Universidad del Bío-Bío

Facultad de Ciencias Empresariales

Departamento de Sistemas de Información

Profesor Guía: Sr. Pedro Campos

"REVISIÓN, SELECCIÓN E IMPLEMENTACIÓN DE UN ALGORITMO DE RECOMENDACIÓN DE MATERIAL BIBLIOGRÁFICO UTILIZANDO TECNOLOGÍA J2EE"

Trabajo de Titulación presentado en conformidad a los requisitos para obtener el título de Ingeniero Civil en Informática

Concepción, Agosto 2008

Alumno: Fernando Andrés Peña Henríquez

Ricardo Elías Riffo Carrillo

Agradezco…

A Dios, por ser mi guía en la vida y darme la paz y tranquilidad que necesito para mantenerme vivo. “Gracias Señor”

A mis padres, Juan y Mireya, y a mi hermana Carolina, porque gracias a su gran apoyo y esfuerzo pude llegar donde estoy ahora, por su comprensión, paciencia y amor, porque confiaron en mí y me dieron la oportunidad de estudiar para llegar a ser alguien en la vida. “Papá, Mamá, Carolina, los amo mucho.”

A mi polola Paula y su familia, Don Gregorio, Señora Silvia, Mauricio y Rodrigo, por apoyarme siempre. Mi amor gracias por darme ese apoyo y esa convicción de que las cosas aunque a veces no salen como esperamos, al final siempre encontramos la forma de finalizar con éxito lo que empezamos, por tu amor incondicional y por estar ahí siempre ahí. “Paula, mi amor, Te amo mucho.”

A nuestro profesor, Pedro Campos, por aceptar ser nuestro guía en este proyecto, por su disposición, ayuda y paciencia. “Profesor, lo estimo mucho”

A mi compañero en este proyecto, gran amigo y colega, Ricardo, por ser mi amigo y compañero en esta etapa de mi vida. “Ricardo, amigo, te estimo mucho”

A mis amigos, Rodrigo, Gonzalo, Jimmy, Herman y Marcos quienes confiaron en mí y en lo que he hecho durante estos años de estudio, por su apoyo y principalmente por su amistad, que valoro mucho. “Amigos, los quiero mucho”

A todas las personas que no nombro aquí, pero que de alguna u otra forma me han ayudado a llegar a este instante de mi vida.

A todos ellos, les agradezco con todo mi corazón y espero que sigan formando parte de mi vida por siempre, por ello, les dedico este proyecto y deseo lo mejor del mundo para ustedes.

Muchas Gracias.

Fernando.

Agradezco...

En primer lugar a Dios por permitirme llegar hasta nuevamente a estas instancias, darme salud, vida y fuerza para seguir luchando día a día.

A mis padres, Berta y Edgardo, a mi abuela, hermano, y a toda mi familia, a quienes quiero y amo mucho, y son los que me han apoyado desde el primer día con todo su esfuerzo y ayuda incondicional y me han otorgado la dicha que lamentablemente no todos tienen, estudiar y ser una persona con principios y valores.

A mi novia Marjorie, a quien amo infinito, por otorgarme su apoyo, amor incondicional y comprensión, a Michael, Don Carlos y Señora Adelina, por darme fuerza y ánimo en aquellos momentos difíciles que tuve durante el transcurso de la carrera y, en los cuales he encontrado una segunda familia.

A mi amigo, compañero y colega Fernando por acompañarme en ésta segunda etapa universitaria. A mis amigos, Susana y Herman, quienes siempre me otorgaron su ayuda, y con los cuales he tenido la suerte de compartir en mi vida y, espero seguir haciéndolo. Gracias por aceptarme tal como soy.

Al profesor Pedro Campos, quien aceptó ser nuestro profesor guía en este proyecto y nos proporcionó siempre buena disposición y ánimo de colaborar.

A ellos y a todos quienes hicieron posible esto, les dedico este Proyecto y solo me resta decir...
Muchas gracias

Ricardo

RESUMEN

El presente proyecto continua con el trabajo realizado por alumnas tesistas de la carrera de Ingeniería Civil Informática en el año 2007 (Pérez y Romero, 2007), el cual tuvo como objetivo la implementación de un sistema de recomendación para guiar a los estudiantes de las Carreras de Informática de la Facultad de Ciencias Empresariales de la Universidad del Bío-Bío en la búsqueda de material educativo.

Para llevar a cabo la investigación se estudiaron diferentes algoritmos generadores de recomendaciones, los cuales permiten realizar la búsqueda de materiales de estudio de forma distinta al que ya posee el mencionado sistema, ya que se requería mejorar la eficiencia y la fiabilidad en la búsqueda y selección del material existente.

Este trabajo comienza con una descripción general del estudio realizado, ayudando a comprender claramente el beneficio de Implementar un algoritmo de Recomendación para el sistema ya existente. Luego se realiza una investigación y análisis de los sistemas de Recomendación definiendo su estructura, diseño, aspectos que se deben considerar, problemas presentes y ejemplos. Describiendo a continuación las técnicas que emplean los sistemas de recomendación, tales como las basadas en contenido y en filtrado colaborativo dentro de las más importantes, señalando ventajas y desventajas presentes en cada una. Como complemento a lo anterior se describen los diferentes tipos de Algoritmos, basados en redes neuronales, basados en vecinos más cercanos, contenido y herramientas slope-one, que permitirán llevar a cabo la selección de los algoritmos que serán implementados e integrados al sistema de recomendación. Posteriormente se especifican las métricas que son utilizadas para realizar la medición de los sistemas y/o algoritmos de recomendación.

Considerando lo anterior, se desarrollaron dos algoritmos de recomendación para apoyar la búsqueda y selección de material educativo, con el fin de beneficiar a todos los alumnos que deseen ser orientados en la selección de cualquier material de estudio de su área u otras áreas de interés.

Finalmente se presentan los resultados obtenidos con los algoritmos sobre un conjunto de datos de prueba, los datos obtenidos permiten concluir que el uso de adaptaciones de algoritmos utilizados exitosamente en el comercio electrónico, tales como la búsqueda de vecinos más cercanos, permiten mejorar ostensiblemente la calidad de las recomendaciones generadas por el sistema.

INDICE DE CONTENIDOS

INDICE DE FIGURAS, TABLAS Y GRÁFICOS

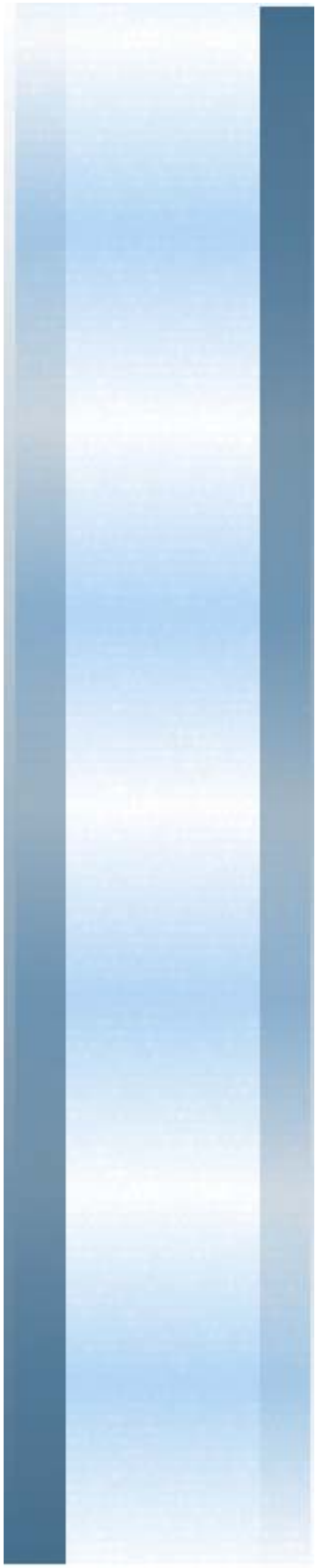
Figuras

Tablas

Gráficos

LISTADO DE ABREVIATURAS UTILIZADAS

- **J2EE:** Java 2 Platform, Enterprise Edition
- **RI:** Recuperación de la Información
- **SR:** Sistemas de Recomendación o Sistemas Recomendadotes
- **CDs:** Compact Disk
- **USA:** United States of America.
- **TC:** Text Categorization
- **CBR:** Case Based Reasoning
- **TV:** Televisión
- **BNB:** Branch-and-Bound
- **MAE:** Error Medio Absoluto
- **ROC:** Receiver Operating Characteristic.
- **AUC:** área bajo la curva ROC
- **CU:** Casos de Uso
- **EJB:** Enterprise JavaBeans.
- **MVC:** Modelo-Vista-Controlador



CAPITULO I:

"INTRODUCCION"

1.1 Introducción al proyecto:

Las universidades de todo el mundo tienen día a día la gran tarea de mejorar sus métodos de enseñanza y orientar de forma correcta el desarrollo estudiantil de sus alumnos, durante este proceso es necesario contar con asesoría y retroalimentación de contenido estudiantil no sólo de profesores hacia los alumnos sino que también de alumno a alumno, ya sea de forma física o virtual.

Hoy en día la forma más conocida y más utilizada de asesoría en la búsqueda de material bibliográfico dentro de las universidades es la física, generalmente con recomendaciones hechas por parte de los profesores hacia los alumnos. Esto hace indispensable buscar alguna forma de virtualizar la asesoría y retroalimentación para proporcionar a los alumnos una forma eficaz y eficiente en la búsqueda de material educativo para el estudio de temas específicos y generales del área de interés (Geyer-Schulz et. al., 2001).Los sistemas de recomendación ayudan a encontrar nuevo contenido, productos o servicios que pueden gustar o que se pueden necesitar. En general, los sistemas de recomendación se podrían definir como un fiel compañero que entiende gustos y ayuda en la selección de nuevos videos, nuevas canciones, nuevos juegos, nuevos anuncios, nuevos productos, nuevos servicios, etc. También permiten, siempre que así se quiera, compartir gustos con otras personas e incluso ayudar a encontrar otras personas con gustos similares.

En éste contexto los sistemas de recomendación, de acuerdo con (Wang, 1998), tienen como principal tarea seleccionar ciertos tipos de objeto de acuerdo a los requerimientos del usuario, dado que esos objetos están almacenados y caracterizados en base a sus atributos. Así, utilizando un sistema de recomendación, estudiantes interesados en una materia, por ejemplo, podrían ver facilitada la búsqueda de documentos, libros, e informes, por mencionar algunos, sobre dicha materia.

Cabe destacar, que los sistemas de recomendación van unidos a la implementación de algoritmos de recomendación, que se ve masificada en la industria del comercio electrónico, existiendo diversas técnicas para el diseño de los mismos (ver por ejemplo Schafer et. al., 2001; Sarwar et. al., 2001; Deshpande y Karypis, 2004). Estos algoritmos son usados con bastante éxito en diversos sistemas de empresas de comercio electrónico, tal

es el caso de Amazon.com (Linden et. al., 2003); Sin embargo existe una cierta carencia en el estudio y la implementación de éstos algoritmos en el área educacional.

En la problemática ya mencionada se presenta el dilema de cómo virtualizar la asesoría en el ámbito educacional utilizando el concepto de sistema de recomendación adoptado en el comercio electrónico, así como también qué arquitectura utilizar, qué lenguaje, qué enfoque, etc. Para ello existe una arquitectura denominada J2EE, que define de forma flexible, escalable y portable el desarrollo de cualquier aplicación de este tipo.

Finalmente, teniendo clara la problemática y conociendo los estándares a seguir para dar una solución al problema, se pretende con este trabajo estudiar los sistemas de recomendación, las técnicas empleadas, los tipos de algoritmos con los que se puede llevar a cabo el desarrollo un sistema de recomendación, las métricas empleadas para la evaluación de éstos y la implementación propiamente tal de algoritmos de recomendación para incorporarlos al sistema ya existente, cubriendo todas las necesidades para orientar, de forma correcta y eficiente, al estudiante o docente en la selección de material educativo.

Para lograr todo lo mencionado anteriormente se definen los siguientes objetivos generales y específicos:

1.2 Objetivo general:

Estudiar e implementar algoritmos de recomendación para apoyar la selección de material educativo de un sistema de búsqueda de recursos educacionales.

1.3 Objetivos específicos:

- Estudiar la especificación J2EE como base para el desarrollo basado en componentes.

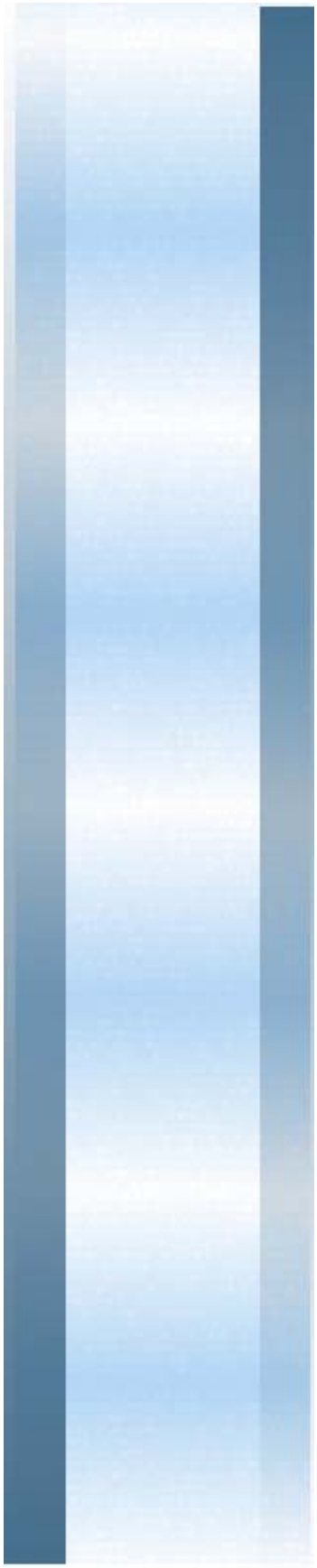
- Estudiar los distintos tipos de recomendación existentes

- Estudiar y analizar los distintos algoritmos de recomendación existentes

- Seleccionar y adaptar uno o varios algoritmos de recomendación estudiados con anterioridad

- Implementar un algoritmo de recomendación de material educativo utilizando uno o varios de los algoritmos antes seleccionados

- Integrar el algoritmo de recomendación creado al sistema de recomendación actual.



CAPITULO II:

"SISTEMAS DE RECOMENDACION"

2.1 Los Sistemas de Recomendación

A menudo es necesario seleccionar una entre varias alternativas sin tener un conocimiento exacto de cada una de ellas. En estas situaciones, la decisión final puede depender de las recomendaciones de otras personas. En Recuperación de la Información (RI) los Sistemas de Recomendación (SR) son herramientas cuyo objetivo es asistir a los usuarios en sus procesos de búsqueda de información, ayudando a filtrar los ítems de información recuperados, usando recomendaciones propuestas sobre esos ítems. Dichas recomendaciones se generan a partir de las opiniones proporcionadas por otros usuarios sobre ciertos ítems, tales como documentos, libros e informes en búsquedas previas o bien a partir de las preferencias del usuario objeto de la recomendación (en adelante, *usuario activo*), dando lugar a los dos grandes grupos de SR (Yager, 2003), *los colaborativos* y *los no colaborativos o basados en contenidos*. El uso de estos sistemas se está poniendo cada vez más de moda debido a su utilidad para evaluar y filtrar la gran cantidad de información disponible en la Web para asistir a los usuarios en sus procesos de búsqueda y recuperación de información (Herrera-Viedma, 2003).

La idea principal de este capítulo es presentar un estudio sobre los SR para la RI, describiendo los aspectos más significativos de su diseño y problemas fundamentales encontrados a la hora de diseñar un sistema de este tipo. Se analizarán los elementos que intervienen en el esquema de funcionamiento de un SR y se usarán como criterios de clasificación.

2.2 Historia de los Sistemas de Recomendación

Los sistemas de recomendación datan de principios de los años 90 y en un comienzo eran conocidos tan sólo como filtros colaborativos. El término fue acuñado en 1992 para un sistema de filtrado de correo electrónico no automatizado. En 1994 se desarrolló el primer "workshop" en Berkeley donde se vio la utilidad en diversas áreas de los primeros algoritmos simples de éste tipo (Foltz y Dumais, 1992) (Resnick et al., 1994) (Stodolsky, 1990). También se identificaron algunas cuestiones importantes para el desarrollo de éstos algoritmos:

- Escalabilidad
- Viabilidad económica
- Puntuaciones implícitas y explicitas

Uno de los grupos de investigación pioneros en el desarrollo del filtrado colaborativo fue el proyecto GroupLens de la Universidad de Minesota (Galán, 2007) que aún permanece muy activo y que ha proporcionado una gran parte de la base algorítmica de muchos sistemas de recomendación. Fueron los primeros en introducir el filtro colaborativo automático (Galán, 2007), usando un algoritmo de búsqueda de vecinos para proporcionar predicciones en los grupos de noticias de USENET[1]. De este grupo de investigación partió también la iniciativa empresarial NetPerceptions[2], despejando gran parte de las dudas acerca de la viabilidad económica de estos proyectos. En la actualidad es un campo que se encuentra muy activo y genera un gran número de publicaciones y congresos todos los años. Algunos ejemplos actuales de uso son:

- Recomendaciones comerciales en tiendas on-line. Partiendo de un producto se recomiendan otros productos que han interesado a los usuarios que compraron dicho producto. La Web pionera en este tipo de recomendaciones fue Amazon.com para la recomendación de libros, películas y música, Sleeper (www.pmetric.com) para

[1] Acrónimo de Users Network (red de usuarios), sistema que se encarga de gestionar el flujo de información generado por el envío, recepción y distribución de documentos, entre un grupo de personas, con intereses afines, tales como política, religión, pasatiempos, profesiones etc.
[2] NetPerceptions es la aplicación utilizada para implementar filtrado colaborativo en modelos de negocio en Internet.

libros; RatingZone's QuickPicks (www.ratingzone.com) para libros, películas, música y programas de televisión; Movie Critic (www.moviecritic.com) para películas; Reel (www.reel.com) para películas; Media Unbound (www.mediaunbound.com) para música y CDNow (www.cdnow.com) que realmente es la parte destinada a CDs dentro de Amazon. Los sistemas difieren apreciablemente respecto al número de ítems que le presentan al usuario para calificar, en la escala de calificación, en el uso de géneros o categorías como ayuda para la recomendación y en el tipo de información que se agrega al ítem recomendado (Swearingen y Sinha, 2002).

- Filtrado de noticias. Se construye un perfil que almacena las noticias que un usuario consulta.

- Plataforma experimental, tal es el caso de Movielens (http://movielens.org/login)**,** que es un sitio Web de recomendación de películas desarrollado por GroupLens Research del Department of Computer Science and Engineering at the University of Minnesota (USA). Es un sitio gratuito, utilizado para conducir estudios sobre sistemas de recomendación y filtrado de información social.

- Búsqueda de personas afines en comunidades. En Webs como meneame.net se tienen en cuenta las noticias que cada usuario ha votado para generar una lista de vecinos con similares intereses.

2.3 Definición

Los SR fueron originalmente definidos como sistemas en los cuales los usuarios explicitaban aquellas recomendaciones que serían útiles para ellos en un formulario, y el sistema almacenaba esta información para utilizarla en el momento adecuado y para los usuarios adecuados (Resnick y Varian, 1997). No obstante, el término tiene ahora una connotación mucho más amplia, describiendo cualquier sistema que ofrece recomendaciones individualizadas o bien que tiene el efecto de guiar al usuario de manera personalizada, escogiendo para él los servicios más útiles entre una colección que puede llegar a ser bastante grande. Estos sistemas tienen un evidente atractivo en un marco donde el usuario tiene muchas más opciones de las que realmente puede examinar para comprobar si le interesan o no (Blanco, 2007).

Un sistema de recomendación es una tecnología de filtrado de información que permite resolver dos problemas específicos: (i) el problema de predicción, en el cual se predice si a un usuario en particular le gustará un ítem en particular; y (ii) el problema de recomendación, en el cual se determina un conjunto de n ítems a recomendar a un usuario a partir de sus preferencias personales (Sarwar et al., 2001).En ambos casos se utiliza el concepto de filtrado colaborativo, puesto que los anteriores usuarios del sistema colaboran con sus preferencias para predecir las preferencias desconocidas del nuevo usuario, y cada colaboración adicional mejora el rendimiento del sistema, es decir, los Sistemas de Recomendación proporcionan sugerencias personalizadas acerca de ítems o productos que un usuario puede llegar a encontrar interesantes. Para esto, el sistema debe estar en capacidad de hacer predicciones sobre la preferencia (o no) de un usuario sobre cierto ítem. El proceso básico es hacer un proceso de concordancia entre la información que se tiene acerca del perfil del usuario actual y los perfiles de los otros usuarios que se encuentran ya almacenados y de cuyas preferencias se tiene conocimiento, a menudo esto se conoce como "filtrado colaborativo de vecindad más cercana", como se muestra en la figura 2.1 (Vélez y Santos, 2006); En general, los Sistemas de Recomendación son un intento por automatizar las conocidas recomendaciones tipo "boca a boca" en que las personas se recomiendan productos unos a otros (Vélez y Santos, 2006). En estos sistemas los usuarios son individuos y a su vez son miembros de un grupo (que es conocido como vecindad), y

modelan el comportamiento del usuario y en base a él aplican los diversos mecanismos para facilitarle la búsqueda de los objetos que le son de interés.

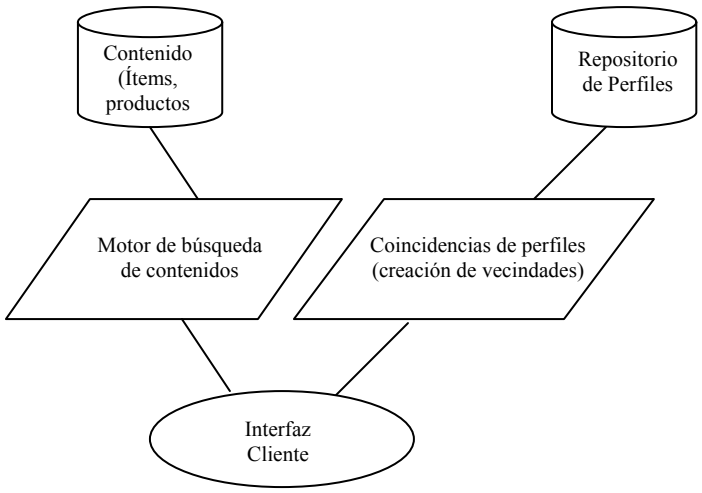


Figura 2.1: Esquema de Funcionamiento Básico de un Sistema de Recomendación (Vélez y Santos, 2006)

La diferencia entre un sistema de recomendaciones y un motor de búsqueda o un sistema de recuperación de información estriba en que la información que ofrece un sistema de recomendaciones se intenta que sea "individualizada" y además, "interesante y útil" (Blanco, 2007).

Los sistemas de recomendación son una alternativa al proceso social de recomendación, es decir, es una opción al proceso de consultar las opiniones de otras personas. Son guías que orientan para tomar decisiones relacionadas con las preferencias del usuario. Estos sistemas modelan el comportamiento del usuario y, en base a él, aplican los diversos mecanismos para facilitarle la búsqueda de los objetos que le son de interés.

2.4 Aspectos a Considerar en el Diseño de un Sistema de Recomendación

Un Sistema de Recomendación está asociado con un conjunto de ítems I = { i_1...,i_n } y su objetivo es recomendar a los usuarios ítems de I que les puedan ser de interés (Yager, 2003). Por ejemplo, se podría diseñar un SR para la recomendación de películas; de hecho en el transcurso de la investigación se han encontrado numerosos ejemplos de SR de películas, tales como Film-Conseil, MovieFinder, Reel o MetaLens entre otros. En el caso de un SR para la recuperación de documentos los ítems serían los documentos almacenados.

La implementación de técnicas para el desarrollo de los SR está íntimamente relacionada con el tipo de información que se vaya a utilizar. Una primera fuente de información a tener en cuenta es el tipo de ítems con los que se va a trabajar. Habrá situaciones en las que únicamente se conozca un identificador de cada ítem. Por ejemplo, en el caso de la recomendación de películas se suele conocer únicamente el título. En otras situaciones, se dispone de más información sobre los ítems, a través de una serie de atributos. En el caso de la recomendación de películas, podrían ser el año en que se hizo la película, el género, el director, protagonistas, etc. En el caso de recuperación de documentos, la información con la que se puede contar son los términos índice usados en su representación. En general, cuanto más sofisticada es la representación de los ítems mejor se puede desarrollar la actividad de los SR.

Las recomendaciones se pueden definir como el acto habitual que todos practican al recurrir a las opiniones de conocidos expertos cuando se tiene que tomar una decisión para adquirir algo sin tener la suficiente información para ello, en este caso, la búsqueda de algún material específico de estudio. Generalmente, las recomendaciones son un listado de ítems que le interesan al usuario. Estas recomendaciones se generan, básicamente, solicitando al usuario, ya sea de forma implícita o explícita, su necesidad de información y sus preferencias, cruzando estas últimas con las de otros usuarios similares.

Algunos aspectos que se deben considerar sobre las recomendaciones en el diseño de SR son:

- Representación de las recomendaciones: Los contenidos de una evaluación pueden venir dados por un único bit (recomendado o no) o por comentarios de texto sin estructurar.

- Expresión de las recomendaciones: Las recomendaciones pueden ser introducidas de forma explícita o bien de forma implícita.

- Aspectos de identificación de la fuente: Las recomendaciones pueden realizarse de forma anónima, identificando la fuente, o bien usando un pseudónimo.

- Forma de agregar las evaluaciones: Se refiere a cómo se van a ir agregando las evaluaciones disponibles sobre los ítems de cara a generar las recomendaciones. Más adelante se entrará en más detalle sobre este aspecto que es de suma importancia de cara a clasificar los SR.

- Uso de las recomendaciones: Las recomendaciones se pueden usar de distintas formas. Por ejemplo, se podrían mostrar los ítems en forma de lista ordenada según las recomendaciones de cada uno, o a la hora de visualizar los ítems que se muestre también su recomendación.

2.5 Estructura de los Sistemas de Recomendación

A continuación se presentan los elementos fundamentales que intervienen en el esquema de funcionamiento de un SR. Dichos elementos se pueden usar como criterios de clasificación y son los siguientes:

- Las entradas / salidas del proceso de generación de la recomendación
- El método usado para generar las recomendaciones (Proceso de Recomendación)
- El grado de personalización.

2.6 Entradas / Salidas

Los SR para generar recomendaciones, usan las entradas del usuario activo, pero también información sobre los ítems o información del resto de usuarios del sistema, que actúan como colaboradores. En este sentido, la realimentación por parte de los usuarios es muy importante de cara a albergar una información más completa ante futuros procesos de generación de recomendaciones. La figura 2.2 refleja el proceso de generación de las recomendaciones.

Para poder realizar una recomendación a un usuario, es necesario conocer algún tipo de información sobre sus preferencias. Además, dependiendo del tipo de sistema también se necesita información sobre los ítems a recomendar o información reunida sobre el resto de usuarios del sistema (comunidad de usuarios o colaboradores). Esta información que se necesita para realizar las recomendaciones constituye la entrada o entradas del sistema.

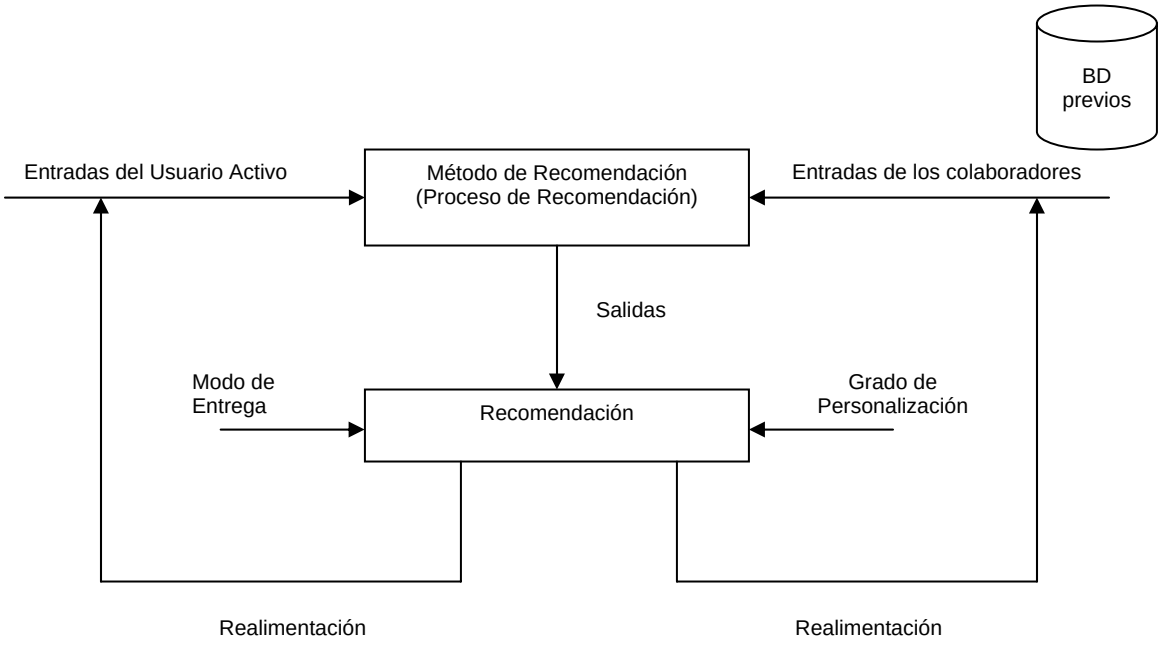


Figura 2.2: Esquema del proceso de generación de una recomendación

La información sobre los usuarios puede venir dada de dos formas (Yager, 2003) que no tienen porque ser mutuamente exclusivas: por extensión o intencionalmente. Por *extensión* se refiere a información que se tenga sobre las experiencias pasadas del usuario con respecto a los ítems encontrados. Es lo que también se conoce como navegación implícita pues el usuario no es consciente de estos seguimientos. Por información expresada *intencionalmente* se entiende alguna especificación de los ítems deseados por los usuarios. También se le llama navegación explícita y consiste en que el usuario expresa intencionalmente (de forma explícita) al SR información sobre sus preferencias.

La salida del sistema está constituida por las recomendaciones generadas por el sistema, que variarán dependiendo del tipo, cantidad y formato de la información proporcionada al usuario.

Algunas de las formas más comunes de representar la salida son las siguientes:

- *Sugerencia o lista de sugerencias* al usuario de una serie de ítems.

- Presentar al usuario *predicciones* del grado de satisfacción que se asignará al ítem concreto. Estas estimaciones pueden ser presentadas como personalizadas al usuario o como estimaciones generales del conjunto de colaboradores.

- Cuando la comunidad de usuarios es pequeña o se conocen bien los miembros de dicha comunidad, podría ser útil *visualizar las valoraciones individuales* de los miembros que permitiría al usuario activo obtener sus propias conclusiones sobre la efectividad de una recomendación.

Independientemente de estos formatos de salida, puede resultar muy interesante incluir una breve *descripción o explicación* sobre el ítem recomendado a modo de justificación del por qué de dicha recomendación.

2.7 Método de Generación de Recomendaciones

En esta sección se describen una serie de métodos de recomendación que se usan habitualmente en los SR, pero se debe tener en cuenta que no son mutuamente exclusivos entre sí, sino complementarios, es decir, que en un mismo SR se pueden usar uno o varios de estos métodos. En principio los tres métodos más simples son:

i) Recuperación pura o recomendación nula, en la que el sistema ofrece a los usuarios una interfaz de búsqueda a través de la cual pueden realizar consultas a una base de datos de ítems. Se trata, pues, de un sistema de búsqueda por lo que técnicamente no es un método de recomendación, aunque ante los usuarios aparece como tal.

ii) Otros sistemas usan recomendaciones seleccionadas manualmente por expertos, como por ejemplo editores, artistas o críticos en el caso de recomendaciones de películas o CDs de música. Los expertos identifican ítems basándose en sus propias preferencias, intereses u objetivos, y crean una lista de ítems que esté disponible para todos los usuarios del sistema. A menudo acompañan estas recomendaciones de comentarios de texto que puedan ayudar a los usuarios a evaluar y entender la recomendación.

iii) En otros casos, los sistemas ofrecen resúmenes estadísticos calculados en función de las opiniones del conjunto de usuarios, por lo que tampoco son personalizados. Por ejemplo, se podrían tener en cuenta el porcentaje de usuarios a los que ha satisfecho o han comprado un artículo, número de usuarios que recomiendan un ítem, o una evaluación media de todos los usuarios con respecto al ítem.

Estos métodos de generación de recomendaciones, por su simplicidad no son considerados propiamente métodos de generar recomendaciones en la literatura. Para generar las recomendaciones hay dos posibilidades comúnmente aceptadas que dan lugar a dos grandes grupos de Sistemas de Recomendación: los SR colaborativos y los no colaborativos o basados en contenidos.

- Los SR *no colaborativos* realizan las recomendaciones usando únicamente las preferencias del usuario activo y los atributos de los ítems a recomendar. Estos sistemas usan correlaciones entre ítems para identificar ítems asociados frecuentemente a un ítem por el que el usuario ha mostrado interés y, por tanto, recomendarle dichos ítems al usuario. Como ejemplo de esta idea, suponga que se tiene un SR de libros y se tienen dos libros 'La clave está en Rebeca' y 'La isla de las tormentas', ambos del mismo autor Ken Follet, pero que además ambos son de intriga y están ambientados en la 2ª Guerra Mundial; por tanto, se podrían considerar en cierto sentido similares, es decir, existe una alta correlación entre ambos. Por esa razón, si un usuario del sistema está interesado en 'La clave está en Rebeca', por lo que se podría recomendar también la lectura de 'La isla de las tormentas'.

- Por otro lado, la mayoría de SR existentes son *colaborativos* (Bafoutsou y Mentzas, 2002). Un SR se dice colaborativo si usa la información conocida sobre las preferencias de otros usuarios para realizar la recomendación al usuario que la precise. Los SR colaborativos identifican usuarios cuyas preferencias sean similares a las de otros usuarios dados y recomiendan a los primeros los elementos que hayan satisfecho a los otros; de esta forma, si dos usuarios *U1* y *U2*, comparten el mismo sistema de valores (tienen las mismas preferencias) y al usuario *U1* le ha satisfecho un ítem *i*, probablemente este ítem también satisfaga al usuario *U2* por lo que debería ser recomendado. Por ello, en estos SR la definición de medidas de similitud entre preferencias es un punto crítico. La situación puede ser representada como una matriz de usuarios e ítems, donde cada celda representa la valoración de un usuario con respecto a un ítem concreto (tabla 2.1). Así visto, el problema consiste en predecir valores para las celdas que estén vacías. En la tabla 2.2 se puede ver un ejemplo de matriz que representa valoraciones de usuarios con respecto a una serie de lenguajes de programación.

Item / Usuario	x_1	x_2	...	x_b	...	x_{n-1}	x_n
u_1							
u_2							
...							
u_a				?			
...							
u_{m-1}							
u_m							

Tabla 2.1. Matriz de Valoraciones Usuario/Ítem

	J2EE	**Perl**	**.NET**	**PHP**
Fernando	5	2	4	5
Ricardo	2	5		3
Carlos	3	2	4	2
Susana	?	1	4	5

Tabla 2.2. Ejemplo de Matriz de Valoraciones

Los algoritmos que se suelen usar para implementar las técnicas de filtrado colaborativo se llaman métodos basados en vecindad. Funcionan seleccionando un conjunto apropiado de usuarios, según la similitud de los mismos con respecto al usuario activo, y usan las valoraciones de dichos usuarios para generar la valoración del usuario activo. Concretamente, los tres pasos a seguir para realizar esto son los siguientes:

i) Medir la similitud de todos los usuarios con respecto al usuario activo.

ii) Seleccionar un subconjunto de usuarios cuyas valoraciones se van a usar y, por tanto, tendrán influencia en la generación de la predicción para el usuario activo.

iii) Normalizar las puntuaciones de los distintos usuarios y calcular una predicción a partir de algún tipo de combinación pesada de las puntuaciones asignadas al ítem por los usuarios seleccionados en el paso anterior.

Como ejemplo, consideremos de nuevo la tabla 2.2; se puede predecir que a Susana le gustará el lenguaje de programación 'J2EE'. Se puede observar que Fernando es el que tiene preferencias más parecidas a Susana, puesto que ambos tienen unas valoraciones muy similares de los lenguajes de programación que ya han visto. Por tanto, la valoración de Fernando sobre el lenguaje de Programación 'J2EE' tendrá gran influencia en la predicción que se le haga a Susana sobre dicho lenguaje. Por el contrario, Ricardo y Carlos tienen opiniones más dispares con respecto a Susana, por lo que tendrán una influencia mucho menor en las recomendaciones que se hagan a dicho usuario.

Los SR colaborativos tienen importantes ventajas con respecto a los no colaborativos (Herrera-Viedma et al., 2004):

- Dan un mayor soporte para el filtrado de ítems cuyo contenido no es fácil de analizar por procesos automatizados,
- la posibilidad de filtrar ítems basándose en su calidad o preferencias,
- y la posibilidad de realizar recomendaciones válidas, que no se esperaban, lo cual puede resultar de gran utilidad.

Sin embargo, también presentan inconvenientes, como por ejemplo que no trabajan bien a la hora de filtrar información para necesidades de contenido específicas, o cuando el número de usuarios es bajo. Por ello, en muchas ocasiones la mejor opción es adoptar un enfoque híbrido entre colaborativo y no colaborativo y, de esta forma, disfrutar de las ventajas de ambos.

2.8 Grado de Personalización

Los SR también se pueden clasificar según su grado de personalización, como se señala a continuación:

i) Cuando los SR proporcionan las mismas recomendaciones a todos los usuarios, son clasificados en este ámbito como no personalizados. Dichas recomendaciones estarán basadas en selecciones manuales, resúmenes estadísticos u otras técnicas similares.

ii) Los SR que tienen en cuenta la información actual del usuario objeto de las recomendaciones, proporcionan personalización efímera, puesto que las recomendaciones son repuestas al comportamiento y acciones del usuario en su sesión actual de navegación.

iii) Los SR que ofrecen el mayor grado de personalización son los que usan personalización persistente ofreciendo recomendaciones distintas para distintos usuarios, incluso cuando estén buscando el mismo ítem. Estos sistemas están basados en el perfil de los usuarios, por lo que hacen uso de métodos de filtrado colaborativo, filtrado basado en contenidos o correlaciones entre ítems.

2.9 Problemas Asociados a los Sistemas de Recomendación

En los SR aparecen una serie de problemas que habrá que considerar en su diseño (Herrera-Viedma et al., 2004) y que se describen a continuación:

- Primero, una vez que se ha establecido un perfil de intereses, es fácil considerar las evaluaciones suministradas por otros. Sin embargo, en algunos casos se hace necesario recurrir a incentivos para la provisión de recomendaciones, puesto que los usuarios no suelen estar muy dispuestos a colaborar proporcionando información personal sobre sus preferencias. Estos incentivos podrían consistir en forzar al usuario a introducir sus preferencias a cambio de recibir recomendaciones, o bien asignarle otro tipo de compensaciones.

- Un segundo problema a solucionar es que si cualquiera puede recomendar, los propietarios de determinados productos podrían generar recomendaciones positivas de los mismos y negativas de otros. En los SR también habrá que tener en cuenta aspectos de privacidad y debido a que algunas personas no quieren que se conozcan sus hábitos o preferencias, se permite la participación anónima o bajo un pseudónimo.

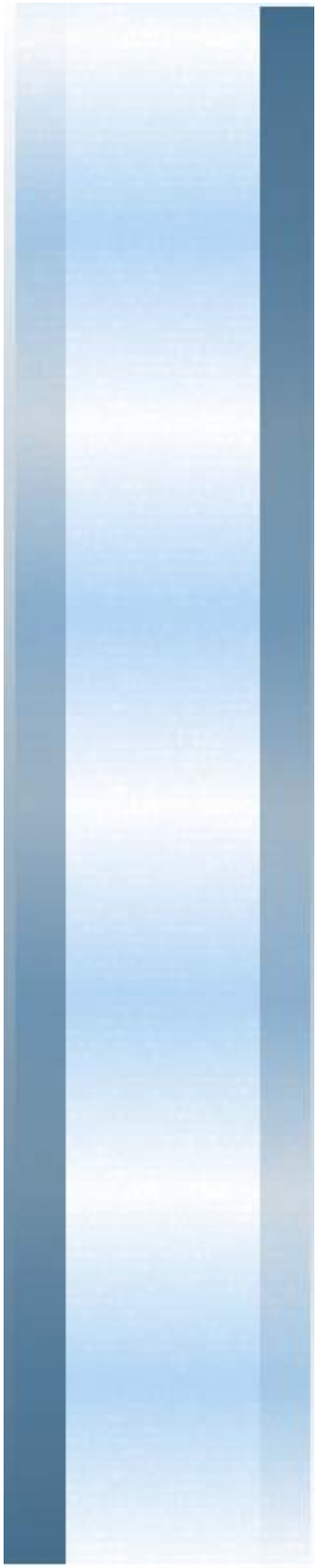
- Como tercer problema, se puede destacar quizás como el más importante el problema de la falta de datos cuando se empieza a utilizar el sistema, conocido en la literatura como el problema del ramp-up (Konstan et al., 1999). Este término realmente se refiere a dos problemas diferentes, aunque relacionados:
    - Nuevos usuarios. Debido a que las recomendaciones son el resultado de la comparación entre el usuario activo y otros usuarios, basándose solamente en anteriores interacciones de los usuarios con el sistema, un usuario del que se disponen pocos datos resulta difícil de clasificar, y por tanto, de recomendar.
    - Nuevos servicios. Análogamente, un nuevo servicio del que no se disponen todavía suficientes datos de accesos de los usuarios a éste, puede ser complicado de recomendar. Este problema se manifiesta en mayor medida, por ejemplo, en un campo como el de los periódicos digitales donde constantemente aparecen nuevas noticias pero solamente unas pocas son accedidas.

- Un último problema es dotar a los SR de mejores técnicas de representación de las preferencias o recomendaciones de los usuarios que permitan captar verdaderamente su concepto del objeto recomendado y mejorar la interacción SR-usuario.

2.10 Ejemplos de Sistemas de Recomendación

Algunos ejemplos de Sistemas de Recomendación en la Web son los siguientes:

Aggregate Knowledge (recomendaciones y descubrimientos)	http://www.aggregateknowledge.com/
Baynote (recomendación del servicio Web)	http://www.baynote.com/
ConfigWorks (Venta de Soluciones Interactivas)	http://www.configworks.com/
Criteo (Tecnología de Recomendación)	http://www.criteo.com/
Criticker (motor de recomendación de películas)	http://www.criticker.com/
Daily Me (sistema de recomendación de noticias (hipotética))	http://www.dailyme.com/
Foodio54 (Recomendación de servicio de restaurante)	http://foodio54.com/
FreshNotes (motor de recomendación)	http://www.freshnotes.com/
Heeii (recommendation plugin)	http://www.heeii.com/
iGoDigital (motor de recomendación)	http://www.igodigital.com/
iLike (servicio de música)	http://www.ilike.com/
Last.fm (servicio de música)	http://www.lastfm.es/
MyStrands (desarrollador de tecnologías de recomendación social)	http://www.strands.com/
Pandora (servicio de música)	http://www.pandora.com/
Peerius (servicio de música)	http://www.peerius.com/
prudsys RE (sistema de recomendación)	http://www.prudsys.com/Software/ Komponenten/RecommendationEngine/
Photoree (sistema de recomendación de imágenes)	http://www.photoree.com/
Slacker (servicio de música)	http://www.slacker.com/
StumbleUpon (Servicio Web de Descubrimientos)	http://www.stumbleupon.com/



CAPITULO III:

"TECNICAS DE RECOMENDACION"

3.1 Las Técnicas de Recomendación

Teniendo en cuenta las partes descritas en el capítulo anterior, se puede hablar de cinco diferentes técnicas de recomendación:

- recomendación colaborativa
- basada en contenido
- demográfica
- basada en la utilidad
- basada en el conocimiento

Para explicar en rasgos generales cada una de estas técnicas, se va a suponer que S es el conjunto de servicios que se pueden recomendar, U el conjunto de usuarios cuyas preferencias son conocidas, u el usuario para el cual tiene que generarse la recomendación, y por último, s es un servicio para el cual se desea saber el grado de preferencia del usuario u.

En la tabla 3.1 se resume el funcionamiento de las cinco técnicas de recomendación consideradas. En particular, en "background" se muestra la información almacenada por la técnica en cuestión, y que es lo que posteriormente utilizará para hacer la recomendación a un usuario determinado tras un proceso de extracción de conocimiento, comparación, etc. En "Dato entrada" se muestra el dato del usuario que va a ser recomendado que necesita la técnica para poder ofrecer la recomendación, y en "Procesado" el procesado llevado a cabo por la técnica para, finalmente, ofrecer la recomendación.

Técnica	Background	Dato entrada	Procesado
Colaborativa	Selecciones realizadas por U en S.	Selecciones realizadas por U en S.	Identificar aquellos usuarios en U que son similares a u y extrapolar a partir de sus selecciones en S.
Basada en contenido	Características de los servicios en S	Selecciones realizadas por U en S.	Generar un clasificador que relaciona el comportamiento de u y lo usa en S.
Demográfica	Inform. Demográfica sobre U y sus selecciones en S.	Inform. Demográfica de u.	Identificar usuarios demográficamente parecidos a u, y extrapolar a partir de sus selecciones en S.
Basada en utilidad	Características de los servicios en S.	Función de utilidad sobre los servicios en S que describen las preferencias en u.	Aplicar la función a los servicios determinando la significancia de s
Basada en conocimiento	Características de los servicios en S, y conocimiento de cómo los servicios se corresponden con las necesidades del usuario.	Descripción de las necesidades o intereses de u.	Inferir una correspondencia entre S y las necesidades de u.

Tabla 3.1: Comparativa de cinco diferentes técnicas de recomendación en cuanto a la información que almacenan (background), la información de entrada que necesitan para ofrecer una recomendación (Dato entrada) y el procesado llevado a cabo por la técnica en cuestión.

3.2 Técnica Basada en Contenido

En los sistemas de recomendaciones basados en contenido, los objetos de interés se definen en función de las características asociadas a éstos. Por ejemplo, los sistemas de recomendación de texto como el sistema implementado en el grupo de noticias NewsWeeder, utilizan las palabras de sus textos como características (Lang, 1995).Esta técnica consiste en recomendar a un usuario activo productos que son *similares* a los que le han gustado en el pasado. Para poder comparar las preferencias del usuario activo frente a los productos objetivo, los métodos basados en contenido consideran sus principales características o atributos. Si bien, por su propia naturaleza, es una estrategia muy precisa, son varias las limitaciones reconocidas en el filtrado basado en contenido:

- Por una parte, esta técnica suele recomendar productos excesivamente parecidos a los que el usuario activo ya conoce, e incluso, demasiado similares entre sí (Smyth y McClave, 2001). En este último caso, si alguno de los productos sugeridos no interesa al usuario su confianza en el sistema se verá comprometida (debido a que el resto de las sugerencias ofrecidas también se parecen a dicho producto).

  Esta limitación, acuñada en la literatura bajo el término de *sobreespecialización*, tiene su origen en las métricas de similitud adoptadas en los enfoques convencionales basados en contenido. Tales métricas únicamente detectan similitud entre dos productos cuando se advierte cierto solapamiento entre sus características o atributos, de ahí la falta de diversidad de las recomendaciones sugeridas.

El problema de la *sobreespecialización* en los sistemas de recomendación basados en contenido ha sido combatido en algunos enfoques propuestos en la literatura (Smyth, 2000). Bajo una filosofía común, estas propuestas tratan de diversificar las recomendaciones ofrecidas a los usuarios sin necesidad de comprometer la personalización. Su objetivo es, por tanto, alcanzar un equilibrio justo entre dos parámetros: (i) la similitud existente entre las preferencias del usuario y las sugerencias ofrecidas, y (ii) la diversidad relativa entre cada producto sugerido y los restantes incluidos en la recomendación final. En consecuencia, tales enfoques seleccionan un conjunto de productos que no sólo son parecidos a los que interesan al usuario activo, sino que además son diferentes entre sí, evitando así la posible pérdida de confianza del usuario en el sistema, mencionada al comienzo de la sección.

- De lo anterior se desprende que este tipo de enfoques combaten la *sobreespecialización* desde un único frente: aseguran que los productos sugeridos son diferentes entre sí, pero no contemplan la posibilidad de que éstos aún pueden ser demasiado similares a las preferencias del usuario activo. En este escenario, el usuario podría llegar a cuestionar la utilidad de un sistema de personalización que únicamente sugiere productos muy parecidos a aquellos que él ya conoce (Blanco, 2007).

- La segunda de las limitaciones de los métodos basados en contenido tiene su origen en la especificación de los atributos de los productos considerados, una tarea costosa que, en algunos casos, incluso requiere la participación de un experto de conocimiento, capaz de describir el dominio del sistema de forma precisa.

- Por último, destacar el llamado *new user ramp-up (Burke, 2002)*, un problema relacionado con la llegada de nuevos usuarios al sistema. En este escenario, el recomendador dispone, típicamente, de muy poco conocimiento sobre sus preferencias personales, de ahí la baja precisión de las recomendaciones ofrecidas.

Una vez identificadas las limitaciones de las que adolecen los sistemas basados en contenido actuales, el énfasis se centra en las técnicas que se utilizan para comparar las preferencias de los usuarios activos frente a los productos objetivos.

3.2.1 Comparación sintáctica basada en palabras claves

Este método consiste en comparar sintácticamente las características definidas en el perfil del usuario activo, frente a los atributos de los productos objetivo considerados. Tiene las limitaciones propias de los enfoques que no razonan sobre el significado de los términos a través de su semántica (palabras con múltiples significados, sinónimos, etc.).

A pesar de ello, esta técnica se ha aplicado exitosamente en diferentes herramientas de personalización Web, como los buscadores de Internet convencionales, y los sistemas ifWeb[3] y SiteIf[4] (Blanco, 2007).

[3] Prototipo de usuario basados en modelos de Agentes inteligentes capaza de soportar a recuperación y el filtrado de documentos teniendo en cuenta las necesidades específicas de información del usuario.
[4] SiteIF es un agente personal que toma en cuenta la navegación del usuario por la Web, tomando en cuenta las páginas visitadas

3.2.2 Categorización de texto (TC - Text Categorization)

El área de la categorización o clasificación textual surge con el propósito de salvar las limitaciones de los enfoques sintácticos convencionales. Sus técnicas se basan en aprender múltiples características a partir de descripciones textuales, y aplicarlas en la construcción de clasificadores eficientes. Entre las aplicaciones de estos últimos, se pueden destacar el filtrado de información, el agrupamiento de documentos relacionados entre sí, la clasificación de los mismos en categorías predefinidas, etc.

Tales capacidades han sido aprovechadas en algunos de los sistemas recomendadores basados en contenido propuestos en la literatura. Estos sistemas procesan las descripciones textuales de las preferencias de los usuarios, identificando las categorías genéricas a las que pertenecen. Este modelo únicamente recomienda un producto objetivo al usuario, cuando el proceso de clasificación determina que está incluido en sus categorías de interés (Blanco, 2007), tal es el caso del sistema de recomendación de películas INTIMATE (Mak, 2003).

3.2.3 Similitud basada en coseno

Esta técnica .tradicionalmente vinculada al campo de la recuperación de información se utiliza en sistemas que representan las preferencias de los usuarios (y también los productos objetivo) mediante los vectores de características semánticas. El enfoque, propuesto por Salton en 1983, calcula la similitud entre dos vectores como el coseno del ángulo que forman, tal como se muestra en la Ecuación 3.1. Así, si ambos vectores son idénticos, la similitud toma el valor 1, mientras que si son completamente diferentes y, por tanto ortogonales, la similitud entre ambos se anula (Salton, 1986).

$$sim_{Cos}(A,B) = \frac{\vec{A} * \vec{B}}{|\vec{A}| * |\vec{B}|}$$

Ecuación 3.1. Similitud Basada en Coseno

La literatura revela la adopción de este enfoque en diferentes dominios de aplicación: filtrado de noticias (NewT (http://newt.org/)), TV personalizada (Movielens (http://movielens.org)), entre otros (Blanco, 2007).

3.2.4 Vecinos más cercanos

Los algoritmos que utilizan este enfoque calculan la distancia entre los productos objetivo y las preferencias del usuario activo, de forma que sólo sugieren aquellos productos que son más parecidos a los que le interesaron en el pasado (por ejemplo: que *están más cerca* de estos últimos).

Un caso particular de las técnicas basadas en la búsqueda de vecinos más cercanos es el razonamiento basado en casos (CBR - *Case Based Reasoning*). En este escenario, los perfiles de los usuarios se representan mediante una colección de experiencias o casos pasados. Dependiendo del dominio de aplicación del sistema, dichos casos pueden ser programas de TV vistos por el usuario, noticias y documentos consultados en la Web, sitios Web visitados o los últimos libros que compró. A la hora de recomendar un producto objetivo concreto, CBR aplica sobre este conjunto de entrenamiento diferentes métricas de similitud, para valorar si dicho producto se ajusta a las preferencias personales de cada usuario activo. Dependiendo del mecanismo elegido para representar los perfiles de los usuarios, dichas métricas pueden incluir: comparaciones sintácticas entre palabras clave, cálculo de similitud basada en coseno o procesos en los que se comparan las preferencias del usuario con los productos objetivo, y se ponderan los valores de similitud resultantes mediante el nivel de interés del usuario en relación a cada uno de los casos definidos en su perfil. Entre los sistemas que utilizan este enfoque se pueden destacar LaboUR (Schwab et al., 2001), Web-Sell (Cunningham, 2001), Re:Agent (Boone, 1998) y Entree (Burke, 2002). Este último sistema merece una mención especial, ya que la estrategia de recomendación empleada lo diferencia, en algunos aspectos, del resto de enfoques descritos en este capítulo. Dicho sistema sugiere restaurantes a sus usuarios utilizando CBR como método para valorar la similitud entre las preferencias de los mismos y los casos (restaurantes) disponibles en la herramienta. La diferencia mencionada antes es debida a que el proceso de personalización no trata de minimizar la interacción entre el sistema y el usuario, sino que la herramienta dialoga activamente con el mismo para así precisar las características de

los restaurantes recomendados. En cada interacción, el sistema presenta una selección de restaurantes al usuario, y éste tiene la posibilidad de establecer ciertos parámetros de búsqueda (como tipo de cocina, atmósfera del restaurante, precio, localización, etc.) para que el sistema Entree refine sus recomendaciones (Burke, 2002). Si bien esta interacción entre sistema y usuario redunda en la precisión y calidad de las recomendaciones ofrecidas, no es menos cierto que, en algunos dominios de aplicación, puede convertirse en una tarea tediosa y molesta para el usuario, lo que limita considerablemente la utilidad de este tipo de enfoques (Blanco, 2007).

3.2.5 Clasificación automática

En los sistemas basados en contenido, el proceso de elaboración de recomendaciones personalizadas se reduce a una simple tarea de clasificación. Para adoptar este enfoque, los sistemas deben representar los perfiles de los usuarios mediante los modelos basados en clasificadores automáticos, ya sean redes Bayesianas, árboles de decisión, reglas de asociación o redes neuronales .En este tipo de propuestas, el sistema dispone de un conjunto de entrenamiento en el que se incluyen las características de los productos que interesaron al usuario en el pasado. Basándose en la frecuencia de aparición de dichas características en el conjunto, el recomendador predice un modelo de sus preferencias (su perfil), que permite clasificar los productos objetivo en categorías genéricas. Este proceso conduce a la recomendación de productos que comparten *exactamente* las características que interesan al usuario activo, de ahí el carácter *sobreespecializado* de este tipo de sugerencias (basadas en contenido) (Blanco, 2007).

Como ejemplos más representativos, se destaca el sistema de filtrado de correo electrónico Re:Agent (Boone, 1998), que emplea una red neuronal para clasificar los mensajes recibidos en dos carpetas etiquetadas como *trabajo* y *otras*; el recomendador Syskill & Webert (Blanco, 2007), que emplea un árbol de decisión para decidir si las páginas Web son *interesantes* o *no interesantes* para los usuarios, y los sistemas de TV personalizada Recommender (Basu et al., 1998) y Movielens (Galán, 2007), que clasifican películas mediante reglas de aprendizaje inductivo.

3.3 Técnica de Filtrado Colaborativo

Es una de las técnicas más empleadas en los sistemas recomendadores propuestos en la literatura. Un recomendador de este tipo se basa en almacenar accesos de usuarios anteriores y reconocer parecidos entre los usuarios del portal basándose en los servicios que son accedidos por éstos. Estas comparaciones Inter-usuario son utilizadas para generar las recomendaciones; la idea es recomendar aquellos servicios que son habitualmente accedidos por aquellos usuarios que son parecidos al actual. La ventaja más significativa de las técnicas colaborativas es que son completamente independientes de cómo se representen los servicios que van a ser recomendados, funcionando de manera correcta en situaciones complejas como puedan ser la recomendación de música o películas, donde la variación de los gustos de los usuarios es la principal razón para la variación en sus preferencias. Es decir, que realmente las técnicas colaborativas pasan por alto la influencia de los servicios que deben ser recomendados para centrarse en los usuarios objeto de esas recomendaciones; esto es lo que se conoce como "correlación usuario a usuario" (Schafer et al., 2002).

A diferencia de los métodos basados en contenido, a la hora de ofrecer una recomendación a un usuario activo, el filtrado colaborativo no considera únicamente sus preferencias personales, sino también las de otros usuarios con intereses similares a los suyos (llamados en adelante vecinos). Tal similitud se estima a partir de las clasificaciones (o índices de interés) definidas en sus perfiles personales; de ahí que los enfoques colaborativos no requieran las descripciones semánticas consideradas en los métodos basados en contenido.

En la literatura se han definido dos técnicas diferentes dentro de las estrategias colaborativas (Blanco, 2007):

3.3.1 Filtrado colaborativo basado en usuario: Sugiere a cada usuario activo aquellos productos que han interesado a sus vecinos. Para formar este vecindario, la estrategia considera que dos usuarios tienen preferencias similares si han clasificado los mismos productos en sus perfiles y les han asignado índices de interés parecidos.

3.3.2 Filtrado colaborativo basado en ítem: Mediante este método, un producto es recomendado a un usuario activo si es similar a los definidos en su perfil personal. En este caso, se considera que dos productos son similares (o vecinos) si los usuarios que han clasificado uno de ellos tienden a clasificar el otro, asignándole índices de interés parecidos. Esta técnica da mejores resultados que la variante basada en usuario cuando el número de productos disponibles en el sistema recomendador es mucho menor que el número de usuarios (Sarwar, 2001).

En ambas técnicas, el objetivo es predecir el (nivel de) interés del usuario activo en relación a un producto objetivo dado y, en función de éste, sugerirlo (o no) a dicho usuario. En este proceso, se identifican tres fases:

1. En los enfoques colaborativos basados en usuario, la primera fase selecciona los usuarios cuyas preferencias son similares a las del usuario activo. Por el contrario, los enfoques basados en *ítem* extraen las preferencias del usuario que son más similares al producto objetivo.

2. A continuación, es necesario formar el vecindario .ya sea del usuario activo o de los productos definidos en su perfil., a partir de la selección realizada en la etapa anterior.

3. Finalmente, tal como se mencionaba previamente, el sistema debe predecir el nivel de interés del usuario activo en relación al producto objetivo. Para ello, los enfoques basados en usuario consideran el nivel de interés de los vecinos del usuario activo en relación al producto objetivo. Por el contrario, la versión colaborativa basada en *ítem* considera el nivel de interés del usuario activo en relación a aquellos productos de su perfil que son más similares al objetivo.

3.4 Fases del Filtrado Colaborativo

3.4.1 Fase 1: Selección de preferencias similares

Los métodos más utilizados en la literatura a la hora de medir la similitud entre los perfiles de varios usuarios, son las técnicas de selección de los vecinos más cercanos, el *clustering* y los modelos basados en clasificadores. Además, la primera es también la estrategia más extendida entre los enfoques colaborativos basados en *ítem*.

Vecinos más cercanos: De forma análoga a la explicada en los métodos basados en contenido, esta técnica también se utiliza para detectar similitud entre las preferencias de los usuarios en los enfoques colaborativos (tanto basados en usuario como basados en *ítem*).

En los enfoques de filtrado colaborativo basado en usuario, cada perfil se representa mediante un vector, cuyas componentes son las clasificaciones que el usuario ha asignado a cada uno de los productos incluidos en el mismo. Una vez modelados todos los usuarios, se aplican sobre sus respectivos vectores métricas como la similitud basada en coseno (ver Ecuación 3.1) o la correlación Pearson-r (ver Ecuación 3.2), detectando así parecidos entre sus preferencias.

$$corr_{Pea}(\vec{A},\vec{B}) = \frac{\sum_r (\vec{A}[r]-\overline{A})*(\vec{B}[r]-\overline{B})}{\sqrt{\sum_r (\vec{A}[r]-\overline{A})^2 * \sum_r (\vec{B}[r]-\overline{B})^2}}$$

Ecuación 3.2: Correlación de Pearson

Donde $\overline{A}$ y $\overline{B}$ son los valores medios de las clasificaciones definidas por los dos usuarios cuyas preferencias son comparadas (y cuyos niveles de interés se registran en los vectores $\vec{A}$ y $\vec{B}$, respectivamente). De acuerdo a la Ecuación 3.2, es evidente que cuantos más productos hayan clasificado a la vez los usuarios comparados, y cuanto más parecidas sean las clasificaciones asignadas a los mismos, mayor será la correlación detectada entre sus preferencias.

Por su parte, en lugar de computar la similitud entre dos usuarios concretos, los enfoques colaborativos basados en *ítem* calculan la similitud entre dos productos (*ítems*). Para ello, este tipo de propuestas seleccionan los usuarios que han clasificado a la vez los productos comparados, y construyen sendos vectores a partir de los niveles de interés que éstos han asignado a ambos productos. Finalmente, basta aplicar sobre estos vectores las métricas antes mencionadas (coseno, Pearson-r), para así obtener los valores de similitud concretos entre los productos considerados. Razonando de forma análoga a la descrita en los sistemas colaborativos basados en usuario, es evidente que cuantos más usuarios hayan clasificado a la vez los dos productos comparados y cuanto más parecidos sean los niveles de interés asignados a los mismos, más significativo será el valor de similitud medido entre ambos. Entre los sistemas colaborativos basados en la selección de los vecinos más cercanos se destacan los recomendadores de películas Bellcore Video Recommender (Hill, 1995) y Movielens (Galán, 2007), los sistemas de filtrado de noticias personalizadas GroupLens (Galán, 1997) y NewsWeeder (Lang, 1995), los recomendadores de listas musicales Ringo (Shardanand, 1994) y SmartRadio (Hayes, 1999) y los sistemas Web WebWatcher (Joachims et al., 1997) y WebSell (Cunningham, 2001).

Clustering: Mediante el método de estereotipos introducido por Rich (Rich, 1989), es posible identificar diferentes grupos o categorías a las que pueden pertenecer los usuarios en función de sus características y preferencias. Como ya ha señalado anteriormente, la identificación de estos estereotipos es una tarea laboriosa, ya que supone gran aporte de información por parte de los usuarios (Blanco, 2007).

Como alternativa a este método, Orwant propuso en 1995 un método de *clustering*, basado en definir grupos de usuarios de forma dinámica a partir de los perfiles individuales disponibles en el sistema (Orwant, 1995). Las características que comparten varios usuarios son utilizadas precisamente como estereotipos, de forma que una vez identificados los grupos de usuarios comunes, el sistema predice el interés de cada uno de ellos promediando las clasificaciones del resto de usuarios pertenecientes a su mismo grupo.

De acuerdo a los resultados obtenidos por Breese (Breese et al., 1998), las recomendaciones elaboradas mediante los métodos basados en vecinos más cercanos son más precisas que las obtenidas mediante el *clustering*, hecho que limita la aplicación de esta última técnica en los enfoques colaborativos actuales.

Clasificadores: El filtrado colaborativo está estrechamente relacionado con la clasificación automática. Típicamente, este tipo de enfoques representan los datos de entrada sobre los que trabaja el clasificador mediante las matrices descritas en la Tabla 2, en las que se definen los niveles de interés que cada usuario (filas) asigna a los productos (columnas) registrados en su perfil. Dado que es poco probable que todos los usuarios clasifiquen exactamente los mismos productos, dicha matriz tendrá muchos elementos vacíos. Tal como se describió previamente, la tarea del filtrado colaborativo es precisamente eliminar estos huecos, prediciendo el interés de cada usuario en relación a todos aquellos productos a los que les corresponde una entrada vacía en la matriz de clasificaciones.

En la literatura, existen enfoques que asignan un clasificador a cada usuario del sistema para poder predecir sus niveles de interés (en función de los gustos de los restantes), detectando con ello similitud entre sus respectivas preferencias. El trabajo presentado en (Breese et al, 1998) adopta un modelo basado en redes Bayesianas (Ver Anexo A), el enfoque descrito en (Billsus y Pazzani, 1998) se basa en los árboles de decisión, mientras que los cinco enfoques restantes utilizan reglas de asociación para cuantificar la similitud entre los productos definidos en la matriz de clasificaciones del sistema.

3.4.2 Fase 2: Creación del vecindario de un usuario o de un producto

Son tres las técnicas utilizadas tradicionalmente a la hora de determinar el tamaño del vecindario en los enfoques colaborativos:

Combinación de correlación y umbral: En los enfoques basados en usuario, este método establece que sólo aquellos usuarios cuya correlación con el activo supere un cierto umbral, podrán formar parte de su vecindario. De forma análoga, los enfoques basados en *ítem* consideran que únicamente aquellos productos del perfil del usuario activo cuya similitud con el objetivo es mayor que el umbral establecido, pueden estar incluidos en su vecindario.

Los n mejores vecinos: Según la técnica colaborativa empleada (basada en usuario o basada en *ítem*), este método selecciona los n usuarios más parecidos al activo, o los n productos de su perfil más similares al objetivo.

Es obvio que si n es muy elevado, se reduce la calidad de las recomendaciones ofrecidas, mientras que si se asumen valores reducidos, se limita mucho la capacidad predictiva del sistema. La principal ventaja de este método es que permite obtener recomendaciones más precisas que el anterior, aun cuando el solapamiento entre las preferencias de los usuarios es muy reducido (recordar la matriz de valoraciones explicada en la Sección anterior).

Centroide: En primer lugar, este enfoque selecciona el usuario más cercano al activo y calcula su centroide (Schmitz, 2006). A continuación, incluye otros usuarios en el vecindario del activo, utilizando como criterio de selección la mínima distancia entre éstos y el centroide, que es recalculado cada vez que se incorporan nuevos vecinos. Su comportamiento en escenarios con matrices de clasificaciones poco pobladas es peor que el de la estrategia de selección de los n vecinos más parecidos al usuario activo, razón por la que los enfoques basados en el cálculo de centroides no gozan de gran popularidad en la literatura.

De forma análoga a la comentada en los enfoques previos, este método también puede aplicarse en las técnicas colaborativas basadas en *ítem* para comparar los productos en lugar de los usuarios que los consumen (Blanco, 2007).

3.4.3 Fase 3: Predicción basada en el vecindario creado

Una vez formado el vecindario del usuario activo, el sistema colaborativo debe predecir su (nivel de) interés en relación al producto objetivo considerado. También en este contexto, se han propuesto técnicas de diversa naturaleza:

Recomendación de los productos más frecuentes: En los enfoques basados en usuario, este método selecciona, de entre todos los productos que interesan a los vecinos del usuario activo, aquellos que aparecen con mayor frecuencia en sus perfiles. Es decir, el recomendador sugiere al usuario activo un producto objetivo si éste ha sido muy relevante para la mayoría de sus vecinos.

En los enfoques basados en *ítem* también se premian los productos más frecuentes en los perfiles de los usuarios, ya que éstos ayudan a incrementar los valores de similitud (entre las preferencias del usuario activo y el producto objetivo) utilizados en el proceso de predicción.

Recomendación basada en reglas de asociación: Este método se utiliza en las técnicas colaborativas basadas en *ítem* y consiste en extraer reglas de asociación a partir de un conjunto de entrenamiento que contiene los perfiles de todos los usuarios del sistema. Estos enfoques cuantifican la similitud entre los productos disponibles en dichos perfiles a partir de la confianza de las reglas de asociación descubiertas entre ellos. Gracias a estos valores de similitud, el sistema colaborativo puede seleccionar los productos del perfil del usuario activo que más se parecen al producto objetivo y predecir su nivel de interés (O'Sullivan, 2004).

Los enfoques basados en reglas de asociación comparten la filosofía apuntada en la primera técnica, dado que también sugieren productos que aparecen en los perfiles de los usuarios con mucha frecuencia. Sólo en este caso, es posible

detectar las reglas que utilizan los enfoques colaborativos durante el proceso de predicción.

Media ponderada de clasificaciones: En las propuestas colaborativas basadas en usuario, un método alternativo para predecir el interés del usuario activo en relación a un determinado producto objetivo, consiste en promediar las clasificaciones que sus vecinos han asignado a dicho producto, empleando como pesos los valores de correlación entre sus respectivas preferencias. Esta idea aparece reflejada en la Ecuación 3.3, donde se muestra el nivel de interés $I_{U,i}$ que predice el enfoque para un usuario activo U y un producto objetivo i.

$$I_{U,i} = \frac{\sum_{K=1}^{M} sim(U, V_k) * R_{V_K}(i)}{M} \quad \textbf{Ecuación 3.3}$$

donde:

- M es el tamaño del vecindario de U,
- Vk es el k-ésimo vecino de U,
- sim(U; Vk) es la similitud (correlación) entre las preferencias de los usuarios U y Vk,
- RVk (i) es el nivel de interés del vecino Vk en relación al producto objetivo i.

De acuerdo a esta expresión, este modelo sugiere al usuario activo aquellos productos que hayan despertado mayor interés entre los vecinos con preferencias más similares a las suyas. Este enfoque asume que todos los usuarios emplean el mismo rango de valores a la hora de asignar clasificaciones a sus preferencias. Para poder contemplar el caso en que esta premisa no se cumpla, las propuestas existentes calculan la desviación de las clasificaciones de cada usuario respecto a su valor medio, tal como se mostró en la expresión de la correlación Pearson-r (ver Ecuación 3.2). El recomendador de noticias Grouplens es uno de los sistemas que

adopta este enfoque. En el dominio de la TV personalizada, algunos ejemplos representativos son MovieLens, Moviefinder, TiV, ReplaTV y TV Scout.

La media ponderada de clasificaciones también puede ser utilizada en los sistemas colaborativos basados en *ítem*. Este enfoque predice el interés del usuario activo, calculando la suma de las clasificaciones que haya asignado a los productos vecinos del objetivo. Dichas clasificaciones son ponderadas mediante los valores de similitud medidos entre tales vecinos y el producto objetivo, tal como muestra la ecuación 3.4.

$$I_{U,i} = \frac{\sum_{j=1}^{N} sim(i, v_j) * R_U(v_j)}{N}$$ **Ecuación 3.4**

donde:

- N es el tamaño del vecindario de i,
- vj es el j-ésimo vecino de i,
- sim(i; vj) es la similitud entre los productos i y vj , y
- RU(vj) es la clasificación asignada por U al producto vecino v_j.

3.4.4 Ventajas e inconvenientes del filtrado colaborativo

Por su propia naturaleza, los enfoques colaborativos permiten superar la falta de diversidad asociada a los métodos basados en contenido, ya que las recomendaciones elaboradas no se basan únicamente en las preferencias personales del usuario activo, sino que consideran los intereses del resto de usuarios del sistema. Esta cualidad se aprecia fácilmente en el método colaborativo basado en *ítem*, donde dos productos pueden ser similares (y por tanto recomendados al usuario activo) aún cuando no compartan ningún atributo semántico; simplemente es necesario que otros usuarios del sistema los hayan clasificado simultáneamente en sus perfiles (Blanco, 2007).Sin embargo, en el filtrado colaborativo también es posible identificar algunas limitaciones importantes:

En primer lugar, el *new user ramp-up* sigue estando presente en este tipo de sistemas, lo que dificulta enormemente la formación del vecindario asociado a un usuario que acaba de llegar al sistema, cuyo perfil registrará, típicamente, un número muy reducido de preferencias.

La segunda de las limitaciones (conocida en la literatura como *sparsity problem*) está relacionada con la noción de vecindario adoptada en los enfoques colaborativos tradicionales. Sus efectos se manifiestan a medida que aumenta el número de productos disponibles en el sistema recomendador. Ante tal diversidad, es menos probable que dos perfiles contengan exactamente los mismos productos (preferencias) y, por consiguiente, es más difícil encontrar vecinos tanto para el usuario activo como para el producto objetivo, fase crucial en los enfoques colaborativos basados en usuario y en *ítem*, respectivamente.

El tercero de los problemas ligados a los sistemas colaborativos es el llamado *gray sheep*, especialmente perjudicial para aquellos usuarios que, al tener preferencias muy diferentes a las del resto de usuarios, no pueden recibir recomendaciones colaborativas (debido a la imposibilidad de crear su vecindario) (Blanco, 2007).

Los problemas de escalabilidad también limitan los enfoques colaborativos tradicionales. Es evidente que a medida que aumenta el número de productos y usuarios en el sistema, también se incrementa el coste computacional del proceso de cálculo del vecindario (del usuario activo o de los productos definidos en su perfil), etapa crucial en este tipo de enfoques.

Otra de las limitaciones reconocidas en los enfoques colaborativos tiene su origen en la incorporación de nuevos productos al recomendador (por esta razón, denominada en inglés *new ítem ramp-up*). De lo anterior, se desprende que los enfoques colaborativos convencionales sólo sugieren productos incluidos en los perfiles de los usuarios del sistema. Como consecuencia de esto último, es posible identificar un tiempo de latencia desde que el sistema conoce nuevos productos hasta que éstos son recomendados a los usuarios activos (dado que deben ser clasificados por un número representativo de usuarios antes de ser incluidos en alguna recomendación) (Blanco, 2007).Esta limitación es crítica en algunos dominios de aplicación (véase la televisión digital), en los que no sólo hay un flujo constante de productos nuevos, sino que, en algunos casos, éstos sólo están disponibles durante un tiempo limitado. En este escenario, es especialmente importante reducir los tiempos de latencia mencionados antes, para así poder ofrecer a los usuarios los productos más novedosos del momento sin retardos innecesarios.

Por último, merece la pena destacar el llamado *cold start*, cuyos efectos se ponen de manifiesto durante las primeras etapas de funcionamiento de los sistemas colaborativos. Así, hasta que no se alcanza un número suficientemente elevado de usuarios registrados, el recomendador no dispone de información suficiente para crear de una forma precisa el vecindario (del usuario activo o del producto objetivo), minando de esta forma la calidad de las sugerencias ofrecidas (Blanco, 2007).

3.5 Técnica de Filtrado Basado en Reglas

Consiste en un tipo de filtrado colaborativo basado en observaciones, está centrado en observaciones implícitas del comportamiento cotidiano del usuario. Ofrecen información según reglas predefinidas. Confía en perfiles estáticos obtenidos a través de realimentación explícita. En los Sistemas de Recomendación basados en este filtraje, se observa lo que hace el usuario activo junto con lo que otros usuarios han hecho con anterioridad, para así poder dar la recomendación (Joachims et al. 1997). Dentro de esta técnica destacan dos tipos de filtrados: Activo y Pasivo, como se describen a continuación.

3.5.1 Filtrado Colaborativo Pasivo

Según la metodología de filtrado de información es posible distinguir entre sistemas que realizan un filtrado pasivo (Rafter et al, 1999) (cuando se genera una única recomendación que es válida para todos los usuarios del sistema), y sistemas de filtrado activo (Boutilier et al, 2003), en el que la recomendación se genera a partir del historial de recomendaciones de los usuarios para generar nuevas recomendaciones personalizadas.

El filtrado pasivo trata de reunir información implícita, para esto se mantiene un registro de las preferencias del usuario de manera de seguir sus acciones. Este filtro implícito usa lo antes grabado para poder determinar qué es lo que al usuario le gustaría y así poder recomendárselo. El filtraje implícito confía en las acciones de los usuarios para valorar y crear un ranking de los objetos a recomendar. La mayor fortaleza de este sistema es que se utilizan variables de análisis que normalmente estarían presentes en la filtración activa, pero sin el inconveniente de pedir a los usuarios que midan los ítems. Por ejemplo, sólo ciertos tipos de personas tomarán el tiempo para "evaluar" algún material, en la filtración pasiva la colaboración viene de alguien que teniendo acceso al sitio, automáticamente da los datos al Sistema de Recomendación (Hirsh et al., 2000) a través de la cantidad de descargas efectuadas sobre un material o la cantidad de veces que el material ha sido visitado.

3.5.2 Filtrado Colaborativo Activo

Una de las diferencias con los otros métodos de filtraje cooperativo es el hecho que éste utiliza el "acercamiento" entre pares. Esto significa que compañeros y personas con interés similares valoran productos, artículos o cualquier otro material y luego comparten esa información en Internet para que otras personas puedan verla. Este tipo de filtraje puede ser muy útil y efectivo en la situación de una búsqueda no guiada con miles de resultados (Shardanand & Maes 1995).

Ventajas

1. La recomendación dada de una persona hacia un objeto es de mucho valor para alguien que no tiene información al respecto. La transparencia de este método es su mayor ventaja.
2. Otra ventaja es que últimamente a la gente no le molesta proporcionar información para ayudar a otros.

Desventajas

1. La opinión personal puede influir. La subjetividad puede ser una desventaja en este método.
2. Otra desventaja es el hecho de que algunas personas no apoyen o colaboren con algún tema, dejándolo sin "recomendaciones".

Utilizando estas técnicas se cumple con el objetivo de considerar el universo de posibles respuestas, que hasta el momento, no han sido tomadas en cuenta, sólo por el hecho de que los usuarios, a pesar de descargar el archivo, no han entregado una valoración o comentario acerca del documento.

3.6 Técnica de Filtrado Demográfico

Los sistemas de recomendación demográficos tienen como objetivo clasificar al usuario en función de sus características demográficas, realizando a continuación las recomendaciones basándose en clases demográficas. Un ejemplo de este tipo de recomendación lo constituía Grundy (Rich, 1989), que era un sistema que recomendaba libros basándose en la información personal que se almacenaba en el sistema a través de un diálogo interactivo.

El filtrado demográfico emplea las características personales de los usuarios (edad, sexo, estado civil, ocupación profesional, historiales de compra, aficiones, etc.) proporcionadas durante la fase de registro en el sistema para descubrir las relaciones existentes entre un determinado producto y el tipo de usuarios a quien éste interesa (Krulwich, 1997). Por su naturaleza, este método modela las preferencias de los usuarios como un vector de características demográficas, y recurre a la técnica de estereotipos para inicializar sus perfiles. Así, todos aquellos usuarios que pertenezcan al mismo estereotipo recibirán las mismas sugerencias por parte del sistema.

El filtrado demográfico adolece de dos limitaciones principales: por una parte, puede conducir a recomendaciones demasiado generales e imprecisas para los usuarios, por considerar únicamente sus características demográficas; además, este método no permite que las sugerencias ofrecidas se adapten a posibles cambios en las preferencias de los usuarios, dado que sus datos personales suelen permanecer invariables a lo largo del tiempo (Blanco, 2007). A pesar de los inconvenientes comentados, el filtrado demográfico puede ser una estrategia útil si se combina con otros métodos.

3.7 Técnica de Filtrado por Utilidad

Los recomendadores basados en utilidad y en conocimiento no intentan realizar modelos a largo plazo, sino que basan su recomendación en analizar la correspondencia que existe entre las necesidades del usuario y el conjunto de opciones disponibles. Los recomendadores basados en utilidad recomiendan utilizando el cálculo de la utilidad de cada uno de los servicios para el usuario (Blanco, 2007).

3.8 Técnica de Filtrado por Conocimiento

Los sistemas de recomendación basados en conocimiento sugieren servicios basándose en inferencias respecto de las necesidades del usuario y de sus preferencias. Los sistemas basados en conocimiento se distinguen del resto en que poseen conocimiento funcional, es decir, estos sistemas tienen conocimiento acerca de cómo un servicio en particular se corresponde con las necesidades individuales de un usuario y, por tanto, tiene capacidad de "pensar" sobre la relación entre una necesidad y una posible recomendación (Blanco, 2007).

Además de las técnicas ya descritas anteriormente se definen técnicas hibridas que mezclan la funcionalidad de dos o más técnicas de recomendación y se definen como sigue:

3.9 Enfoques Híbridos

Las limitaciones identificadas tanto en el filtrado demográfico, como en los métodos basados en contenido y el filtrado colaborativo, plantearon la necesidad de combinar varias de estas estrategias para así aunar sus ventajas y mitigar sus deficiencias. Bajo esta premisa, surgen los denominados sistemas híbridos.

Uno de los modelos que goza de mayor popularidad es el que combina el filtrado colaborativo y los métodos basados en contenido. La mayoría de las propuestas existentes adoptan un paradigma conocido en la literatura como *collaboration via content*, definido por Pazzani en 1999. Este autor propone calcular la similitud entre los usuarios del sistema utilizando tanto las descripciones semánticas de los productos definidos en sus perfiles (empleadas en los métodos basados en contenido), como los niveles de interés asignados a los mismos (considerados en el filtrado colaborativo).

Son muchos los sistemas de personalización que han adoptado este enfoque en dominios de aplicación muy diversos: Amazon (http://www.amazon.com), CDNow (http://www.cdnow.com) y WebSell (Cunningham, 2001) (comercio electrónico); Fab (Balabanovic y Shoham, 1997), WebWatcher (Armstrong et al., 1995) y LaboUr (Schwab

et al., 2001) (Web personalizada); Anatagonomy (Kamba et al., 1997) y NewsWeeder (Lang, 1995) (filtrado de noticias);

Tal como apuntábamos al comienzo, los enfoques híbridos permiten neutralizar algunas de las limitaciones (individuales) asociadas a los métodos basados en contenido y al filtrado colaborativo:

- Por una parte, estos modelos permiten superar el excesivo parecido existente entre las recomendaciones basadas en contenido y las preferencias del usuario activo, gracias a que el enfoque colaborativo considera la experiencia del resto de usuarios del sistema, diversificando con ello las sugerencias ofrecidas.

- Por otro lado, el filtrado basado en contenido también contribuye a aliviar, en algunos casos, los tiempos de latencia que retrasan la recomendación de los productos que acaban de llegar a un sistema colaborativo. Esto es debido a que en un esquema híbrido, cualquier producto puede ser recomendado al usuario activo sin retardo alguno, siempre que los métodos basados en contenido consideren que se adapta a sus preferencias. Dadas las limitaciones de las métricas de similitud utilizadas en estos métodos, el problema de la latencia sólo será eliminado cuando los productos considerados sean muy parecidos a dichas preferencias; en caso contrario, serán sugeridos con retardos innecesarios debido a la inflexibilidad de las métricas adoptadas en los enfoques actuales (Blanco, 2007).

- Por último, destacar que los enfoques híbridos también mitigan, en cierta medida, los efectos del *sparsity problem* ya que, gracias a las descripciones semánticas consideradas en el filtrado basado en contenido, pueden detectar que dos productos son similares aun cuando no hayan sido clasificados simultáneamente en los perfiles de los usuarios. Sin embargo, esta solución aún no es efectiva, debido a las limitaciones de las métricas de similitud (sintácticas) adoptadas tradicionalmente (Blanco, 2007).

Para una revisión exhaustiva de los enfoques híbridos se recomienda la consulta de (Burke, 2002), donde Robin Burke define hasta siete métodos diferentes para combinar estrategias de personalización de diferente naturaleza:

Ponderado (*Weighted*): En un sistema híbrido ponderado se decide si un determinado producto objetivo es sugerido o no al usuario considerando las salidas de *todas* las estrategias de recomendación que conviven en el mismo. Como su nombre indica, algunas de las estrategias pueden tener más peso que otras durante el proceso de decisión, de forma que si las más determinantes deciden recomendar el producto, éste será ofrecido al usuario.

Conmutación (*Switching*): En este caso, en lugar de ejecutar todas las estrategias simultáneamente, el sistema emplea algún criterio para conmutar entre ellas: cuando se cumplen ciertas condiciones el sistema emplea una estrategia y, en caso contrario, recurre a las restantes.

Mixto (*Mixed*): Este esquema reúne en una misma recomendación productos que han sido sugeridos mediante las diferentes estrategias implementadas en el sistema híbrido.

Combinación de características (*Feature combination*): En este modelo híbrido se funden en un único conjunto los datos que utilizan las diferentes estrategias de personalización, y con éste se ejecuta un solo algoritmo de recomendación. Por ejemplo, si se combinan los métodos basados en contenido y el filtrado colaborativo, este esquema podría utilizar la información colaborativa (clasificaciones de los usuarios) como una característica más de las preferencias de los usuarios, sobre las que finalmente se ejecutaría el filtrado basado en contenido.

Cascada (*Cascade*): Un sistema híbrido en cascada funciona en dos etapas: primero se ejecuta una de las estrategias de recomendación sobre las preferencias del usuario, obteniendo un primer conjunto de productos candidatos a ser incluidos en la recomendación final; a continuación, una segunda estrategia refina la recomendación y selecciona sólo algunas de las sugerencias obtenidas en la primera etapa.

Incorporación de características (*Feature augmentation*): En este esquema, una de las estrategias se emplea para calcular una clasificación para un producto concreto. A continuación, esa información se incorpora como dato de entrada para las siguientes técnicas de recomendación; en otras palabras, la salida de una de las estrategias de recomendación se utiliza como entrada en las siguientes.

La diferencia fundamental entre este esquema híbrido y el modelo en cascada, es que, en este último, la segunda estrategia sólo trabaja sobre el conjunto de productos candidatos obtenidos por la primera de ellas, prescindiendo de cualquier tipo de información adicional calculada por ésta. Por el contrario, en el modelo basado en incorporación de características, toda la información que proporcione la primera de las técnicas, se utiliza en la(s) siguiente(s).

Metanivel (*Meta-level*): En este caso, el modelo *completo* generado por una de las estrategias se utiliza como entrada en las restantes. La diferencia entre este sistema híbrido y el basado en incorporación de características, es que en este último, el modelo aprendido sólo se utiliza para generar características que se usan como entrada en las siguientes estrategias, mientras que en el híbrido de metanivel se utiliza todo el modelo como dato de entrada.

Un esquema de este tipo estaría presente en un sistema híbrido que combine una primera etapa basada en contenido y una segunda colaborativa de la siguiente forma: la primera etapa construye un modelo de las preferencias del usuario. A continuación, este modelo es empleado en la fase colaborativa para comparar las preferencias de los usuarios y formar los vecindarios oportunos, antes de elaborar las recomendaciones personalizadas.

Todas las técnicas de recomendación, sin tomar en cuenta los enfoques híbridos, tienen sus puntos fuertes y débiles. De entre todos los problemas, se pueden destacar quizás como el más importante el problema de la falta de datos cuando se empieza a utilizar el sistema, conocido en la literatura como el problema del ramp-up (Konstan et al., 1999). Este término realmente se refiere a dos problemas diferentes, aunque relacionados (Martín, 2004):

- Nuevos usuarios. Debido a que las recomendaciones son el resultado de la comparación entre el usuario objetivo y otros usuarios, basándonos solamente en anteriores interacciones de los usuarios con el sistema, un usuario del que se disponen pocos datos resulta difícil de clasificar, y por tanto, de recomendar.
- Nuevos servicios. Análogamente, un nuevo servicio del que no se disponen todavía suficientes datos de accesos de los usuarios a éste, puede ser complicado de recomendar. Este problema se manifiesta en mayor medida, por ejemplo, en un campo como el de los periódicos digitales donde constantemente aparecen nuevas noticias pero solamente unas pocas son accedidas.

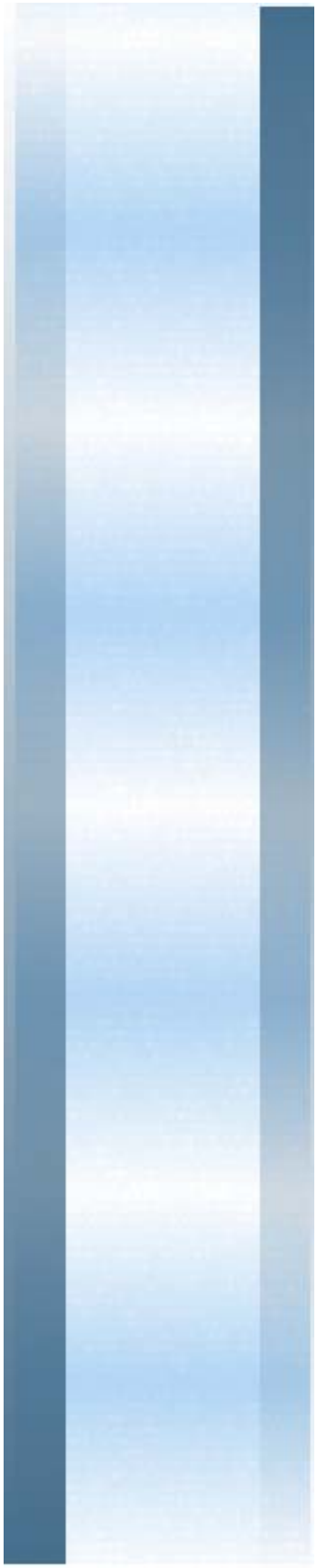
En la Tabla 3.2 se resumen las ventajas e inconvenientes de cada una de las cinco técnicas de recomendación explicadas. La notación seguida en esta tabla es la siguiente (Martín, 2004):

- A: Capacidad para identificar usuarios similares que acceden a servicios de diferente tipo; en otras palabras, capacidad para identificar usuarios similares a pesar de que puedan parecer heterogéneos analizando estrictamente los accesos a servicios que realizan.
- B: No necesita conocimiento del campo que tratan los servicios.
- C: Adaptabilidad, es decir, capacidad de mejorar con el tiempo.
- D: El sistema es capaz de auto realimentarse.
- E: No presenta problemas de ramp-up.
- F: Sensibilidad a cambios en las preferencias.
- G: Puede incluir características que no tienen que ver, estrictamente, con el servicio a recomendar.
- H: Capacidad de relacionar las necesidades de los usuarios con los servicios a recomendar.
- I: Problema de ramp-up para nuevos usuarios.
- J: Problema de ramp-up para nuevos servicios.

- K: No funciona bien para usuarios cuyo comportamiento pueda considerarse como difuso, en el sentido de que pertenece a dos grupos de usuarios distintos, cada uno de los cuales define un claro comportamiento.
- L: Depende de disponer de un histórico de datos grande.
- M: Problema de la "estabilidad-plasticidad".
- N: Debe obtener información demográfica.
- O: El usuario debe introducir una función de utilidad.
- P: No posee capacidad de aprendizaje.
- Q: Se necesita conocimiento.

Técnica	**Ventajas**	**Inconvenientes**
Colaborativa	A,B,C,D	I,J,K,L,M
Basada en Contenido	B,C,D	I,L,M
Demográfica	A,B,C	I,K,L,M,N
Basada en Utilidad	E,F,G	O,P
Basada en Conocimiento	E,F,G,H	P,Q

Tabla 3.2: Comparativa de las ventajas e inconvenientes de las diferentes técnicas de recomendación (Martín, 2004).



CAPITULO IV:

"ALGORITMOS DE RECOMENDACION"

4.1 Algoritmos basados en clasificadores automáticos.

A fin de aprovechar las capacidades predictivas durante el proceso de recomendación, algunos enfoques optan por modelar las preferencias de los usuarios mediante clasificadores automáticos. En el estado del arte, son dos los algoritmos que gozan de mayor popularidad, denominados de forma genérica *Branch-and-Bound* y *Hopfield Net*. Ambos algoritmos trabajan sobre un conjunto de conceptos especificados por el usuario, a partir de los cuales se extraen aquellos nodos de la red que están relacionados con los primeros de forma significativa.

4.1.1 El algoritmo Branch-and-Bound

Este algoritmo adopta como modelo de representación de conocimiento una red semántica, de ahí que los enlaces tengan asociadas etiquetas semánticas que identifican las relaciones existentes entre los conceptos definidos en la misma (Chen y Lynch 1992). Basándose en el principio de búsqueda *branch-and-bound y red bayesiana (ver Anexo A)*, este algoritmo (en adelante BNB) explora los caminos de longitud mínima establecidos entre los conceptos que el usuario ha especificado y aquellos que son finalmente recuperados. De esta forma, el algoritmo garantiza que dichos conceptos son los más relevantes para la consulta del usuario, ya que son los que están relacionados de forma más significativa (a través de los caminos más cortos) con los indicados por el mismo (Blanco, 2007).

Inicialmente, el algoritmo BNB activa el conjunto de nodos correspondientes a la consulta (que en adelante serán $\{S_1; S_2;\ldots; S_m\}$), a los que asigna un nivel de activación de valor 1, y construye dos colas:

- La cola de prioridad ($Q_{priority}$): En la primera iteración del algoritmo, esta cola registra el conjunto de nodos inicialmente activados, ordenados de acuerdo a su nivel de activación, de forma que la cabeza de la misma la ocupa siempre el nodo de mayor nivel.

$$Q_{priority} = \{S_1; S_2;\ldots; S_m\}$$

- La cola de salida (Q_{out}): Esta cola, inicialmente vacía, almacena todos los nodos que se vayan activando durante el proceso de propagación, junto a sus respectivos niveles.

$$Q_{out} = \{ \}$$

En cada iteración, el algoritmo:

1. Extrae de la cola de prioridad el nodo de mayor nivel de activación. En caso de que sean varios los nodos que registran este valor máximo, todos ellos son extraídos en la misma iteración.

2. Activa únicamente los vecinos de este nodo, unidos directamente a él mediante enlaces en la red de propagación.

3. Calcula los nuevos niveles de activación de dichos vecinos. Para ello, en cada iteración el nivel del vecino se calcula multiplicando el nivel que tenía en la iteración anterior el nodo que lo activó (seleccionado en 1) por el peso del enlace que une ambos nodos. Así, siendo j el nodo vecino e i el nodo extraído de la cola de prioridad, en t + 1 se cumple que $Aj(t + 1) = A_i(t) * W_{ij}$.

4. Añade dichos vecinos a la cola de prioridad, colocándolos en el lugar que les corresponda según su nuevo nivel de activación.

5. Dado que los nodos vecinos han sido activados, deben ser añadidos a la cola de salida. Destacar que, en caso de que dichos nodos ya se hubiesen almacenado en iteraciones previas en esta cola, sus niveles de activación son actualizados sumando al valor que ya tenían, el calculado mediante el procedimiento descrito en el paso 3.

El algoritmo itera hasta que se alcanza una determinada condición de parada, por ejemplo, número fijo de elementos en la cola de salida, cola de prioridad vacía, etc.

El enfoque adoptado en BNB puede tener una utilidad limitada en algunos dominios de aplicación, en concreto, en aquellos en los que las relaciones más significativas para el usuario no sean descubiertas a partir de caminos de longitud mínima. Precisamente para descubrir dichos caminos, el algoritmo activa en cada iteración únicamente los vecinos del nodo más relevante de la red, es decir, aquel que tiene mayor nivel de activación, y que, por tanto, ocupa la cabeza de la cola de prioridad. En caso de que el tipo de relaciones de interés no sean tan directas, parece más adecuado computar y evaluar los niveles de todos los nodos de la red en una misma iteración. Este enfoque es el adoptado en el algoritmo Hopfield Net (Blanco, 2007).

4.1.2 El algoritmo Hopfield Net

El algoritmo Hopfield Net (en adelante, Hopfield) (Chen et al., 1993) está basado en la red neuronal que lleva el mismo nombre (Hopfield, 1982), cuyas principales características han sido revisadas en el Anexo A: "*Preferencias de los usuarios mediante clasificadores automáticos*". Por esta razón, los nodos y enlaces de la red de propagación coinciden con las neuronas y sinapsis, respectivamente, en la red neuronal.

Hopfield propone una alternativa interesante al proceso de activación aplicado en el algoritmo BNB. Para ello, incorpora dos de las características más destacables de la red neuronal Hopfield: su principio de búsqueda en paralelo y sus propiedades de convergencia (Blanco, 2007).

- Búsqueda en paralelo: Por un lado, el algoritmo activa, en cada iteración, todos los nodos de la red en paralelo, calculando los niveles de cada uno de ellos en función de los niveles de los restantes.
- Convergencia: Por otra parte, el enfoque recorre los diferentes nodos de la red, iteración tras iteración, hasta que sus niveles de activación convergen a un valor estable. En este sentido, destacar que una de las principales contribuciones de Hopfield fue demostrar que, dado cualquier estado inicial de la red neuronal y

cualquier combinación de pesos en los enlaces (sinapsis), el algoritmo conducía siempre a un estado estable (Knight, 1990).

El funcionamiento detallado del algoritmo se describe a continuación:

1. Inicialmente, y al igual que en BNB, el algoritmo asigna un peso de valor uno a todos los nodos seleccionados por el usuario, y de valor cero a los restantes. Así, asumiendo una red de propagación con n nodos, se tiene que el nivel de activación de un nodo genérico x en t=0 será:

$$A_x(0) = \begin{cases} 1 & \text{si } x \in \{S_1, \ldots, S_m\} \\ 0 & \text{en otro caso} \end{cases}$$

Ecuación 4.1

2. En la iteración correspondiente a t, se calculan los niveles de todos los nodos de la red, esto es:

$$A_j(t+1) = f_S\left[\sum_{i=0}^{n-1} A_i(t) \cdot w_{ij}\right], \quad 0 \leq j \leq n-1$$

Ecuación 4.2

donde f_s es la función sigmoide (Knight, 1990), cuya expresión analítica se muestra a continuación:

$$f_S(net_j) = \frac{1}{1 + exp\left[\frac{\theta_1 - net_j}{\theta_2}\right]}$$

Ecuación 4.3

Donde:

- $net_j = \sum_{i=0}^{n-1} A_i(t) \cdot w_{ij}$,
- θ_1 un umbral configurable y
- θ_2 es un parámetro usado para modificar la forma de la función sigmoide.

La Ecuación 4.2 muestra la propiedad de búsqueda en paralelo de la red Hopfield, ya que en cada iteración del algoritmo todos los nodos de la red se activan a la vez. Basándose en ese principio de activación en paralelo, el nivel de activación de cada nodo individual se calcula sumando las contribuciones de *todos* sus nodos vecinos (tal como demuestra la expresión netj = Pn_1 i=0 Ai(t) en la Ecuación 4.2). Tal como se adelantaba en la sección previa, este proceso de activación es la principal diferencia entre los dos algoritmos revisados. Así, en BNB si un nodo tiene varios vecinos, en cada iteración su nivel se calcula considerando sólo la contribución del vecino que lo activó, en lugar de considerar el vecindario completo, tal como propone Hopfield (Blanco, 2007).

3. El proceso descrito se repite hasta que los niveles de activación de todos los nodos de la red converjan. Para comprobar esta condición, se consideran las dos últimas iteraciones del algoritmo y se aplica la Ecuación 4.4, donde es obvio que ε debe ser un parámetro configurable que toma valores poco significativos (Blanco, 2007).

$$\sum_{j=0}^{n-1} |A_j(t+1) - A_j(t)| \leq \xi$$ **Ecuación 4.4**

A continuación se señala el pseudocódigo del Algoritmo Hopfield:

```
**************************************************************
    * Entrada: R_U → red de propagación del usuario U
               P_U → perfil de U
    * Salida:  REC → conjunto de contenidos recomendados
**************************************************************
REC  Validación-Hopfield  (R_U, P_U) {
    t = 0;
    Para cada  nodo j  en  R_U  {
     Si  ((j ∈ P_U)  Y  (DOI_{P_U}(j) > 0))  Entonces  A_j(t) = DOI_{P_U}(j);
     En otro caso  A_j(t) = 0;
    }

    Para cada  nodo j  en  R_U  {
     net_j = Σ_{i=0}^{n-1} A_i(t) · w_ij ;

     A_j(t+1) = 1 / (1 + exp((θ_1 - net_j)/θ_2));
    }

    Mientras  (Σ_{j=0}^{n-1} |A_j(t+1) - A_j(t)| > ξ)  Hacer  {
     t++;
     Para cada  nodo j  en  R_U  {
      net_j = Σ_{i=0}^{n-1} A_i(t) · w_ij ;

      A_j(t+1) = 1 / (1 + exp((θ_1 - net_j)/θ_2));
     }
    }

    REC = Comprobar_Niveles_Activación (clase, δ_{Val_U});
}
```

Finalmente, en la Tabla 4.1 se resumen las principales características de los dos algoritmos descritos en esta sección.

Algoritmo BNB	Algoritmo Hopfield
Basado en redes semánticas.	Basado en redes neuronales.
Activación en serie: en cada iteración se activan sólo los vecinos del nodo extraído de Qpriority.	Activación en paralelo: en cada iteración se activan todos los nodos de la red.
El nivel de un nodo activado se calcula multiplicando el nivel del nodo que lo activó por el peso del enlace que los une.	El nivel de activación de un nodo se calcula considerando los niveles de todos sus vecinos y los pesos de los enlaces que los unen.
Condiciones de parada: - Número determinado de nodos en Q_{out}. - Cola $Q_{priority\ vacía.}$	Condición de parada: - Convergencia de los niveles de activación de los nodos de la red.

Tabla 4.1: Comparativa entre los algoritmos Branch-and-Bound y Hopfield Net

4.2 Algoritmos basados en vecinos cercanos.

Fueron los primeros algoritmos de filtrado colaborativo en implementarse. En primer lugar es necesario medir los parecidos de todos los usuarios con el usuario actual. Pueden utilizarse distintas medidas (Konstan et al., 1999):

4.2.1 Coeficiente de Correlación de Pearson.

Se deriva de las formulas de regresión lineal, y asume que la relación entre elementos es lineal, los errores independientes y la distribución tiene varianza constante y media 0. Estas suposiciones normalmente no se producen realmente con lo que hay que valorar como afectan a la bondad de los resultados, pero en un gran número de casos el rendimiento utilizando Pearson es apropiado. El peso que se asigna al usuario u para predecir al usuario activo a viene dado por: ($r_{a,i}$ es la votación del usuario a al elemento i, ver ecuación 4.5) (Galán, 2007).

$$w_{a,u} = \frac{\sum_{i=1}^{m} (r_{a,i} - \bar{r}_a) * (r_{u,i} - \bar{r}_u)}{\sigma_a \sigma_u}$$ **Ecuación 4.5**

Otras formas son las basadas en vectores (medida del coseno), las medidas de correlación basadas en entropía, correlación de Ringo o correlación de Spearman.

Una vez que se tienen los pesos de correlación de cada algoritmo hay que saber cuan confiables son estos pesos. Es posible tener un alto grado de correlación con vecinos con los que se comparten pocos elementos valorados por el usuario actual, pero con igual valoración. El uso de estos pesos proporciona estimaciones malas puesto que para tener una idea real de la correlación mientras más votos compartidos se tenga, resulta ser mejor. En los casos de pocas muestras es recomendable disminuir el factor de correlación en función del número de votos compartidos.

Para intentar mejorar más los pesos de la correlación entre usuarios se puede entrar a trabajar con la varianza de los elementos que cada usuario ha votado. Si un elemento es

votado positivamente por un gran porcentaje de la población, el que dos usuarios compartan esa votación dice poca información acerca de la correlación entre usuarios. Lo contrario pasa en el caso de elementos que sean votados positiva o negativamente por pocos usuarios. Para tomar en cuenta este hecho se añade a la formula de la correlación de Pearson un termino con la varianza del elemento (Galán, 2007).

4.2.2 Selección de Vecinos

En sistemas con pocos usuarios podría ser factible trabajar con todos los usuarios como vecinos tan sólo multiplicando cada uno por el peso de su correlación. Sin embargo en los sistemas actuales que poseen miles de usuarios esta vía no es posible, por ello hay que tomar un subconjunto de los usuarios, lo que no sólo mejora la eficiencia sino también la efectividad. Para elegir el número de vecinos se puede:

1. Establecer un umbral de correlación y tomar todos los que superen dicho umbral. El problema es que puede haber usuarios que no tengan muchos vecinos con correlación alta. Por tanto, el número de vecinos sería bajo y esto provocará que el número de elementos sobre el que la vecindad de un usuario puede opinar es también bajo.

2. Tomar "n" vecinos siempre, teniendo en cuenta que un número demasiado alto puede diluir la influencia de los vecinos con más peso y un número demasiado bajo provoca los mismos problemas que el método anterior.

4.2.3 Recomendación

La forma básica consiste en tomar las notas dadas por cada vecino y proporcionarlas a su correlación con el usuario actual como se muestra en la ecuación 4.6:

$$p_{a,i} = r_a + \frac{\sum_{u=1}^{n}(r_{u,i} - \bar{r}_u) * w_{a,u}}{\sum_{u=1}^{n} w_{a,u}}$$ **Ecuación 4.6**

Donde:

$p_{a,i}$: representa la predicción para el usuario activo a sobre el ítem i.

n : corresponde al número de vecinos.

$W_{a,u}$: es la similitud entre el usuario activo y el vecino u (ej.: correlación de Pearson).

$r_{u,i}$: es la valoración asignada por el vecino u al ítem i.

$\bar{r}_u$: corresponde al promedio total de valoraciones del vecino u.

Una posible mejora es la de introducir una variable que dependa de lo optimista o pesimista que sea cada usuario (ecuación 4.7). Unos usuarios votan por norma más alto que otros. Para proporcionarlos basta con normalizar respecto a la media de puntuación de cada usuario. De esta forma el predictor quedaría como (Galán, 2007):

$$p_{a,i} = r_a + \sigma_a \frac{\sum_{u=1}^{n} \frac{(r_{u,i} - \bar{r}_u)}{\sigma_u} * w_{a,u}}{\sum_{u=1}^{n} w_{a,u}}$$ **Ecuación 4.7**

4.3 Algoritmos basados en elementos.

En lugar de buscar similitudes entre usuarios buscan cercanías entre elementos (Sarwar et al, 2005). El procedimiento consiste en seleccionar los elementos que un usuario determinado ha votado y después comprobar como de similar es cada uno del resto de los elementos del sistema, para terminar recomendando los más parecidos. Existen distintas formas de evaluar la similitud entre elementos pero el procedimiento genérico consiste en tomar dos elementos x1, x2 y después calcular su similitud a partir de todos los usuarios que han votado ambos elementos. En teoría es la misma aproximación que la que se tenía con algoritmos basados en vecinos cercanos. La ventaja es que en el caso de los elementos la similitud entre ellos es menos variable que la similitud entre usuarios, lo que permite pre-computar estas similitudes y hace el proceso mucho más rápido.

4.3.1 Similitudes Basadas en Coseno.

Se considera cada elemento como un vector dentro de un espacio vectorial de m dimensiones y se calcula la similitud como el coseno del ángulo que forman como se señala en la ecuación 4.8. Es decir, si se tienen dos vectores x1, x2 consistentes en un arreglo, cuyos elementos son las votaciones recibidas de cada usuario, como se menciona en el capítulo II. Su similitud será entonces (Galán, 2007):

$$cos(x1, x2) = \frac{\vec{x1} \cdot \vec{x2}}{\|\vec{x2}\| \Delta \|\vec{x1}\|}$$ **Ecuación 4.8**

4.3.2 Similitudes Basadas en Correlación.

Considerando el conjunto de votaciones de los usuarios con votos sobre el elemento x1 y x2, se utiliza el coeficiente de correlación de Pearson (ecuación 4.9). Siendo U ese conjunto, u cada usuario R la valoración sobre un elemento y $\overline{R_x}$ la valoración media de ese elemento (Sarwar et al., 2005).

$$sim(x1,x2) = \frac{\sum_{u \in U} \left(R_{u,x1} - \bar{R}_{x1}\right)\left(R_{u,x2} - \bar{R}_{x2}\right)}{\sqrt{\sum_{u \in U} \left(R_{u,x1} - \bar{R}_{x1}\right)^2} \sqrt{\sum_{u \in U} \left(R_{u,x2} - \bar{R}_{x2}\right)^2}}$$ **Ecuación 4.9**

4.3.3 Similitudes Basadas en Coseno Ajustado.

En las similitudes basadas en coseno no se tienen en cuenta las diferencias entre las escalas de los usuarios. Para eso se incluye en la fórmula el parámetro $\overline{R_u}$ que indica la valoración media de ese usuario (ecuación 4.10). También se denomina transformación z (Sarwar et al., 2005):

$$sim(x1,x2) = \frac{\sum_{u \in U} \left(R_{u,x1} - \bar{R}_{u}\right)\left(R_{u,x2} - \bar{R}_{u}\right)}{\sqrt{\sum_{u \in U} \left(R_{u,x1} - \bar{R}_{u}\right)^2} \sqrt{\sum_{u \in U} \left(R_{u,x2} - \bar{R}_{u}\right)^2}}$$ **Ecuación 4.10**

4.3.4 Cálculo de la predicción.

Una vez que se elige un método para computar la relación entre elementos quedan por calcular los elementos que son más parecidos a los del usuario. De nuevo existen varias aproximaciones.

4.3.5 Suma con pesos (Weighted Sum).

Se toman todos los elementos que el usuario ha votado. Se toma un elemento "x1" y para ese elemento se suman todos los coeficientes de similitud entre ese elemento y los elementos votados por el usuario, proporcionados al valor del voto. Siendo N cada elemento votado por el usuario, $S_{i,N}$ la similitud entre los elementos i y N y, $R_{u,N}$ la valoración del usuario del elemento N (ecuación 4.11) (Sarwar et al., 2005):

$$Pu,i = \frac{\sum_N (S_{i,N} * R_{u,N})}{\sum_N (|S_{i,N}|)}$$ **Ecuación 4.11**

4.3.6 Regresión.

Similar al modelo anterior, pero en lugar de sumar directamente las notas de los elementos similares se utiliza una aproximación basada en la recta de regresión. Con este método se intenta compensar un problema que se da al evaluar las similitudes mediante medidas del coseno o la correlación, y es que vectores con alta similitud pueden encontrarse distantes en sentido euclídeo. Se utiliza la misma fórmula que en el caso de la suma proporcionada pero sustituyendo $R_{u,N}$ por (ecuación 4.12) (Sarwar et al., 2005):

$$\overline{R}'_N = \alpha \overline{R}_i + \beta + \varepsilon$$ **Ecuación 4.12**

4.4 Predictores "Slope-one"

Slope one es un algoritmo sencillo pero eficaz de recomendaciones ítem a ítem basado en el filtrado colaborativo, es decir, en el rating proporcionado por los usuarios. Su principio básico es comparar 2 ítems en función de cuantos usuarios han valorado estos dos ítems para determinar su similitud. A su vez permite predecir en función de las valoraciones dadas por un usuario a un ítem cómo valorará otro ítem de la colección. Un ejemplo lo constituyen Taste, Cofe y Duine, que son tres frameworks orientados al trabajo con sistemas de recomendación, cada uno de ellos con características propias de distintos métodos de recomendación (Ver Anexo B).

A la hora del cálculo de la predicción para un usuario U se tiene en cuenta tanto la información de usuarios que tienen en común la votación de algún elemento U como la información del resto de los elementos votados (Lemire y Maclachlan, 2005).

Partiendo de dos Arrays v_i y w_i de longitud n se busca obtener la mejor predicción de w a partir de v (ecuación 4.13). Tendrá la forma f(x) = x + b y deberá minimizar el error cuadrático medio $\sum_i (v_i + b - w_i)^2$ de donde derivando se obtiene que:

$$b = \frac{\sum w_i - v_i}{n}$$ **Ecuación 4.13**

Es decir, la diferencia media entre ambos arrays. Partiendo de esto se pueden diseñar las predicciones. A partir de un conjunto X de datos de entrenamiento se toman dos elementos cualquiera i,j con votaciones u_i u_j y se calcula la desviación media entre ambos (ver ecuación 4.14) (Sólo se consideran usuarios que hayan votado tanto i como j):

$$desv_{j,i} = \sum \frac{u_j - u_i}{n}$$ **Ecuación 4.14**

Con lo que se tiene un array simétrico precalculado que es posible actualizar con cada nuevo elemento que se añada al sistema. La predicción para un usuario u_j a partir del resto de usuarios será (Con R_j el conjunto de ítems relevantes):

$$P(u)_j = \frac{1}{total(R_j)} \sum_{i \in R_j} (desv_{j,i} + u_i)$$

Ecuación 4.15

4.4.1 Predictor "Slope-one" con pesos.

El predictor anterior (ecuación 4.15) no tiene en cuenta el número de notas que se han tomado. No es igual predecir la nota de un usuario sobre un ítem L a partir de los ratings de ese usuario de otros elementos J K si hay muchos más usuarios que tienen el par de votos J-L que el par K-L. El elemento J es mucho mejor predictor que el elemento K en este caso. Analíticamente esta idea se introduce en la ecuación 4.16 con el factor $c_{j,i}$ que es el número de evaluaciones de los ítems j e i (Galán, 2007).

$$P(u)_j = \frac{\sum_{i \in S(u) - \{j\}} (desv_{j,i} + u_i) c_{j,i}}{\sum_{i \in S(u) - \{j\}} c_{j,i}}$$

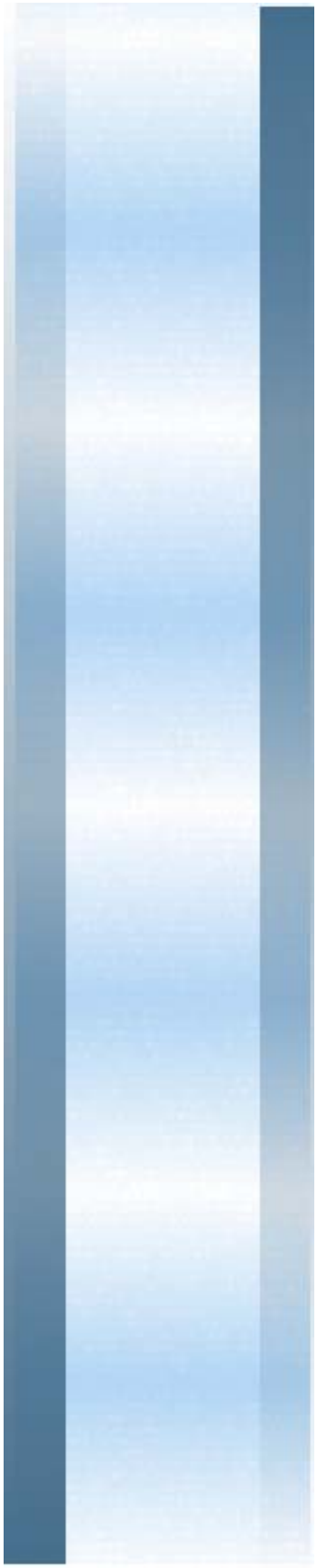
Ecuación 4.16

4.4.2 Predictor "Slope-one" Bi-Polar.

Trabaja dividiendo la predicción en dos partes. Usa el algoritmo anterior una vez para obtener una predicción de los elementos que han gustado al usuario y de los que no han gustado.

El primer problema que plantea este sistema es establecer el umbral a partir del que se considera que un elemento gusta o disgusta. La idea intuitiva es establecer un umbral que sea la mitad de la escala de evaluación. Si la escala va de 1 a 10 los elementos por debajo de 5 se considerarían como evaluados negativamente y los otros son evaluados positivamente. Esta aproximación sería la adecuada si las evaluaciones de los usuarios

fueran distribuidas uniformemente, sin embargo el comportamiento real de los usuarios indica que existe un porcentaje elevado de votaciones superiores a la mitad de la escala. Por ello el valor del umbral se establece como la media de todas las notas dadas por el usuario (Galán, 2007). En la práctica este procedimiento supone doblar el número de usuarios, pero a la vez también reduce el número de elementos en el cálculo de las predicciones.



CAPITULO V:

"METRICAS DE EVALUACION"

5.1 Medición de la calidad de las Recomendaciones

Después del diseño y desarrollo de un sistema de Recomendación, como en cualquier otro sistema, se hace necesaria la medición de la calidad de sus resultados con el fin de analizar los puntos fuertes y débiles de su funcionamiento, visualizar las variables involucradas, diagnosticar las causas de posibles fallas y descubrir la forma de afectar dichas variables para optimizar los resultados.

Evaluar el rendimiento de los algoritmos de recomendación no es trivial. Primero porque diferentes algoritmos pueden ser mejores o peores dependiendo del dataset[5] elegido. Según Herlocker (Herlocker et al., 2004), una métrica para la exactitud de un sistema de recomendación mide, empíricamente, que tan cerca está el ordenamiento de ítems predichos para un usuario por el sistema con respecto al ordenamiento verdadero que el usuario haría, según su preferencia, de los mismos ítems. Aunque los objetivos del sistema de recomendación pueden ser diversos, un sistema puede diseñarse para estimar con exactitud la nota que daría un usuario a un elemento, mientras otro puede tener como principal objetivo el no proporcionar recomendaciones erróneas. Es decir, puede haber múltiples tipos de medidas: Que las recomendaciones cubran todo el espectro de elementos del conjunto (cobertura), que no se repitan, que sean explicables. Sin embargo, el principal objetivo de un sistema de recomendación no es directamente cuantificable: la satisfacción del usuario. En muchos casos conseguir un error cuadrático menor al elegir un algoritmo u otro no es apreciado por el usuario. Sin embargo, hay muchos otros parámetros que pueden influir en esa satisfacción: La sensación de credibilidad que ofrezca el sistema, la interfaz de usuario y la mejora del perfil al incluir nuevos votos. En cualquier caso las medidas de precisión pueden dar una primera idea de cuán bueno es el algoritmo nuclear del sistema de recomendación (Galán, 2007).

En el presente capítulo se presentan varias métricas, como opción a ser empleadas en la evaluación de un Sistema de recomendación y, que son categorizadas en tres clases (Herlocker et al., 2004): métricas de exactitud predictiva, métricas de exactitud en clasificación y métricas de exactitud en ordenamiento.

[5] Es una colección de datos, suele presentarse en forma de cuadro. Cada columna representa una variable y cada fila corresponde a un determinado miembro del conjunto de datos en cuestión. En ella se incluyen los valores para cada una de las variables, por ejemplo como la altura y el peso de un objeto o valores de números aleatorios. Cada valor es conocido como un dato.

5.2 Métricas de exactitud predictiva

En la clase de métricas de exactitud predictiva están aquellas métricas que miden qué tan cerca están los puntajes de relevancia predichos por el sistema con respecto a los puntajes reales dados por un usuario. Entre estas métricas destacan los llamados métodos estadísticos tales como: el error medio absoluto (MAE por sus siglas en inglés) y otras métricas relacionadas como el error medio absoluto normalizado y el error medio cuadrado (Salazar y Ortega, 2006).

5.2.1 Error Medio Absoluto (MAE)

Mide la desviación de las recomendaciones predichas (p_i) y los valores reales (q_i) (ecuación 5.1). A menor MAE mejor predice el sistema las evaluaciones de los usuarios.

$$MAE = \sum_{i=1}^{N} \frac{| p_i - q_i |}{N} \qquad \textbf{Ecuación 5.1}$$

El MAE sin embargo puede dar una idea distorsionada del algoritmo para el caso de sistemas que tienen como objetivo encontrar una lista de buenos elementos recomendables. El usuario tan sólo está interesado en los N primeros elementos de la lista. El error que se cometa al estimar el resto le es indiferente. Tampoco es recomendable en sistemas en los que la salida deba de ser una decisión binaria de si/no. Por ejemplo, con una escala de 1 a 10 si el umbral está situado en 5, utilizando MAE se obtendría un mayor error al errar de 9 a 5 que al errar de 5 a 4, lo cual no es cierto a la hora de medir el error de salida (Galán, 2007).

Sin embargo, es un tipo de error estadísticamente muy estudiado y sencillo de comprender. Posee muchas variaciones, como el error cuadrático medio que persigue penalizar los mayores errores o el error absoluto normalizado que

facilita la tarea de establecer comparaciones entre pruebas con diferentes datasets.

5.2.2 Error Medio Absoluto Normalizado

Para tener en cuenta el peso del error respecto al valor de la variable medida se normaliza el valor absoluto, teniendo el valor absoluto medio normalizado (ecuación 5.2) (Stauffer y Seaman, 1990):

$$NMAE = \sum_{i=1}^{N} \frac{| p_i - q_i | / q_i}{N}$$

Ecuación 5.2

5.2.3 Error Medio Cuadrático

Para el cálculo de precisión se utiliza el error medio cuadrático definido como (Pielke, 1984) (ecuación 5.3):

$$RMSE = \sqrt{\sum_{i=1}^{N} \frac{(| p_i - q_i |)^2}{N}}$$

Ecuación 5.3

El error medio cuadrático proporciona la medida de las diferencias en promedio entre los valores pronosticados y los observados.

5.3 Métricas de exactitud en clasificación

Las métricas de exactitud en clasificación o también llamadas métricas de decisión, evalúan cuán efectivo es un sistema de predicción, ayudando al usuario a seleccionar los elementos de mayor calidad, es decir, con qué frecuencia el sistema de recomendación efectúa recomendaciones correctas. Para ello asumen que el proceso de predicción es binario: o el elemento recomendado agrada al usuario o no lo agrada. Sin embargo, en la práctica se plantea el problema de evaluar esto. Una posible solución es la de dividir el conjunto de datos en dos conjuntos, entrenamiento y test. Se trabaja con el conjunto de entrenamiento y posteriormente se evalúa el resultado comparando las recomendaciones proporcionadas con las del conjunto de test. Aun siendo a veces útil esta técnica, hay que tener en cuenta que los resultados dependen fuertemente del porcentaje de elementos relevantes que el usuario haya votado. Entre estas métricas están Precision and Recall y las curvas ROC (Receiver Operating Characteristic) (Salazar, 2006).

5.3.1 "Precisión and Recall"

Precisión and Recall es una de las métricas más conocidas y utilizadas en muchos tipos de sistemas de recuperación de información. Precisión es la probabilidad de que un elemento seleccionado sea relevante y Recall es la probabilidad de que sea seleccionado un elemento relevante, aunque en los sistemas de recomendación la "relevancia" es algo totalmente subjetivo. De cara al usuario esta métrica es más intuitiva, puesto que establecer que un sistema tiene una precisión del 90% significa que de cada 10 elementos recomendados 9 serán buenas recomendaciones, algo que no queda claro proporcionando valores de error cuadrático medio (Galán, 2007).

La evaluación a nivel del sistema se realizará en casos en los que los investigadores identifiquen indicadores que se puedan medir y que evidencien correlación con la efectividad del sistema independientemente de la interacción de los usuarios. Este tipo de evaluación es la más usada en Filtrado de

Información, porque ofrece un análisis económico y fácilmente repetible, donde los datos de los usuarios son recogidos una sola vez.

Gwizdka y Chignell (Gwizdka y Chignell, 1999) mencionan cinco **métricas** a tener en cuenta:

- ***Retardo***. Intervalo de tiempo transcurrido desde que se hace la demanda hasta que se da la respuesta.
- ***Presentación***. El formato físico de la salida del sistema.
- ***Esfuerzo del usuario***. El esfuerzo, intelectual o físico que se demanda del usuario.
- ***Exhaustividad***. Capacidad del sistema de presentar todos los ítems relevantes.
- ***Precisión***. Capacidad del sistema de ocultar ítems que no sean relevantes.

De estas métricas, se pueden destacar las dos últimas. Se calculan a partir de una tabla de contingencia que clasifica los ítems con respecto a las necesidades de información distinguiendo dos grupos: *relevantes* o no relevantes. Además, también se clasifican los ítems según se hayan recomendado al usuario (seleccionados) o no (no seleccionados).

Con estas cuatro categorías, se construye la tabla de contingencia 5.1

	Seleccionados	**No Seleccionados**	**Total**
Relevantes	N_{rs}	N_m	N_r
Irrelevantes	N_{is}	N_{in}	N_i
Total	N_s	N_n	N

Tabla 5.1: Tabla de contingencia Precisión and Recall

La *precisión (P)* se define como la proporción de ítems relevantes seleccionados (Nrs) con respecto al total de ítems seleccionados (Ns), es decir, mide la probabilidad de que un ítem seleccionado sea relevante (Ecuación 5.4):

$$P = \frac{N_{rs}}{N_s}$$ **Ecuación 5.4**

La exhaustividad (R) se define como la proporción de ítems relevantes seleccionados (Nrs) con respecto al total de ítems relevantes existentes (Nr) (Ecuación 5.5), es decir, representa la probabilidad de que un documento relevante sea seleccionado.

$$R = \frac{N_{rs}}{N_r}$$ **Ecuación 5.5**

5.3.2 Curvas ROC (Receiver Operating Characteristic).

ROC (Reciver operating characteristic) es otra medida muy utilizada. Proporciona una idea de la potencia de diagnóstico de un sistema de filtrado. Las curvas ROC dibujan la especifidad (Probabilidad de que un elemento malo del conjunto sea rechazado por el filtro) y la sensitividad (probabilidad de que un elemento bueno al azar sea aceptado). Si un elemento es bueno o malo viene dado por las valoraciones de los usuarios. Las curvas se dibujan variando el umbral de predicción a partir del cual se acepta un elemento. El área bajo la curva se va incrementando cuando el filtro es capaz de retener más elementos buenos y menos malos (Galán, 2007).

El modelo de las curvas ROC trata de medir el grado con el cual un sistema de filtrado de información puede distinguir de manera exitosa entre relevancia y ruido (Herlocker et al., 2004).Una curva ROC representa una gráfica del recall versus el fallout, donde recall es el porcentaje de ítems relevantes (proporción

de verdaderos positivos) recomendados a los usuarios (ecuación 5.6) y el fallout es el porcentaje de ítems no relevantes (proporción de falsos positivos) recomendados (ecuación 5.7).

$$recall = TP\_rate = \frac{TP}{P} \qquad \textbf{Ecuación 5.6}$$

$$fallout = FP\_rate = \frac{FP}{N} \qquad \textbf{Ecuación 5.7}$$

Cada punto en la curva ROC corresponde a un valor límite de predicción de relevancia a partir del cual el sistema considera que la predicción de un ítem es positiva o negativa (relevante o no relevante respectivamente). Los puntos ubicados en la parte más a la izquierda de la curva corresponden con los valores límite mayores de predicción de relevancia, por lo cual es la parte más exigente de la curva. Los puntos ubicados más a la derecha de la curva corresponden con los valores límite menores, por lo cual es la parte menos exigente de la curva (Salazar, 2006). Los valores de recall y fallout, se calculan con base en las llamadas matrices de confusión o tablas de contingencia como la presentada en la Tabla 5.2

		Verdadera clasificación dada por el usuario	
		Positivo	Negativo
Clasificación dada por el sistema de recomendación	Positivo	Verdaderos Positivos (TP)	Falsos Positivos (FP)
	Negativo	Falsos Negativos (FN)	Verdaderos Negativos (TN)
	Totales	Total Positivos (P)	Total Negativos (N)

Tabla 5.2: Tabla de contingencia o matriz de confusión con base en la cual se calculan los valores de recall y fallout para las gráficas ROC.

Un ítem es considerado verdadero positivo cuando dicho ítem es en realidad positivo para el usuario y es también clasificado como positivo por el sistema de recomendación. Si tal ítem es positivo para el usuario pero es clasificado como negativo por el sistema de recomendación, es considerado un falso negativo (ver Tabla 5.2).

Un ítem es considerado verdadero negativo cuando dicho ítem es en realidad negativo para el usuario y es también clasificado como negativo por el sistema de recomendación. Si tal ítem es negativo para el usuario pero es clasificado como positivo por el sistema de recomendación, es considerado un falso positivo (ver Tabla 5.2).

El área bajo una curva ROC es una medida que resume el desempeño del sistema de filtrado y es equivalente a la probabilidad de que el sistema esté en capacidad de elegir correctamente entre dos ítems, uno seleccionado aleatoriamente a partir del conjunto de ítems no relevantes y el otro seleccionado aleatoriamente a partir del conjunto de ítems relevantes (Salazar, 2006), como se muestra en la figura 5.1.

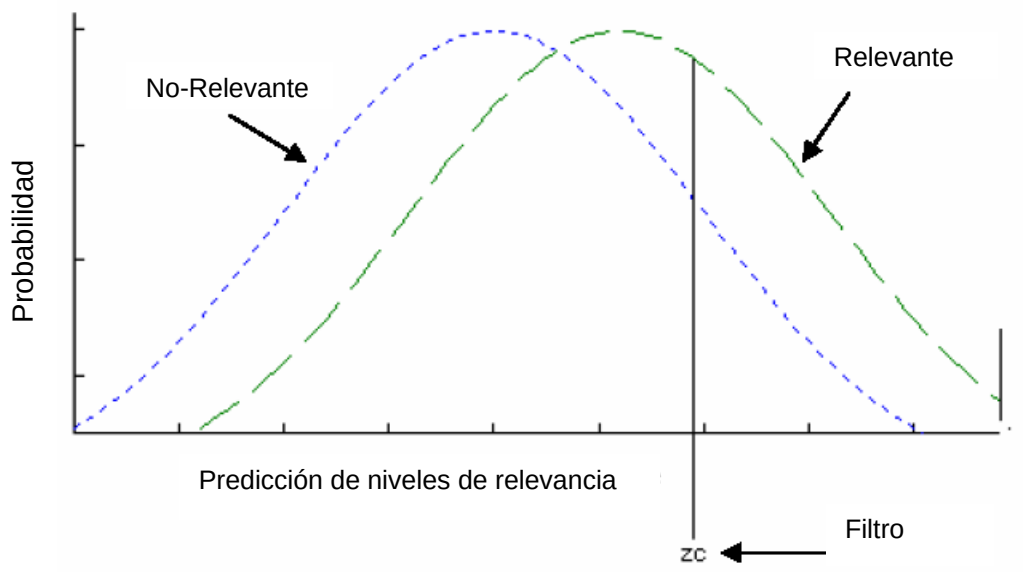


Figura 5.1: Posible representación de las funciones de densidad para ítems relevantes y No-relevantes (Herlocker el al., 2004)

El modelo ROC asume que existe un valor de afinación de filtro zc, de tal manera que todos los ítems que el sistema rankea sobre el corte serán revisados por el usuario y los que se encuentren bajo el corte no serán revisados por el usuario. Este corte define el tamaño de la búsqueda. Como se muestra en la figura 5.1, para cada valor de zc existirá un valor diferente de memoria (porcentaje de ítems buenos devuelto, o el área bajo la distribución de probabilidad relevante a la derecha de zc) y de caída (porcentaje de ítems malos devuelto, o el área bajo la distribución de probabilidad no-relevante a la derecha de zc). La curva ROC representa un grafico de memoria versus caída, donde los puntos en la curva corresponden a cada valor de zc (Herlocker et al., 2004).

A continuación se muestra un algoritmo común para crear una curva ROC (Herlocker et al., 2004):

1. Determinar de qué forma se identificará un ítem como relevante o no relevante.
2. Generar un ranking predicho de ítems para cada ítem predicho, en orden descendente de relevancia predicha (comenzando la gráfica en el origen)

    a. Si el ítem predicho es efectivamente relevante, dibujar la curva un paso verticalmente.
    b. Si el ítem predicho no es relevante, dibujar la curva horizontalmente un paso a la derecha.

    Si el ítem predicho no ha sido calificado (Ej. Su relevancia no se conoce), entonces el ítem es simplemente descartado y no afecta la curva ni de forma positiva ni negativa.

De manera intuitiva, según (Herlocker et al., 2004), la medida AUC (área bajo la curva ROC) captura el recall del sistema a diversos niveles de fallout. Según esta interpretación del área bajo la curva, un sistema de filtrado de información perfecto generaría una curva ROC tal que su área AUC es igual a 1

(ver Figura 5.2); y un sistema de filtrado que genera predicciones aleatorias, el cual representa el peor caso, produciría una curva ROC tal que su área AUC es igual a 0,5 (ver Figura 5.3).

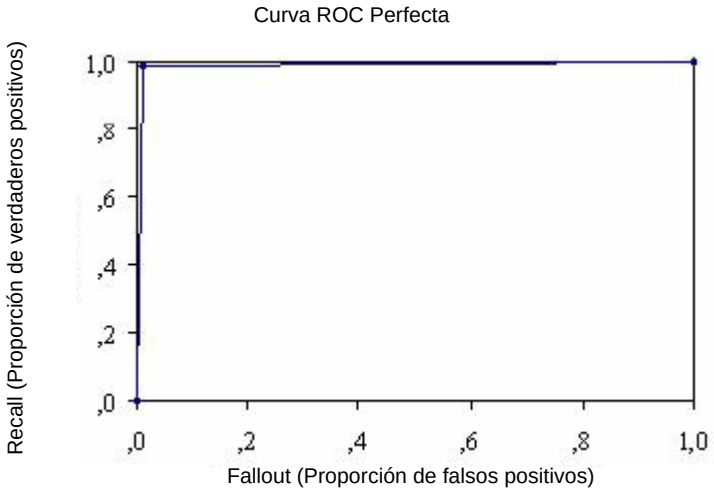


Figura 5.2: Curva ROC para un sistema de filtrado perfecto.

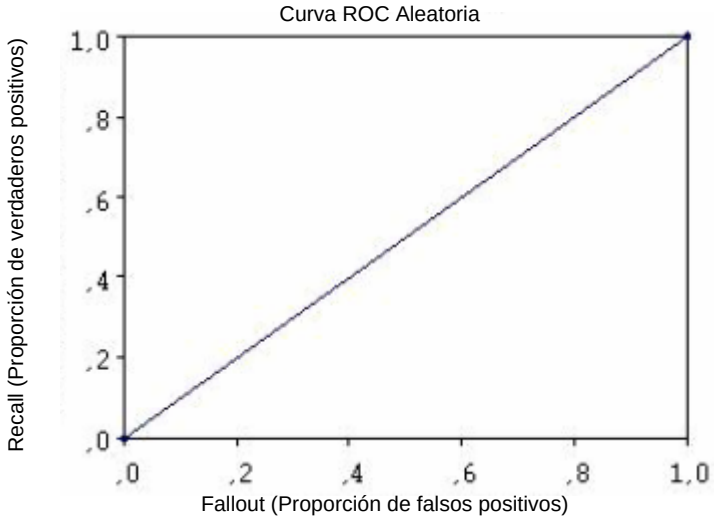


Figura 5.3: Curva ROC para un sistema de filtrado que genera predicciones aleatorias.

5.4 Métricas de exactitud de ordenamiento

En la clase de métricas de exactitud de ordenamiento están aquellas que miden la habilidad de un sistema de recomendación para producir un ordenamiento de ítems recomendados que coincida con el ordenamiento que un usuario haría de los mismos ítems. Entre estas métricas están la correlación predicción-evaluación, métrica de utilidad half-life y la medida NDPM (Salazar, 2006).

5.4.1 Correlación predicción-evaluación

Dos variables están correlacionadas si la varianza en una variable puede ser explicada por la varianza de la segunda. La correlación producto-momento de Pearson, la ρ de Spearman y el Tau de Kendall son 3 de las medidas de correlación más conocidas (Herlocker et al., 2004).

La correlación de Pearson mide la extensión que existe en una relación lineal entre dos variables. Esto está definido como la covarianza de las anotaciones z y se muestran en la ecuación 5.8.

$$c = \frac{\sum \left(x - \bar{x} \right) \left(y - \bar{y} \right)}{n * stdev(x) stdev(y)}$$ **Ecuación 5.8**

Las correlaciones de ranking, como la ρ de Spearman (ecuación 5.9) y el Tau de Kendall miden la extensión que existe cuando dos rankings diferentes concuerdan independiente de los valores actuales de las variables. La ρ de Spearman se computa de la misma forma que la correlación producto-momento de Pearson, a diferencia que x e y son transformadas en rankeos y las correlaciones son computadas sobre esos rankeos (Herlocker et al., 2004).

$$\rho = \frac{\sum\left(u - \bar{u}\right)\left(v - \bar{v}\right)}{n * stdev(u)stdev(v)} \qquad \textbf{Ecuación 5.9}$$

El Tau de Kendall representa un enfoque diferente para computar la correlación de los rankings, que es independiente de los valores de las variables. En la ecuación 5.10 se muestra una aproximación al Tau de Kendall. La C representa el número de pares concordantes – pares de ítems que el sistema predice en el orden de rankeo apropiado. D representa el número de pares discordantes – pares que el sistema predice en el orden incorrecto. TR es el número de pares de ítems en el orden real (el ranking determinado por la calificación del usuario) que poseen rankeos empatados (ej. La misma nota) y TP es el número de pares de ítems en el orden predicho que poseen rankeos empatados (el mismo valor de predicción)

$$Tau = \frac{C - D}{sqrt((C + D + TR)(C + D + TP))} \qquad \textbf{Ecuación 5.10}$$

A pesar de su simplicidad, las métricas de correlación vistas no han sido usadas de forma extensiva en la medida de sistemas de recomendación o sistemas de obtención de información. La correlación de Pearson fue utilizada por Hill (Hill, 1995) para evaluar el desempeño de sus sistemas recomendadores.

Las ventajas de las métricas de correlación son:

(a) Comparan un ranking de sistema no binario a un ranking de usuario no binario.
(b) Son bien entendidas por la comunidad científica.
(c) Proporcionan una evaluación de medida única para el sistema completo.

Puede existir debilidad en la forma en que lo "malo" de un cruce es calculado con diferentes métricas de correlación. Por ejemplo, la métrica de Tau de Kendall aplica el mismo peso a cualquier cruce de igual distancia, sin importar donde ocurra (similar a la métrica de área ROC). Por lo tanto, un cruce entre las recomendaciones 1 y 2 será tan malo como un cruce entre las recomendaciones 1000 y 1001. Sin embargo, si es más probable que el usuario considere las primeras cinco y probablemente nunca examine los ítems rankeados en los miles, el cruce entre 1 y 2 debería tener un impacto negativo substancialmente mayor en el resultado de la métrica (Herlocker et al., 2004).

La métrica de correlación de Spearman no soporta bien los órdenes débiles (parcialmente). Los órdenes débiles ocurren cuando existen por lo menos dos ítems en el ranking donde ninguno de esos ítems es preferido sobre el otro. Si un ranking no es de orden débil entonces se llama orden completo. Si el ranking del usuario (basado en sus calificaciones) es un orden débil y el ranking del sistema es un orden completo, entonces la correlación de Spearman será penalizada por cada par de ítems en los que el usuario ha calificado igual y el sistema rankea en niveles diferentes. Esto no es ideal, ya que al usuario no debería importarle como el sistema ordena ítems que el usuario ha calificado al mismo nivel. La métrica de Tau de Kendall sufre del mismo problema, aunque en menor medida que la correlación de Spearman (Herlocker et al., 2004).

5.4.2 Métrica de utilidad Media-Vida (half-life)

Breese (Breese et al., 1998), presentó una nueva métrica de evaluación para sistemas recomendadores que está diseñada para las tareas donde el usuario es presentado con una lista rankeada de resultados y es poco probable que mire muy profundo en la lista rankeada. Otra descripción de esta métrica puede ser encontrada en (Heckerman, 2000). La tarea para la que la métrica fue diseñada es un recomendador de páginas de Internet. Ellos reclaman que la mayoría de los usuarios de Internet no miraran muy profundo en los resultados de los motores de búsqueda.

Esta métrica de utilidad de media-vida intenta evaluar la utilidad que tiene una lista rankeada para el usuario. La utilidad está definida como la diferencia entre las calificaciones del usuario para un ítem y la "calificación por defecto" para un ítem. La calificación por defecto es generalmente una calificación neutra o ligeramente negativa. La probabilidad de que un usuario verá cada ítem sucesivo es descrita con una función de descomposición exponencial, donde la fuerza de la descomposición es descrita por un parámetro de media-vida. La utilidad esperada (R_a) se muestra en la ecuación 5.11; $r_{a,j}$ representa la calificación del usuario a en el ítem j de la lista rankeada, d es la calificación por defecto, y α es la media-vida. La media-vida es el rankeo del ítem en la lista que posee un 50% de oportunidad que el usuario lo vea. Breese utilizó una Media-vida de 5 para su experimento, pero notó que utilizando una media-vida de 10 proporcionaba una pequeña sensibilidad adicional al resultado (Herlocker et al., 2004).

$$R_a = \sum_j \frac{\max(r_{a,j} - d, 0)}{2^{(j-1)/(\alpha-1)}} \qquad \textbf{Ecuación 5.11}$$

La puntuación total para un conjunto de datos a través de todos los usuarios (R) se muestra en la ecuación 5.12. R_a^{max} es la utilidad máxima alcanzable si el sistema rankea los ítems en el orden exacto que el usuario los rankeo.

$$R = 100 \frac{\sum_a R_a}{\sum_a R_a^{\max}} \qquad \textbf{Ecuación 5.12}$$

La métrica de utilidad de media-vida es mejor para dominios de tareas donde existe una disminución exponencial en la utilidad real (uno podría considerar la utilidad desde un análisis de radio de costo/beneficio) a medida que el largo de la búsqueda aumenta, asumiendo que la media-vida a y el voto por defecto d son

escogidos de forma apropiada en la métrica de utilidad. La métrica de utilidad aplica la mayoría del peso a los ítems más nuevos, con cada rankeo sucesivo que posea exponencialmente menos peso en la medida. Para obtener valores altos de la métrica, los primeros rankings predichos deben consistir en ítems calificados de forma alta por el usuario. El inconveniente es que si la función real que describe la probabilidad de acceder cada rankeo es significativamente diferente a la exponencial usada en la métrica, si esto ocurre los resultados medidos podrían no ser indicativos del desempeño actual. Por ejemplo, si el usuario casi siempre busca 20 ítems en la lista de recomendación rankeada, entonces la verdadera probabilidad es una función que es constante para los primeros 20 ítems y 0 más tarde (Herlocker et al., 2004).

La métrica de utilidad de media-vida puede ser demasiado sensible en dominios con preferencias de usuarios binarias donde el usuario solo necesita que las recomendaciones principales sean "los suficientemente buenas" o para tareas de usuarios como "Encontrar todos los ítems buenos" donde el usuario desea ver todos los ítems buenos.

Existen otras desventajas de la métrica de utilidad de media-vida. Primero, los órdenes débiles creados por el sistema generan diferentes posibles puntuaciones para el mismo ranking de sistema. Suponiendo que el sistema genera una lista de recomendación, con tres ítems compartiendo el rankeo mas alto. Si el usuario califica esos tres ítems de forma distinta, entonces dependiendo del orden en que el recomendador pone esos ítems, la métrica podría tener valores muy diferentes (si las evaluaciones fueran significativamente diferentes).

Segundo, debido a la aplicación de la función max() en la métrica (ecuación 4.11), todos los ítems que están calificados menor al voto por defecto contribuyen de igual forma al puntaje. Por lo tanto, un ítem evaluado en el ranking del sistema con un 2 que está calificado ligeramente menor a la calificación por defecto (que usualmente indica ambivalencia) tendrá el mismo efecto en la utilidad que un ítem que posea la peor calificación posible. La aparición de predicciones extremadamente incorrectas en los niveles altos de un

ranking de sistema puede socavar la confianza del usuario en el sistema. Las métricas que penalizan estos errores más severamente son preferidas.

Para resumir, la métrica de utilidad de media-vida es la única que hemos examinado que cuenta utilidad no uniforme. De esta manera podría ser apropiada para la evaluación de las tareas "Encontrar Ítems Buenos" en dominios donde se crea que pueda existir utilidad no uniforme. Por otro lado posee muchas desventajas, particularmente cuando se considera una estandarización entre diferentes investigadores. Distintos investigadores podrían usar valores de alfa significativamente diferentes o del voto por defecto – haciendo difícil comparar resultados entre investigadores y haciendo fácil manipular resultados. Además., el parámetro media-vida es poco probable que sea el mismo para todos los usuarios (usuarios diferentes necesitan/desean números de resultados diferentes) (Herlocker et al., 2004).

5.4.3 La medida NDPM

La NDPM fue utilizada para evaluar la precisión del sistema recomendador FAB (Balabanovic y Shoham 1997). Fue propuesto originalmente por Yao (Yao, 1995). Yao desarrolló la NDPM de forma teórica utilizando un enfoque de teoría de decisión y medida. La NDPM adopta una postura de "medida de desempeño basada en distancia normalizada". La NDPM (Ecuación 5.13) puede ser usada para comparar dos rankings ordenados-débilmente diferentes.

$$NDPM = \frac{2C^{-} + C^{u}}{2C^{i}} \qquad \textbf{Ecuación 5.13}$$

C^{-} es el número de relaciones de preferencia contradictorias entre el ranking del sistema y el ranking del usuario. Una relación de preferencia contradictoria se da cuando el sistema dice que el ítem 1 será preferido sobre el ítem 2, y el ranking del usuario dice lo contrario. C^{u} es el numero de relaciones de preferencia compatibles, donde el usuario califica al ítem 1 más alto que el ítem

2, pero el ranking del sistema tiene al ítem 1 y al ítem 2 en el mismo nivel de preferencia. C^i es el número total de relaciones "preferidas" en el ranking del usuario (Ej. Pares de ítems calificados por el usuario en los que uno es calificado más alto que el otro). Esta métrica es comparable entre diferentes conjunto de datos (está normalizada), ya que el numerador representa la distancia y el denominador representa la peor distancia posible (Herlocker et al., 2004).

La NDPM es similar en forma a las correlaciones de Spearman y Tau de Kendall, pero proporciona una interpretación más precisa al efecto de rankeos empatados. Sin embargo, sufre de la misma debilidad de cruces que las métricas de correlación de rankeo (cruces al principio del ranking poseen el mismo peso que los cruces al final del ranking).

Como la NDPM no penaliza al sistema para los órdenes de sistema donde los rankeos del usuario están empatados, ésta puede ser más apropiada que las métricas de correlación para dominios donde el usuario está interesado en ítems que son "lo suficientemente buenos". Las calificaciones de los usuarios podrían ser transformadas a calificaciones binarias (si actualmente no lo son), así la NDPM podría ser usada para comparar el resultado con el ranking del sistema.

Como la NDPM sólo evalúa el orden y no el valor de la predicción, no es apropiada para evaluar tareas donde la meta es predecir un valor de calificación actual (Herlocker et al., 2004).

CAPITULO VI:

"ANALISIS, DISEÑO E IMPLEMENTACION DE LOS ALGORITMOS DE RECOMENDACION"

6.1 Análisis del Problema

El Análisis tiene como finalidad indicar los lineamientos en los cuales se basa la creación y posterior desarrollo de los Algoritmos de recomendación para material educativo. Dentro de éstos indicaremos requerimientos funcionales y técnicos que éstos deben cumplir, además de los casos de uso que satisfacen dichos requerimientos.

6.2 Requerimientos Funcionales

Los Requerimientos funcionales definen el comportamiento interno del software: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que muestran cómo los casos de uso serán llevados a la práctica. En el contexto de este trabajo se presentan a continuación los requisitos con los que debe satisfacer la implementación del algoritmo:

IDENTIFICACION	NOMBRE O ESPECIFICACIÓN	DESCRIPCIÓN
RF_01	Generar Recomendación	Genera una Recomendación de materiales de acuerdo al criterio de cada algoritmo de recomendación y sus criterios de Búsqueda de material.
RF_02	Listar Recomendación	Presenta los resultados de la(s) recomendación(es) generada(s).
RF_03	Emplear Filtros en las Recomendaciones.	Permite utilizar filtro en la generación de recomendaciones.
RF_04	Buscar material para las Recomendaciones.	Permite la búsqueda de material para la generación de recomendaciones.

6.3 Requerimientos Técnicos

Éstos dan el soporte de hardware para que el sistema pueda ejecutarse. En el caso específico de este sistema, se debe contar con los siguientes requisitos para el buen funcionamiento de él, cabe señalar que estos requisitos se subclasificaron en requerimientos de servidor y cliente, los cuales son:

- Requerimientos de Servidor

Para que el sistema funcione se debe tener un computador como servidor, el cual debe contar con las siguientes características:

- Funcionar bajo ambiente Windows 98, ME, 2000, XP, Server 2003 o Vista.
- Tener instalado y funcionando el servidor de aplicaciones JBoss.
- Tener instalado el motor de base de datos Postgres 8.1 o superior.
- Tener instalado el servidor Web apache.
- Tener acceso a Internet.

- Requerimientos de Cliente

En el caso del computador que actuará como cliente, éste debe tener las siguientes características:

- Tener acceso a Internet
- Tener Internet Explorer 5.0 o superior, o Mozilla 1.0 o superior.

6.4 Casos de Uso

Los Casos de Uso (CU) son descripciones narrativas que describen el comportamiento de un sistema desde el punto de vista del usuario. Ayudan a tener una mejor comprensión de los requerimientos del sistema., definir los límites y las relaciones entre el sistema y el entorno (Pérez y Romero, 2007).

El formato utilizado para representar los CU consta de los siguientes elementos:

- Nombre: identifica al Caso de Uso.
- Actores: representan el rol que cumplen las entidades que intervienen en el Caso de Uso.
- Descripción: es un breve relato de lo que se realiza en el Caso de Uso.
- Pre-Condiciones: indica los requisitos necesarios para la ejecución del Caso de Uso.
- Flujo Evento: indica el curso normal de las acciones del Caso de Uso.
- Flujo Evento Alternativos: indica los cursos alternos del Caso de Uso.
- Post- Condiciones: indica que el caso de Uso se realizó con éxito. Con respecto al sistema de recomendación, la postura para señalar el éxito del Caso de Uso será expresada con respecto al estado en que queda la base de datos.

A continuación se detallan los casos de uso asociados a los requisitos que el Algoritmo de recomendación debe satisfacer. Cabe señalar que se tomó como base los usuarios (Profesor y Alumno) descritos en la memoria de título "***Desarrollo de un sistema de recomendación para la búsqueda de material educativo" (***Pérez y Romero, 2007) para el modelo de casos de Uso.

Caso de Uso Buscar Material:

Nombre:	BUSCAR MATERIAL

Descripción:

Permite que el usuario realice la búsqueda de algún material educativo, ingresando un criterio para ello. Esta búsqueda genera un listado de materiales, que sirve como entrada al caso de usos Generar Recomendación.

Actores:

1. Profesor
2. Alumno

Precondiciones:

No tiene

Flujo Normal:

1. El Sistema despliega un formulario para el ingreso de los criterios de búsqueda:

   - Palabra(s) Clave(s)
   - Filtro

2. El usuario presiona el botón "Buscar Material", para comenzar la búsqueda.

3. El sistema utiliza los materiales seleccionados para iniciar la recomendación.

Flujo Alternativo:

Línea 3: El sistema informa al usuario que la búsqueda no arrojó ningún resultado

Poscondiciones:
Los materiales encontrados se utilizan para realizar la recomendación

Caso de Uso Generar Recomendación:

Nombre:	GENERAR RECOMENDACIÓN
Descripción: Genera una recomendación de materiales en base a una búsqueda	
Actores: 3. Profesor 4. Alumno	
Precondiciones: El sistema debe haber generado una lista de materiales a través de la búsqueda.	
Flujo Normal 1. El sistema muestra al usuario activo el listado de recomendaciones generado.	
Flujo Alternativo Línea 1: El sistema muestra un mensaje señalando que no hay recomendación.	
Poscondiciones: Recomendación generada	

6.4.1 Diagrama de Caso de Usos

En la figura 6.1 se presenta el diagrama de casos de uso que representa los requerimientos que el Algoritmo de recomendación debe satisfacer para la búsqueda de material educativo.

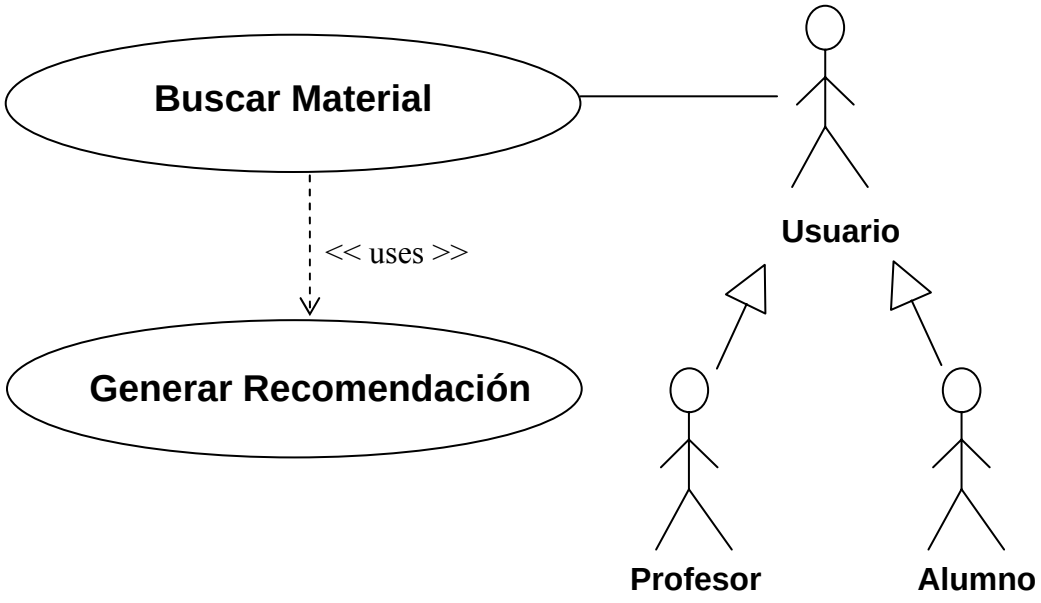


Figura 6.1: Diagrama de Casos de uso

6.4.2 Relación Caso de Uso y Requerimientos Funcionales

		CASO DE USO	
ID	**REQUERIMIENTO**	Buscar Material	Generar Recomendación
RF_01	Generar Recomendación		X
RF_02	Listar Recomendación		X
RF_03	Emplear Filtros en las Recomendaciones.	X	X
RF_04	Buscar material para las recomendaciones	X	

6.4.3 Diagramas de Secuencia

Los diagramas de secuencia muestran gráficamente las operaciones que van desde los actores al sistema en un orden temporal (qué pasa primero, qué pasa después) y las cuales serán implementadas durante la fase de construcción de la aplicación. Este tipo de diagrama se origina a partir de los casos de Uso desde donde se obtienen las operaciones que deben indicarse en él.

La figura 6.2 presenta un diagrama de secuencia general para los casos de Uso del sistema de recomendación.

Generar Recomendación para Ambos Algoritmos

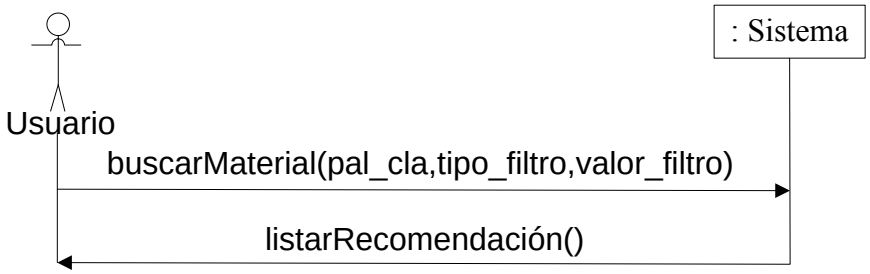


Figura **6.2:** Diagrama general de Secuencia para los casos de uso del sistema de recomendación

6.5 Modelo Conceptual del Dominio

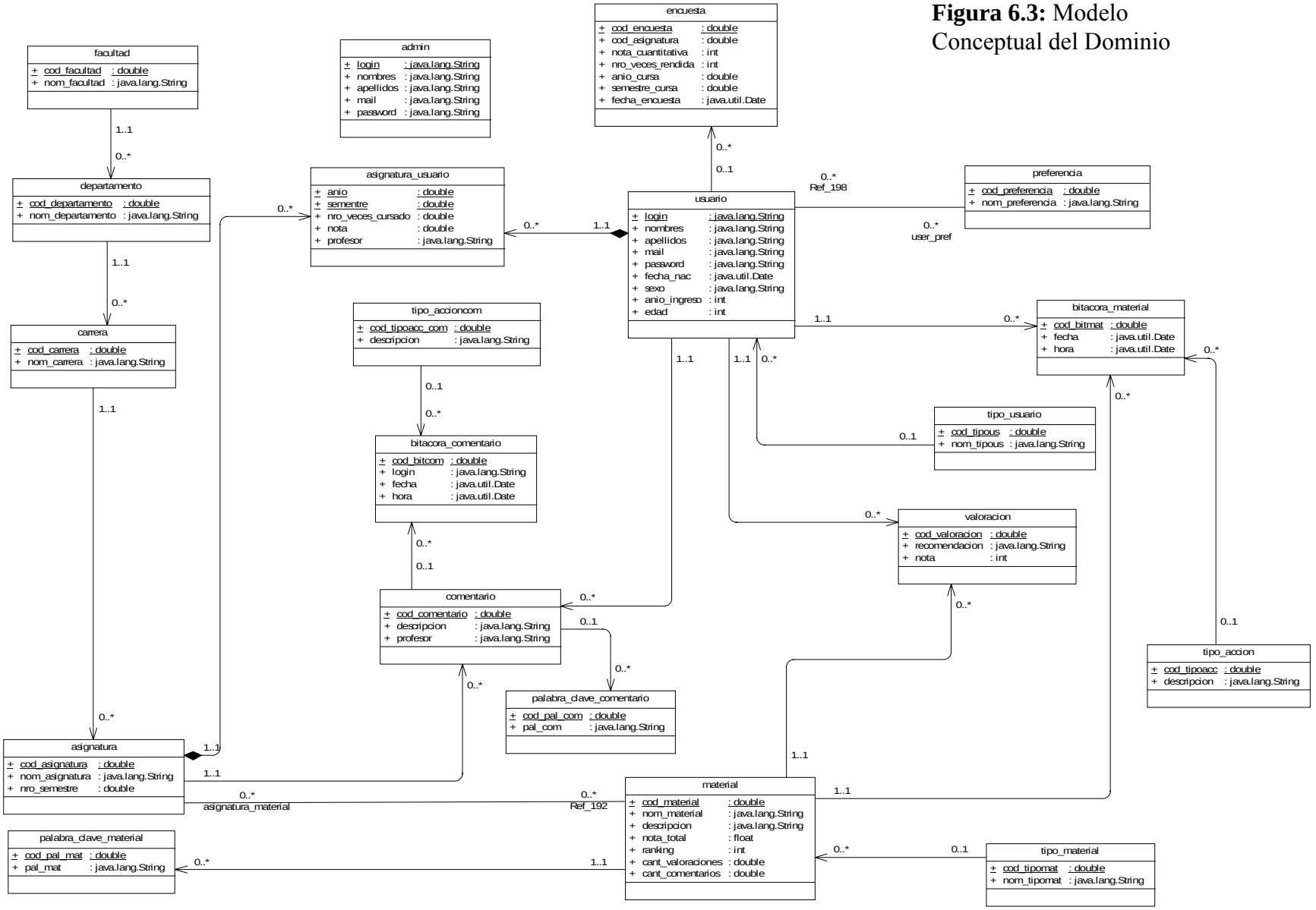


Figura 6.3: Modelo Conceptual del Dominio

El diagrama anterior (Figura 6.3) especifica los conceptos presentes en el dominio de la aplicación representado mediante objetos, que en conjunto con sus atributos y relaciones existentes entre ellos, proporcionan la información requerida para dar cumplimiento al objetivo de la aplicación.

Los conceptos principales de este dominio para el desarrollo de este proyecto son:

Concepto	: material
Descripción	: mantiene información asociada a los materiales que están en el sistema.
Atributos	• cod_material: identificador del material dentro del sistema. • nom_material: nombre o título del material. • descripción: breve reseña del contenido del material. • nota_total: promedio de calificaciones cuantitativas del material. • ranking: clasificación del material en relación a otros materiales. • cant_valoraciones: numero de calificaciones cuantitativas efectuadas al material. • cant_comentarios: cantidad de calificaciones cualitativas realizadas al material.

Nombre	: valoracion
Descripción	: mantiene información de las recomendaciones efectuadas a un material.
Atributos	• cod_valoracion: identificación de la valoración en el sistema. • recomendación: valoración narrativa de un material. • nota: valoración de un material, mediante una nota.

Concepto	: bitácora_material
Descripción	: mantiene información historica asociada a los materiales que están en el sistema.
Atributos	• cod_bitmat: identificador de la bitácora del material dentro del sistema. • fecha: fecha del evento historico del material. • hora: hora del evento historico del material.

Nombre	: palabra_clave_material
Descripción	: mantiene información de las palabras clave asignadas a cada material que está en el sistema.
Atributos	• cod_pal_mat: identificación de la palabra clave del material. • pal_mat: palabra clave del material.

Nombre	: asignatura_material
Descripción	: Ralacion de asignatura con material.
Atributos	• cod_asignatura: identificación de la asignatura en el sistema. • cod_material: identificación del material en el sistema.

6.6 Diseño de los algoritmos de recomendación para la búsqueda de material educativo

El proceso de desarrollo de software es aquel en el que las necesidades del Usuario son traducidas en requisitos de software, estos transformados en diseño y el diseño implementado en código. ¿En qué se basa un buen diseño? Obviamente, no hay un modo sencillo y objetivo de decidir si un diseño es mejor o peor que otro, pero sí que hay ciertas propiedades clave que permiten evaluar su calidad. Lo ideal sería un diseño que funcionara bien en todos los aspectos, es decir, balancear la "rapidez del tiempo de respuesta" y "espacio de almacenamiento" con la "facilidad de uso", pero en la práctica normalmente es necesario sacrificar algún aspecto a cambio de otro.

Un sistema dinámico y automatizado requiere un análisis exhaustivo y preciso, para lograr una óptima definición del diseño que se implementará. Para lograr esto, la siguiente sección da a conocer el diseño elaborado para los Algoritmos de recomendación que serán integrados al Sistema de Recomendación de material educativo.

Antes de realizar el diseño de los algoritmos de recomendación debemos señalar que se implementarán dos algoritmos basados en técnicas distintas, el primero, basado en el filtrado colaborativo pasivo y el segundo, basado en la teoría de las técnicas de filtrado colaborativo, por usuarios e ítems (combinadas), utilizando el concepto de vecinos más cercanos, como se describen el capítulo III. Además, para el diseño de ambos algoritmos se utilizarán patrones de diseño con el fin de ser integrados al sistema de recomendación actual.

A continuación se detallan los casos de uso asociados a los requisitos que cada algoritmo de recomendación específico debe satisfacer.

6.6.1 Algoritmo 1: Basado en Descargas

Nombre:	BUSCAR MATERIAL

Descripción:

Permite que el usuario realice la búsqueda de algún material educativo, ingresando un criterio para ello. Esta búsqueda genera un listado de materiales, que sirve como entrada al caso de usos Generar Recomendación.

Actores:

5. Profesor
6. Alumno

Precondiciones:

No tiene

Flujo Normal:

1. El Sistema despliega un formulario para el ingreso de los criterios de búsqueda, los que pueden ser:

   Para el Algoritmo 1 "Por Descargas de material":

   - Palabra(s) Clave(s)
   - Asignatura

   Para el Algoritmo 2 "Selección de Vecinos Más Cercanos":

   - Palabra(s) Clave(s)
   - Criterio de Filtrado de Vecinos

2. El usuario presiona el botón "Buscar Material", para comenzar la búsqueda.

3. El sistema utiliza los materiales seleccionados para iniciar la recomendación.

Flujo Alternativo:

Línea 3: El sistema informa al usuario que la búsqueda no arrojó ningún resultado

Poscondiciones:
Los materiales encontrados se utilizan para realizar la recomendación

6.6.2 Algoritmo 2: Basado en Vecinos más Cercanos

Nombre:	GENERAR RECOMENDACIÓN

Descripción:

Genera una recomendación de materiales en base a una búsqueda

Actores:

7. Profesor
8. Alumno

Precondiciones:

El sistema debe haber generado una lista de materiales a través de la búsqueda.

Flujo Normal Para el Algoritmo 1 "Por Descargas de material":

2. El Sistema genera un listado ordenado de los materiales que no se descartaron y lo presenta al usuario como una recomendación.

Flujo Normal Para el Algoritmo 2 "Selección de Vecinos Más Cercanos":

El sistema muestra al usuario activo el listado de recomendaciones generado.

Flujo Alternativo Para el Algoritmo 1 "Por Descargas de material":

Línea 3: El sistema muestra un mensaje señalando que no hay recomendación si se descartaron todos los materiales arrojados por la búsqueda.

Flujo Alternativo Para el Algoritmo 2 "Selección de Vecinos Más Cercanos":

Línea 5: El sistema muestra un mensaje señalando que no hay recomendación si no existe correlación entre el usuario activo y los demás usuarios del sistema.

Poscondiciones:

Recomendación generada

6.7 Diagramas de Colaboración

Los diagramas de colaboración representan la interacción entre objetos para efectuar alguna de las operaciones representadas en los diagramas de secuencia. Esta interacción se representa mediante el flujo de mensajes intercambiados entre los objetos.

A continuación se presentan los diagramas de colaboración del sistema de recomendación, específicamente del funcionamiento de los Algoritmos de Recomendación.

Diagrama de colaboración algoritmo basado en descargas:

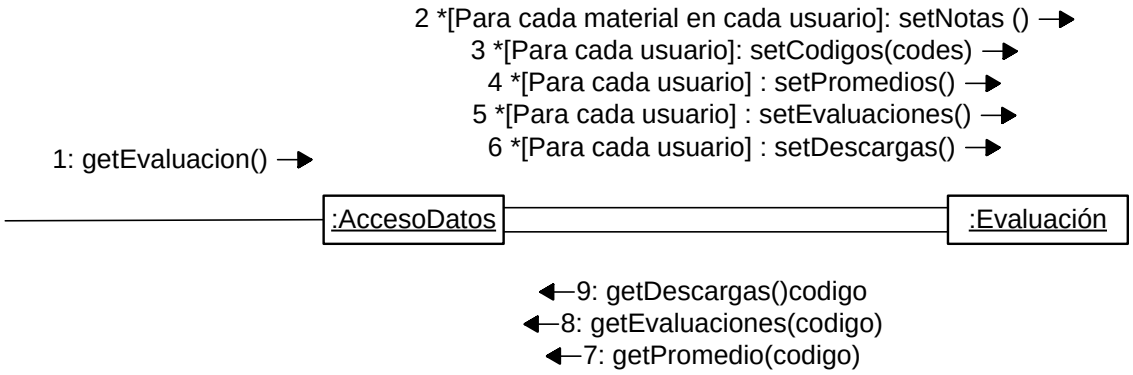


Figura 6.4: Diagrama de colaboración algoritmo basado en descargas.

Diagrama de colaboración algoritmo basado en vecinos más cercanos:

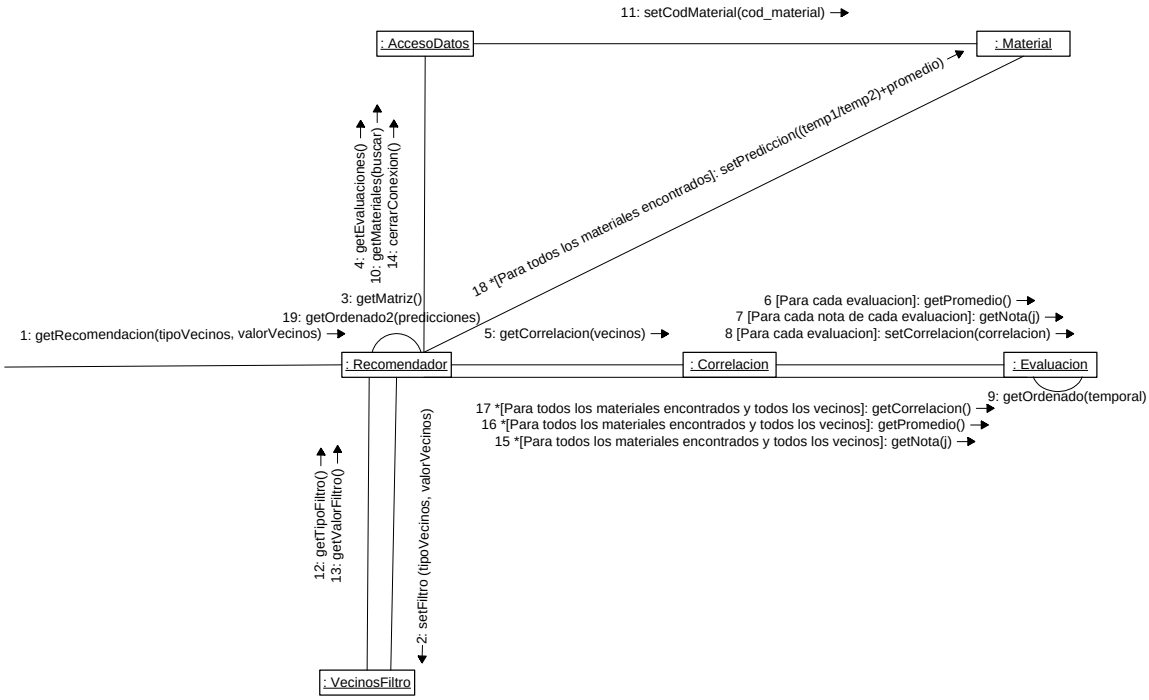


Figura 6.5: Diagrama de colaboración algoritmo basado en vecinos más cercanos**.**

6.8 Diseño de Datos

Para que los Algoritmos puedan generar una Recomendación y, al igual como lo plantean Pérez y Romero en su memoria de título, se necesita mantener en él Sistema los datos "básicos" asociados a:

Entrada de Datos

CÓDIGO	NOMBRE ÍTEM	DETALLE DE DATOS
EE_01	Datos Usuario	Nombres, apellidos, ramos en curso, materias preferidas, correo electrónico
EE_02	Datos Búsqueda	Palabra clave, criterio, materia o ramo, tipo de respuesta esperada(documento, link, trabajo, certamen, comentario por material, comentario por asignatura)
EE_03	Datos Material	Nombre, tipo, materia, ramo, valoración (ranking)
EE_04	Datos Asignatura	Nombre, código

Salida de Datos

Código	Nombre Ítem	Detalle De Datos
ES_01	Recomendación	Nombre material, materia o ramo, tipo de material (documento, link, trabajo, certamen, comentario por material, comentario por asignatura),link para descargarlo (si es aplicable)

6.9 Diseño basado en J2EE

J2EE es una plataforma que nace como respuesta a la necesidad del mercado relacionada con el desarrollo del software, la cual era el contar con herramientas y medios que permitan construir aplicaciones empresariales. J2EE es una tecnología versátil, ya que los componentes de las aplicaciones que se construyen con J2EE se comunican entre sí detrás de las cámaras mediante métodos estándar de comunicación como son HTTP, SSL, XML, RMI E IIOP (Keogh, 2003).

Esta nueva plataforma, ha sido diseñada para aplicaciones distribuidas que son construidas en base a componentes[6], los cuales interaccionan entre sí para formar parte de una aplicación J2EE. Un componente de esta plataforma debe formar parte de una aplicación y ser desplegado en un contenedor, o sea en la parte del servidor J2EE que le ofrece al componente ciertos servicios de bajo nivel y de sistema.

J2EE trabaja bajo el modelo de multicapas, es decir, que realiza una división de la aplicación en diferentes niveles, según su tarea particular. Típicamente una aplicación J2EE, puede tener 4 capas: cliente, negocio, presentación y datos.

En la siguiente sección se indica cómo la arquitectura J2EE (figura 6.6) está presente en el desarrollo de la aplicación. Como se mencionó anteriormente, J2EE consta de cuatro capas, de las cuales se trabajo con:

- Capa Negocio: define el modelo a utilizar para EJB. En esta capa se encuentra la lógica que resuelve o cumple las necesidades de un negocio particular, se maneja mediante Enterprise JavaBeans. Esta capa se encuentra alojada en el servidor de aplicaciones y es sobre ésta donde se acoplan los Algoritmos de recomendación Implementados, cuyos modelos de clases señalan en las figuras 6.9 y 6.10.

[6] Un componente es una unidad funcional de software

La figura 6.6 representa la implementación de los algoritmos integrados en la capa lógica de negocio sobre la arquitectura J2EE, que es la que utiliza el sistema actual de recomendación de material educativo.

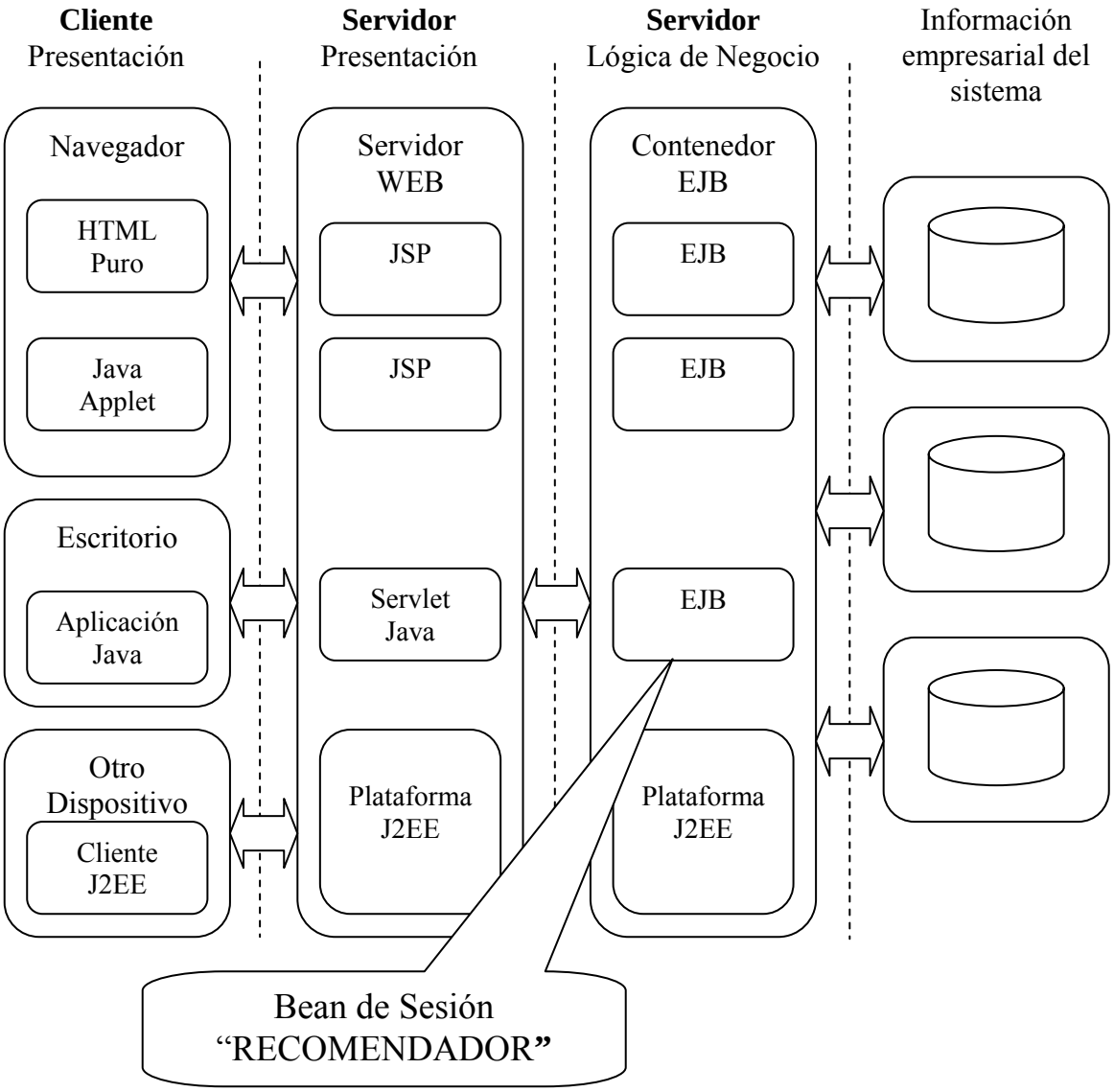


Figura 6.6: Integración de Algoritmos a la Arquitectura J2EE

Cabe destacar que sólo se implementarán como un Bean de Sesión que utiliza clases específicas, no requiriendo ningún tipo de Bean de entidad, ya que por las características de los algoritmos esto es innecesario.

Para el diseño de ambos algoritmos se utilizaron patrones, ofreciendo a cada uno una mayor claridad, portabilidad, capacidad de integración y escalabilidad. Los patrones de diseño utilizados en cada algoritmo se describen a continuación:

Modelo-Vista-Controlador: La mejor práctica para simplificar la distribución de la funcionalidad de la Aplicación es utilizar la estrategia Modelo-Vista-Controlador (MVC) que apoya Sun Microsystems[7] y que tiene sus fundamentos en la tecnología Smalltalk[8], que ya tiene varias décadas de antigüedad (Keogh, 2003).

La estrategia MVC fundamentalmente divide la aplicación en tres grandes componentes (Ver figura 6.7), estos son:

- **El Modelo**: en donde se encuentran los componentes que controlan los datos que utiliza la aplicación.
- **La Vista**: en donde se encuentran los componentes que presentan datos al cliente.
- **El Controlador**: que es responsable de la gestión de eventos y de coordinar las actividades del modelo y la vista.

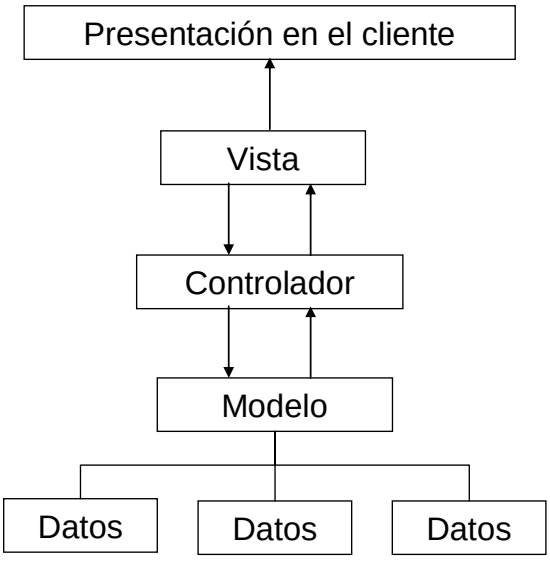


Figura 6.7: Patrón de Diseño Modelo-Vista-Controlador

[7] Empresa informática creadora de la plataforma de programación Java
[8] sistema informático que permite realizar tareas de computación mediante la interacción con un entorno de objetos virtuales

Patrón Estrategia: permite mantener un conjunto de algoritmos de los que el objeto cliente puede elegir aquel que le conviene e intercambiarlo según sus necesidades (Dodero y Fernández, 2002).

Los distintos algoritmos se encapsulan y el cliente trabaja contra un objeto contexto o Context. Como hemos dicho, el cliente puede elegir el algoritmo que prefiera de entre los disponibles o puede ser el mismo objeto Context el que elija el más apropiado para cada situación (ver figura 6.8).

Cualquier programa que ofrezca un servicio o función determinada, que pueda ser realizada de varias maneras, es candidato a utilizar el patrón estrategia. Puede haber cualquier número de estrategias y cualquiera de ellas podrá ser intercambiada por otra en cualquier momento, incluso en tiempo de ejecución.

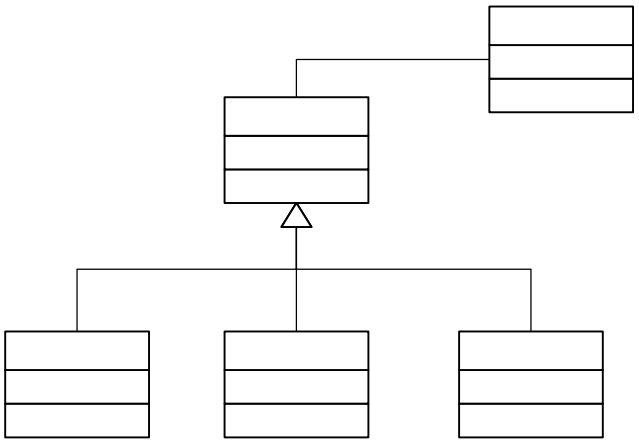


Figura 6.8: Patrón de Diseño Estrategia

Las figuras 6.9 y 6.10 muestran el diagrama de clases de cada algoritmo con los respectivos patrones de diseño aplicados a su estructura.

6.10 Modelos de Clases de los Algoritmos de Recomendación

Modelo de Clases Algoritmo de recomendación Basado en Descargas

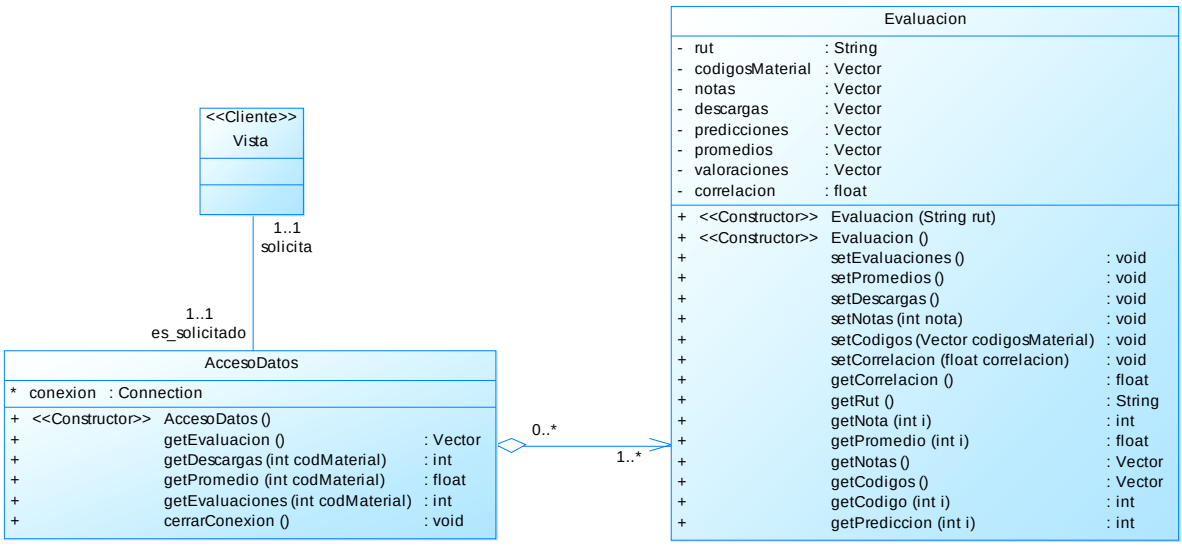


Figura 6.9: Modelo de clases para la Implementación del Algoritmo de Recomendación Basado en Descargas

Modelo de Clases Algoritmos de Recomendación Basado en la Búsqueda de Vecinos Más Cercanos

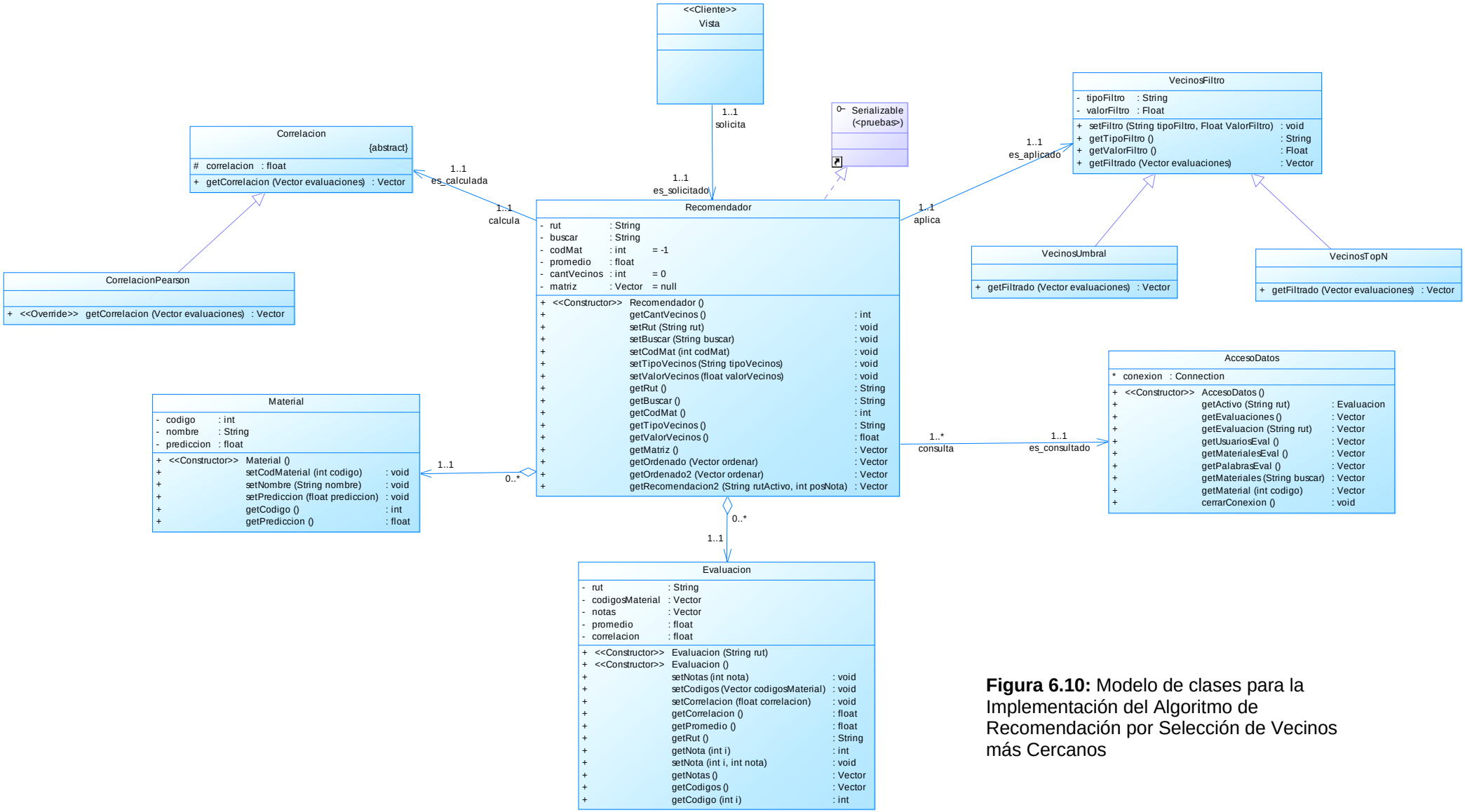


Figura 6.10: Modelo de clases para la Implementación del Algoritmo de Recomendación por Selección de Vecinos más Cercanos

Para el diseño de ambos algoritmos se utilizó el patrón Modelo-Vista-Controlador (figura 6.7), ya que se trata de realizar un diseño que desacople la vista del modelo, con la finalidad de mejorar la reusabilidad. De esta forma las modificaciones en las vistas impactan en menor medida en la lógica de negocio o de datos. Además, gracias a este patrón, la integración de los algoritmos al Sistema de Recomendación es mucho más fluida.

Por otra parte, el patrón Estrategia sólo se utilizó para el algoritmo de recomendación por selección de vecinos más cercanos (figura 6.10), ya que éste utiliza, en primer lugar, un cálculo de correlación entre usuarios (clase "Correlacion"), el que puede ser implementado de varias formas dependiendo de la ecuación de correlación que se desee utilizar (subclases de "Correlacion"), y en segundo lugar, un filtrado de vecinos más cercanos (Clase "VecinosFiltro"), que al igual que el caso anterior se puede aplicar de más de una forma (subclases de "VecinosFiltro").

Otro punto de vital importancia es la integración de ambos algoritmos a la arquitectura J2EE. En el caso de ambos algoritmos la integración es a través de la capa de lógica de negocio definida en la arquitectura, en esta capa se aloja la parte principal de cada algoritmo, definida como la que ejecuta sólo la lógica de negocio dentro del patrón MVC en el modelo de cada algoritmo, como un bean de sesión que implementa la funcionalidad para realizar la recomendación. Para el caso del algoritmo basado en descargas (figura 6.9) la clase utilizada como bean de sesión es la clase "AccesoDatos" y para el algoritmo basado en vecinos más cercanos es la clase "Recomendador". Además, cabe destacar que las demás clases serán utilizadas por este bean de sesión y no formarán parte de él.

6.11 Implementación de los Algoritmos de Recomendación

En el desarrollo de éste trabajo se adaptaron una mezcla de técnicas de filtrado colaborativo, para complementar las técnicas que el Sistema de Recomendación utilizaba originalmente (Algoritmo de Recomendación basado en promedio). Particularmente se adaptaron técnicas generales de *"filtrado colaborativo, basado en filtraje pasivo"* (Algoritmo por descargas), y luego se utilizaron técnicas de *"filtrado colaborativo basado en usuarios y contenidos o ítems*" (Algoritmo por selección de vecinos más cercanos), ya que de acuerdo a la información con la que cuenta el sistema y a lo estudiado en los capítulos anteriores resultaron ser una buena alternativa de desarrollo, no dejando de existir otras que pueden ser implementadas en trabajos posteriores.

Pasos Generales del Algoritmo Basado en Descargas para Realizar Recomendaciones

1. Identificar la carrera de usuario activo (usuario que realiza la búsqueda y que solicita la recomendación): esto es importante, porque, dependiendo de la carrera, cada usuario buscará temas y documentos con distintos grados de detalle.

2. Luego para poder realizar una búsqueda el sistema solicita la asignatura de la cual desea buscar información, asociada a esto, se ingresa una palabra clave que permite acotar la búsqueda y se ingresa el tipo de material que se desea buscar, este puede ser: link, trabajo, documento y pauta certamen. Con esta información se busca en forma histórica al interior de la base de datos, todos los usuarios que sean de la misma carrera, y que en algún momento hayan realizado la misma búsqueda.

3. Una vez que se tiene este grupo de usuarios, se busca los materiales que ellos han recuperado cuando realizaron la misma búsqueda, independiente si han entregado alguna valoración o comentario.

4. Se calcula la cantidad de veces que ha sido descargado cada documento

5. Posteriormente se descartan aquellos documentos que tengan un promedio de valoraciones muy baja, y en donde este promedio ha sido obtenido por las valoraciones de una cantidad significativa de usuarios, es decir que más de la mitad de los usuarios que han descargado el documento lo hayan calificado con baja nota.

6. Finalmente se calcula una valoración a cada uno de los documentos obtenidos, en base a las descargas registradas, y se recomienda al usuario el que tenga la valoración más alta, asumiendo que, lo que les ha servido a los demás usuarios es probable que también le sirva al usuario que está realizando la búsqueda o solicitando la recomendación (usuario activo).

Es importante destacar que los documentos que no registran valoración ni comentarios, tienen un promedio de valoraciones igual a cero, y que para efectos en las pruebas se asoció una nota por descarga, es decir, para materiales cuya cantidad de descargas fueron 4 se le asoció una nota 4 y así sucesivamente hasta evaluar con nota 10 aquellos materiales que registraron ésta cantidad de descargas o superior.

Considerando que las valoraciones fluctúan entre 1 y 10, se considera como un promedio de valoraciones muy bajo cualquier promedio inferior a 4.

Se considera una cantidad significativa de usuarios que ha valorado un documento, cuando (a modo de ejemplo), un documento ha sido descargado por 20 personas, y ha sido valorado por más de 10 personas, que equivalen a más de la mitad de usuarios que ha descargado el documento (cantidad de personas que han valorado > al total de descargas que registra el documento / 2)

Pseudo-código del Algoritmo por Descargas

INICIO

Capturar la asignatura y palabra clave de la búsqueda

SI (palabra clave es distinta de vacío**) ENTONCES**

{

/* Buscar si la palabra clave ingresada existe en la base de datos */

SI (palabra clave existe en la base de datos**) ENTONCES**

{

Ejecutar consulta que realiza la búsqueda de los documentos descargados por usuarios de la misma carrera que el usuario activo, y que han buscado la misma asignatura y palabra clave.

Guardar el resultado de la consulta en un arreglo que contiene el documento (código del documento), cantidad de veces que se ha descargado, promedio de valoraciones y cantidad de usuarios que han registrado valoraciones.

Sumar el total de descargas que se han efectuado cuando se ha realizado la misma búsqueda

RECORRER MIENTRAS (el arreglo sea distinto de vacío)

{

SI (0 < promedio de valoraciones < 4 **Y** cantidad de usuarios que han registrado valoración > cantidad de veces que se ha descargado / 2**) ENTONCES**

{

Eliminar documento del arreglo

/* Calcular el porcentaje de veces que se ha descargado cada documento con respecto al total de descargas */

Cantidad de veces que ha sido descargado cada documento * 100 / total de descargas

Traspasar los datos a otro arreglo en donde se guarden los siguientes datos: documento (código), porcentaje de descargas y promedio de valoraciones.

} //FIN DE CONDICION QUE ELIMINA DOCUMENTOS

} //FIN CICLO

} //FIN DE CONDICION QUE BUSCA LA PALABRA CLAVE

SI (no se encuentra la palabra clave en la base de datos) ENTONCES

{

No Se Encontraron Documentos Con Esas Características.

}

} //FIN DE CONDICION QUE REVISA QUE EL CAMPO DE PALABRA CLAVE ES DISTINTO DE VACIO

SI (el campo de palabra clave es vacío**) ENTONCES**

{

/* Realizar búsqueda solo por asignatura */

Ejecutar consulta que realizar la búsqueda de los documentos descargados por usuarios de la misma carrera que el usuario activo, y que han buscado la misma asignatura.

Guardar el resultado de la consulta en un arreglo que contiene el documento (código del documento), cantidad de veces que se ha descargado, promedio de valoraciones y cantidad de usuarios que han registrado valoración.

Sumar el total de descargas que se han efectuado cuando se ha realizado la misma búsqueda

RECORRER MIENTRAS (el arreglo sea distinto de vacío)

{

SI (0 < promedio de valoraciones < 5 Y cantidad de usuarios que han registrado valoración > cantidad de veces que se ha descargado / 2**)**

ENTONCES

{

Eliminar documento del arreglo

/* Calcular el porcentaje de veces que se ha descargado cada documento con respecto al total de descargas */

Cantidad de veces que ha sido descargado cada documento * 100 / total de descargas

Traspasar los datos a otro arreglo en donde se guarden los siguientes datos: documento (código), porcentaje de descargas y promedio de valoraciones.

} //FIN IF QUE ELIMINA DOCUMENTOS

} //FIN CICLO

Se recomienda al usuario los 3 documentos más descargados por otros usuarios que han hecho la misma búsqueda, con su respectivo promedio de valoraciones

FIN

Pasos Generales del Algoritmo por Selección de Vecinos Más Cercanos para Realizar Recomendaciones

1. Definir Matriz Usuarios v/s Materiales y llenarla con la nota asignada a cada material por cada usuario que ha evaluado al menos un material.

2. En base a la matriz calcular, para el usuario activo la correlación de Pearson (Ecuación 3.2) que posee con los demás usuarios.

3. Seleccionar los vecinos más cercanos según corresponda (cantidad máxima o correlación mínima).

4. Seleccionar los materiales que coincidan con el parámetro de búsqueda.

5. Para cada material encontrado calcular la predicción de evaluación para el usuario activo sobre los vecinos más cercanos.

6. Ordenar las predicciones de forma descendente y recomendar según ese orden.

Debido a que el algoritmo utiliza el concepto de Predicción asociado a una búsqueda de los "Vecinos Más Cercanos" para realizar las recomendaciones el uso de la variable asignatura lo limita enormemente en el cálculo de éstas predicciones, ya que, aunque encontremos algunos vecinos más cercanos en el cálculo de la predicción para los materiales encontrados en la búsqueda, la recomendación dependerá netamente de cuántos de éstos vecinos encontrados evaluaron dichos materiales asociados a la asignatura seleccionada, lo que, para casos prácticos, es muy difícil de lograr. En otras palabras, mientras más filtros se aplican a este tipo de algoritmos, más difícil es lograr una recomendación, por ésta razón se decidió eliminar el filtrado de la búsqueda por asignatura, para brindar al algoritmo un mayor universo de datos y precisa al momento de ejecutar las pruebas.

Pseudo-código del Algoritmo por Selección de Vecinos Más Cercanos

INICIO

Capturar la palabra clave de la búsqueda y el tipo de filtro para los vecinos más cercanos

SI (palabra clave es distinto de vacío) **ENTONCES**

{

/*Buscar si la palabra clave ingresada existe en la base de datos*/

SI (palabra clave existe en la base de datos) **ENTONCES**

{

Obtener una matriz Usuarios vs Materiales (tabla 2.1) en donde se especifique la evaluación de cada usuario a cada material existente en el sistema (La matriz se representa en base a un Vector de evaluaciones por usuario con las distintas evaluaciones para cada material).

RECORRER MIENTRAS (existan usuarios en la matriz Usuarios vs Material)

{

Calcular la correlación (Ecuación 3.2) que posee el usuario con respecto al usuario activo y agregar la información al Vector de evaluaciones.

}

Ordenar el Vector de evaluaciones según la correlación en forma descendente.

RECORRER MIENTRAS (existan usuarios en la matriz Usuarios vs Material)

{

SI (correlación usuario es igual a cero)

{

Eliminar al usuario del Vector de evaluaciones.

}

}

SI (tipo filtro es igual a cantidad)

{

SI (Valor de filtro > 0 **Y** Tamaño del vector de evaluaciones < Valor filtro)

{

Eliminar del vector de evaluaciones los elementos que sobrepasen la cantidad especificada por el filtro

}

}

EN CASO CONTRARIO SI (tipo filtro es igual a factor correlación)

{

Eliminar del vector evaluaciones los elementos que posean una correlación menor al valor del filtro

}

RECORRER MIENTRAS (existan materiales encontrados en la búsqueda inicial)

{

Calcular la predicción de evaluación del usuario activo para el material encontrado (Ecuación 4.6)

Agregar la predicción del material al material encontrado dentro del vector de materiales encontrados

}

Ordenar el vector de materiales encontrados, por la predicción calculada, de forma descendentemente.

Listar el vector de materiales encontrados a modo de recomendación.

} **EN CASO CONTRARIO**

{

Imprimir mensaje que especifica que no se encontraron materiales

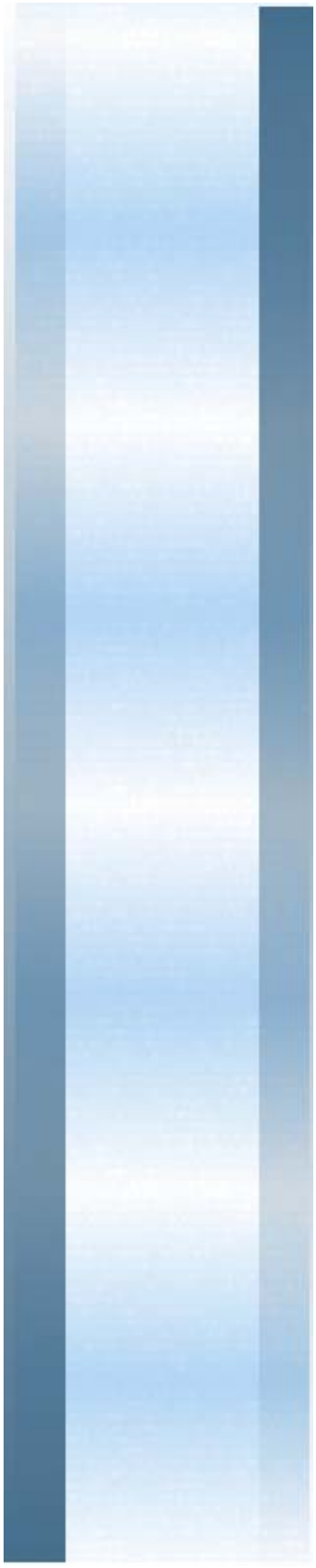
}

} **EN CASO CONTRARIO**

{

Imprimir mensaje que especifica que la búsqueda no puede ser vacía

}



CAPITULO VII:

"RESULTADOS"

Para poder llevar a cabo las pruebas de los algoritmos fue necesario implementar un programa que generara datos aleatorios (ver Anexo C), utilizando como parámetros configurables la cantidad de usuarios, materiales, bitácoras y promedio de valoraciones, y los almacenara en la base de datos de la aplicación. De esta forma, se asegura la generación de valoraciones de distintos materiales para cada alumno (elegidos al azar para cada alumno), y así se generen perfiles de alumnos con gustos distintos para diferentes materiales. Es importante mencionar que se genera una cantidad distinta de descargas y valoraciones para cada alumno. Debido a que el sistema de recomendación no se encontraba en marcha y, por lo tanto, no contaba con información referente al registro de usuarios, almacenamiento de materiales y votaciones como también descargas hechas sobre éstos por parte de los usuarios, información necesaria para aplicar los algoritmos de recomendación.

Actualmente el sistema cuenta con un Algoritmo de recomendación basado en promedios, cuya función era la de calcular el promedio de las votaciones de un material y asignarle un ranking de acuerdo al resultado que arrojaba el promedio, y de acuerdo a este ranking el sistema recomendaba algún material al usuario que hacía uso del sistema.

Las pruebas hechas al sistema comprometieron los tres algoritmos: basado en el promedio, basado en descargas y basado en la selección de vecinos más cercanos, para ello se trabajó en base a cuatro posibles escenarios, utilizando para ello como métrica de evaluación el MAE, que es el valor obtenido por cada algoritmo reflejado en el eje Y de cada gráfico.

1. La primera prueba consistió en evaluar el comportamiento de los algoritmos con un número variable de alumnos registrados en él, partiendo con un total de 5 hasta llegar a 1000 alumnos, manteniendo constante la cantidad de materiales disponibles en el sistema (300), la cantidad de asignaturas(40) y la cantidad máxima de descargas por alumno(10) y como se aprecia en el gráfico N°1 resulta ser mejor el algoritmo basado en vecinos más cercanos cuyo valor MAE es menor al de los otros dos algoritmos que presentan un comportamiento similar, lo que se verificó para todos los valores de alumnos probados.

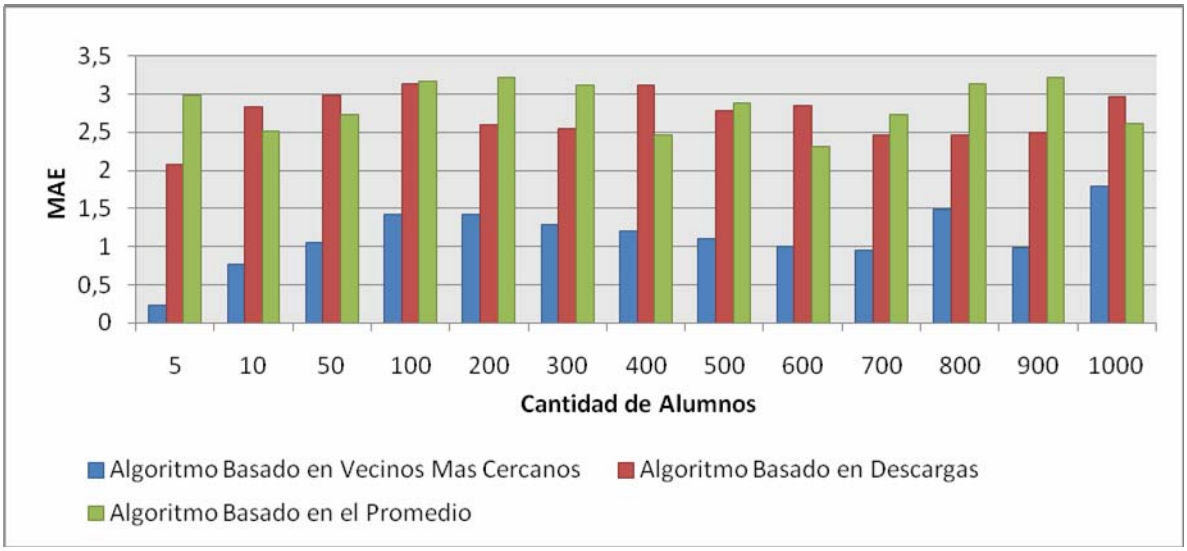


Gráfico Nº 1: Resultados de la Evaluación de Algoritmos por cantidad de Alumnos

2. La segunda prueba consistió en evaluar el comportamiento de los algoritmos con un número variable de materiales, partiendo con un total de 100 hasta llegar a 900 materiales, manteniendo constante la cantidad de alumnos o usuarios registrados en el sistema (300) que evaluaron al menos un material, la cantidad de asignaturas (40) y la cantidad máxima de descargas por alumno (10), y como se aprecia en el gráfico Nº 2 resulta ser más efectivo en la recomendación de material el algoritmo basado en Vecinos más cercanos, aunque con 900 materiales el algoritmo basado en el promedio se asemeja al algoritmo basado en la búsqueda de vecinos más cercanos.

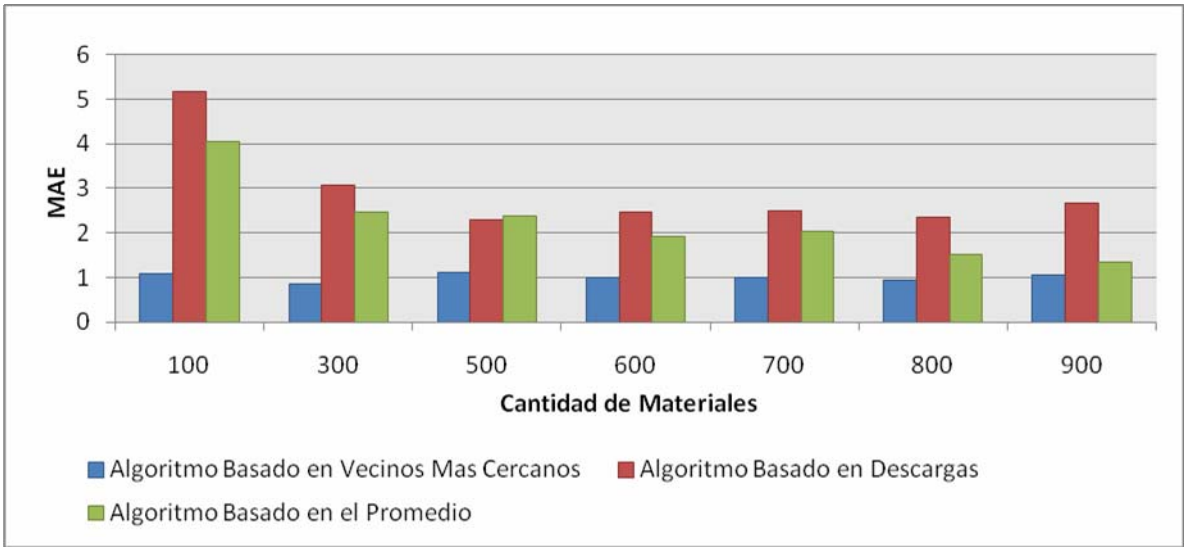


Gráfico Nº 2: Resultados de la Evaluación de Algoritmos por cantidad de Materiales

3. La tercera prueba consistió en evaluar el comportamiento de los algoritmos con un número variable de alumnos y materiales presentes en el sistema, manteniendo constante los demás valores, tales como la cantidad de asignaturas (40) y la cantidad máxima de descargas por alumno(10), y como se aprecia en el gráfico Nº 3, nuevamente resulta ser más efectivo el algoritmo basado en vecinos más cercanos, sin embargo, cuando se igualan la cantidad de alumnos y materiales el valor MAE para el algoritmo basado en descargas y el algoritmo basado en promedio resulta ser igual. Ahora bien, si se mantiene una cantidad de alumnos por sobre los 200 y una cantidad de materiales que fluctúe entre los 300 y 600 el valor del MAE para los tres algoritmos no resulta ser muy alto, lo que significa, por consiguiente, que se obtienen recomendaciones más acertadas.

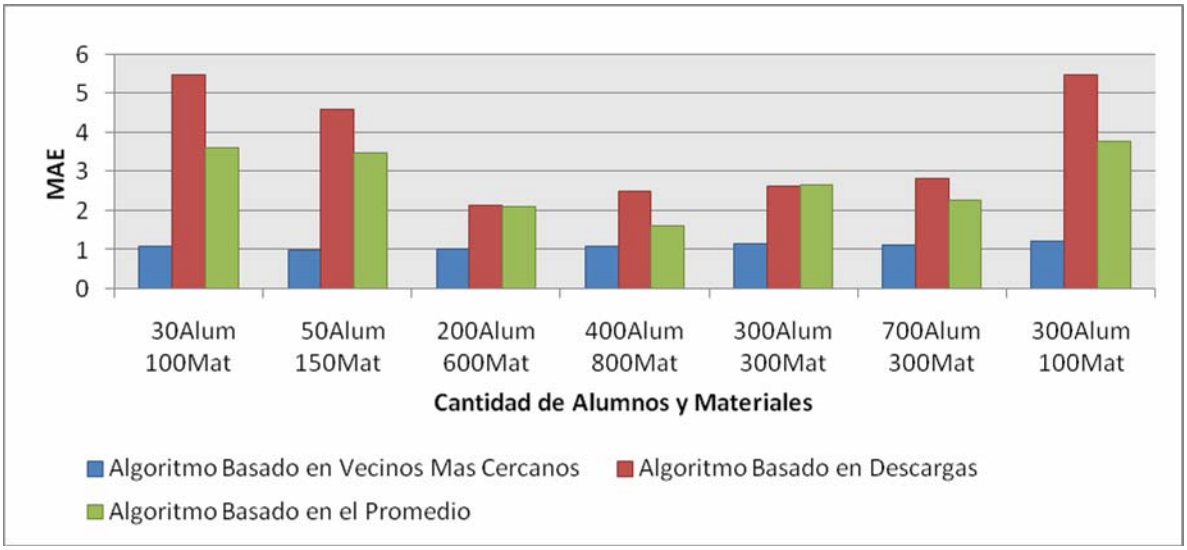


Gráfico Nº 3: Resultados de la Evaluación de Algoritmos por cantidad de Alumnos y Materiales

4. La cuarta prueba consistió en evaluar el comportamiento de los algoritmos con un número variable en el promedio de la recomendación, partiendo desde 0,3 hasta 0,6, manteniendo constante los demás valores, tales como la cantidad Alumnos (100), asignaturas (40), materiales (300) y la cantidad máxima de descargas por alumno(10), y como se aprecia en el gráfico Nº 4 sigue siendo más efectivo el algoritmo basado en vecinos más cercanos, a diferencia de los otros dos que, nuevamente, se muestran similares.

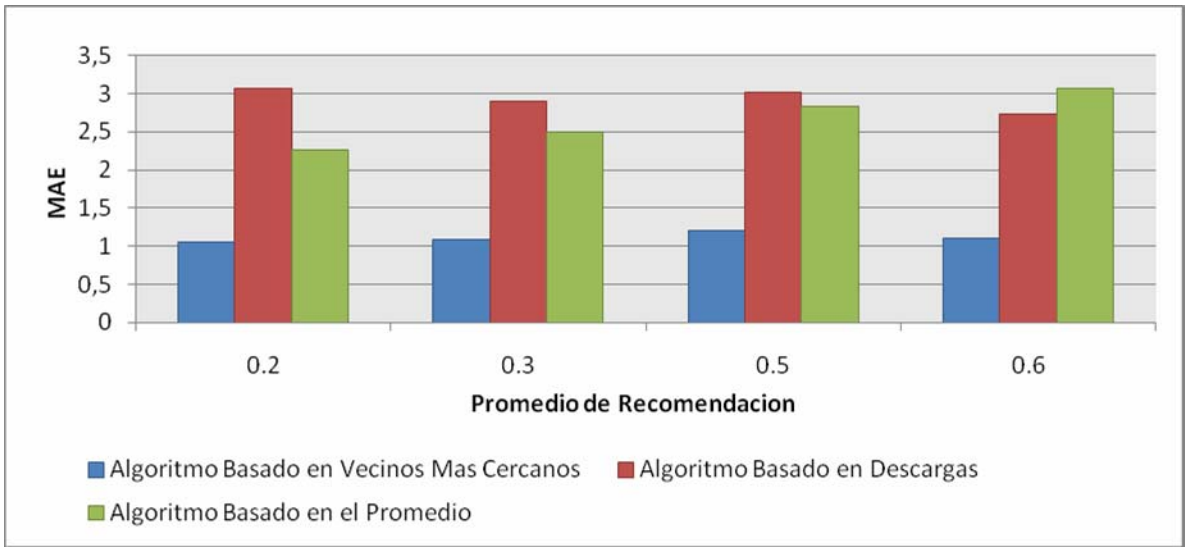


Gráfico Nº 4: Resultados de la Evaluación de Algoritmos por Promedio de Recomendación

Finalmente, de acuerdo a los resultados obtenidos en cada gráfico (cuyo detalle se señala en el Anexo D) para cada caso distinto, resulta ser más efectivo el algoritmo basado en la selección de vecinos más cercanos, independiente de la cantidad de alumnos y materiales que maneje el sistema, ya que presenta menor MAE con respecto a sus pares, lo que trae como consecuencia una mejor predicción del sistema con respecto a las evaluaciones de los usuarios, otorgando de esta manera una recomendación más acertada de algún material.

CONCLUSIÓN

Conclusión del proyecto

En el desarrollo de este trabajo se amplió un Sistema de Recomendación existente desarrollado por alumnas tesistas de la carrera de Ingeniería Civil en Informática (Pérez y Romero, 2007). Dicho sistema es accedido mediante Internet, sirviendo como un repositorio Web que almacena y comparte documentos de interés académico para sus usuarios, en donde, para implementar un nuevo método de recomendación se llevó a cabo una investigación a fondo sobre los sistemas de recomendación.

Bajo estas características se estudiaron diferentes técnicas de recomendación como una alternativa viable en el diseño de herramientas que puedan optimizar las búsquedas, obteniendo resultados personalizados para cada usuario. Se utilizó una base de datos que permite registrar la información del comportamiento de los usuarios al utilizar el Sistema de Recomendación y se seleccionaron e implementaron dos algoritmos, uno basado en descargas y el otro basado en la detección de vecinos más cercanos, consiguiendo, de esta manera, potenciar el Sistema de Recomendación de Material Educativo (SRME), ya que se incorporaron y adaptaron nuevos métodos de recomendación.

Los resultados de las pruebas muestran que el uso del algoritmo de Filtrado Colaborativo basado en la detección de vecinos (usuarios) más cercanos permite mejorar ostensiblemente la calidad de las recomendaciones generadas por el SRME. El uso de información implícita fue un elemento importante a considerar, ya que muchas veces es difícil contar con información explícita por parte de los alumnos. En dicho contexto, el algoritmo basado en descargas realizadas, puede ser de utilidad, pero falta determinar una manera adecuada de utilizarlo, o complementarlo, con otros algoritmos.

Es importante destacar que el desarrollo de este trabajo permitió aplicar los conocimientos adquiridos durante los años de estudio, específicamente, todo lo relacionado a la tecnología de objetos a través del lenguaje de modelamiento unificado (UML) unido al lenguaje de programación Java y a su arquitectura J2EE, siendo una experiencia enriquecedora, ya que no teníamos conocimiento alguno sobre los sistemas de recomendación, sus distintas aplicaciones y lo útiles que resultan ser al momento de realizar una búsqueda. Este trabajo a su vez deja abierta una línea de investigación para la implementación de nuevos algoritmos de recomendación que utilicen técnicas más

complejas (ej. Redes neuronales), además de la posibilidad de mejorar la eficiencia y eficacia de los presentados en este estudio.

Finalmente es importante mencionar que el trabajo realizado sirvió como base para la elaboración de un artículo titulado "Evaluación de Algoritmos de Filtrado Colaborativo para la búsqueda de materiales educativos", y que fue sometido al Encuentro Chileno de Computación 2008, en el marco de las Jornadas Chilenas de la Computación (JCC) 2008.

REFERENCIAS BIBLIOGRAFICAS

Referencias

(Herrera-Viedma et al., 2003) E. Herrera-Viedma, F. Herrera, L. Martínez, J.C. Herrera, A.G. López. "*Incorporating Filtering Techniques in a Fuzzy Linguistic Multi-Agent Model for Information Gathering on the Web*. Fuzzy Sets and Systems", 2003

(Yager, 2003) R.R. Yager. *Fuzzy Logic Methods in Recommender Systems*, Fuzzy Sets and Systems, Volume 136, pp 133-149, 2003.

(Herrera-Viedma et al., 2004) Enrique Herrera-Viedma & Carlos Porcel & Lorenzo Hidalgo. *Sistemas de recomendaciones: herramientas para el filtrado de información en Internet* [on line]. "Hipertext.net", núm. 2, ISSN 1695-5498, 2004.

(Foltz y Dumais, 1992) Foltz, P.W. y Dumais, S.T. "Personalized information delivery: an analysis of information filtering methods". Communications of the ACM, v. 35, pp. 51-60, 1992.

(Resnick et al., 1994) *Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, John Ried,* "GroupLens: An open architecture for collaborative filtering of netnews". Proc. of the ACM Conference on Computer Supported Cooperative Work, pp. 175-186, 1994.

(Resnick y Varian, 1997) P. Resnick, H.R. Varian, Guest Editors. *Recommender Systems*. Communications of the ACM, pp. 56-89, 1997.

(Stodolsky, 1990) Stodolsky, D.S. "Invitational Journals Based Upon Peer Consensus". Psycoloquy, v. 1, 1990.

(Galán, 2007) Sergio Manuel Galán Nieto, Tesis doctoral:"Filtrado Colaborativo y Sistemas de Recomendación", IRC'07, Leganés, Madrid, España, 2007.

(Vélez-Langs y Santos, 2006) Vélez-Langs O.E., SANTOS C."Sistemas Recomendadores: Un Enfoque Desde Los Algoritmos Genéticos". Industrial Data, v.9, n.1, pp. 20 - 27, 2006.

(Bafoutsou y Mentzas, 2002) G. Bafoutsou, G. Mentzas. *Review and Functional Classification of Collaborative Systems.* International Journal of Information Management, pp. 281-305, 2002.

(Sarwar et al., 2001) Sarwar B., Karypis G., Konstan J. y Riedl J. Item-based collaborative ltering recommendation algorithms. En *10th International Conference,* pp. 285.295, 2001.

(Lang, 1995) Lang K. NewsWeeder: Learning to Filter News. En *12t International Conference on Machine Learning (ICML-95)*, pp. 331-339, 1995.

(Smyth y Cotter, 2000) Smyth B. y Cotter P. "A personalized TV listing service for the Digital TV age. *Journal of Knowledge-Based Systems"*, pp. 53.59, 2000.

(Smyth y McClave, 2001) Smyth B. y McClave P. Similarity vs. diversity. En *4th International Conference on Case-Based Reasoning (ICCBR-01)*, págs. 347.361, 2001.

(Burke, 2002) Burke R. Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*, pp. 331-370, 2002.

(Schwab et al., 2001) Schwab I., Kobsa A. y Koychev I. *Learning user interests through positive examples using content analysis and collaborative- ltering*. Draft from Fraunhofer Institute for Applied Information Technology, Germany, 2001.

(Cunningham et al., 2001) Cunningham P., Bergmann R., Schmitt S., Traphöner R., Breen S. y Smyth B. WebSell: Intelligent Sales Assistants for the World Wide Web. En *eBusinees and eWork Conference and Exhibition (E-01)*, pp. 31-41, 2001.

(Boone, 1998) Boone G. Concept Features in RE: Agent, an Intelligent Email Agent. En *2nd International Conference on Autonomous Agents (Agents-98))*, pp. 141.148, 1998.

(Good et al., 1999) Good N., Schafer J., Konstan J., Borchers A., Sarwar B., Herlocker J. and Riedl J. Combining Collaborative Filtering with Personal Agents for Better Recommendations. En *17th National Conference on Arti_cial Intelligence (AAAI-99)*, págs. 439.446, 1999.

(Schafer y Konstan, 2002) J.B. Schafer, J.A. Konstan, J. Riedl. *MetaRecommendation Systems: User-controlled Integration of Diverse Recommendations* .ACM Conference on Information and Knowledge Management (CIKM-02), November 5-7, 2002.

(Basu et al, 1998) Basu C., Hirsh H. y Cohen W. Recommendation as Classi_cation: Using Social and Content-based Information in Recommendation. En *15th National Conference on Artificial Intelligence (AAAI-98)*, págs. 714.720, 1998.

(Hill et al., 1995) Hill W., Stead L., Rosenstein M. y Furnas G. Recommending and Evaluating Choices in a Virtual Community of Use. En *International Conference on Human Factors in Computing Systems (CHI-95)*, págs. 194.201, 1995.

(Shardanand, 1994) Shardanand U. *Social Information Filtering for Music Recommendation*. PhD thesis, Program in Media Arts and Sciences, Massachusetts Institute of Technology, 1994.

(Hayes C. y Cunningham, 1999) Hayes C. y Cunningham P. Smart Radio - a Proposal. En *Technical Report, Trinity College Dublin, Computer Science, TCD-CS-1999-24*, 1999.

(Joachims et al., 1997) Joachims T., Freitag D. y Mitchell T. WebWatcher: A Tour Guide for the World Wide Web. En *16th International Joint Conference on Artificial Intelligence (IJCAI-97)*, págs. 770.777, 1997.

(Orwant, 1995) Orwant L. J. Heterogeneous Learning in the Doppelganger User Modelling System. *User Modeling and User-Adapted Interaction*, 42(2):107.130, 1995

(Breese, 1998) Breese J., Heckerman D. y Kadie C. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. En *14th Conference on Uncertainty in Artificial Intelligence (UAI-98)*, págs. 43.52, 1998.

(Billsus y Pazzani, 1998) Billsus D. y Pazzani M. Learning Collaborative Information Filters. En *15th International Conference on Machine Learning (ICML-98)*, págs. 46.54, 1998.

(Schmitz et al., 2006) Schmitz C., Hotho A., Jäschke R. y Stumme G. Content Aggregation on Knowledge Bases using Graph Clustering. En *3rd European SemanticWeb Conference (ESWC-06)*, págs. 61.75, 2006.

(O'Sullivan et al., 2004) O'Sullivan D., Smyth B., Wilson D. y McDonald K. Improving the Quality of the Personalized Electronic Program Guide. *User Modeling and User-Adapted Interaction*, 14(1):5.36, 2004.

(Han y Karypis, 2005) Han S. y Karypis G. Feature-based recommendation system. En *14th International Conference on Information and Knowledge Management (CIKM-05)*, págs. 446. 452, 2005.

(Rich, 1989) Rich E. Stereotypes and User Modelling *in* Kobsa A and Wahlster W (eds) *User Models in Dialog Systems* Springer, Berlin,1989.

(Krulwich, 1997) Krulwich B. LifeStyle Finder: Intelligent User Profiling Using Large-Scale Demographic Data. *AI Magazine*, 18(2):37.45, 1997.

(Balabanovic y Shoham, 1997) Balabanovic M. y Shoham Y. "Combining Content Based and Collaborative Recommendation.*Communications of the ACM"*, 40(3):1.9, 1997.

(Armstrong et al., 1995) Armstrong R., Freitag D., Joachims T. y Mitchell T. WebWatcher: A Learning Apprentice for the World Wide Web. En *AAAI Spring Symposium on Information Gathering from Heterogeneous Distributed Environments*, págs. 6.12, 1995.

(Kamba et al., 1997) Kamba T., Sakagami H. y Koseki Y. ANATAGONOMY: A Personalized Newspaper on the World Wide Web. *International Journal of Human-Computer Studies*, 46(6):789.803, 1997.

(Martın, 2004) Jose David Martın Guerrero. "Tesis Doctoral: Determinacion de tendencias en un portal web utilizando técnicas no supervisadas. Aplicación a sistemas de recomendaciones basados en filtrado colaborativo.", Valencia – Julio, 2004.

(Chen y Lynch, 1992) Chen H. y Lynch K. J. Automatic construction of networks of concepts characterizing document databases. *IEEE Transactions on Systems, Man and Cybernetics*, 22(1):885.902, 1992.

(Chen et al., 1993) Chen H., Lynch K. J., Basu K. y Ng T. Generating, integrating, and activating thesauri for concept-based document retrieval. *IEEE Experts (special series on Artificial Intelligence in Text-based Information Systems)*, 8(1):25.34, 1993.

(Hopfield, 1982) Hopfield J. J. Neural network and physical systems with emergent collective computational abilities. *National Academy of Science*, 79(4):2254.2258, 1982.

(Knight, 1990) Knight K. Connectionist ideas and algorithms. *Communications on the ACM*, 33(11):59.74, 1990.

(Badrul et al., 2005) Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-Based Collaborative Filtering Recommendation Algorithms WWW10, ACM 1-58113-348-0/01/0005, 2005.

(Lemire y Maclachlan, 2005) Daniel Lemire, Anna Maclachlan Slope One Predictors for Online Rating-Based Collaborative Filtering, SDM05. February 7, 2005

(Herlocker et al., 2004) Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. ACM Trans. Inf. Syst., 22(1):5–53, 2004.

(Salazar G., Oscar Ortega, 2006) Erika J. Salazar G. y Oscar Ortega L, Sistema de Búsqueda Personalizada y Recomendación de Documentación Científica. Universidad de Antioquia Medellin, Colombia Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial. ISSN: 1137-3601 No.30, pp. 25-42, 2006.

(Stauffer y Seaman, 1990) Stauffer, D. R., and N. L. Seaman, 1990: Use of four-dimensional data assimilation in a limited-area mesoscale model. Part I: Experiments with synoptic-scale data. *Mon. Wea. Rev.*, 118, 1250-1277, 1990.

(Pielke, 1984) Pielke, R.A., Mesoscale Meteorological Modeling. Orlando, Academic Press, 611 pp., 1984.

(Hill et al., 1995) Hill, W., Stead, L., Rosenstein, M., Furnas, G. W. Recommending and Evaluating Choices in a Virtual Community of Use. In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems,* ACM Press, 194-201, 1995.

(Breese et al, 1998) Breese, J. S., Heckerman, D., Kadie, C. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98),* COOPER, G. F. and MORAL, S., Eds. Morgan Kaufmann, San Francisco. 43-52, 1998.

(Heckerman et al., 2000) Heckerman, D., Chickering, D. M., Meek, C., Rounthwaite, R., and Kadie, C., Dependency Networks for Inference, Collaborative Filtering, and Data Visualization. *Journal of Machine Learning Research 1*, 49-75, 2000.

(Jensen, 2001) Jensen F. V. *Bayesian Networks and Decision Graphs*. Springer-Verlag, 2001.

(Yao, 1995) Yao, Y. Y., Measuring Retrieval Effectivenes Based on User Preference of Documents. *Journal of the American Society for Information Science 46*, 133-145, 1995.

(Quinlan, 1983) Quinlan J. R. Learning Efficient Classification Procedures and their Application to Chess End Games. En *Machine Learning: An Artificial Intelligence Approach*, págs. 463.482, 1983.

(Shannon y Weaver, 1948) Shannon C. y Weaver W. The mathematical theory of communication. En *University of Illinois Press*, 1948.

(Mobasher et al., 2000) Mobasher B., Cooley R. y Srivastava J. Automatic Personalization Based on Web Usage Mining. *Communications of the ACM*, 43(8):142.151, 2000.

(Arbib, 1995) Arbib M. A. *The handbook of brain theory and neural networks*. Cambridge, 1995.

(Moody y Darken, 1989) Moody J. y Darken C. J. Fast Learning in Networks of Locally Tuned Processing Units. *Neural Computation*, 1(2):281.294, 1989.

(Rosenblatt, 1962) Rosenblatt F., *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan Books, 1962.

(Hampshire y Perlmutter, 1990) Hampshire J. B. y Perlmutter B. A. Equivalence proofs for multilayer perceptron classifiers and the Bayesian discriminant function. En *Connectionist Models Summer School*, págs. 159.173, 1990.

(Sahami, 1993) Sahami M. Learning Non-Linearly Separable Boolean Functions With Linear Threshold Unit Trees and Madaline-Style Networks. En *11th National Conference on Artificial Intelligence (AAAI-93)*, págs. 335.341, 1993.

(Hinton y Sejnowski, 1986) Hinton G. E. y Sejnowski T. J. Learning and relearning in Boltzmann Machines. En *Parallel Distributed Processing*, págs. 282.317. MIT Press, 1986.

(Melville et al., 2002)| Melville P., Mooney R. y Nagarajan R. Content-Boosted Collaborative Filtering for Improved Recommendations. En *18th National Conference on Artificial Intelligence (AAAI-2002)*, págs. 187.192, 2002.

(Rumelhart y MacClelland, 1986) Rumelhart D. E. y MacClelland J. L. *Parallel Distributed Processing. Vol 1,* MIT Press, 1986.

(White y Elmasry, 1992) White B. A. y Elmasry M. I. The digi-neocognitron: a digital neocognitron neural network model for VLSI. *IEEE Transactions on Neural Networks*, 3(1):167-172, 1992.

(Jennings y Higuchi, 1993) Jennings A. y Higuchi H. A User Model Neural Network for a Personal News Service. *User Modeling and User-Adapted Interaction*, 3(1):1.25, 1993.

(Framework Taste) Sean Owen Taste: Collaborative Filtering for Java. http://sourceforge.net/projects/taste/

(Blanco, 2007) Blanco Y., *Tesis doctoral propuesta metodológica para el razonamiento semántico en sistemas de recomendación personalizada y automática. Aplicación al caso de contenidos audiovisuales, 2007,15 -228.*

(Swearingen y Sinha, 2002) Swearingen, K., Sinha, R. (2002). *Interaction Design for Recommender Systems*. DIS2002, London.

(Rafter et al., 1999) Rafter, R. Bradley, K., and Smyth, B. Passive Profiling and Collaborative Recommendation. Proceedings of the 10th Irish Conference on Artificial Intelligence and Cognitive Science, Cork, Ireland, September. 1999.

(Boutilier et al., 2003) Boutilier, C.; Zemel, R.S. y Marlin, B. "Active collaborative filtering". Proc. Of the 19th Annual Conference on Uncertainty in Artificial Intelligence, pp. 98-106, 2003.

(Carrión y Torres, 2007) Carolina Carrión D. y Roger Torres P." Utilización de Minería de Datos, Para Búsquedas en un Repositorio Web", Universidad del Bio Bio, 2007.

(Pérez y Romero, 2007) María José Pérez T. y María José Romero F., "Desarrollo de un Sistema de Recomendación para la Búsqueda de Material Educativo" Universidad del Bio Bio, 2007.

ANEXOS

ANEXO A: Preferencias de los usuarios mediante clasificadores automáticos

Los clasificadores son modelos computacionales capaces de asignar categorías específicas a cada una de las entradas que se les presentan. En el caso de los sistemas de personalización basados en clasificadores, dichas entradas identifican información sobre los productos objetivo y las preferencias de los usuarios activos, mientras que las categorías de salida deciden si estos productos deben ser sugeridos a dichos usuarios, clasificándolos como *interesantes* o *no interesantes*.

A fin de aprovechar tales capacidades predictivas durante el proceso de recomendación, algunos enfoques optan por modelar las preferencias de los usuarios mediante clasificadores automáticos. Éstos están basados en diferentes técnicas propuestas en el campo del aprendizaje de máquinas, tales como redes Bayesianas, árboles de decisión, reglas de aprendizaje inductivo o redes neuronales (Blanco, 2007).

Redes Bayesianas

Una red Bayesiana es un grafo acíclico dirigido (Diestel, 2000) en el que los nodos representan variables proposicionales y los arcos identifican dependencias causales entre ellos. Así, un arco entre dos nodos de la red significa que el nodo origen (también llamado nodo padre) tiene un impacto causal sobre el nodo destino (nodo hijo), o lo que es lo mismo, que este último depende del primero. Para poder cuantificar el efecto que los nodos padres tienen sobre un nodo específico de la red, se asocia a cada uno de éstos una tabla de probabilidades condicionales (calculadas mediante el teorema de Bayes), de forma que el valor de cada nodo es función de los valores de sus nodos padres. Además, los nodos hojas representan proposiciones cuyos valores son determinados por observación, a partir de los datos almacenados en un conjunto de entrenamiento del que se extrae la información representada en la red (Jensen, 2001).

Dada su utilidad en el campo de la clasificación automática, las redes Bayesianas son herramientas adecuadas tanto para modelar las preferencias de los usuarios, como para utilizarlas posteriormente durante el proceso de recomendación. El perfil de usuario resultante sería un grafo en el que los nodos y arcos identificarían los conceptos/atributos que interesan a los usuarios y las relaciones causales establecidas entre ellos, respectivamente. A la hora de elaborar recomendaciones personalizadas, el clasificador automático decide si un producto objetivo es interesante para el usuario activo, basándose en las relaciones probabilísticas existentes entre los atributos de dicho producto y las preferencias representadas en su red Bayesiana (Blanco, 2007).

Como principales ventajas, podemos destacar que las redes Bayesianas son modelos de clasificación muy rápidos, sencillos y precisos (Breese, Heckerman y Kadie, 1998). Además, las probabilidades condicionales que definen las dependencias causales entre los nodos de la red pueden ser calculadas *off-line*, lo que contribuye a reducir el coste computacional asociado al proceso de recomendación. Para poder aprovechar tales ventajas (computacionales), es necesario que las preferencias de los usuarios no varíen bruscamente a lo largo del tiempo, preservando con ello la validez del modelo calculado inicialmente; es este requisito el que limita la aplicación de las redes Bayesianas en el campo del modelado de usuarios (Blanco, 2007).

Árboles de decisión

Como su propio nombre indica, un árbol de decisión es una estructura en forma de árbol formada por un conjunto de nodos y arcos dirigidos que los conectan entre sí. Como modelo predictivo, su objetivo es clasificar una entrada dada en una categoría específica (típicamente, en los sistemas recomendadores, *interesante* o *no interesante*). Para ello, los nodos internos del árbol identifican preguntas sobre las variables consideradas en el problema de clasificación, mientras que los arcos representan las respuestas a esas preguntas, esto es, todos los posibles valores que pueda tomar cada variable en un conjunto de entrenamiento previamente definido. Por su parte, los nodos hoja se refieren a la clasificación final correspondiente a la variable de entrada considerada (Blanco, 2007).

En la literatura se han propuesto diferentes algoritmos recursivos para construir árboles de decisión a partir de la información almacenada en el conjunto de entrenamiento. De todos ellos, el algoritmo más conocido es ID3 (Quinlan, 1983), que construye un árbol de decisión de manera descendente, comenzando por identificar el atributo del conjunto de entrenamiento que debe utilizarse como raíz. Para este propósito, ID3 analiza cada atributo utilizando un test estadístico para evaluar la clasificación que establece dicho atributo en relación a los ejemplos de entrenamiento. Tal como describe (Blanco, 2007), una manera de cuantificar la relevancia de un atributo en este contexto se basa en medir la cantidad de información que éste proporciona, utilizando la teoría de información propuesta por Claude E. Shannon (Shannon, 1948).

Una vez seleccionado el mejor atributo, éste se coloca en la raíz del árbol y se crea un arco para cada uno de los posibles valores del atributo en cuestión. A continuación, los ejemplos de entrenamiento deben repartirse en los nodos en los que terminan los arcos generados, construyendo así el árbol de decisión de forma incremental. Para ello, el algoritmo ID3 considera los valores que toman dichos ejemplos para el atributo raíz y, en función de los mismos, selecciona el atributo que debe ocupar el siguiente nivel del árbol. Este proceso se repite, típicamente, hasta que todos los ejemplos de entrenamiento comparten un mismo valor para el atributo que está siendo evaluado. En ese caso, se crea un nodo hoja en el árbol que decide la clasificación final correspondiente.

Al igual que las redes Bayesianas, los árboles de decisión pueden ser utilizados como técnicas de modelado de usuarios en un sistema de personalización, gracias a su capacidad de clasificación automática. Mediante las capacidades predictivas de este modelo, los recomendadores aprenden a clasificar los productos objetivo (como *interesante o no interesante*), en función de sus atributos y de las preferencias representadas en el árbol correspondiente a cada usuario activo.

Reglas de aprendizaje inductivo

El descubrimiento de reglas de asociación mediante aprendizaje inductivo (Mobasher, 2000) se ha relacionado tradicionalmente con la realización de estudios de mercado. Esto es debido a que dichas reglas identifican patrones comunes entre los datos contenidos en diferentes transacciones. Mediante estos patrones es posible formular declaraciones del tipo .el 80% de los usuarios que compran un producto A, también compran B y C., de innegable valor a la hora de conocer la demanda de los usuarios en las transacciones comerciales (Blanco, 2007).

En la literatura se han propuesto algoritmos que extraen automáticamente las reglas de asociación a partir de un conjunto de entrenamiento, formado por varias transacciones que contienen atributos concretos. Tales reglas, de la forma X) Y (donde X e Y, identifican conjuntos de atributos definidos en el conjunto de entrenamiento), tienen dos valores asociados (Blanco, 2007):

- Confianza: Es el porcentaje de las transacciones que contienen el conjunto de atributos X, en las que también está presente Y.

- Soporte: Es el número de transacciones en las que se definen tanto X como Y.

Con el propósito de evitar el descubrimiento de reglas poco significativas, el algoritmo A priori utiliza dos parámetros de configuración, referidos a los valores mínimos de confianza y soporte exigidos a las reglas de asociación extraídas.

Conscientes de la eficacia del aprendizaje inductivo en el descubrimiento de conocimiento, son varios los sistemas de personalización que se basan en este enfoque a la hora de aprender y representar las preferencias de sus usuarios. En este escenario, los perfiles simplemente almacenan una relación de los atributos que resultan interesantes para los usuarios. Éstos son descubiertos mediante reglas de asociación extraídas por A priori a

partir de bases de datos donde se registran las preferencias de un elevado número de usuarios (Blanco, 2007).

Entre los sistemas que adoptan este enfoque destacan Movielens (Galán, 2007) y Recommender (Basu, Hirsh y Cohen, 1998), en el dominio de la TV personalizada.

Redes neuronales

Las redes neuronales artificiales (RNA) son sistemas de aprendizaje y procesamiento automático de la información, inspirados en el funcionamiento del sistema nervioso de los animales (Arbib, 1995). Están formadas por un gran número de unidades de procesamiento, llamadas neuronas, organizadas en varias capas. Cada neurona recibe una serie de entradas a través de enlaces de comunicación, denominados sinapsis, y emite una salida. Estos enlaces tienen asociado un peso numérico, en el que se codifica el conocimiento que la RNA tiene acerca de un determinado problema. La salida producida por la neurona viene dada por dos funciones:

- una *función de propagación* que, por lo general, es la suma de cada entrada multiplicada por el peso de su enlace o sinapsis (aunque también puede ser no lineal (Moody, 1989)), y

- una *función de activación* que, opcionalmente, modifica la anterior determinando el valor de la salida. Tal como muestra la Figura A1, en la literatura se han propuesto diversos tipos de funciones de activación: funciones lineales, en las que la salida es proporcional a la entrada; funciones umbrales, donde la salida es un valor discreto que depende de si la estimulación total supera o no un determinado umbral; y funciones gaussianas no lineales (Knight, 1990).

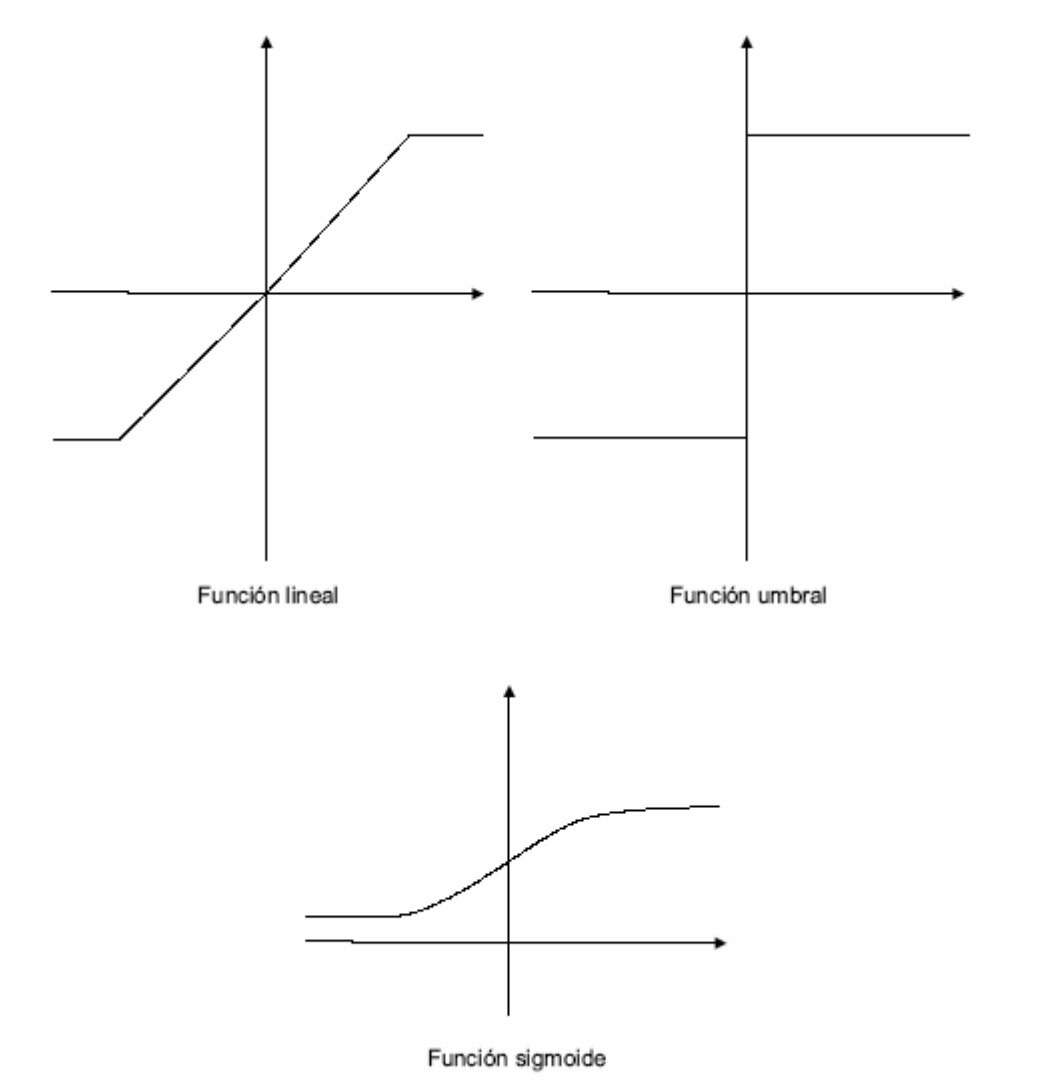


Figura A1: Funciones de activación utilizadas en las redes neuronales artificiales

Típicamente, las RNA han sido clasificadas en el estado del arte atendiendo a dos criterios: (i) su topología, relacionada con el número de capas disponibles en la red, el número de neuronas por capa y el grado o tipo de conectividad entre las mismas, y (ii) el tipo de aprendizaje que la red es capaz de asumir, según ésta necesite o no un conjunto de entrenamiento inicial (Blanco, 2007).

Primer criterio de clasificación: Topología de la RNA

Según su topología, podemos distinguir los siguientes tipos de RNA:

Redes monocapa: Sólo cuentan con una capa de neuronas, que intercambian señales con el exterior y que funcionan a la vez como entrada y salida del sistema. En la literatura, las redes más representativas en esta categoría son el perceptrón simple (Rosenblatt, 1962) y la red Hopfield (Hopfield, 1982).

Redes multicapa: Disponen de conjuntos de neuronas jerarquizadas en distintos niveles o capas, con al menos una capa de entrada y otra de salida y, eventualmente, una o varias capas ocultas. Dentro de éstas, podemos identificar dos tipos básicos de RNA, (Blanco, 2007):

- **Redes en cascada (*feedforward*)**: En estas redes, la información fluye unidireccionalmente de una capa a otra (desde la capa de entrada a las capas ocultas y desde éstas a la capa de salida), y además, no se admiten conexiones intracapa. Ejemplos representativos de este tipo de red son el perceptrón multicapa (MLP) (Hampshire, 1990), la red Madaline (Sahami, 1993), la máquina de Boltzmann (Hinton, 1986), los modelos de Kohonen (Melvilla, 2002) y las redes *backpropagation (*Rumelhart, 1986*)*.

- **Redes recurrentes (*feedback*)**: En este tipo de redes, la información puede volver a lugares por los que ya había pasado, formando bucles. Además, es posible el establecimiento de conexiones intracapa, e incluso la conexión de una neurona consigo misma. Estas últimas propiedades permiten que también puedan ser definidos modelos recurrentes para las redes monocapa descritas previamente (por ejemplo, la red Hopfield). Por su parte, entre las redes recurrentes multicapa propuestas en la literatura, destacan el Cognitrón (Blanco, 2007), el Neocognitrón (White, 1992) y las redes de resonancia adaptativa (Blanco, 2007).

Mientras las redes en cascada dan siempre soluciones estables, los modelos recurrentes proporcionan soluciones inestables o dinámicas, lo que no siempre es aconsejable. La principal aportación de John Hopfield, creador de la red que hereda su nombre, consistió precisamente en dotar de estabilidad a tales redes recurrentes (Blanco, 2007).

La red neuronal Hopfield

La red Hopfield está formada por un conjunto de unidades de procesamiento que están en estado activo o inactivo, conectadas tanto consigo mismas como entre sí a través de enlaces (bidireccionales) ponderados de forma simétrica. Su funcionamiento se basa en elegir una unidad de forma aleatoria y comprobar si alguno de sus nodos vecinos está activo. En caso afirmativo, se suman los pesos de todos los enlaces que conectan esta unidad con sus vecinos activos. Si esta suma es positiva, la unidad se activa; en caso contrario, se desactiva. Este proceso se repite, eligiendo otra unidad al azar, hasta que la red alcanza un estado estable, en el cual no varían los estados de activación de las unidades. Este proceso de búsqueda recibe el nombre de *parallel relaxation* en la literatura (Hopfield, 1982).

Hopfield desarrolló tanto un modelo continuo como discreto para su red neuronal, definido en función de los valores que pudiesen tomar sus entradas y salidas. En el modelo discreto, estos valores son 0 y 1, y su función de activación es un escalón entre ambos valores; por el contrario, en el modelo continuo, dichas funciones son típicamente sigmoides cuyas salidas oscilan entre -1 y 1, o incluso, entre 0 y 1.

La principal contribución de Hopfield fue demostrar que, para cualquier combinación inicial de pesos y de estados de activación/desactivación de las unidades, el algoritmo propuesto convergía siempre a un estado estable. Con ello, se eliminaban las oscilaciones e inestabilidades asociadas tradicionalmente a los modelos recurrentes (Blanco, 2007).

Segundo criterio de clasificación: Tipo de aprendizaje de la RNA

En lo que al tipo de aprendizaje se refiere, han sido varios los modelos de RNAs propuestos por diferentes autores (Blanco, 2007):

- **Aprendizaje supervisado**: Las redes que adoptan este enfoque necesitan un conjunto de datos de entrada previamente clasificado o cuya respuesta objetivo sea conocida. Esta fase de entrenamiento consiste en aplicar las entradas a la RNA y observar la salida que produce. Si esta salida no coincide con la esperada, es necesario ajustar los pesos de cada neurona para, iterativamente, ir obteniendo las respuestas adecuadas del sistema. Los ejemplos presentados deben ser suficientemente representativos en el dominio de aplicación concreto, de forma que sea posible que la RNA *aprenda* a partir de ellos. Entre las redes que requieren este entrenamiento podemos destacar el perceptrón simple (Rosenblatt, 1962), el perceptrón multicapa (Hampshire, 1990), y las redes *backpropagation* (Rumelhart, 1986).

- **Aprendizaje no supervisado o autoorganizado**: En este caso es posible prescindir de la fase de entrenamiento inicial. Es el aprendizaje en el que se basan las memorias asociativas, las redes de Hopfield (Hopfield, 1982), los modelos de Kohonen, Cognitrón y Neocognitrón (Blanco, 2007).

- **Redes híbridas**: Se basan en un enfoque mixto en el que se utiliza una función de mejora para facilitar la convergencia. Un ejemplo de este último tipo de RNA son las redes de base radial (Blanco, 2007).

Gracias a sus capacidades de aprendizaje y clasificación automática, las RNA se han aplicado también en el campo de los sistemas recomendadores. Prueba de ello son los enfoques presentados en el sistema Re:Agent (Boone, 1998), encargado de filtrar los correos electrónicos de los usuarios en función de los mensajes leídos por éste

recientemente, y en el recomendador de noticias personalizadas descrito en (Jennings y Higuchi, 1993), donde sus autores entrenan una red neuronal para cada usuario, con el fin de aprender y representar sus preferencias. De forma análoga a la comentada en el resto de modelos basados en clasificadores, en la red usada en (Jennings y Higuchi, 1993) los nodos identifican las palabras que aparecen en los artículos consultados por el usuario en el pasado, y las conexiones neuronales ponderadas representan la fuerza de las asociaciones existentes entre las palabras que aparecen en estos documentos. De esta forma, a la hora de decidir si una noticia debe ser sugerida al usuario, el sistema extrae los términos más significativos de la misma y los coloca en las entradas de la red neuronal. Basándose en los ejemplos aprendidos durante una fase de entrenamiento supervisado, la red trata de descubrir patrones comunes entre dicha noticia y las preferencias del usuario. Obviamente, en caso de que éstos existan, esta noticia es finalmente sugerida al usuario (Blanco, 2007).

Ya en el dominio de la TV personalizada, podemos destacar la herramienta presentada en y el sistema TV Recommender Show en el que se emplea una red neuronal RBF para combinar las sugerencias elaboradas por dos estrategias diferentes, construyendo así la lista final de programas mostrados al espectador. Ambas estrategias están basadas en clasificadores, concretamente en un árbol de decisión y una red Bayesiana (Blanco, 2007).

ANEXO B: Frameworks Orientados al Trabajo con Sistemas de Recomendación

Framework Taste

Taste es un motor de filtrado colaborativo rápido y flexible. El motor toma las preferencias del usuario para los ítems ("tastes") y retorna preferencias estimadas para otros ítems. Por ejemplo, un sitio que vende libros o CDs podría utilizar Taste para conocer, a partir de datos de ventas anteriores, que CDs le puedes interesar escuchar a un comprador.

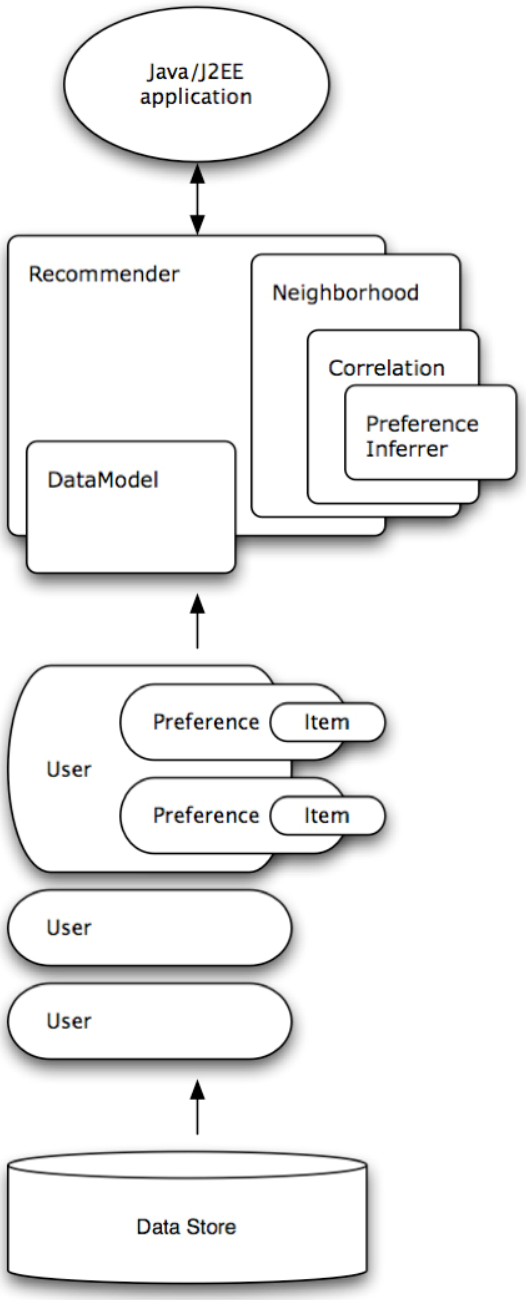
Taste proporciona un set importante de componentes con los cuales se puede construir un sistema de recomendación desde una selección de algoritmos. Taste está diseñado para ser una aplicación lista para la empresa; está diseñado para el desempeño, escalabilidad y flexibilidad. Soporta una interfaz estándar EJB para aplicaciones basadas en J2EE, pero taste no es solo para Java; puede ser ejecutado como un servidor externo que provee la lógica de recomendación para una aplicación vía Web services y http.

Se definen interfaces Taste, a través de paquetes de alto nivel, para estas claves de abstracción:

- DataModel
- UserCorrelation e ItemCorrelation
- UserNeighborhood
- Recommender

La implementación de estos paquetes está contenida en los subpaquetes de "comp.planetj.taste.impl". Esas son las piezas con las que se puede construir un motor de recomendaciones propio. Además, para un enfoque académico, Taste soporta sistemas basados en memoria y basados en ítem, recomendadores slope one y un par de otras implementaciones experimentales.

Arquitectura:



Este diagrama muestra la relación entre los distintos componentes de Taste en un recomendador basado en usuario. Un sistema recomendador basado en ítem es similar, con la excepción de que no se utilizan algoritmos de PreferenceInferres o Naighborhood.

Taste proporciona distintas clases e interfaces para realizar las recomendaciones, a continuación se definen las más importantes:

Recommender

Un recommender es la abstracción núcleo en Taste. A través de un DataModel puede producir recomendaciones. Las aplicaciones, en su mayoría, utilizarán la implementación de GenericUserBasedRecommender o GenericItemBasedRecommender, posiblemente decorada por CachingFRecommender.

DataModel

Un DataModel es la interfaz hacia la información de las preferencias del usuario. Una implementación podría obtener estos datos desde cualquier fuente, pero lo más común es desde una base de datos. Como muchas aplicaciones desearán escribir sus propios datos Taste proporciona MySQLJDBCDataModel para acceder a los datos de las preferencias desde una base de datos vía JDBC. Taste además proporciona un FileDataModel.

Junto con DataModel, Taste utiliza las abstracciones User, Item y Preference para representar a los usuarios, ítems y preferencias para los ítems del motor de recomendaciones. Implementaciones personalizadas de DataModel podrían retornar implementaciones de estas interfaces que sean apropiadas para la aplicación – quizás una implementación de OnlineUser que representa un usuario de una tienda online, y una implementación de un BookItem representando un libro.

UserCorrelation, ItemCorrelation

Un UserCorrelation define una noción de similitud entre dos usuarios. Esta es una parte crucial de un motor de recomendación. Estos están ligados a la implementación de un Neighborhood. Los ItemCorrelations son análogos, pero encuentran la similitud entre ítems.

UserNeighborhood

En un recomendador basado en usuario las recomendaciones son producidas encontrando un "neighborhood" de usuarios similares cercanos a un usuario dado. Un UserNeighborhood define un significado de determinar ese neighborhood – por ejemplo, los 10 usuarios más cercanos. Las implementaciones comúnmente requieren un UserCorrelation para operar.

Framework COFE

COFE es un motor de recomendaciones. Con los datos correctos COFE predecirá exactamente que ítems serán del gusto de un usuario. COFE puede ser utilizado en una amplia variedad de ambientes. Algunos ejemplos incluyen:

- Reducir información sobrecargada – cuando existen muchos ítems a considerar, un sistema recomendador puede predecir qué ítems poseen la preferencia más alta de ser útiles, interesantes, entretenidos o de valor. Por ejemplo, el sitio Web MovieLens (MovieLens.org) predice que películas serán del gusto del usuario entre miles de películas y videos disponibles.

- Mejorar usabilidad – incluso en casos donde no se cuenta con un número enorme de ítems de los cuales seleccionar, los sistemas recomendadores pueden mejorar la usabilidad de un sistema prediciendo los ítems que serán más útiles para un usuario y organizando la interfaz de usuario para colocar esos ítems en los lugares más fáciles de encontrar.

- Mejorar numero de ventas – Los motores recomendadores pueden predecir que ítems serían más interesantes para comprar por los clientes, permitiendo al vendedor hacer que esos ítems estén siempre disponibles para ese usuario, o quizás proporcionar cupones para ese ítem.

El filtrado colaborativo se refiere a un ambiente en el que la comunidad de personas se juntan para compartir la necesidad de filtrar información. Veamos el siguiente ejemplo. Considerando un periódico online con cincuenta artículos de noticias. Cualquier persona no posee el tiempo para leer 50 artículos de noticias, pero si tienes una comunidad de 50 personas, cada miembro de la comunidad puede leer un artículo y determinar qué valor proporciona el artículo. Se dice que la persona evalúa el articulo – le da una evaluación. Si la evaluación de una persona para un artículo es lo suficientemente alta, el articulo es recomendado al resto de la comunidad. Consultando las recomendaciones de todos se

construye una lista de los top 10 de los artículos que vale la pena leer. En retorno de solo leer un ítem, el usuario guarda el tiempo de escaneo entre cincuenta artículos para encontrar los más interesantes. En realidad, no todos comparten el mismo gusto, intereses o necesidades. De cualquier forma, se supone que cada miembro primero lee y evalúa cinco artículos en vez de uno. Ahora, podemos examinar las evaluaciones de un miembro hipotético Joe, y encontrar los otros top 10 miembros de la comunidad con las notas más similares a Joe. Eso es – las personas que leen alguno de los mismos 5 artículos que Joe y los evalúan con valores similares. Por ejemplo, Joe puede disfrutar la sección internacional, lee y evalúa altamente cinco artículos de esa sección. Joe puede ahora ser comparado con otras 10 personas que también leyeron artículos internacionales y los evaluaron altamente. Se llama a esas personas vecinos de Joe. Una vez que se han identificado los vecinos de Joe podemos buscar que artículos han sido evaluadas altamente por los vecinos de Joe. Luego podemos recomendar estos artículos a Joe para que los lea. Una lista de ítems que se acerca a las necesidades de un usuario es conocida como un set de recomendaciones.

Como extensión a este ejemplo, consideremos un servicio de noticias online que muestra títulos de artículos de noticias gratuitamente, pero cobra al mostrar el texto completo del articulo. Joe podría querer asegurarse que un artículo valdrá la pena antes de pagarlo. Utilizando el mismo enfoque descrito anteriormente, Joe podría consultar a sus vecinos que ya han leído el artículo para predecir su evaluación. Cuando un usuario quiere saber no solo cuales son los mejores ítems, sino cuales son las evaluaciones predictivas se le llama predicción.

Los sistemas recomendadores de filtrado colaborativo son sistemas de software que habilitan a las comunidades a realizar filtrado colaborativo. En el centro de un sistema de filtrado colaborativo esta un motor de recomendaciones de filtrado colaborativo – el sistema de software responsable de la computación involucrada en el filtrado colaborativo. El motor de recomendación es responsable de analizar las evaluaciones, determinando quien es vecino de quien y computando las predicciones y las recomendaciones. COFE es un motor de recomendaciones.

A las personas que proporcionan evaluaciones al motor de recomendaciones se les conoce como usuarios.

Evaluaciones

Como se describió anteriormente el sistema recomendador toma evaluaciones como entrada y puede dar como salida recomendaciones y predicciones. ¿Que son exactamente estas recomendaciones?
COFE opera sobre evaluaciones numéricas de una dimensión. Existen tres clases de evaluaciones numéricas:

1. Datos de evaluaciones multi-valuables: cada evaluación es un número en una escala predefinida. La parte baja de la escala indica que un ítem es pobre – el usuario siente que el ítem no proporciona valor o no es interesante o apropiadamente entretenido. La parte alta de la escala significa que el ítem es excelente – posee un alto valor. Un valor múltiple es similar a lo que se esperaría en una encuesta "En una escala de 1 a 5 valore cuanto disfruto esta película…" Para el ejemplo una escala discreta podría ser los valores 1, 2, 3, 4 o 5. Unan escala continua podría ser cualquier número real entre 1 y 5.

2. Datos binarios: Cada evaluación es 0 o 1. Por ejemplo, podemos contar con datos de una encuesta donde preguntamos a cada usuario "¿Le gusto el producto X? Si o No" Si dicen Si, tenemos una evaluación 1 y si dicen No tenemos una evaluación 0.

3. Datos Unarios: Estos datos son más difíciles. Imaginemos que se tienen un vendedor de e-commerce. Se ha guardado información de todos los compradores. Se conoce exactamente qué productos han comprado. El hecho de que un comprador ha comprado un producto indica un alto valor de gusto para el ítem comprado. Por lo tanto se obtiene una evaluación positiva. Por otro lado, solo porque un comprador no ha comprado un ítem en particular no se puede inferir que

no le gusta dicho producto. Por esto existe solo una evaluación y el dato no es binario. A esto se le llama situación unaria.

COFE está diseñado y testeado actualmente para trabajar de mejor forma con datos de evaluaciones multi-valuables. Por otra parte, también puede ser satisfactorio con datos binarios. Los datos unarios son un problema muy difícil. COFE funcionará con datos unarios sólo si se convierten antes en evaluaciones de datos binarios o multi-valuables.

Funcionalidad detallada de COFE

Una vez que se puebla COFE con evaluaciones de todos los usuarios, este proporciona soporte para las siguientes operaciones:

- Para un usuario dado, lista de los top N ítems que el usuario debería evaluar de forma alta. Este es el pan y mantequilla de COFE.

- Para un usuario dado, lista de los top N ítems de una categoría que el usuario debería evaluar de forma alta. En muchos casos no se quiere recomendaciones para un tipo de ítem – se quiere limitar a un subset de ítems. COFE proporciona soporte para esto vía categorías. Se puede definir cualquier set de ítems para formar categorías y luego obtener las recomendaciones top para la categoría. Poro ejemplo, se podría querer la recomendación para libros de misterio.

- Para un usuario dado y un ítem dado, predecir el valor de la evaluación que el usuario le dará al ítem. Útil para evaluar en una base ítem-a-ítem. Por ejemplo, un usuario podría estar viendo la descripción de un producto online. Se podría usar esta funcionalidad para predecir cuánto valor dará el usuario al ítem que está viendo. Si la predicción de valor es alta se podría ofrecer al usuario un cupón. O quizás solo si la predicción de valor es medianamente alta, ya que con una predicción de valor alto el usuario probablemente comprará el ítem sin el incentivo de un cupón.

Existe un gran número de otras funciones soportadas, pero son secundarias a las funciones listadas anteriormente, que son el núcleo de la funcionalidad del sistema.

Arquitectura de alto nivel de una aplicación utilizando COFE

COFE guardará valoraciones numéricas y proveerá recomendaciones basadas en esas valoraciones. En COFE cada usuario e ítem es identificado como un único número. COFE no soporta guardar u obtener cualquier información que no sea evaluaciones numéricas y predicciones/recomendaciones numéricas.

Como resultado, una aplicación común que utilice CODE necesitará mantener sus propias bases de datos de la información del usuario e ítem específicos del dominio. Por ejemplo, se puede querer mantener información demográfica de cada usuario o información de catalogo de producto de cada ítem. La figura A.2 ilustra una arquitectura ejemplo de una aplicación de administración de contenido Web del lado del servidor que utiliza COFE para predecir que ítems del contenido deberían ser mostrados a cuales usuarios. En esta figura un usuario – a través de su navegador Web – ingresa a un sitio Web que utiliza COFE. En el lado del servidor Web, la aplicación de administración de contenido localiza la información del usuario en los datos de usuario y de eso determina el id de usuario COFE asociado al usuario. Una solicitud para el id de usuario dado es enviado a COFE, que responde con ids de ítems que representan recomendaciones para ítems que al usuario le gustarán. La aplicación de administración de contenido luego accede a su base de datos de ítems para obtener el contenido asociado con esos ids de ítems y muestra esos ítems al usuario.

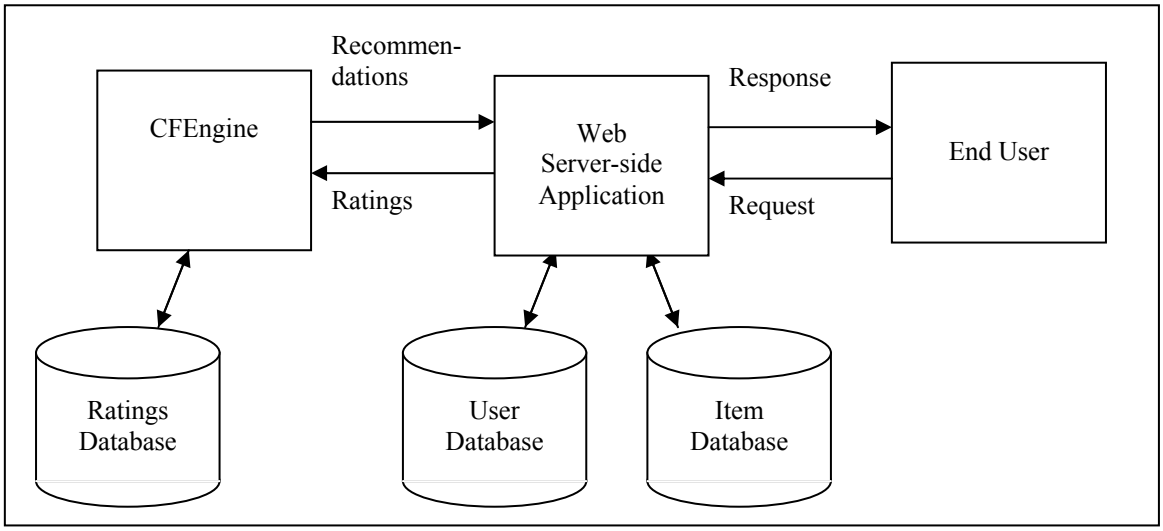


Figura A.2: Ejemplo de la arquitectura de una aplicación de administración de contenido Web del lado del servidor que utiliza COFE para generar recomendaciones.

Framework Duine

El toolkit Duine es un paquete de software que permite a los desarrolladores crear motores de predicción en sus propias aplicaciones. Un motor de predicción es un componente que predice cuan interesados están los usuarios individuales en una pieza de información. Dichas predicciones pueden ser usadas para personalizar información para los usuarios, específicamente en recomendar a éstos qué información es y no es interesante para ellos.

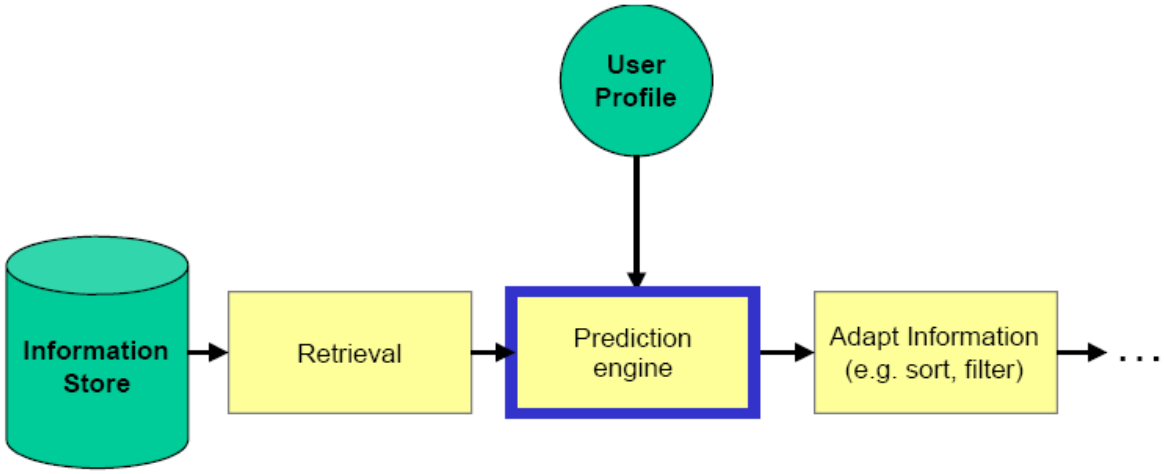


Figura A.3: Motor de predicción: predecir el nivel de interés en información de un usuario.

Un motor de predicción predice el nivel de interés de un usuario para un set de información que es obtenida desde un almacén de información o base de datos. El almacenamiento, indexado u obtención actual de la información no es parte del toolkit (ver figura A.3). La obtención de información puede estar basada en una consulta explicita del usuario o un proceso automatizado que monitorea cierto almacenamiento de información para nuevos ítems. Las predicciones son hechas basándose en información acerca del usuario que está guardado en los perfiles de usuarios. El toolkit incluye un componente para la administración de perfiles de usuarios. El resultado del motor de predicción es el set de información obtenido con datos agregados acerca de cuan interesante es cada pieza de

información para el usuario (las predicciones). Actualmente, los siguientes datos son agregados a cada pieza de información:

El valor de interés predicho: este es un valor entre -1 y +1 (incluidos). Un valor de -1 indica que al usuario probablemente no le gustará nada esa pieza de información. Un valor de +1 indica que probablemente al usuario le gustará mucho esa información y un valor 0 indica que el usuario es indiferente a esa pieza de información.

Un valor confidencial: El valor confidencial indica cuan fiable es la predicción y va de 0 (La predicción no posee valor, ya que pudo generarse de forma incorrecta) a +1 (La predicción es absolutamente confiable, esto es cuanto le gustará al usuario)

Un objeto explicativo: Puede ser usado para proporcionar explicaciones de las razones detrás de la predicción. Los objetos explicativos son actualmente implementados solamente en una forma limitada de la predicción. Las explicaciones son un tema de investigación futura y serán vistos en futuras versiones del toolkit

Utilizando estos datos adicionales, el set de información puede ser presentado al usuario sin cambios en la información o primero ser ordenada en las predicciones. Incluso es posible filtrar el set de información utilizando las predicciones, por ejemplo en un VCR digital personalizado, el grabador puede determinar qué programas de TV debería grabar automáticamente basándose en las predicciones. Las operaciones de orden y filtrado del set de información no es parte de este toolkit, a excepción de que los resultados del motor de predicción son ordenados en la predicción con el ítem más interesante al principio.

Debido a que el predecir cuan interesante es una pieza de información para un usuario depende altamente del dominio de la información y de los tipos de usuarios para los que la predicción debe ser hecha, el toolkit permite a los desarrolladores crear sus propios motores de predicción ajustados a sus situaciones, mientras se reutilizan técnicas de predicción existentes lo mayor posible. La principal diferencia de este toolkit de personalización con otros toolkits de personalización existentes en la actualidad es que el

toolkit Duine puede combinar múltiples técnicas de predicción dentro lo las llamadas estrategias de predicción, en orden a proporcionar predicciones mas exactas.

Base del toolkit Duine

Determinar cuan interesante es una información para un usuario es básicamente una forma de predicción. La mayoría de los sistemas de información personalizada disponible actualmente se enfocan en el uso de una técnica de predicción o una combinación arreglada de técnicas. Se piensa que combinando deferentes técnicas de una forma dinámica e inteligente proporciona predicciones más exactas y estables.

A través de combinaciones dinámicas e inteligentes se sugiere que la combinación de técnicas no debería ser arreglada dentro de un sistema, que la combinación debe estar basada en el conocimiento de las fortalezas y debilidades de cada técnica y que la selección de las técnicas a utilizar debería ser realizada solo al momento en que la predicción es requerida, tomando en cuenta el conocimiento más actual del usuario actual, otros usuarios, la información y el sistema en sí. Dicha combinación de técnicas de predicción es llamada estrategia de predicción. Antes de describir las estrategias de predicción con más detalle discutiremos un modelo genérico para las técnicas de predicción, que permite su uso para crear estrategias de predicción.

Técnicas de predicción

Una técnica de predicción calcula cuan interesado estará un determinado usuario en una pieza de información, utilizando algún tipo de algoritmo. La predicción resultante es un valor número que representa la cantidad del interés esperado para el usuario.

Aunque existen diferentes tipos de técnicas de predicción es posible crear un modelo genérico gracias a la naturaleza básica de cada predicción: cada técnica puede calcular un valor de interés predictivo para una pieza de información para un determinado usuario, basándose en conocimiento almacenado en el perfil de usuario, datos y metadatos

de la información y perfiles de otros usuarios. Eso forma la base de nuestro modelo. Naturalmente cada técnica debe normalizar sus predicciones en orden a ser comparable y combinar predicciones. Utilizamos el rango bipolar desde -1 a +1 (cero es neutro).

Varias técnicas son capaces de aprender de los usuarios para optimizar predicciones futuras. Ellas aprenden de la retroalimentación provista por los usuarios, ya sea retroalimentación explícita o implícita. Por esta razón la retroalimentación también es incorporada en el modelo.

Opcionalmente las técnicas de predicción pueden proporcionar datos explicativos. Las explicaciones proporcionan transparencia, exponiendo la razón y los datos detrás de una predicción y pueden incrementar la aceptación de los sistemas de predicción. Las explicaciones ayudan a los usuarios a decidir si aceptan o no una predicción y a entender las razones incorrectas cuando una predicción es imprecisa. Esto también ayuda a incrementar la confianza que tiene un usuario en el motor de predicción. Nuestro foco actual se basa en las predicciones y no en las explicaciones.

Para poder realizar decisiones informadas acerca de cuándo las técnicas son utilizables cada técnica expone los llamados indicadores de validez. Debido a las diferencias en las técnicas de predicción, la mayoría de las técnicas poseen indicadores de validación únicos y diferentes. Por ejemplo mientras que el filtrado social proporciona el número de usuarios similares que evaluaron la información, CBR proporciona el número de ítems evaluados similarmente por el usuario. Estos indicadores pueden ser utilizados por estrategias para decidir para qué caso debe ser usada una técnica.

El modelo genérico de las técnicas de predicción mostrado en la figura A.4 es el resultado de los seis aspectos discutidos de las técnicas de predicción.

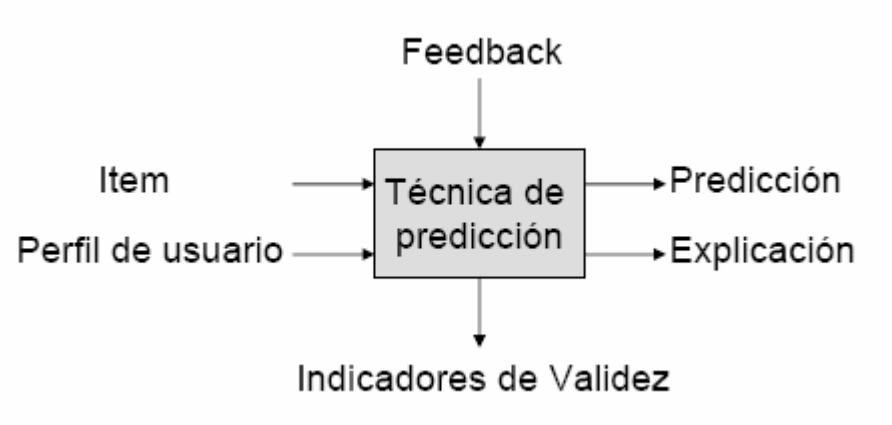


Figura A.4: Modelo genérico de una técnica de predicción

Estrategias de predicción

Las estrategias de predicción generan una predicción de interés para una determinada pieza de información y un usuario, seleccionando entre los algoritmos de predicción y combinando las técnicas de predicción basadas en los conocimientos más actuales del usuario actual, otros usuarios, la información y el sistema de información personalizada. Para realizar decisiones acerca de que técnicas de predicción utilizar y como combinarlas, una estrategia emplea un enfoque estratégico. Esta versión del toolkit Duine utiliza reglas de decisión estrictas (if… then… else…) para decidir que técnicas de predicción usar y combinar. Los enfoques estratégicos utilizan datos almacenados en el perfil de usuario, datos y metadatos de la información, datos del estado del sistema y más importante los indicadores de validez de las técnicas de predicción para decidir que técnicas de predicción seleccionar y/o combinar.

Desde una perspectiva de caja negra, las estrategias de predicción no son diferentes a las técnicas de predicción. Ambas predicen el interés en una pieza de información para un usuario: ambas son predictores (ver figura A.5). Esto quiere decir que el modelo genérico de una técnica de predicción de aplica también a las estrategias de predicción. Esto significa también que las estrategias pueden ser anidadas cuando sea necesario, creando una

jerarquía. Por ejemplo una estrategia puede ser creada para que sea mejor para usuarios nuevos. Dicha estrategia puede ser invocada por otras estrategias cuando determinen que otras técnicas de predicción no son capaces de proveer una predicción exacta.

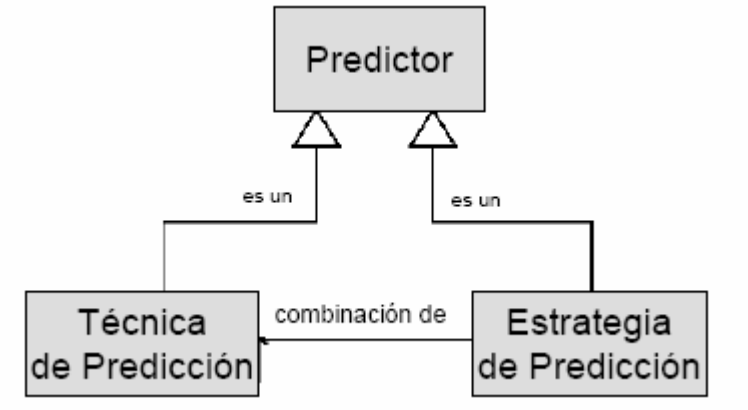


Figura A.5: Predictor es una superclase para las técnicas de predicción y estrategias de predicción.

Por otra parte, cuando se mira dentro de la caja negra, las técnicas de predicción actualmente generan predicciones basándose en el perfil de usuario y su información, mientras que las estrategias de predicción solo seleccionan uno o más predictores para generar las predicciones para ellos.

Motores de predicción

Un motor de predicción es un componente que procesa cada solicitud de predicciones que llega y es el punto de entrada principal para procesar la retroalimentación. El motor es la interfaz para las predicciones a otros componentes de un sistema de información personalizada.

Un motor de predicción no hace selecciones acerca de que estrategias de predicción usar: posee una estrategia arreglada que invoca. Mientras que las estrategias de predicción y

técnicas trabajan en generar una predicción para una sola pieza de información para un usuario los motores de predicción pueden procesar un set de solicitudes para múltiples piezas de información y para múltiples usuarios o grupos de usuarios. Es una tarea del motor de predicción separar el set en solicitudes individuales para la estrategia de predicción principal y después combinar los resultados de la predicción en el set nuevamente. La segunda tarea principal del motor de predicción es transferir toda la retroalimentación de los usuarios a las técnicas y estrategias de predicción para aprender en un orden correcto. Las relaciones entre un motor de predicción, estrategias de predicción y técnicas de predicción están sumadas en la figura A.6.

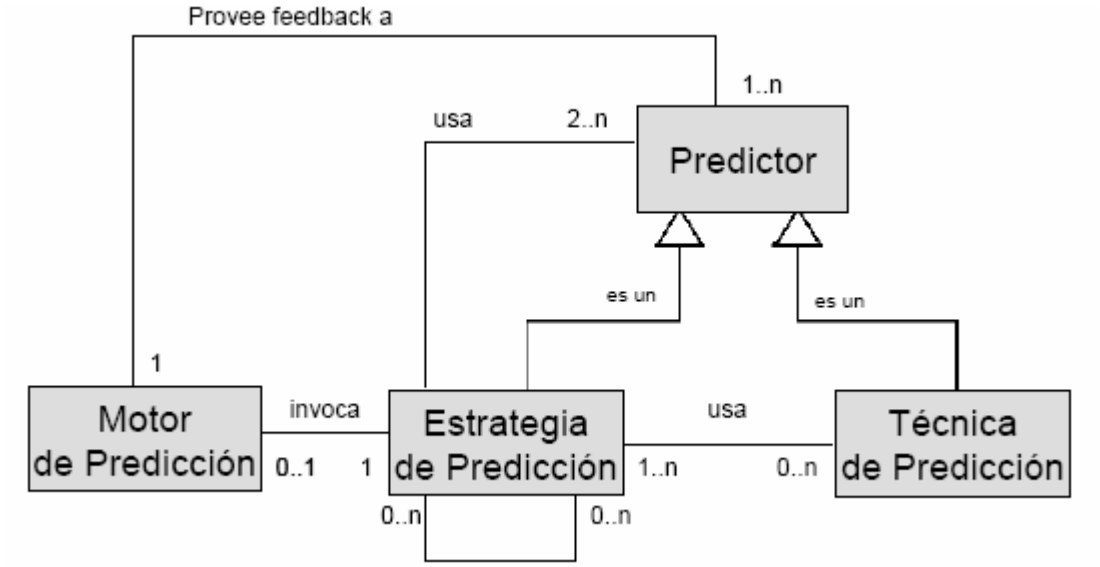


Figura A.6: Relaciones entre un motor de predicción, estrategias de predicción y técnicas de predicción. (La indicación 1..n muestra que por ejemplo el motor de predicción proporción retroalimentación a 1 o mas predictores y que cada técnica de predicción es usada por 1 o más estrategias).

Contenido

Las ideas básicas descritas en la sección anterior generan los siguientes módulos del toolkit:

- Núcleo de predicción: Consiste en las técnicas de predicción (incluyendo un set por defecto de técnicas de predicción y formas de desarrollar técnicas de predicción propias), formas de combinar las técnicas de predicción en estrategias de predicción y componentes para crear los motores de predicción.

- Administrador de perfiles: Es necesario para el almacenamiento y mantenimiento de los perfiles de usuario.

- Modulo de validación y desempeño: Puede ser utilizado para testear la exactitud de las técnicas de predicción y su desempeño. Son usadas también para personalizar y optimizar las técnicas y estrategias de predicción.

ANEXO C: Algoritmo de Generación de datos Aleatorios

Para evaluar los algoritmos, se generó un conjunto de datos de manera aleatoria, a través del siguiente algoritmo:

1. Generar 1 carrera *C* de *n*0 semestres y un conjunto *As* de *n*1 asignaturas para dicha carrera, asignando a cada asignatura un semestre, correspondiente a un número aleatorio entre 1 y *n*0.
2. Generar un conjunto *Al* de *n*2 alumnos para *C*, asignándoles como semestre cursado actual un número aleatorio entre 1 y *n*0, y asociando a cada alumno de *Al*, de manera aleatoria, con distribución normal, a diferentes asignaturas de *As*, verificando que se cumpla que el semestre cursado actual por cada alumno no pueda superar en 1 al semestre de cada asignatura asociada a él.
3. Generar un conjunto *Mat* de *n*3 Materiales, asociando a cada material, de manera aleatoria, con distribución normal, a diferentes asignaturas de *As*.
4. Generar un conjunto aleatorio de descargas y valoraciones realizadas por alumnos elegidos aleatoriamente de *Al*, sobre un conjunto de materiales elegidos aleatoriamente de *Mat*. Se definió que el 50% de las veces que un alumno revisara (descargara) un material, se generaría una valoración. La generación de la valoración también fue de forma aleatoria, de la siguiente forma.
    a. Determinar un subconjunto *SAl* de alumnos de *Al* elegidos aleatoriamente.
    b. Para (cada alumno *Ai* de *SAl*)
        i. Seleccionar subconjunto *SAl-M.at* de materiales de *Mat* elegidos aleatoriamente
        ii. Para (cada material *Mj* de *SAl-M.at*)
            Registrar descarga del material *Mj* por parte del alumno *Ai*
            Cada 2 descargas, generar una valoración entre 1 y 10, de manera aleatoria con distribución normal, y registrar la valoración al material *Mj* por parte del alumno *Ai*

ANEXO C: Tablas de Resultados