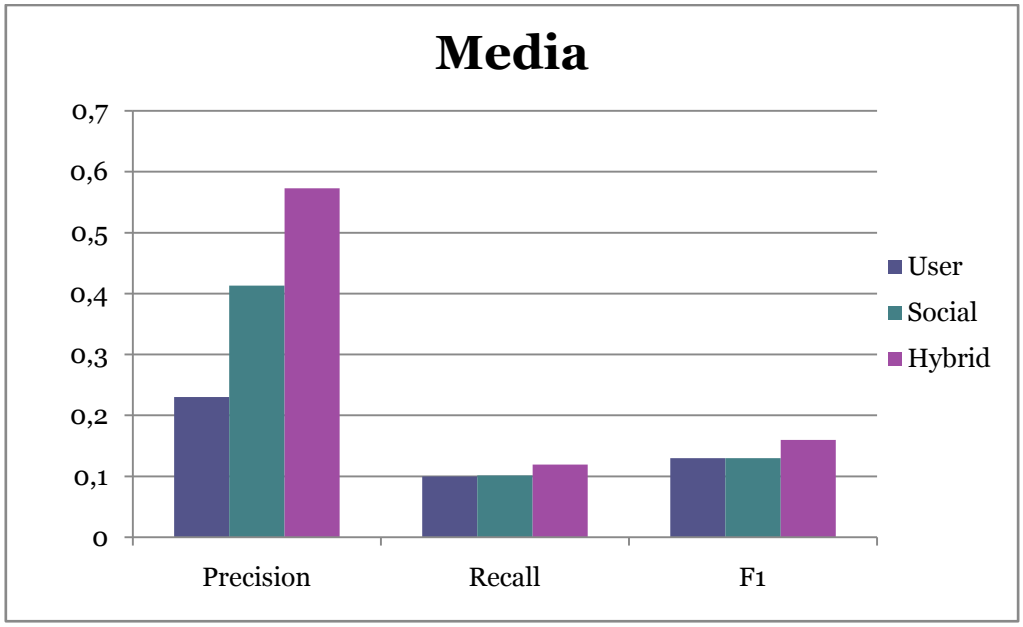
Como última métrica, podemos ver los resultados obtenidos para F1 Score de estos dos recomendadores, en la tabla 10 y gráfica 11. Apreciamos que los valores obtenidos en el recomendador híbrido son ligeramente superiores a los del recomendador social, por tanto, podemos concluir que en la recomendación de 5 canciones, el sistema de recomendación híbrido obtiene mejores resultados.

5.2.4 Comparación entre los recomendadores basado en usuario, social e híbrido

Por último, se desea realizar una comparación de los valores medios obtenidos para cada recomendador y métrica, descartando el basado en ítem por sus malos resultados en nuestra red social. Para ello, se han lanzado los recomendadores utilizando los parámetros óptimos, establecidos anteriormente, para los usuarios correspondientes a cada tipo de recomendación. Una vez obtenidas las canciones recomendadas, se ha procedido a calcular las diferentes métricas y medias correspondientes.

	Precisión	Recall	F1
Basado en usuario	0,23	0,1	0,13
Social	0,4133	0,1012	0,13
Híbrido	0,573	0,1194	0,16

Tabla 11. Valores medios de las métricas para los recomendadores basado en usuario, social e híbrido.



Gráfica 12. Valores medios de las métricas para los recomendadores basado en usuario, social e híbrido.

Observando la tabla 11 y gráfica 12 sobre las recomendaciones de 5 canciones, podemos afirmar que el recomendador que mejores resultados proporciona sería el híbrido, aunque en las métricas recall y f1 proporcionan resultados muy similares, en precisión es mucho mejor que los anteriores. Comparando cada métrica tenemos:

- Con respecto a la precisión, el peor recomendador es el basado en usuario con un 0.23, lo que significa que acierta en 1 de cada 5 canciones recomendadas. Luego vendría el basado en la interacción social con un valor de 0.41, cuyo significado es que acierta en 2 de cada 5 canciones recomendadas. Por último, como mejor recomendador, tenemos el híbrido, que combina las técnicas de los anteriores y obtiene un 0.57, lo que quiere decir que, acierta casi 3 de cada 5 canciones recomendadas.

- En cuanto al recall, los valores de los recomendadores basado en usuario y social son prácticamente idénticos con un 0.1, mientras que el valor correspondiente al recomendador híbrido se sitúa casi en un 0.12. Por lo tanto, aunque no con mucha diferencia, también es mejor en cuanto a esta métrica el recomendador híbrido.

- Por último, sobre el F1 Score, vemos que los valores obtenidos para el basado en usuario y el social son idénticos, con un 0.13, mientras que el híbrido los supera con un 0.16.

Mediante este conjunto de pruebas realizadas podemos concluir que el recomendador híbrido obtiene mejores resultados que los demás recomendadores, esto sucede porque este utiliza mayor información sobre los usuarios e interacción entre ellos que los anteriores, de este modo, es capaz de predecir con mayor exactitud los gustos del usuario y así obtener canciones con mayor posibilidad de acertar.

5.3 Pruebas con una recomendación de 10 canciones

Ahora procedemos a comparar de nuevo los recomendadores realizando las mismas pruebas pero con una recomendación de 10 canciones. Esto nos permitirá saber qué métricas mejoran al aumentar el número de canciones recomendadas y cuales disminuyen, ya que cuantas más canciones se recomienden, más difícil será acertar con todas. Al igual que para las recomendaciones de 5 canciones, primero se estimarán los mejores valores para el parámetro de número de usuarios a realizar y luego se realizaran diversas comparaciones entre los recomendadores.

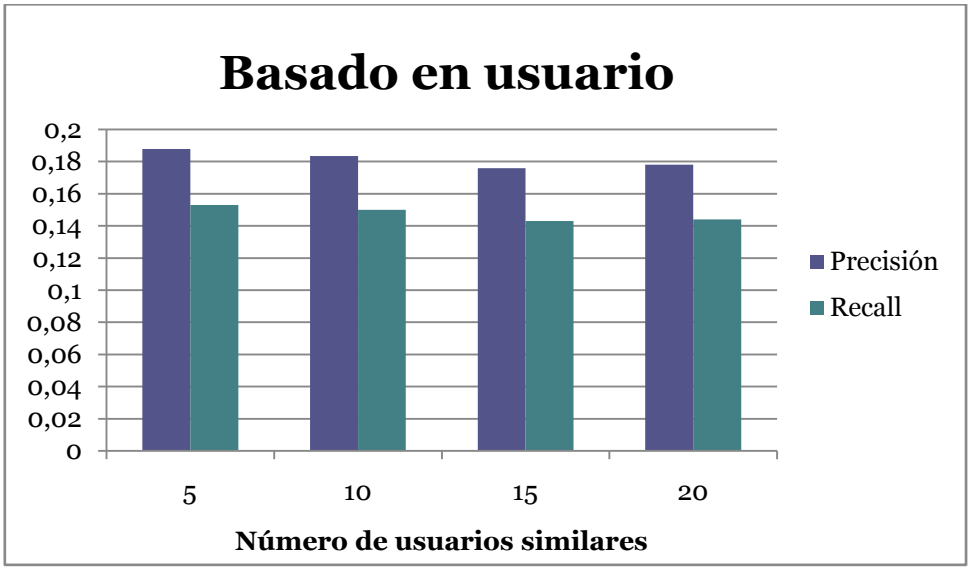
5.3.1 Estimación de los mejores parámetros para los recomendadores basado en usuario, social e híbrido.

Al igual que en la recomendación de 5 canciones hay que realizar las diferentes pruebas para saber qué parámetros son mejores para la recomendación de 10 canciones, ya que seguramente al dar una recomendación de un mayor número de canciones, sea necesario aumentar la lista de usuarios similares o amigos para tener una cantidad mayor de canciones sobre las que elegir. Para ello se lanzaran ejecuciones de los tres recomendadores con los valores 5, 10, 15, 20, y 25 si se cree oportuno, usuarios similares o amigos y se calcularán las métricas de precisión y recall correspondientes, de la misma manera que se hizo para las recomendaciones de 5 canciones.

5.3.1.1 Basado en usuario

	5	10	15	20
Precision	**0,1879**	**0,1835**	**0,1758**	**0,178**
Recall	**0,1529**	**0,15**	**0,143**	**0,144**

Tabla 12. Precisión y recall para el recomendador basado en usuario con diferentes cantidades de usuarios.



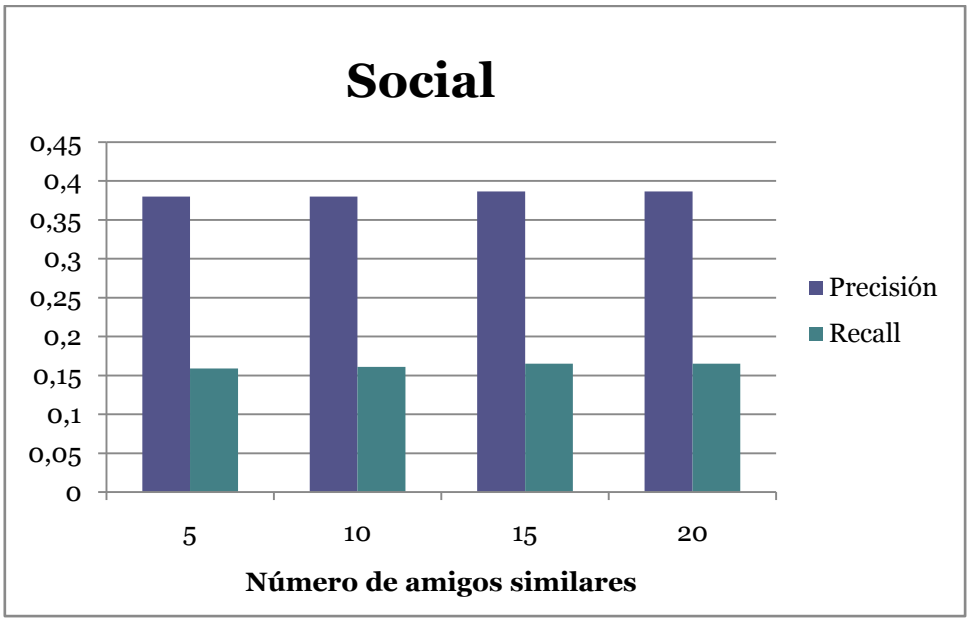
Gráfica 13. Precisión y recall para el recomendador basado en usuario con diferentes cantidades de usuarios.

Como vemos en la tabla 12 y gráfica 13, los resultados obtenidos son muy parecidos, pero se puede apreciar una ligera diferencia que nos afirma que, para el recomendador basado en usuario, el valor correspondiente al número de usuarios similares utilizados para realizar la recomendación de 10 canciones sigue siendo 5, al igual que en la recomendación de 5 canciones. El hecho de que no se den resultados totalmente idénticos quiere decir que el recomendador sí utiliza los últimos usuarios de la lista y al hacerlo, empeora ligeramente las recomendaciones.

5.3.1.2 Social

	5	10	15	20
Precisión	**0,38**	**0,38**	**0,38666**	**0,38666**
Recall	**0,159**	**0,161**	**0,165**	**0,165**

Tabla 13. Precisión y recall para el recomendador social, utilizando diferentes números de amigos.



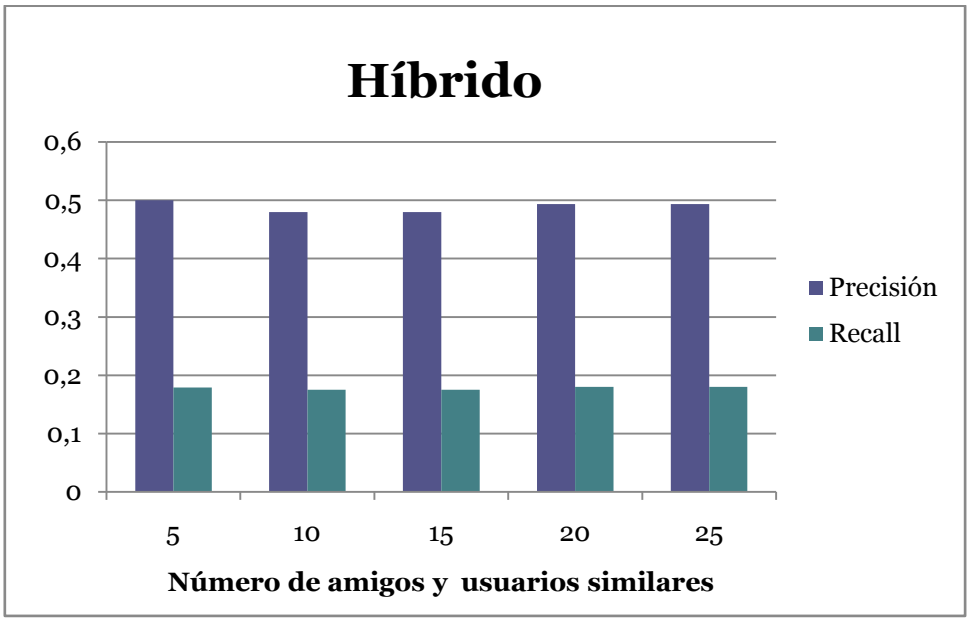
Gráfica 14. Precisión y recall para el recomendador social, utilizando diferentes números de amigos.

Como podemos observar en la tabla 13 y gráfica 14, aunque los resultados son muy similares, para el recomendador social el valor de amigos similares que mejor resultados proporciona es 15. Con 20 proporciona los mismos resultados, por lo que podemos afirmar que no utiliza los 5 últimos usuarios. Por lo tanto, nos quedaremos con el valor 15, ya que disminuye el coste de ejecución al calcular menos usuarios similares y realizar una búsqueda de canciones menor. A diferencia del recomendador basado en usuario ha sido necesario aumentar la lista de usuarios con respecto al utilizado en la recomendación de 5 canciones, que utilizaba una lista con 5 usuarios.

5.3.1.3 Híbrido

	5	10	15	20	25
Precisión	**0,5**	**0,48**	**0,48**	**0,4933**	**0,4933**
Recall	**0,179**	**0,175**	**0,175**	**0,18**	**0,18**

Tabla 14. Precisión y recall para el recomendador híbrido con diferentes cantidades de usuarios



Gráfica 15. Precisión y recall para el recomendador híbrido, utilizando diferentes números de amigos.

Se puede apreciar en la tabla 14 y gráfica 15 que la variación de los resultados en este recomendador también es muy pequeña. Con estos resultados tenemos la duda de que valor escoger si 5, 20 o 25. Por una parte, escoger 5 usuarios similares nos parece que es un número demasiado pequeño, a pesar de sus resultados, ya que para obtener 10 canciones a recomendar, puede que con 5 usuarios solamente, se tengan muy pocas canciones sobre las que elegir. Por otra parte, que los resultados obtenidos entre usar 20 y 25 usuarios sean iguales nos indica que de esos últimos 5 usuarios ya no se ha escogido ninguna canción. Por lo tanto, para ahorrar en coste de ejecución, se ha decidido establecer el valor de usuarios y amigos similares sobre los que buscar las canciones a recomendar en 20.

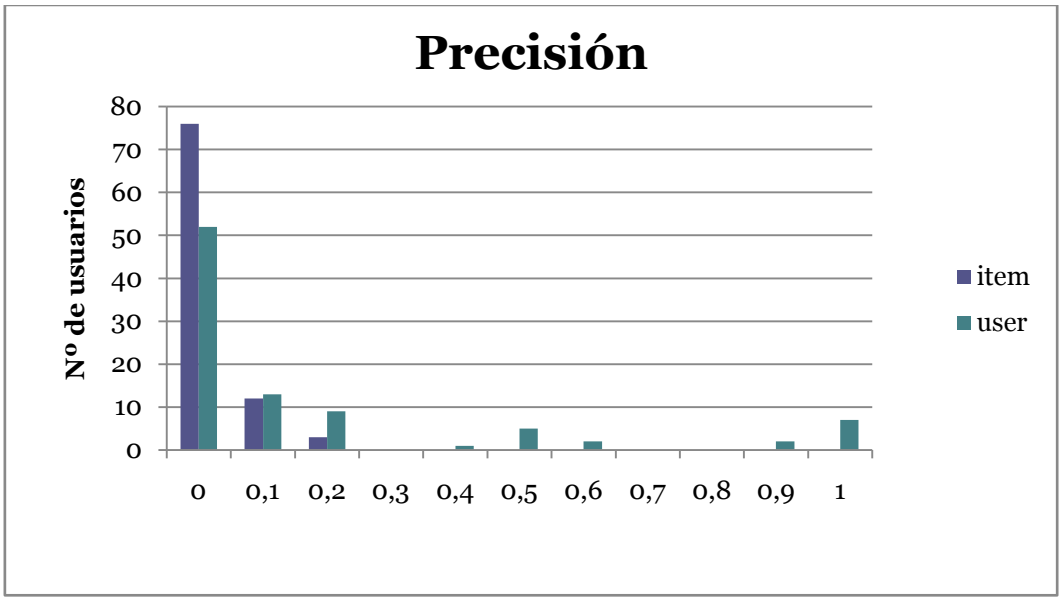
5.3.2 Comparación entre los recomendadores basados en ítem y usuario

Al igual que antes, comenzamos con una comparación entre los recomendadores basados en ítem y usuario, utilizando las métricas de precisión, recall y F1 Score.

5.3.2.1 Precisión

		Precisión										
	%	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
Usuarios	Basado en ítem	**76**	**12**	**3**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	Basado en usuario	**52**	**13**	**9**	**0**	**1**	**5**	**2**	**0**	**0**	**2**	**7**

Tabla 15. Número de usuarios con los diferentes valores de precisión, para los recomendadores basado en ítem y usuario.



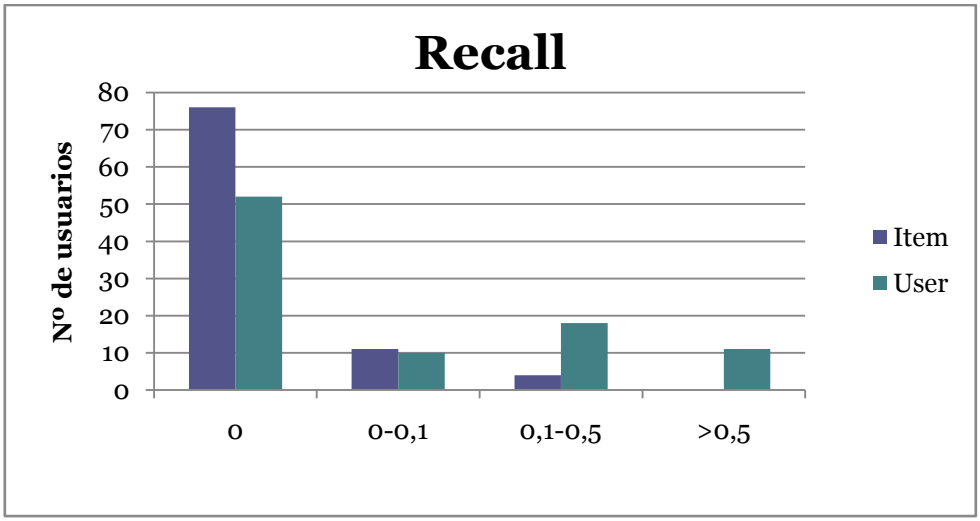
Gráfica 16. Número de usuarios con los diferentes valores de precisión, para los recomendadores basado en ítem y usuario.

En la tabla 15 y gráfica 16 se muestra el número de usuarios con los que se obtienen los valores 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 y 1 en la métrica de precisión, al realizar una recomendación de 10 canciones para ambos recomendadores. Estos valores indican que de las 10 canciones recomendadas se han acertado 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 o 10 canciones respectivamente. Vemos que el recomendador basado en ítem sigue siendo bastante peor que el basado en usuario, ya que con el primero tenemos 76 usuarios en los que no se acierta ninguna canción frente a las 52 del segundo.

5.3.2.2 Recall

		Recall			
	%	0	0-0,1	0,1-0,5	>0,5
Usuarios	Basado en ítem	**76**	**11**	**4**	**0**
	Basado en usuario	**52**	**10**	**18**	**11**

Tabla 16. Número de usuarios con los diferentes valores de recall, para los recomendadores basado en ítem y usuario.



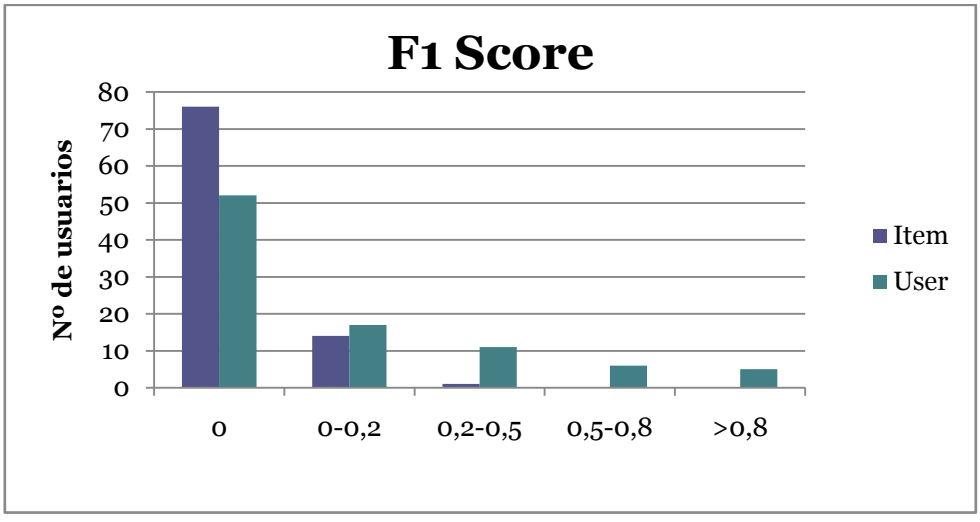
Gráfica 17. Número de usuarios con los diferentes valores de recall, para los recomendadores basado en ítem y usuario.

En la tabla 16 y gráfica 17 vemos el número de usuarios con los que el recall de la recomendación ha obtenido los valores de 0, 0-0.1, 0.1-0.5, >0.5. Podemos afirmar que en esta métrica el recomendador basado en usuario también supera los resultados del basado en ítem, lo que significa que el primero obtiene más canciones relevantes que el segundo.

5.3.2.3 F1 Score

		F1 Score				
	%	0	0-0,2	0,2-0,5	0,5-0,8	>0,8
Usuarios	Basado en ítem	**76**	**14**	**1**	**0**	**0**
	Basado en usuario	**52**	**17**	**11**	**6**	**5**

Tabla 17. Número de usuarios con los diferentes valores de F1 Score, para los recomendadores basado en ítem y usuario.



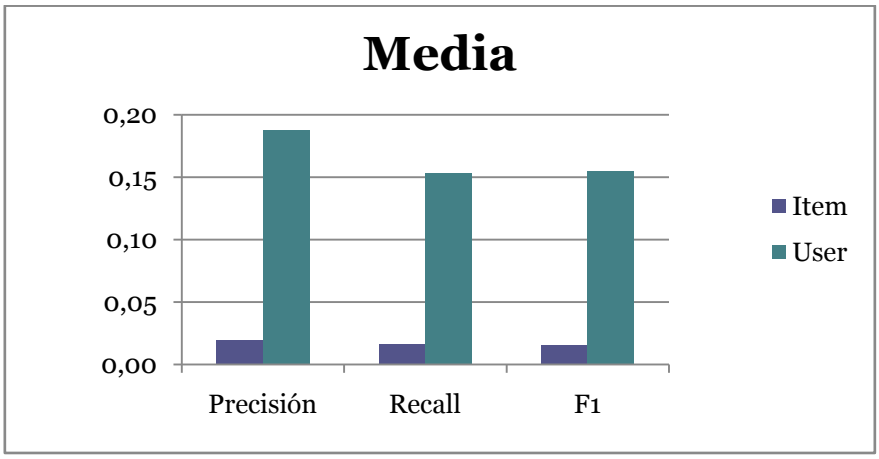
Gráfica 18. Número de usuarios con los diferentes valores de F1 Score, para los recomendadores basado en ítem y usuario.

La tabla 17 y gráfica 18 muestran la cantidad de usuarios sobre los que se obtienen los valores 0, 0-0.2, 0.2-0.5, 0.5-0.8, >0.8 de F1 Score. En ellas podemos ver que, como era de esperar, en esta métrica también proporciona mejores resultados el algoritmo basado en usuario, ya que la métrica consiste en una combinación de las dos anteriores.

5.3.2.4 Valores medios

	Precisión	Recall	F1
Ítem	0,020	0,016	0,016
Usuario	0,188	0,153	0,155

Tabla 18. Valores medios de las tres métricas, para los recomendadores basado en ítem y usuario.



Gráfica 19. Valores medios de las tres métricas, para los recomendadores basado en ítem y usuario.

Como última prueba para la comparación entre los dos sistemas de recomendación podemos ver en la tabla 18 y gráfica 19 la media de cada una de las métricas obtenida al realizar las recomendaciones. Con ella podemos comparar con exactitud la diferencia entre ambos recomendadores:

- En cuanto a la precisión, tenemos un valor de 0.02 en el basado en ítem frente al 0.188 del basado en usuario, lo cual nos índica la existencia de una diferencia bastante alta entre ellos.

- Sobre el recall, vemos que el basado en ítem obtiene un valor de 0.016 frente al 0.153 obtenido por el basado en usuario. En esta métrica también hay una diferencia bastante significativa entre los dos recomendadores.

- Por último, en la métrica F1 Score podemos ver que, como era de esperar, sigue habiendo una gran diferencia entre ellos, el valor del sistema basado en ítem es de 0.016, mientras que el del basado en usuario es de 0.155.

Mediante estas pruebas podemos concluir que el recomendador basado en ítem proporciona unos resultados bastante malos para nuestra red social. Mientras que los proporcionados por el

sistema de recomendación basado en usuario, aunque no son unos resultados muy buenos, son superiores, ya que aciertan en casi 2 de cada 10 canciones recomendadas.

5.3.3 Comparación entre el recomendador social e híbrido

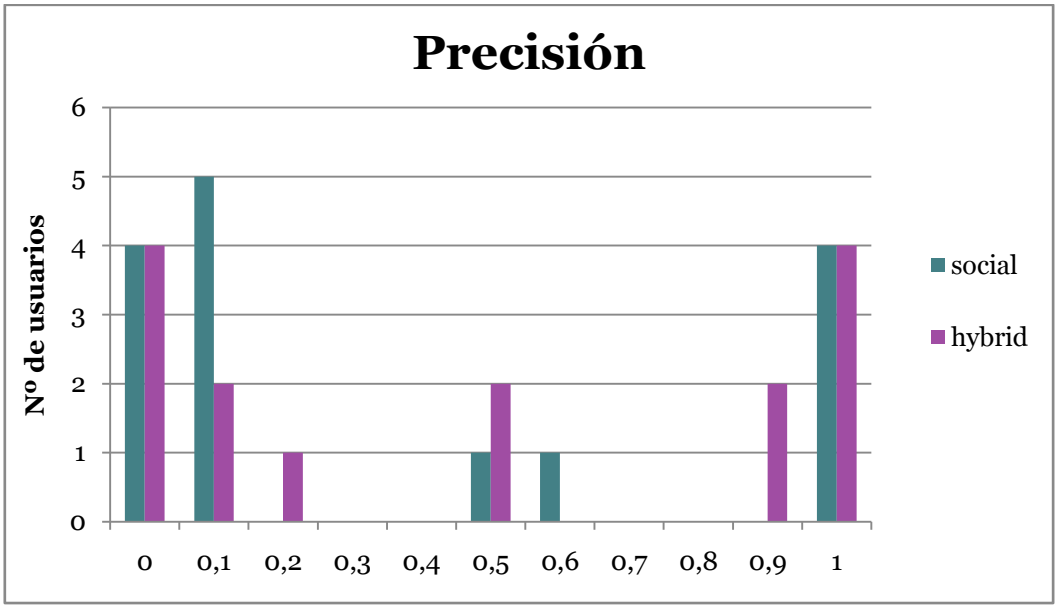
Ahora se procede a realizar una comparación entre el recomendador basado en el grado de amistad entre los usuarios y el recomendador híbrido que combina las técnicas utilizadas en el social y el basado en usuario.

Mediante esta prueba se pretende descubrir si el recomendador híbrido, al combinar las técnicas, produce mejores resultados que los anteriores.

5.3.3.1 Precisión

		Precisión										
	%	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
Usuarios	Social	**4**	**5**	**0**	**0**	**0**	**1**	**1**	**0**	**0**	**0**	**4**
	Híbrido	**4**	**2**	**1**	**0**	**0**	**2**	**0**	**0**	**0**	**2**	**4**

Tabla 19. Número de usuarios con los diferentes valores de precisión para el recomendador social e híbrido.



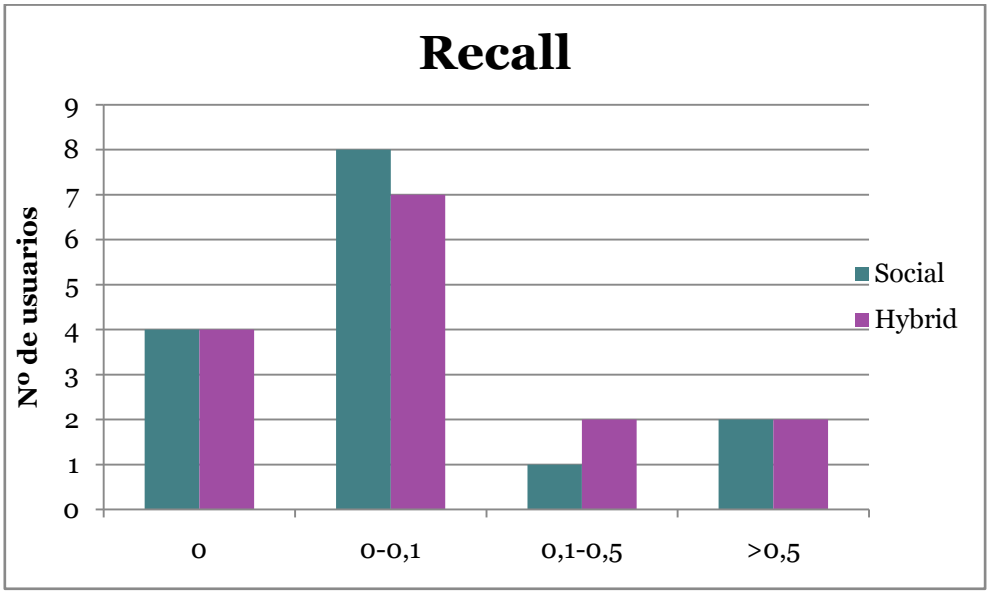
Gráfica 20. Número de usuarios con los diferentes valores de precisión para el recomendador social e híbrido.

En la tabla 19 y gráfica 20 se muestran el número de usuarios sobre los que se obtienen unos valores de precisión de 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 y 1. Como podemos ver, aunque son resultados bastante similares, el recomendador híbrido obtiene un mayor número de usuarios con precisiones más altas. Lo que quiere decir que, el híbrido acierta en un mayor número de canciones recomendadas.

5.3.3.2 Recall

		Recall			
	%	0	0-0,1	0,1-0,5	>0,5
Usuarios	Social	**4**	**8**	**1**	**2**
	Híbrido	**4**	**7**	**2**	**2**

Tabla 20. Número de usuarios con los diferentes valores de recall para el recomendador social e híbrido.



Gráfica 21. Número de usuarios con los diferentes valores de recall para el recomendador social e híbrido.

En cuanto al los valores de recall obtenidos, como podemos ver en la tabla 20 y gráfica 21, los recomendadores están más igualados. Aunque sigue obteniendo mejores resultados el recomendador híbrido, ya que el social tiene un usuario más con valor entre 0 y 0.1 y el híbrido tiene un usuario más con un valor entre 0.1 y 0.5.

5.3.3.3 F1 Score

		F1 Score				
	%	0	0-0,2	0,2-0,5	0,5-0,8	>0,8
Usuarios	Social	**4**	**7**	**2**	**1**	**1**
	Híbrido	**4**	**7**	**2**	**1**	**1**

Tabla 21. Número de usuarios con los diferentes valores de F1 para el recomendador social e híbrido.

Gráfica 22. Número de usuarios con los diferentes valores de F1 para el recomendador social e híbrido.

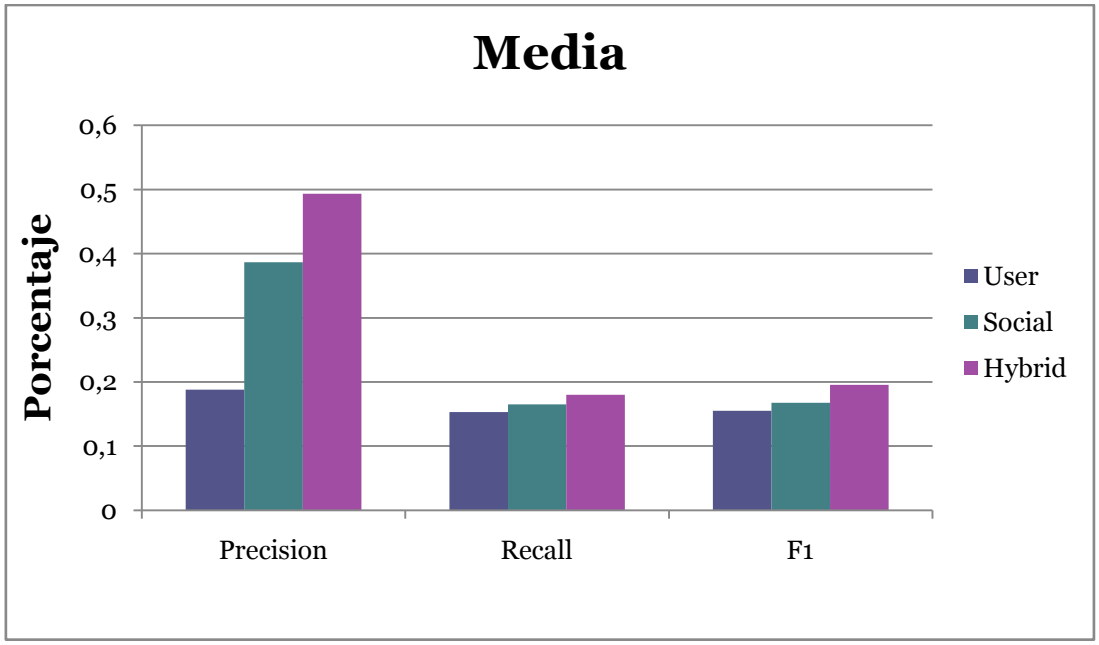
La gráfica 22 y tabla 21 no resultan muy significativas, ya que las diferencias para esta métrica son pequeñas y no aparecen en estos resultados.

5.3.4 Comparación entre los recomendadores basado en usuario, social e híbrido

Como última prueba para este apartado, se ha realizado una comparación de las medias de las métricas utilizadas, obtenidas para los 3 últimos recomendadores. Con esta prueba se pretende establecer cuál es el mejor recomendador para nuestra red social musical.

	Precisión	Recall	F1
Basado en usuario	**0,188**	**0,153**	**0,155**
Social	**0,39**	**0,1651**	**0,1678**
Híbrido	**0,4933**	**0,18**	**0,1953**

Tabla 22. Valores medios de las métricas para los recomendadores basado en usuario, social e híbrido.



Gráfica 23. Valores medios de las métricas para los recomendadores basado en usuario, social e híbrido.

Observando los datos obtenidos en la tabla 22 y gráfica 23, podemos ver claramente, que al igual que cuando se trataba de una recomendación de 5 canciones, el recomendador que mejores resultados proporciona, en base a las 3 métricas utilizadas es el recomendador híbrido. Podemos analizar cada métrica con exactitud:

- En cuanto a la precisión obtenida por cada recomendador, tenemos que el que más precisión proporciona es el recomendador híbrido con un 49%, frente al 39% obtenido del recomendador social y al 19% del basado en usuario.

- Sobre el recall, se aprecian diferencias más pequeñas pero el orden de los recomendadores dependiendo de su valor sigue siendo el mismo. El mejor es el híbrido con un 18%, seguido del social con un 16.5% y por último, como peor recomendador, el basado en usuario con un 15%.

- En cuanto al F1 Score, obtenemos los mismos resultados, el mejor es el híbrido, con un 19.5%, frente al 16.7% del social y el 15.5% del basado en usuario.

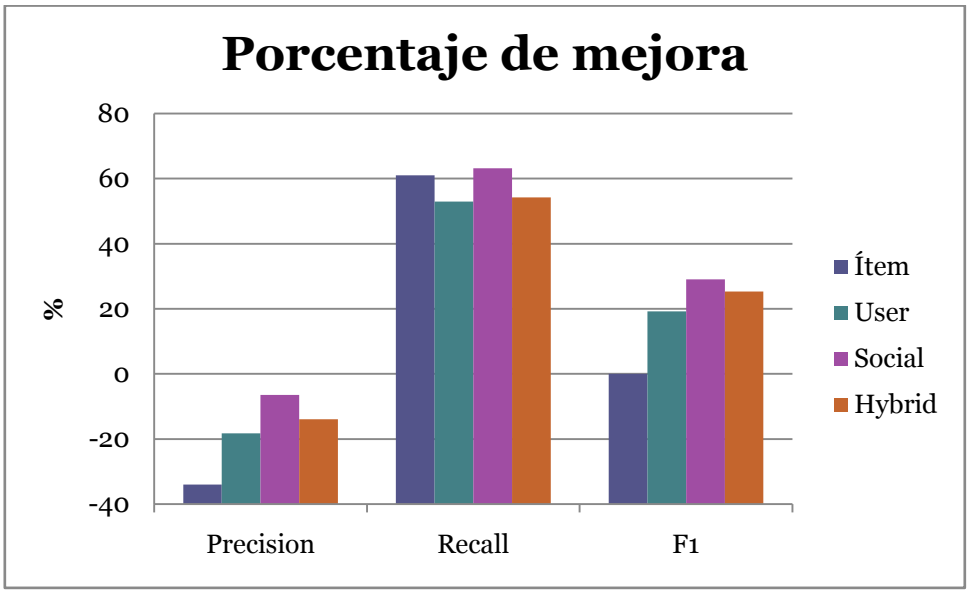
Con esta comparación podemos concluir que el recomendador híbrido es mucho más preciso que los otros dos, quedando muy por debajo el recomendador basado en usuario.

5.4 Comparación al cambiar el nº de canciones recomendadas

Como última prueba se procede a comparar los porcentajes de mejora de los valores de las diferentes métricas para cada recomendador al cambiar de 5 a 10 canciones recomendadas.

	Precision	Recall	F1
Basado en ítem	-34	61	0
Basado en usuario	-18	53	19
Social	-6	63	29
Híbrido	-14	54	25

Tabla 23. Porcentajes de mejora de las métricas al aumentar las canciones recomendadas para los recomendadores basado ítem, usuario, social e híbrido.



Gráfica 24. Porcentajes de mejora de las métricas al aumentar las canciones recomendadas para los recomendadores basado ítem, usuario, social e híbrido.

Podemos comparar los resultados obtenidos en la tabla 23 y gráfica 24 centrándonos en cada una de las métricas:

- En cuanto a precisión tenemos que, el basado en ítem es el que más empeora, mientras que el que menos lo hace es el social. Como podemos ver todos los recomendadores empeoran en esta métrica. Esto era algo de esperar, dado que al aumentar el número de canciones recomendadas, es más difícil acertarlas todas y conforme se añaden más canciones a la recomendación van siendo menos similares al usuario, por lo que es más difícil que sean de su gusto.

- Sobre el recall podemos decir que todos aumentan sus valores, ya que al aumentar el número de canciones recomendadas, es lógico que también aumente el número de canciones relevantes obtenidas. En esta métrica los porcentajes de mejora son muy similares para los recomendadores, aunque podemos ver que el que más mejora es el recomendador social, mientras que el que menos lo hace es el basado en usuario.

- Los porcentajes de mejora para la métrica F1 Score son similares entre los recomendadores basado en usuario, social y híbrido, mientras que el basado en ítem no mejora. Podemos ver que el que más mejora sus resultados es el social, luego vendría el híbrido y por último el basado en usuario.

6. Conclusiones

El trabajo realizado en este proyecto se enfocó en el diseño e implementación de un recomendador social y posteriormente, en la comparación con otros recomendadores. Este se basa en la idea de que la persuasión que puede tener un amigo del usuario es mucho mayor a la de un usuario cualquiera. Por ello se calculan las canciones a recomendar escogiéndolas de los usuarios amigos del usuario activo, que es aquel al que se le va a realizar la recomendación.

Para realizar este objetivo principal, fue necesario el diseño e implementación de cuatro recomendadores para poder comparar los resultados de dichos sistemas. En primer lugar, se consiguió de forma satisfactoria diseñar e implementar un recomendador basado en ítem – en nuestro caso se basa en las canciones –, que busca en el conjunto de canciones la más similar al usuario activo. En segundo lugar, se logró diseñar un recomendador basado en usuario, que trata de buscar las canciones a recomendar en los usuarios más similares al usuario activo. En tercer lugar, se diseñó e implementó un recomendador social, que busca las canciones entre los amigos del usuario. En último lugar, se diseñó e implementó un recomendador híbrido, que mezcla las técnicas del recomendador basado en usuario y el recomendador social.

Una vez diseñados e implementados todos los recomendadores propuestos, se procedió a realizar diferentes pruebas y comparaciones:

- Se realizó una prueba para descubrir con qué número de usuarios almacenados en la base de datos y con canciones en sus *playlist*, los resultados del recomendador basado en usuario son óptimos. Para ello, se lanzaron diferentes ejecuciones escogiendo diferentes cantidades de usuarios de forma aleatoria. En los resultados obtenidos pudimos ver que el recomendador aumentaba sus valores de precisión y recall conforme aumentaba el número de usuarios. Además, pudimos ver que nuestra base de datos se queda corta en cuanto al número de usuarios sobre los que probar, ya que únicamente pudimos utilizar 90 usuarios como máximo y no se apreció ningún límite en los resultados.

- Se compararon los diferentes recomendadores entre sí, utilizando una recomendación de 5 y 10 canciones. En estas comparaciones se utilizaron las métricas precisión, recall y F1 Score. En ellas se vio que para la recomendación de 5 y 10 canciones, el orden de los recomendadores de acuerdo a cual funciona mejor es el mismo. Como mejor recomendador se sitúa el híbrido, después le sigue el recomendador social, luego el basado en usuario y por último, como peor recomendador se sitúa el basado en ítem.

- Se hizo una comparación entre el porcentaje de mejora que producía sobre cada recomendador incrementar el número de canciones recomendadas de 5 a 10. En esta se observó que el recomendador que más mejoraba era el recomendador social, al que le seguía el recomendador híbrido, luego el basado en usuario y por último el basado en ítem.

En resumen, se ha comprobado que el uso de la información social para realizar una recomendación mejora los resultados que proporcionan los recomendadores que utilizan un filtrado basado en contenido y los que utilizan un filtrado colaborativo. Aunque para obtener unos mejores resultados todavía, se pueden mezclar estas técnicas y crear un recomendador

hibrido, que mantiene la idea de la fuerza social y añade la potencialidad de los usuarios similares, dado que en algunos casos en número de amigos es reducido. En base a los objetivos planteados al inicio del proyecto y realizando un análisis general del trabajo realizado, se puede concluir que estos han sido alcanzados de forma satisfactoria.

6.1 Trabajo futuro

A lo largo del proyecto se han ido descubriendo mejoras y trabajos que se podrían realizar en el futuro. A continuación se describirán los más relevantes.

Dado que la base de datos que se nos ha proporcionado resulta algo carente de información, se podrían probar los diferentes recomendadores implementados en una base de datos con más información. Esta podría referirse a: un mayor número de usuarios con canciones en sus *playlist*; más datos pertenecientes a la interacción entre usuarios, como podrían ser mensajes o *likes*, que nos servirían para mejorar el grado de amistad a calcular en el recomendador social.

Si tenemos una base de datos con mayor información sobre la interacción de los usuarios se podría cambiar el recomendador social para que añadiera esa información en el proceso de recomendación. Así, podríamos calcular de mejor forma el grado de amistad entre dos usuarios y escoger aquellos que tengan un mayor grado.

Se podrían crear diferentes funciones de similitud para utilizarlas en los diferentes recomendadores y compararlas entre sí para determinar cuál de ellas proporciona mejores resultados.

7. Bibliografía

[1] Ricci, F., Rokach, L., Shapira, B. (2011). Introduction to Recommender Systems Handbook.

[2] Herlocker, J., Konstan, J., Terveen, L. y Riedl, J. Evaluating Collaborative Filtering Recommender Systems.

[3] Vicente, M. (2012). Estrategias para el incremento de la fiabilidad Y confianza en sistemas de recomendación Colaborativa. Un enfoque semántico para la personalización De la publicidad en comercio electrónico.

[4] Balabanovic, M. y Shoham, Y. (1997). Content-based, collaborative recommendation.

[5] Claypool, M. G. (Agosto 1999). Combining content-based and collaborative filters in an online newspaper.

[6] Burke, R. (2000). Knowledge-based recommender systems. Encyclopedia of library and information systems.

[7] M.J. Barranco, L.G. Pérez, L. Martínez. (2006). Un Sistema de Recomendación Basado en Conocimiento con Información Lingüística Multigranular.

[8] Fink J., Kobsa A. A review and analysis of commercial user modeling servers for personalization on the World Wide Web.

[9] Burke, R. (2002). Hybrid recommender systems: Survey and experiments. User modeling and user-adapted interaction.

[10] Burke, R. (2007). Hybrid web recommender systems. In The adaptive web. Springer Berlin Heidelberg.

[11] Palanca, J., Heras, S., Botti, V., Julián, V. receteame.com: a Persuasive Social Recommendation System.

[12] Palanca, J., Heras, S., Jorge, J., Julián, V. Using Graph-Based Models in a Persuasive Social Recommendation System.

[13] Tiroshi, A., Berkovsky, S., Ali Kaafar, M., Vallet, D., Kuflik, T. Graph-Based Recommendations: Make the Most Out of Social Data.