UNIVERSIDAD CENTRAL "MARTA ABREU" DE LAS VILLAS
FACULTAD DE MATEMÁTICA, FÍSICA Y COMPUTACIÓN
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

Métodos de preprocesamiento de datos para sistemas recomendadores de filtrado colaborativo, con aplicación en un escenario de e-learning

Tesis presentada en opción al grado científico de Doctor en Ciencias Técnicas

Autor: MSc. Raciel Yera Toledo

Santa Clara, 2015

UNIVERSIDAD CENTRAL "MARTA ABREU" DE LAS VILLAS
FACULTAD DE MATEMÁTICA, FÍSICA Y COMPUTACIÓN
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

Métodos de preprocesamiento de datos para sistemas recomendadores de filtrado colaborativo, con aplicación en un escenario de e-learning

Tesis presentada en opción al grado científico de Doctor en Ciencias Técnicas

Autor: MSc. Raciel Yera Toledo

Tutor: Dra. Yailé Caballero Mota

Cotutor: Dr. Luis Martínez López

Santa Clara, 2015

Agradecimientos

Esta tesis no hubiera podido hacerse sin la ayuda decisiva de muchas personas. Primero que todo quisiera agradecer a mi familia, por su sacrificio, por siempre estar ahí disponible para hacer cualquier cosa en pos del avance en el doctorado, inclusive sacrificando y condicionando sus planes. Por acompañarme en las madrugadas, los fines de semana y las vacaciones. A mi esposa. A mis padres. A mis suegros.

Agradezco a mi tutora, la Dra. Yailé Caballero Mota, por su guía y ejemplo, por confiar en mí, por creer desde un principio en el tema de investigación, por dedicarme una gran parte de su tiempo, en cualquier horario, a cualquier fecha. Por su incondicionalidad y optimismo. Por estar siempre disponible para atenderme. Por su paciencia. Al Dr. Luis Martínez López, cotutor de esta tesis, por su apoyo a lo largo de la investigación, por su eterno optimismo cuando las cosas iban mal. Por su sabia constancia. Por creer en el tema de investigación y defenderlo hasta las últimas consecuencias. Por su guía. Por sus enseñanzas. Por su confianza. Al Dr. Rafael Bello Pérez, por abrirme las puertas del Centro de Estudios de Informática en la UCLV, por abrirme las puertas de su oficina. Por permitirme entrar en el camino de hacerse doctor.

A la Dra. Leticia Arco, por su ayuda incondicional y decisiva, desde la presentación del tema hasta la terminación del documento, por siempre transmitir optimismo. A la Dra. María Matilde García y el Dr. Carlos Morell, por siempre estar disponibles para recibirme, por estar siempre ahí. Al Dr. Abel Rodríguez por su esmero en la revisión del documento, por sus consejos. A los profesores Dr. Ricardo Grau, Dra. Gladys Casas, Dr. Carlos Pérez, Dra. Luisa González, Dra. Ana María García y Dr. Ramiro Pérez, por todas sus enseñanzas. A los profesores Dra. Gheisa Ferreira, Dra. Yailén Martínez, y Dra. Martha Boggiano, por sus valiosos criterios sobre el trabajo. A Magaly Flora Prombol, por su seriedad y constancia. Agradezco a mis amigos de la Universidad de las Ciencias Informáticas, José Ernesto Lara, Dr. Enrique José Altuna, MSc. Leandro González, y Jorge Amado Soria, y en especial al MSc. Tomás Orlando Junco Vázquez, por su pasión por la resolución de problemas de programación, pasión gracias a la cual ha podido hacerse esta tesis. A Dovier Antonio Ripoll, director general de la Región Caribeña del ACM-ICPC. Agradezco a mis amigos y compañeros de la Universidad de Ciego de Ávila. A mis compañeros de trabajo, por darme la posibilidad de realizar la investigación, y por darme ánimo en todo momento. Por ayudarme en la parte logística. Al Dr. Raúl Fernández Aedo, a Loriet Gómez, al Dr. Reemberto Naranjo, a la Dra. María Emilia Aspiolea. Al Dr. Milton García Borroto por iniciarme en el camino de la investigación. A Ángel, "el chino".

A muchas personas que por razón de espacio no puedo mencionar, pero cuya ayuda ha sido decisiva para desatascar cosas sin las cuales este trabajo no hubiera podido avanzar. A todos, muchas gracias.

Dedicatoria

A mis padres

A mi esposa

A mis suegros

A toda mi familia

SÍNTESIS

Los sistemas recomendadores se centran en ayudar a los usuarios a encontrar aquella información que mejor se corresponda con sus necesidades y preferencias, en un espacio de búsqueda sobrecargado de posibles opciones. El grueso de las investigaciones en esta área se centra en la propuesta de nuevos métodos de recomendación que mejoren la eficacia de trabajos previos. Sin embargo, se han detectado muy pocos trabajos enfocados en preprocesar datos inconsistentes en sistemas recomendadores, con vistas a elevar su eficacia. A raíz de lo anterior, el objetivo de esta investigación es el de desarrollar nuevos métodos de preprocesamiento para la eliminación de inconsistencias de tipo ruido natural en sistemas recomendadores de filtrado colaborativo, que contribuyan a mejorar la eficacia de las recomendaciones generadas. Los resultados principales son, como aporte teórico, un método de corrección de preferencias inconsistentes que se destaca en el estado del arte por no depender de información adicional más allá de los propios valores de preferencia, y como aporte práctico un método para recomendar problemas a resolver en un Juez en Línea de Programación, y un método para preprocesar comportamientos inconsistentes en este recomendador, basado en el método de corrección inicialmente propuesto.

ÍNDICE

INTRODUCCIÓN

Desde la última década del pasado siglo la información disponible en Internet ha estado constantemente creciendo de forma exponencial. Este desbordamiento ha ocasionado que los usuarios tiendan a sentirse agobiados y tomen decisiones malas al identificar aquella información que mejor se corresponda con sus necesidades y preferencias. La disponibilidad de numerosas opciones, lejos de producir un beneficio, tiende a crear incomodidad entre los usuarios finales. De esta forma se comenzó a comprender que si bien en este escenario tener muchas opciones es positivo, tener un número excesivo de opciones no siempre resulta algo mejor (Schwartz, 2004a).

Durante un tiempo, servicios tradicionales de búsqueda tales como Yahoo y Google han ayudado a los usuarios a discernir entre toda la información disponible en la World Wide Web, mitigando así este problema. Sin embargo, se ha considerado que estos todavía resultan ineficientes de cara al usuario final.

En esta dirección, muchos expertos de Internet han planteado que los servicios de acceso a la información deben incluir un mayor nivel de personalización, de manera que no sean los usuarios los que tengan que buscar y acceder a la información; sino que sea esta la que, de alguna forma, fluya hacia los usuarios. Específicamente, criterios como el de Anderson (2004), han manifestado que se ha terminado la etapa de la búsqueda, y se ha iniciado la de la "recomendación", en la cual, "como por arte de magia", es la información la que descubre a los usuarios que la necesitan.

Desde mediados de los años 90, inclusive previamente a la formulación de estos planteamientos, ya se venía formalizando el concepto de sistema recomendador, así como desarrollándose como una línea de investigación independiente como respuesta a estas necesidades (Goldberg, Nichols, Oki, & Terry, 1992). Burke (2002) considera a un sistema recomendador como cualquier sistema que produce recomendaciones individualizadas como salida, o tiene el efecto de guiar al usuario, de una forma personalizada, hacia ítems

interesantes o útiles dentro de un espacio sobrecargado de posibles opciones. Según el Recommender Systems Handbook (Ricci, Rokach, Shapira, & Kantor, 2011), publicado por Springer, los sistemas recomendadores están conformados por métodos y herramientas que brindan sugerencias a un usuario, sobre los ítems a ser utilizados por este. El propósito de estas sugerencias es servir de intermediario en el mencionado proceso de toma de decisiones por parte de un usuario al acceder a la información. Entre estas decisiones pudieran encontrarse las de qué ítems comprar, qué música oír, qué noticias leer, qué contenidos estudiar, qué artículos científicos leer, o qué problemas de una guía de estudio resolver. En este tipo de escenarios, estos sistemas han pasado a jugar un papel importante, a tal grado que en la actualidad, numerosos sitios de Internet incluyen sistemas recomendadores como parte fundamental de su estructura. Entre estos puede citarse a Amazon.com[1] para la recomendación de productos en general, MovieLens[2] para la recomendación de películas, y Spotify[3] para la recomendación de música.

De manera general, los dominios iniciales de aplicación han sido el e-commerce y el e-learning, aunque en los últimos tiempos estos sistemas están siendo aplicados a escenarios cada vez más diversos (Cappella, Yang, & Lee, 2015; Li, Yuan, & Xu, 2015; Lu, Wu, Mao, Wang, & Zhang, 2015). Particularmente, con el rápido crecimiento de las aplicaciones de e-learning sobre la web, la recomendación de recursos en ambientes de aprendizaje electrónico se ha convertido en un problema de creciente interés y por tanto una importante línea de investigación (Chen, Niu, Zhao, & Li, 2014). En consonancia con esto, en los últimos tiempos en Cuba han proliferado varios escenarios de e-learning con un rápido desbordamiento de información, destacándose entre estos los jueces en línea de programación (JLP) (Altuna Castillo & Guibert Estrada, 2013; Kurnia, Lim, & Cheang, 2001; Leal & Silva, 2003), en los que un sistema recomendador jugaría un importante papel para mitigar el efecto negativo de dicho desbordamiento.

Desde su surgimiento, la comunidad científica internacional ha prestado especial atención a los sistemas recomendadores (Balabanovic, 1998; Contreras, Salamó, & Pascual, 2015; Ekstrand, 2014; Guo, 2011; Hu, 2014; Parra, 2013; Wu & Niu, 2015; Xin, 2011). Desde el año 2007, se ha venido desarrollado la conferencia ACM Recommender Systems como el evento anual principal en investigación y aplicaciones en sistemas recomendadores. Por otro lado, un buen número de

[1] http://www.amazon.com
[2] http://www.movielens.com
[3] http: //www.spotify.com

revistas académicas han publicado números enteros dedicados a la temática. Entre estas están ACM Transactions on Intelligent Systems and Technology (2014), Intelligent Decision Technologies (2014), Information Processing & Management (2013), User Modeling and User-Adapted Interaction (2012), AI Communications (2008); IEEE Intelligent Systems (2007); y ACM Transactions on Information Systems (2004). Todo lo anterior permite concluir que esta es un área de investigación y desarrollo ya establecida y de gran actualidad.

A pesar de que las tareas de minería de datos a llevar a cabo por un sistema recomendador pueden ser varias (Herlocker, Konstan, Terveen, & Riedl, 2004), Gunawardana y Shani (2009) han considerado que existen tres esenciales. La primera está relacionada con la recomendación de un conjunto "bueno" de ítems, en la que bueno se podría traducir en "útil" o "interesante". Una segunda tarea, menos abordada, es la de recomendar ítems que maximicen la utilidad de la recomendación desde el punto de vista del proveedor de los ítems, la que en la práctica se pudiera ver como recomendar productos de manera que se incrementen las ganancias de los vendedores, o recomendar temas a estudiar de manera que se incremente el aprovechamiento por parte de los estudiantes según los intereses del profesor. La tercera tarea se basa en predecir la opinión o rating[4] de los usuarios acerca de un conjunto de ítems que le resultan desconocidos; y a pesar de que esta no representa un acto explícito de recomendación, un importante esfuerzo de investigación en sistemas recomendadores se ha centrado en ella. El enfoque más popular es el de emplear una combinación entre la tercera y la primera tarea; donde inicialmente para un usuario se predice el rating de los ítems desconocidos y posteriormente se muestra un listado con los ítems de mayor valor de rating (Adomavicius & Tuzhilin, 2005). Esta tesis se centra, precisamente, en la primera y tercera tarea.

La mayoría de los trabajos en sistemas recomendadores giran alrededor de dos enfoques principales, que son la recomendación basada en el contenido (Lops, de Gemmis, & Semeraro, 2011; Pazzani & Billsus, 2007), y la recomendación basada en filtrado colaborativo (Desrosier & Karypis, 2011; Schafer, Frankowski, Herlocker, & Sen, 2007).

La recomendación basada en el contenido utiliza la filosofía de recomendar los ítems que sean semejantes a otros previamente preferidos por el usuario activo. En este enfoque se debe tener como premisa disponer de un conjunto de características asociadas a cada uno de los ítems,

[4] Por ser ampliamente usada la palabra rating en el argot de los sistemas recomendadores, a lo largo del documento se empleará su versión original (en inglés), y no una traducción al español de la misma.

para aprender a partir de estas las preferencias de los usuarios, y posteriormente utilizar dichas preferencias para recomendar nuevos ítems con similares características.

Por otro lado, el filtrado colaborativo asume el principio de que los usuarios que estuvieron de acuerdo en las preferencias manifestadas en el pasado, deben también coincidir en el futuro. A raíz de esto sigue la filosofía de recomendar al usuario activo los ítems preferidos por usuarios que tengan preferencias semejantes a este. Básicamente, en estos métodos se identifica un conjunto de usuarios "similares", que reciben el nombre de vecindario, y posteriormente los ítems que fueron preferidos por los usuarios de dicho vecindario son recomendados al usuario activo. En este enfoque y contrario a la recomendación basada en contenido, solamente se requiere conocer los ratings brindados por los usuarios a los respectivos ítems, no necesitándose información adicional sobre dichos ítems.

Esta última característica ha provocado que en los últimos años los métodos de filtrado colaborativo hayan ganado en popularidad, puesto que disponiendo de un mínimo de información, son capaces de generar recomendaciones eficaces. Así, se han reportado métodos de filtrado colaborativo basados en k-vecinos más cercanos (Konstan et al., 1997; Linden, Smith, & York, 2003), métodos de reducción de dimensión como complemento al filtrado colaborativo (Koren, 2010), métodos donde se incorpora algún enfoque probabilístico tal como el clasificador Naïve-Bayes (Miyahara & Pazzani, 2002), trabajo con agrupamiento buscando escalabilidad (Breese, Heckerman, & Kadie, 1998; Rapečka & Dzemyda, 2015), métodos aplicando el modelo de un proceso de decisión de Markov (Shani, Heckerman, & Brafman, 2005), y métodos basados en la minería de reglas de asociación (Lin, Alvarez, & Ruiz, 2002). A raíz de la positiva reputación ganada por estos métodos, esta tesis se desarrollará tomando como base enfoques de recomendación de filtrado colaborativo.

Amatriain, Jaimes, Oliver, y Pujol, en uno de los capítulos del Recommender Systems Handbook (Amatriain, Jaimes, Oliver, & Pujol, 2011), consideran que no sólo el filtrado colaborativo, sino todo método de recomendación, tiene en su núcleo un algoritmo que puede ser comprendido como una instancia particular de un método de minería de datos. Por tanto, asocia cada uno de los métodos presentes en el estado del arte, a uno de los tres posibles pasos que conforman un problema de minería de datos, que son preprocesamiento de datos, análisis, e interpretación. Mientras que el grueso de los trabajos que aparecen en el marco teórico que proponen están asociados a la segunda etapa, que es la vinculada con la técnica de

minería en sí empleada, llegan a la conclusión de que muy pocos trabajos se identifican dentro del primer grupo, que es el correspondiente a métodos de preprocesamiento.

En contraparte a esto, Jiawei Han y Micheline Kamber en Data Mining: Concepts and Techniques (Han & Kamber, 2001), otorgan vital importancia a los métodos de preprocesamiento, y afirman que las bases de datos del mundo de hoy son altamente susceptibles a la presencia de datos ruidosos, erróneos e inconsistentes. Consideran que esto ocurre debido al gran tamaño que presentan y a que su origen proviene de fuentes múltiples y heterogéneas; y que la baja calidad de los datos generalmente da lugar a malos resultados en la minería de estos, haciendo imprescindible por tanto su preprocesamiento. A su vez, agrupan los posibles tipos de preprocesamiento en tres categorías principales: limpieza de datos, para eliminar ruido y corregir inconsistencias en los datos, integración y transformación de datos, para unir datos de diferentes fuentes, y reducción de datos, para disminuir el tamaño de los datos agregando, eliminando rasgos redundantes, o agrupando.

En (Amatriain, et al., 2011), esencialmente se relacionan dos líneas de preprocesamiento de datos en sistemas recomendadores. Una de estas, perteneciente al grupo de reducción de datos, está conformada por los métodos basados en reducción de dimensión tales como PCA (análisis de componentes principales) y SVD (descomposición en valores singulares), los que son concebidos para atacar el problema puntual de la dispersión de los datos, asociado a la excesiva cantidad de preferencias desconocidas. Sin embargo, a pesar de ser estos considerados como métodos de preprocesamiento en otros escenarios, en el caso particular de los sistemas recomendadores tienden a ser más utilizados por la generalidad de la literatura como un enfoque aislado más que como complemento a otro método de recomendación (Koren, Bell, & Volinsky, 2009). Esto viene dado por el hecho de que transforman los datos (ratings) a un nuevo formato que es prácticamente imposible de utilizar por otros métodos más tradicionales y puramente de recomendación.

La otra línea de preprocesamiento de datos planteada por Amatriain et al. está asociada a la limpieza de los datos, y particularmente se denomina como denoising. Esta asume que en el contexto de un sistema recomendador pueden estar presentes dos tipos de ruido: ruido malicioso y ruido natural. Por una parte, el de tipo malicioso (Mehta & Hofmann, 2008; Mehta & Nejdl, 2009) hace referencia a aquel que se introduce deliberadamente en el sistema, generalmente por un usuario, con vistas a alterar los resultados finales de la recomendación. Sobre este tipo de inconsistencias se han realizado importantes trabajos en los últimos años,

centrados esencialmente en identificar aquellos usuarios ruidosos. Sin embargo, por su propia naturaleza, estas inconsistencias son sólo inherentes a cierto grupo de perfiles, lo que imposibilita la aplicación de los métodos desarrollados para enfrentarlas, en calidad de técnicas de preprocesamiento utilizables en cualquier sistema de filtrado colaborativo. Por esta razón, estos métodos no resultan directamente de interés a los efectos del presente trabajo.

Por otro lado, el ruido natural es el introducido involuntariamente por los usuarios durante su interacción con el sistema. Amatriain, Pujol y Oliver (2009) demostraron que este ruido provoca inconsistencias que afectan la salida de un sistema recomendador, y por ello consideran necesario realizar trabajos para procesarlas y corregirlas, con vistas a mitigar su efecto negativo.

A diferencia del caso del ruido malicioso, resulta mucho menor el número de trabajos que se centran en el tema del ruido natural. Como soluciones con vistas a mitigar este problema tomando como referencia a la recomendación basada en filtrado colaborativo, O'Mahony, et al. proponen un método basado en vecindades para eliminar ruido natural en una base de datos de un sistema recomendador (O'Mahony, Hurley, & Silvestre, 2006). Sin embargo, su evaluación es limitada, perdiendo posteriormente continuidad el trabajo. Amatriain, Pujol, Tintarev y Oliver (2009), proponen una técnica de "re-rating" para eliminar ruido natural, asumiendo que se dispone de varios ratings de un mismo usuario por cierto ítem, y considerando la posibilidad de solicitar nuevamente la valoración sobre un ítem ya valorado previamente. Este método tiene una alta dependencia del usuario activo, lo que constituye una limitante importante. Hwang et al. por su parte en (Hwang, Pham, & Jung, 2011) y posteriormente en (Pham & Jung, 2013), proponen un modelo de corrección de preferencias del usuario que utiliza la información adicional asociada a los ítems, específicamente las categorías. Estas categorías pudieran estar constituidas, en el caso de películas, por el género, el director, o los actores. Dicho enfoque también tiene limitantes; debido a que no siempre se cuenta con este tipo de información. Finalmente, en (Li, Chen, Xingquan, & Chengqi, 2013) se propone un método para la detección de usuarios "ruidosos pero no maliciosos", reportándose además que la no consideración de este tipo de usuarios podría implicar un incremento en la eficacia de la recomendación. A pesar de no necesitar información adicional, este enfoque trata las inconsistencias a nivel de usuario y no de rating, que es donde realmente ocurren. En adición, para lograr una mejora en la eficacia de las recomendaciones se requiere el descarte de un grupo importante de usuarios, y todavía así, dicha mejora resulta pequeña. Por tanto, este enfoque también resulta insuficiente.

En los últimos años varios autores han mostrado que los datos en los que se basan las recomendaciones pueden ser ruidosos, corruptos, o simplemente erróneos, debido a que los usuarios son inherentemente inconsistentes al expresar sus preferencias en forma de ratings (Amatriain, Pujol, & Oliver, 2009; Ekstrand, Riedl, & Konstan, 2010; Konstan & Riedl, 2012; Said, Jain, Narr, & Plumbaum, 2012). A pesar de esto, la revisión anteriormente presentada constituye evidencia de que los trabajos realizados hasta el momento son insuficientes para enfrentar este problema, definiéndose por tanto como **problema de investigación** la presencia de inconsistencias asociadas a las preferencias de los usuarios en sistemas recomendadores de filtrado colaborativo, que afecta la eficacia de las recomendaciones generadas. Como ya se ha evidenciado, la mayoría de los trabajos en esta área del conocimiento están centrados en la identificación de perfiles de usuarios maliciosos o con comportamiento inestable. Existe una importante ausencia de trabajos que realizan este preprocesamiento a nivel de ratings, lo que convierte al presente en problema relevante a resolver en el área de sistemas recomendadores de filtrado colaborativo. Este problema se manifiesta tanto en sistemas recomendadores de propósito general (Adomavicius & Tuzhilin, 2005), como en aplicaciones específicas como los ya mencionados jueces en línea de programación (Kurnia, et al., 2001), escenarios de e-learning actualmente de relevancia en Cuba (Altuna Castillo & Guibert Estrada, 2013).

El **objetivo general** de esta investigación es:

Desarrollar nuevos métodos de preprocesamiento, utilizando técnicas de inteligencia artificial, para la eliminación de inconsistencias en sistemas recomendadores de filtrado colaborativo, que contribuyan a mejorar la eficacia de las recomendaciones generadas.

Para lograr este objetivo, se plantearon los siguientes **objetivos específicos** de la investigación:

1. Desarrollar un nuevo método de preprocesamiento para la eliminación de inconsistencias en los ratings en sistemas recomendadores de filtrado colaborativo.

2. Validar la efectividad del método desarrollado en el incremento de la eficacia en la obtención de mejores recomendaciones, utilizando bases de datos de repositorio y algoritmos clásicos de filtrado colaborativo.

3. Mostrar que el enfoque obtenido es eficaz para el escenario particular de la recomendación de problemas a resolver en un Juez en Línea de Programación (JLP), a través del desarrollo y

la evaluación de un marco de recomendación construido combinando filtrado colaborativo básico y un método de preprocesamiento para este escenario, basado en el enfoque obtenido.

4. Desarrollar una herramienta informática para la recomendación de problemas a resolver en los JLP, basado en el marco de recomendación construido.

Después de revisar la literatura y considerar el problema identificado, se trazaron las siguientes **hipótesis** de investigación:

H1: La caracterización del comportamiento global de los usuarios y los ítems en sistemas recomendadores de filtrado colaborativo, permitirá la detección y procesamiento de preferencias inconsistentes, posibilitando mejorar la eficacia de las recomendaciones generadas[5].

H2: La aplicación de un método para el procesamiento de preferencias inconsistentes inicialmente concebido para un escenario tradicional de recomendación, igualmente repercute en la obtención de recomendaciones más eficaces para el escenario particular de la recomendación de problemas a resolver en un JLP.

Las tareas de investigación trazadas son:

1. La definición de un método general para el procesamiento de ratings ruidosos, que no depende de información adicional más allá de los propios ratings.

2. La definición de un método para clasificar los usuarios y los ítems de un sistema de filtrado colaborativo, basado en su comportamiento global al dar y recibir ratings.

3. La definición de un método que utilice las clasificaciones anteriores para detectar y corregir preferencias inconsistentes en el sistema de filtrado colaborativo.

4. La evaluación del efecto de la propuesta en métodos tradicionales de recomendación, a través de conjuntos de datos de sistemas recomendadores publicados internacionalmente con este fin.

[5] A lo largo de este trabajo, el término "eficacia de la recomendación" será empleado, tal y como explica (Ricci, et al., 2011), para hacer referencia al resultado de la evaluación obtenida mediante métricas tradicionales de recuperación de información, tales como el error medio absoluto (MAE), el precision, y el recall. En el capítulo 1 se hará una referencia más detallada a estas métricas.

5. La definición de un método de recomendación de problemas a resolver en un JLP, que adapte los métodos de filtrado colaborativo a este escenario en particular, e integre el método de preprocesamiento preconcebido.

6. La evaluación del método anterior a través de un conjunto de datos extraído de un JLP.

7. La elaboración de un prototipo de software que funcione como sistema recomendador de problemas a resolver, acoplado a un JLP.

Entre los métodos de trabajo científico utilizados se destacan los siguientes:

- Analítico-sintético, al descomponer el problema de investigación en elementos por separado, tales como filtrado colaborativo, preferencias de los usuarios, y eficacia de la recomendación, para luego sintetizarlos en la presentación de las propuestas.
- Histórico-lógico, para el estudio crítico de los trabajos anteriores, utilizándolos como punto de referencia y comparación de los resultados alcanzados.
- Hipotético-deductivo para elaborar las hipótesis de la investigación, y para proponer nuevas líneas de trabajo a partir de los resultados obtenidos.
- Modelado para el desarrollo de los algoritmos propuestos.
- Experimental para comprobar la utilidad de los resultados obtenidos, y para compararlos con otros métodos reportados previamente en la literatura.

La **novedad científica** del presente trabajo se resumen en:

La propuesta de un método de preprocesamiento de datos para sistemas recomendadores de filtrado colaborativo, cuya eficacia es superior a la de otros métodos del estado del arte, se destaca sobre estos por corregir ruido sin depender de información adicional más allá de los valores de preferencia, y es capaz de dar lugar a recomendaciones eficaces en un escenario puntual de e-learning.

Esta novedad se resume en el siguiente aporte teórico:

-Un método de corrección de ratings inconsistentes basado en la clasificación previa de usuarios e ítems, para un escenario tradicional de recomendación.

Por su parte, el valor práctico de la investigación viene dado con la obtención de un prototipo de aplicación informática, donde se integre un método para recomendar problemas a resolver en

un JLP. Este método de recomendación se construye combinando filtrado colaborativo básico y un método de preprocesamiento para este escenario puntual basado en el resultado teórico obtenido. Con ello, se podrían preprocesar comportamientos inconsistentes en los datos del JLP, con vistas a obtener recomendaciones más eficaces. Con respecto a los JLPs, es válido resaltar que son aplicaciones muy utilizadas para la preparación de estudiantes en la docencia y en los concursos de programación de la ACM-ICPC[6]. Por sólo poner un ejemplo, el Juez en Línea Caribeño de Programación (Caribbean Online Judge), disponible desde la Universidad de las Ciencias Informáticas (UCI), posee en estos momentos (enero 2015), más de 20000 usuarios y 3000 problemas. Por tanto, dotar a los estudiantes de una herramienta que le sugiera automáticamente problemas a resolver acorde a sus intereses, habilidades y preferencias, permitiría un mayor aprovechamiento de los beneficios que brinda el juez en línea, al posibilitar un acceso más personalizado a este.

La tesis se encuentra estructurada en tres capítulos:

El capítulo 1 hace una revisión general de los sistemas recomendadores de filtrado colaborativo, así como de los métodos desarrollados con el objetivo de hacer preprocesamiento de datos en sistemas recomendadores. De manera particular, se hace especial énfasis en los métodos de preprocesamiento de ruido natural.

El capítulo 2 propone un método para el procesamiento de ruido natural en sistemas recomendadores de filtrado colaborativo. Como parte de este desarrollo, se realiza un estudio experimental con vistas a evaluar el efecto de la propuesta sobre diferentes algoritmos de recomendación. Además, se estudian otros criterios de interés como el nivel de intrusión del método y su tiempo de ejecución.

El capítulo 3 se centra en la construcción de un sistema recomendador para un JLP, basado en filtrado colaborativo y técnicas de preprocesamiento de ruido natural. Específicamente, se presenta un método basado en filtrado colaborativo para sugerir problemas a resolver en un JLP. En adición, se hace uso del enfoque presentado en el capítulo anterior, para mejorar el desempeño del sistema recomendador propuesto.

Finalmente el documento culmina con las conclusiones, recomendaciones, bibliografía, producción científica del autor sobre el tema de tesis, y los anexos.

[6] Competencia Internacional Intercolegios de Programación, auspiciada por la ACM. Este tipo de competencias se vienen celebrando en Cuba desde el año 2005, y en el mundo desde el 1970.

1. ACERCA DEL FILTRADO COLABORATIVO Y MÉTODOS DE PREPROCESAMIENTO DE DATOS PARA SISTEMAS RECOMENDADORES

Introducción

En el presente capítulo se hace una revisión general de los sistemas recomendadores de filtrado colaborativo, así como de los métodos desarrollados con el objetivo de hacer preprocesamiento de datos en sistemas recomendadores. De manera particular, se hace especial énfasis en los métodos de preprocesamiento de ruido natural.

La sección 1.1 proporciona una introducción a los sistemas recomendadores, mostrando su esquema de funcionamiento así como las tareas principales que suelen llevar a cabo. En adición, se presenta la taxonomía que usualmente se emplea para agruparlos. La sección 1.2 hace una revisión extensa de los métodos de recomendación basados en filtrado colaborativo, haciéndose hincapié en aquellos de mayor relevancia tanto para el estado del arte de esta área del conocimiento, como para esta tesis en particular. La sección 1.3 presenta brevemente el protocolo de experimentación que se utiliza para evaluar este tipo de métodos. La sección 1.4 se centra en el análisis de los métodos de preprocesamiento de datos en estos sistemas, iniciando por los de ruido malicioso y destacando los que se centran en el ruido natural. La sección concluye con la presentación de una taxonomía que agrupa a estos métodos, y que permite presentar la brecha que se pretende cubrir con la presente tesis. Finalmente, el capítulo termina con las conclusiones parciales.

1.1 Los sistemas recomendadores

Los sistemas recomendadores son herramientas enfocadas a ayudar a los usuarios a obtener aquella información que mejor se corresponda con sus intereses y preferencias. Mientras que un buscador habitual se centra en encontrar aquello que el usuario solicita, un sistema recomendador ayuda al usuario a tomar una decisión, que puede ser la compra de un producto

en un portal de comercio electrónico, la lectura de un libro, la revisión de un artículo científico, el acceso a una página web en específico, o el estudio de determinado recurso educativo en una plataforma virtual de aprendizaje.

Teniendo en cuenta la sobrecarga de información y de alternativas disponibles, en la actualidad la acción de elegir está comenzando a considerarse compleja. Según Formoso (2013), citando a Schwartz (2004b), el proceso de elección generalmente está conformado de forma implícita o explícita, por las siguientes etapas:

1. Conocer y establecer los objetivos.
2. Evaluar la importancia de cada objetivo.
3. Buscar las opciones disponibles.
4. Evaluar el grado en que cada una de las opciones cumple los objetivos.
5. Escoger la opción ganadora.

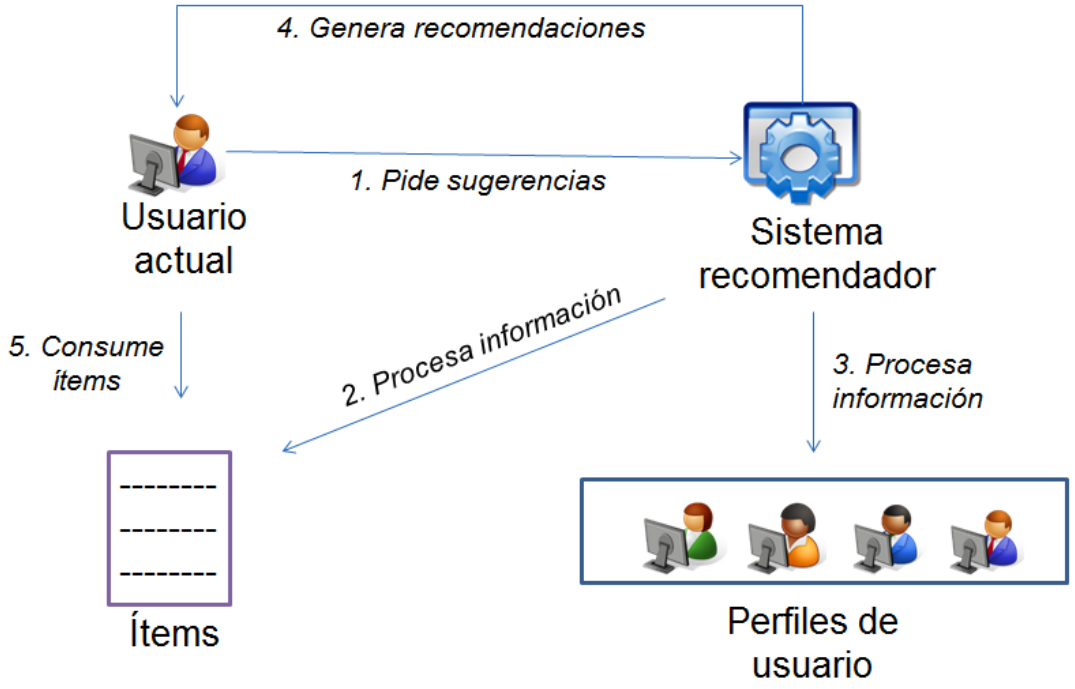


Figura 1.1 Esquema del funcionamiento de un sistema recomendador.

El aumento de la cantidad de opciones disponibles hace necesario un mayor esfuerzo de los usuarios finales con vistas a lograr una buena elección. Sin embargo, por otro lado, un sistema recomendador automatiza la mayoría de los pasos anteriores, permitiendo un importante ahorro

de tiempo, aumentando la satisfacción del usuario, y posibilitando encontrar productos novedosos o inesperados.

La figura 1.1 muestra un esquema del funcionamiento básico de un sistema recomendador, en el que inicialmente un usuario necesita elegir qué ítems consumir dentro de un gran conjunto, y para esto le solicita sugerencias al sistema (1). Tras esto, el recomendador procesa la información asociada a los ítems y a todos los perfiles de usuario (incluyendo el usuario actual) (2, 3), para posteriormente sugerir los ítems más apropiados (4), quedando el usuario libre de realizar la elección que él desee, basándose en estas recomendaciones (5).

Más allá de este esquema, Herlocker et al. (2004) han distinguido hasta seis tareas de minería de datos diferentes asociadas a los sistemas recomendadores, siendo estas:

-Anotación en el contexto (Annotation in context), donde la tarea es la de predecir para un usuario específico su grado de preferencia por un ítem en particular, para posteriormente presentarlo en el sistema de conjunto con la visualización del ítem.

-Encontrar ítems buenos (Find good items), donde el sistema sugiere una lista, de tamaño predefinido, con los ítems más apropiados con respecto al usuario actual.

- Encontrar todos los ítems buenos (Find all good items), donde el sistema sugiere todos los ítems que considere adecuados acorde a las preferencias del usuario.

-Recomendar secuencias (Recommend sequence), donde se centra en la sugerencia de una secuencia de ítems, en vez de la de un ítem en particular.

-Sólo búsqueda (Just Browsing), donde queda en un segundo plano el acceso a algún ítem en específico, siendo el principal objetivo el soporte a la navegación dentro de un gran volumen de información.

-Encontrar recomendaciones creíbles (Find Credible Recommendations), donde el objetivo es que el usuario tenga confianza en las recomendaciones que reciba por parte del sistema.

El grueso de los trabajos ha estado centrado en las tres primeras tareas (Gunawardana & Shani, 2009).

Por otro lado, la clasificación más popular de los sistemas recomendadores está asociada al algoritmo que emplean para realizar la tarea de minería correspondiente, y divide a los métodos

de recomendación en métodos de filtrado basado en el contenido, métodos de filtrado colaborativo, métodos de filtrado demográfico, y métodos híbridos (Bellogín, Cantador, Díez, Castells, & Chavarriaga, 2013; Bernardes, Diaby, Fournier, FogelmanSoulié, & Viennet, 2015; Bobadilla, Ortega, Hernando, & Gutiérrez, 2013; Lü et al., 2012).

Los *recomendadores basados en el contenido* (Cantador, Bellogín, & Vallet, 2010; Lops, et al., 2011; Pazzani & Billsus, 2007; Tkalčič, Burnik, & Košir, 2010) recomiendan basándose exclusivamente en la descripción del ítem y de un perfil con los intereses del usuario activo, teniendo en cuenta la filosofía de "recomiéndame las cosas que he seleccionado o me han gustado anteriormente". Estos sistemas se caracterizan por contener un modelo que describe los ítems que pueden ser recomendados, un modo de crear un perfil de usuario que representa los tipos de ítems que este prefiere, y un modo de comparar cada uno de los ítems a recomendar con el perfil del usuario, para determinar cuáles de ellos serán recomendados.

Los ítems a ser recomendados suelen caracterizarse a través de un conjunto de rasgos o atributos, conociéndose los valores que pueden tomar cada uno de estos atributos. Considerando el grado de preferencia de un usuario por un subconjunto de ítems y utilizando algoritmos de aprendizaje, es posible construir su perfil en términos de los mismos rasgos; y a partir de este, obtener de los ítems restantes aquellos de mayor utilidad para el referido usuario. Las dos cuestiones más importantes en estos sistemas son, por tanto, la representación de los ítems y el aprendizaje del perfil del usuario.

Los *recomendadores basados en filtrado colaborativo* (Desrosier & Karypis, 2011; Ekstrand, et al., 2010; Pham, Cao, Klamma, & Jarke, 2011; Su & Khoshgoftaar, 2009; Zhong & Li, 2010), por otra parte, inicialmente permiten a los usuarios proporcionar valores de preferencias personales acerca de un conjunto de objetos (videos, canciones, filmes), de forma tal que una vez que haya suficiente información almacenada en el sistema, sea posible realizar recomendaciones basándose en la información proporcionada por aquellos usuarios con un comportamiento semejante al del usuario actual. De forma alternativa, las preferencias de los usuarios también pueden ser adquiridas implícitamente.

Los *sistemas recomendadores demográficos* (Pazzani, 1999; Porcel, Tejeda-Lorente, Martínez, & Herrera-Viedma, 2012) tienen como objetivo clasificar al usuario según sus atributos personales y hacer las recomendaciones basándose en sus clases demográficas. El

funcionamiento de estos está justificado bajo el principio de que individuos con ciertos atributos personales en común (sexo, edad, país), también deben tener preferencias comunes.

Finalmente, los *sistemas recomendadores híbridos* (Burke, 2002; Cantador, Castells, & Bellogín, 2011; Martinez, Rodriguez, & Espinilla, 2009) frecuentemente usan una combinación de las técnicas anteriores con vistas a explotar los méritos de cada una de ellas por separado. Así, se han desarrollado enfoques que combinan filtrado colaborativo con información demográfica (Vozalis & Margaritis, 2007), enfoques de filtrado colaborativo con métodos basados en el contenido (Choi, Yoo, Kim, & Suh, 2012), y enfoques que combinan diferentes técnicas de filtrado colaborativo (Su, Greiner, Khoshgoftaar, & Zhu, 2007). Burke (2002) menciona siete formas básicas en las que se pueden combinar varios métodos de recomendación, con vistas a construir métodos híbridos. Estas son las basadas en mezcla (mixed), en pesos (weighted), en intercambio (switching), en cascada (cascade), en combinación de características (feature combination), en aumento de características (feature augmentation), y en meta-nivel (meta-level).

Desde sus inicios, la recomendación basada en filtrado colaborativo ha sido muy popular, puesto que se basa únicamente en los valores de preferencia, y no depende de la disponibilidad de información adicional para la generación de recomendaciones, como es el caso de los sistemas basados en el contenido y los demográficos. Incluso así, es capaz de generar recomendaciones eficaces, mostrándose inclusive por algunos autores que sólo unos pocos ratings en un sistema de filtrado colaborativo pueden dar lugar a una eficacia superior que la obtenida por un sistema basado en contenido con un mayor volumen de información asociado (Pilászy & Tikk, 2009). A raíz de lo anterior, la presente tesis se desarrolla tomando como base este paradigma de la recomendación.

A continuación se hace una revisión más detallada de los métodos de filtrado colaborativo.

1.2 Filtrado colaborativo

El filtrado colaborativo constituye hoy en día la técnica más popular para el desarrollo de sistemas recomendadores (Deshpande & Karypis, 2004; Konstan, et al., 1997; Resnick, Iacovu, Suchak, Bergstrom, & Riedl, 1994). La principal ventaja de este viene dada en que, en su versión más elemental, sólo necesita de los valores de preferencia de los usuarios por un grupo de ítems, para predecir la preferencia por ítems desconocidos y así recomendar.

Varios autores han definido de manera formal el problema general de la recomendación, siendo la más aceptada la dada por Adomavicius y Tuzhilin (2005), específicamente para la tarea de la predicción de preferencias o ratings, en un escenario de filtrado colaborativo. Estos autores asumen la existencia de *U* como el conjunto de usuarios, y de *I* como el conjunto de todos los ítems posibles a ser recomendados tales como libros, películas o restaurantes, pudiendo ser muy grande la cardinalidad de ambos conjuntos. Consideran además a *f*: *U x I* → *R* (Adomavicius & Tuzhilin, 2005) como la utilidad del ítem *i* al usuario *u*, donde *R* es un orden total (enteros no negativos o números reales en un determinado rango). Con estos elementos, se desea acotar para cada usuario $u \in U$, aquel ítem $i' \in I$ que maximice su utilidad al usuario:

$$\forall u \in U, \;\; i'_u = argmax_{i \in I} f(u, i) \qquad (1.1)$$

La utilidad *f* no se define completamente para el espacio U x I, sino únicamente para un subconjunto de este. En otras palabras, el papel del sistema recomendador radica en predecir el valor de la utilidad de manera completa para dicho espacio, tomando como punto de entrada el subconjunto del mismo para el cual sí se encuentra definida. Una vez conocida la utilidad de los ítems desconocidos para un usuario en particular, el objetivo final es brindar recomendaciones de manera personalizada considerando estos.

En este marco, el papel del filtrado colaborativo es el de predecir la utilidad de un ítem para un usuario activo u_a, basándose en la evaluación dada a ese mismo ítem por usuarios similares a u_a. De manera más formal, la utilidad $f(u, i)$ del ítem desconocido i para el usuario u, se estimaría basándose en las utilidades $f(u_j, i)$ asignadas al ítem i por aquellos usuarios $u_j \in U$ cercanos a u_a. Así, la selección de los ítems a recomendar depende casi exclusivamente de las valoraciones realizadas por otros usuarios, en vez de depender directamente del contenido, el que puede ser un indicador erróneo de aptitud.

Tabla 1.1 Escenario de recomendación clásico

	Ítem1	Ítem2	Ítem3	Ítem4	Ítem5
Juan	5		4		
Usuario1		3		2	1
Usuario2	4				
Usuario3			3		4
Usuario4	3	5			
Usuario5	5			2	5

Un escenario clásico de recomendación se muestra en la tabla 1.1, en la que se presentan las utilidades (también identificadas como preferencias o ratings) sobre diferentes ítems (columnas) con respecto a un grupo de usuarios (filas), enmarcándose estas en el rango [1, 5]. Cada fila de la tabla está compuesta por las utilidades, explícitamente especificadas, de un subconjunto del total de ítems con respecto a un usuario puntual.

Considerando lo anterior, el proceso de brindar recomendaciones al usuario Juan vendría dado por inicialmente predecir, con un valor (discreto o continuo), las utilidades de los ítems no evaluados (en este caso Item2, Item4 e Item5) (tarea Annotation in context), y posteriormente recomendar de estos, el de mayor utilidad si se realiza la recomendación ajustada a la definición formal dada por (Adomavicius & Tuzhilin, 2005), o un listado con varios elementos si se desea recomendar más de un ítem (tarea Find good items).

Adomavicious y Tuzhilin (2005) agrupan a los métodos de filtrado colaborativo en dos grandes familias: los métodos de vecindario (también llamados basados en memoria o en heurísticas), y los basados en modelos. A continuación se hará un breve recuento de trabajos asociados a cada uno de estos grupos.

1.2.1 Métodos basados en vecindario

En los métodos basados en vecindario (Deshpande & Karypis, 2004; Konstan, et al., 1997; Linden, et al., 2003; Nakamura & Abe, 1998), la evaluación usuario-ítem almacenada en el sistema se utiliza de manera directa en la predicción de la evaluación para los nuevos ítems. La figura 1.2 muestra el esquema general del funcionamiento de este tipo de sistemas.

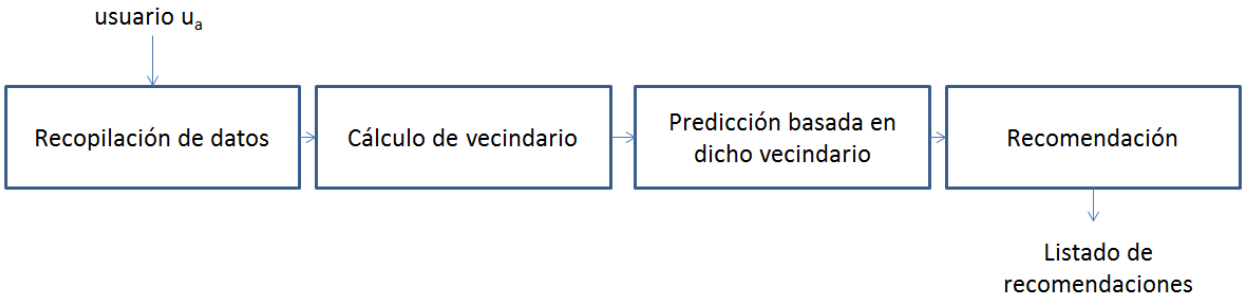


Figura 1.2. Esquema general de los sistemas recomendadores basados en vecindario

Esta predicción puede ser hecha de dos formas: 1) basada en el usuario (Konstan, et al., 1997), en la que el sistema estima el interés de un usuario u por un ítem i utilizando los ratings para

este mismo ítem por parte de otros usuarios con patrones similares de rating, llamados vecinos más cercanos; o 2) basada en el ítem (Deshpande & Karypis, 2004; Linden, et al., 2003), donde la predicción del rating de un usuario u para un ítem i se realiza basándose en la predicción dada por ese mismo usuario u para otros ítems con patrones semejantes a i.

-Métodos de vecinos más cercanos basados en usuarios.

Para realizar un análisis más formal de la manera en que se realiza la predicción basada en el usuario, inicialmente se asume la siguiente notación:

-Para cada usuario u se tienen los valores w_{uv} que indican la similitud de las preferencias entre u y cualquier usuario v ≠ u (el último epígrafe de la presente sección 1.2.1 muestra las formas de calcular estas similitudes)

- r_{vi} representa el rating dado por el usuario v al ítem i,

-$NU^k(u)$, representa los k usuarios v con la mayor similitud w_{uv} con respecto al usuario activo u, y

-$NU_i^k(u)$ representa los k-vecinos más cercanos a u que, en adición, han evaluado el ítem i,

Considerando lo anterior, las dos formas primarias de calcular la predicción del rating r_{ui} dado por el usuario u al ítem i están dadas por el uso del promedio (1. 2) y de un promedio pesado (1. 3).

$$r_{ui} = \frac{1}{|NU_i^k(u)|} \sum_{v \in NU_i^k(u)} r_{vi} \qquad (1.\ 2)$$

$$r_{ui} = \frac{\sum_{v \in NU_i^k(u)} w_{uv} r_{vi}}{\sum_{v \in NU_i^k(u)} |w_{uv}|} \qquad (1.\ 3)$$

Específicamente, la ecuación (1. 2) calcula la predicción a través del simple promedio del rating de los usuarios pertenecientes al conjunto $NU_i^k(u)$, el que representa a los k-vecinos más cercanos a u que han evaluado el ítem i. A su vez, la ecuación (1. 3) considera que no deben tributar de igual manera los ratings de estos usuarios en la estimación del rating del usuario u, y por tanto calcula la predicción como un promedio pesado de los mismos ratings que usa (1. 2),

incorporando el comentado parámetro w_{uv}. De esta forma, se establece un peso acorde a la cercanía, con vistas a maximizar la contribución de los usuarios más semejantes a u dentro de $NU_i^k(u)$, y minimizar la de los menos semejantes. Como los pesos inicialmente no están normalizados, la expresión del denominador se incluye para realizar esta función, y se inserta el signo de módulo para que en caso de que haya pesos negativos el valor obtenido se mantenga en el rango deseado. Esta expresión (1. 3) constituye el basamento de todos los sistemas de filtrado colaborativo que operan sobre vecindarios.

Un tercer enfoque básico consiste en representar el promedio pesado como el producto de la similitud de preferencias entre el usuario activo u y otro usuario (w_{uv}), por la diferencia entre el rating de este último usuario v para el ítem en cuestión y su rating promedio r_v. Este promedio pesado sería entonces adicionado al rating promedio r_u del usuario activo u, como se muestra en la expresión (1. 4). Mientras la expresión (1. 3) tiene la limitante de no considerar el hecho de que cada usuario puede tender a concentrar sus ratings en un intervalo específico, la propuesta (1. 4) sí lo toma en cuenta, al realizar la predicción tomando como base la desviación de los ratings del usuario con respecto a su valor promedio.

$$r_{ui} = r_u + \frac{\sum_{v \in NU_i^k(u)} w_{uv}(r_{vi} - r_v)}{\sum_{v \in NU_i^k(u)} |w_{uv}|} \quad (1.\ 4)$$

-Métodos de vecinos más cercanos basados en ítems.

La definición formal para el filtrado colaborativo basado en ítems es muy semejante a la del basado en usuarios. En este caso se definiría como w_{ij} el valor de la semejanza entre los ítems i y j (en la siguiente sección se mostrará las posibles formas de calcularla), y $NI_u^k(\mathrm{i})$ sería el conjunto de k ítems, evaluados por el usuario u, más similares a i. La predicción del rating del usuario u para el ítem i se obtendría como un promedio pesado de los ratings proporcionados por u para los ítems de $NI_u^k(\mathrm{i})$, tal y como se muestra en (1. 5). En el caso de la expresión (1. 6), homóloga a (1. 4), r_i constituiría el promedio de todos los ratings asociados a i.

$$r_{ui} = \frac{\sum_{j \in NI_u^k(\mathrm{i})} w_{ij} r_{uj}}{\sum_{j \in NI_u^k(\mathrm{i})} |w_{ij}|} \quad (1.\ 5)$$

$$r_{ui} = r_i + \frac{\sum_{j \in NI_u^k(\mathrm{i})} w_{ij}(r_{uj} - r_j)}{\sum_{v \in NI_u^k(\mathrm{i})} |w_{ij}|} \quad (1.\ 6)$$

En los dominios sobre los cuales se insertan sistemas recomendadores, el número de ítems tiende a mantenerse constante, mientras que el de usuarios suele aumentar rápidamente. Por esta razón, utilizar un filtrado colaborativo basado en ítems tiende a ser más efectivo en aras de garantizar un mejor rendimiento (Sarwar, Karypis, Konstan, & Riedl, 2001). Esto hace que esta alternativa sea considerada superior a la anterior.

-Cálculo de semejanza

El cálculo de la semejanza w_{uv} entre dos usuarios u y v (aplicable también a dos ítems i y j de forma directa), se basa en los ratings de los ítems que ambos usuarios han evaluado en común. Con este fin, asumiendo que I_{uv} representa aquellos ítems que han recibido votaciones tanto de u como de v, los dos enfoques más populares para la obtención del grado de semejanza entre ambos perfiles son la similitud basada en el coseno (1. 7) (Billsus & Pazzani, 1998), y el coeficiente de correlación de Pearson (1. 8) (Resnick, et al., 1994). En esta segunda alternativa se consideran los efectos de la media y la varianza de los ratings, en contraposición con la similitud del coseno.

$$\cos(u, v) = \frac{\sum_{i \in I_{uv}} r_{ui} r_{vi}}{\sqrt{\sum_{i \in I_{uv}} r_{ui}^2 \sum_{i \in I_{uv}} r_{vi}^2}} \quad (1.\ 7)$$

$$\mathrm{pearson}(u, v) = \frac{\sum_{i \in I_{uv}} (r_{ui} - r_u)(r_{vi} - r_v)}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - r_u)^2 \sum_{i \in I_{uv}} (r_{vi} - r_v)^2}} \quad (1.\ 8)$$

No obstante, se ha demostrado que el mejor desempeño de una u otra medida, desde el punto de vista de la eficacia de la recomendación, depende en alto grado de los datos sobre los cuales se apliquen (Howe & Forbes, 2008).

Más allá de la popularidad de estas dos medidas de semejanza, existen otras tales como el Simple Matching Coefficient, o el coeficiente de Jaccard, que también se han empleado en el área de los sistemas recomendadores específicamente para el caso de datos nominales o binarios (Amatriain, et al., 2011).

1.2.2 Métodos basados en modelos

Los métodos de filtrado colaborativo basados en modelos se centran en utilizar los ratings conocidos para realizar el aprendizaje de un modelo, el que se emplea para predecir el resto de los ratings (Su & Khoshgoftaar, 2009). En los últimos años se ha desarrollado un grupo importante de trabajos en esta línea, donde priman las hibridaciones al punto de que ya es casi imposible citar algún método puramente basado en vecindario, o puramente basado en modelos. Así, entre los trabajos más relevantes están los que se apoyan en enfoques probabilísticos (Javari & Jalili, 2015; Miyahara & Pazzani, 2002), en agrupamiento (Alam, Dobbie, Koh, & Riddle, 2014; Sarwar, Karypis, Konstan, & Riedl, 2002), en métodos de regresión (Lemire & Maclachlan, 2005; Vucetic & Obradovic, 2005), en procesos de decisión de Markov (Abbasi, Javari, Jalili, & Rabiee, 2014; Tavakol & Brefeld, 2014), y en minería de reglas de asociación (Kiran & Kitsuregawa, 2013; Lucas, Laurent, Moreno, & Teisseire, 2012; Sarwar, Karypis, Konstan, & Riedl, 2000a; Tyagi & Bharadwaj, 2013).

Finalmente, formando un grupo especial dentro de los métodos basados en modelos están los métodos basados en reducción de dimensión (Bobadilla, et al., 2013). Estos enfoques se centran en proponer una solución para los altos niveles de dispersión que afectan el desempeño de los métodos de recomendación, presentando a la vez algoritmos escalables. De manera particular, los mejores resultados globales obtenidos en el área del filtrado colaborativo están vinculados con este grupo (Koren, et al., 2009), destacándose las investigaciones asociadas al Netflix Prize donde para un escenario en específico se logró mejorar la eficacia de la recomendación un 10% con respecto a los métodos de peor desempeño. En esta misma sección, más adelante se analizará un método específico basado en reducción de dimensión.

En adición, el enfoque denominado slope one (Lemire & Maclachlan, 2005), se destaca en la literatura por tener un desempeño semejante a los métodos tradicionales basados en vecindario (analizados en la sección anterior), y una menor complejidad computacional. A raíz de esto, será también analizado con mayor detenimiento a continuación.

-Métodos basados en filtrado colaborativo slope one

Lemire y Mclachlan (2005) desarrollaron un grupo de métodos, denominados predictores slope one, donde toman ideas de las técnicas de regresión para realizar el cálculo de los ratings desconocidos. Estos métodos obtienen resultados comparables con los métodos tradicionales basados en vecindario, y tienen una menor complejidad computacional.

En este caso, la predicción de un rating desconocido (asociado a cierto usuario e ítem) se lleva a cabo basándose en los promedios de las diferencias entre los ratings de dicho ítem, y los ratings asociados a los restantes ítems. La ecuación (1. 9) muestra la forma de obtención de estas diferencias promedio, las que son combinadas a través de un promedio pesado (ecuación (1. 10)), para obtener las predicciones finales. En las ecuaciones, $S_i(u)$ representa el conjunto de ítems proporcionados por el usuario u, $S_u(i)$ y $S_u(j)$ el conjunto de usuarios que evaluaron los ítems i y j, y c(j,i) es la cardinalidad del conjunto $S_u(j) \cap S_u(i)$. Este método ha sido identificado por Lemire y Maclachlan como enfoque slope one pesado.

$$dev_{j,i} = \sum_{u \in S_u(j) \cap S_u(i)} \frac{r_{u,j} - r_{u,i}}{c(j,i)} \tag{1. 9}$$

$$P_{u,j} = \frac{\sum_{i \in S_i(u)} ((dev_{j,i} + r_{u,i}) c(j,i))}{\sum_{i \in S_i(u)} c(j,i)} \tag{1. 10}$$

Estos autores también presentaron una alternativa más sofisticada, denominada bipolar slope one. En este caso, para cada usuario se divide el conjunto de ítems evaluados en dos grupos: aquellos evaluados positivamente y aquellos evaluados negativamente, y para esto se toma como umbral de diferencia el rating promedio del usuario correspondiente. Tras esto, se aplica la técnica slope one pesada, combinándose finalmente ambos resultados. Las ecuaciones (1. 11), (1. 12) y (1. 13) muestran cómo aplicar el bipolar slope one partiendo de la técnica anterior.

$$dev^{like}{}_{j,i} = \sum_{u \in S_u{}^{like}(j) \cap S_u{}^{like}(i)} \frac{r_{u,j} - r_{u,i}}{c(j,i)} \tag{1. 11}$$

$$dev^{dislike}{}_{j,i} = \sum_{u \in S_u{}^{dislike}(j) \cap S_u{}^{dislike}(i)} \frac{r_{u,j} - r_{u,i}}{c(j,i)} \tag{1. 12}$$

$$P_{u,j} = \frac{\sum_{i \in S_i{}^{like}(u)} ((dev^{like}{}_{j,i} + r_{u,i}) c^{like}(j,i)) + \sum_{i \in S_i{}^{dislike}(u)} ((dev^{dislike}{}_{j,i} + r_{u,i}) c^{dislike}(j,i))}{\sum_{i \in S_i{}^{like}(u)} c^{like}(j,i) + \sum_{i \in S_i{}^{dislike}(u)} c^{dislike}(j,i)} \tag{1. 13}$$

En este caso, los valores de $S_i^{like}(u)$, $S_i^{dislike}(u)$, $S_u^{like}(i)$, $S_u^{dislike}(i)$, $c^{like}(j, i)$ and $c^{dislike}(j, i)$ son definidos a continuación en las ecuaciones (1. 14), (1. 15), (1. 16), (1. 17), (1. 18) y (1. 19) respectivamente, tomando como punto de partida el valor de cada uno de los ratings con respecto al rating promedio del usuario actual, tal y como se expresara anteriormente.

$$S_i^{like}(u) = \{i \in S_i(u) \mid r_{u,i} > \overline{r_u}\} \quad (1.\ 14)$$

$$S_i^{dislike}(u) = \{i \in S_i(u) \mid r_{u,i} < \overline{r_u}\} \quad (1.\ 15)$$

$$S_u^{like}(i) = \{u \in S_u(i) \mid i \in S_i^{like}(u)\} \quad (1.\ 16)$$

$$S_u^{dislike}(i) = \{u \in S_u(i) \mid i \in S_i^{dislike}(u)\} \quad (1.\ 17)$$

$$c^{like}(j,i) = \left|S_u^{like}(j) \cap S_u^{like}(i)\right| \quad (1.\ 18)$$

$$c^{dislike}(j,i) = \left|S_u^{dislike}(j) \cap S_u^{dislike}(i)\right| \quad (1.\ 19)$$

-Métodos basados en reducción de dimensión

Los métodos de reducción de dimensión han sido utilizados para enfrentar el problema de la dispersión de los datos que típicamente aparece en las bases de datos de los sistemas recomendadores. Como de una forma u otra se ha evidenciado con anterioridad, estos escenarios se caracterizan por estar representados a través de un espacio de alta dimensión, que en la práctica está conformado por una matriz con varias filas y columnas, representando usuarios e ítems, en las que un número importante de entradas son valores desconocidos. El resultado de la reducción de dimensión en esta matriz es directamente utilizable en la computación de las predicciones, y puede marcar un hito importante en aras de lograr una mayor eficacia en esta tarea.

La técnica más popular de reducción de dimensión para filtrado colaborativo es la descomposición en valores singulares (SVD) (Sarwar, Karypis, Konstan, & Riedl, 2000b). Esta propone encontrar un espacio de ítems de menor dimensión donde los nuevos ítems representen “conceptos”, y el peso de cada concepto sea computable. En este contexto, se ha utilizado el hecho de que cualquier matriz A siempre es posible descomponerla en un producto de la forma $A = U\lambda V^T$. Dada la matriz A de n x m (n usuarios y m ítems), es posible obtener una matriz U de n x r (n usuarios, r conceptos), una matriz diagonal λ de r x r (peso de cada concepto), y una matriz V de m x r (m ítems, r conceptos). La matriz diagonal λ contendría los valores singulares, que serían ordenados de forma decreciente.

Tras obtener esta matriz, el método propone dejar sólo las filas y columnas asociadas a los k mayores valores singulares, eliminando las restantes. Esto daría lugar a una aproximación de

rango k, de la matriz A, de la forma $A_k = U_k \lambda_k V_k^T$. En la práctica, inicialmente los valores desconocidos de la matriz de usuarios e ítems se completan utilizando alguna estrategia global a nivel general, o global a nivel de usuario o ítem, y posteriormente se aplica el método descrito. Utilizar la matriz obtenida A_k para la obtención de las predicciones, mitiga el efecto de la dispersión inicial de la matriz A, y puede contribuir a la eliminación de ruido presente en los ratings.

Más allá de este trabajo inicial, se han propuesto caminos alternativos de menor costo computacional, como el empleo de métodos de gradiente descendente (Koren, et al., 2009) con vistas a obtener dicho espacio. En los últimos años estos métodos han ganado en popularidad en el ámbito de los sistemas recomendadores.

En esta idea, se explora la obtención (basándose en la matriz usuario-ítem) de un espacio k-dimensional, en el que cada ítem i se asocia a un vector $q_i \in R^k$ y cada usuario u a un vector $p_u \in R^k$. Cada uno de los elementos en los vectores de ítem o de usuario representa la magnitud con que su contenido (en el caso de los ítems) o sus intereses (para los usuarios) está asociado con cada factor k. Basándose en esto, se asume que el producto punto $q_i^T p_u$ representa la interacción entre el usuario u y el ítem i, y que aproximadamente se corresponde con el rating del usuario u sobre el ítem i, denotado por $\hat{r}_{u,i}$ (1. 20).

$$\hat{r}_{u,i} = q_i^T p_u \tag{1. 20}$$

Esto implica que en este escenario la principal tarea a realizar es la de mapear los usuarios y los ítems a los vectores de dimensión k, puesto que una vez obtenidos estos, es posible calcular los valores de preferencia de cualquier usuario por cualquier ítem a través del referido producto punto. La opción más popular es la de obtener los vectores q_i y p_u minimizando el error cuadrático regularizado sobre el conjunto de ratings conocidos $r_{u,i}$ (1. 21), donde la constante λ representa la magnitud de la regularización, y debe ser empíricamente determinada.

$$\min_{q^*, p^*} \sum_{(u,i) \in R} (r_{u,i} - q_i^T p_u)^2 + \lambda (\|q_i\|^2 + \|p_u\|^2) \tag{1. 21}$$

Siguiendo esta idea, (Koren, et al., 2009) muestra el empleo del gradiente descendente estocástico simple para encontrar los parámetros q* y p* en este modelo. Para cada caso de entrenamiento (rating conocido $r_{u,i}$), predice el valor $\hat{r}_{u,i}$ usando el producto punto de los valores actuales q_i and p_u, y determina el error de la predicción con respecto al valor ya conocido (1. 22). Tras esto, q_i and p_u se modifican en una magnitud proporcional a un factor de aprendizaje α, en la dirección opuesta de su gradiente, pudiendo ser esto realizado a través de las ecuaciones (1. 23) y (1. 24).

$$e_{u,i} = r_{u,i} - q_i^T p_u \quad (1.\ 22)$$

$$q_i \leftarrow q_i + \alpha(e_{u,i} p_u - \lambda q_i) \quad (1.\ 23)$$

$$p_u \leftarrow p_u + \alpha(e_{u,i} q_i - \lambda p_u) \quad (1.\ 24)$$

Este modelo de factorización es muy flexible, por lo que puede ser extendido fácilmente. Una de las extensiones más populares es la de considerar el sesgo asociado a cada rating (Koren, et al., 2009). Este se modela incluyendo un valor promedio global de rating μ, y los parámetros b_u y b_i representando la desviación, con respecto a la media, de cada usuario u e ítem i (1. 25). En el caso de los ítems, la desviación tendería a ser positiva para aquellos que sean populares, y negativa para aquellos que no lo sean. Por otro lado, los usuarios críticos usualmente tendrían un sesgo negativo, mientras que los poco críticos tendrían uno positivo. Este sesgo se combina con el mencionado producto punto $q_i^T p_u$ para obtener la predicción del rating (1. 26).

$$b_{u,i} = \mu + b_u + b_i \quad (1.\ 25)$$

$$\hat{r}_{u,i} = b_{u,i} + q_i^T p_u \quad (1.\ 26)$$

Basándose en esto, los parámetros b_u y b_i se incorporan al modelo de optimización (1. 27). En este caso, estos serían también calculados a través de una técnica de gradiente, tal como en el caso anterior con q* y p*.

$$\min_{q^*,p^*,b^*} \sum_{(u,i)\in R} (r_{u,i} - \mu - b_u - b_i - q_i^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2 + {b_u}^2 + {b_i}^2) \quad (1.\ 27)$$

1.3 Sobre la evaluación de los métodos de filtrado colaborativo

Varios marcos de trabajo han sido propuestos para la evaluación del desempeño de los sistemas de recomendación (Ekstrand, et al., 2010; Herlocker, et al., 2004; Ricci, et al., 2011). Shani y Gunawardana (2011) han descrito tres tipos diferentes de experimentos que se pueden hacer en este escenario: el experimento offline, el estudio del usuario, y el experimento online.

Los experimentos offline se llevan a cabo usando una base de datos, obtenida con anterioridad, en la que varios usuarios escogen o evalúan ítems. Usando esta base de datos, se trata de simular el comportamiento de los usuarios cuando interactúan con un sistema recomendador. Estos experimentos son muy populares debido a que no requieren la interacción con usuarios reales, y por tanto permiten realizar comparaciones entre varios algoritmos de recomendación candidatos, con un costo muy bajo. Así, su meta principal es la de descartar enfoques inapropiados, dejando solamente un conjunto de algoritmos relativamente pequeño para ser probados en escenarios mucho más costosos como los de estudios de usuarios o de experimentos en línea. Un ejemplo típico de este proceso es cuando los parámetros de los algoritmos se ajustan en un experimento offline, usando en la siguiente fase sólo el algoritmo con los mejores valores de los parámetros.

Por otro lado, un estudio de usuario (Knijnenburg, Willemsen, Gantner, Soncu, & Newell, 2012) generalmente se conduce sobre un grupo de usuarios de prueba, a los que se les solicita desarrollar una serie de tareas que requieren una interacción con el sistema recomendador. El comportamiento de los usuarios es observado y registrado mientras se llevan a cabo las tareas, almacenándose un número importante de medidas cuantitativas. Estas medidas se centran en evaluar qué porción de la tarea fue completada, la eficacia de este completamiento, o el tiempo que se requirió para completarla. En muchas ocasiones también evalúan medidas cualitativas. El objetivo principal de estas preguntas es el de recolectar datos que no son directamente observables, tales como, por ejemplo, si el usuario disfrutó de la interfaz provista, o si percibió la tarea recibida como fácil de completar.

Finalmente, la evaluación online es el experimento que proporciona la evidencia más fuerte sobre el verdadero valor del sistema. Aquí, el sistema se usa por usuarios reales que llevan a

cabo tareas reales. Como ejemplo de estos casos, muchas aplicaciones ya desplegadas utilizan un sistema online de pruebas (Kohavi, Deng, Longbotham, & Xu, 2014) donde se pueden comparar múltiples algoritmos. Típicamente, tales sistemas redirigen un pequeño por ciento del tráfico a diferentes enfoques de recomendación alternativos, grabándose las interacciones de los usuarios con cada uno de estos.

Gunawardana y Shani (2009) especifican un protocolo para llevar a cabo la evaluación offline partiendo de una base de datos con ratings asociados a usuarios e ítems. Con este fin, proponen seleccionar un conjunto de usuarios de dicha base de datos. Luego, aleatoriamente seleccionar la cantidad de ítems n_u a "ocultar" para cada usuario u. Estos ítems "ocultos" pasarían a formar parte del conjunto de prueba, mientras que los restantes conformarían el conjunto de entrenamiento.

Usando estos dos conjuntos, el referido trabajo propone el cálculo de un nuevo valor para cada rating del conjunto de prueba utilizando el algoritmo de predicción a evaluar y basándose en los datos del conjunto de entrenamiento. Estas predicciones se usan para evaluar el desempeño de dicho algoritmo por medio de diferentes métricas. Las dos más populares son:

a) El error medio absoluto (MAE) de todas las predicciones hechas: MAE se define como el error medio absoluto entre los valores reales de los ratings de prueba, y los ratings predichos.
b) La métrica F1 como combinación de precision y recall: Precision y recall miden el grado en el que el algoritmo presenta información relevante, a través del cálculo de la proporción de ítems que son preferidos y recomendados, con respecto al total de ítems preferidos (recall), y al total de ítems recomendados (precision). Con vistas a combinar estas dos métricas, se define F1= 2*precision*recall/(precision+recall). Para calcular el F1 usando las predicciones se requiere establecer un umbral de relevancia. Así un ítem es considerado relevante (preferido en el caso del conjunto de prueba, recomendado en el caso del conjunto de predicciones generadas) si ha sido evaluado con un rating igual o superior a este umbral, mientras que todos los ratings por debajo de dicho umbral son considerados como no relevantes. Bajo estos criterios, se calcula un valor diferente para cada uno de los usuarios de la base de datos, y finalmente estos se promedian para calcular el valor de F1 global.

1.4 Preprocesamiento de datos en sistemas recomendadores

Actualmente las bases de datos asociadas a cualquier escenario son altamente susceptibles a la presencia de datos ruidosos, erróneos o inconsistentes, típicamente debido a su gran tamaño y a su origen a partir de fuentes múltiples y heterogéneas (Nguyen, 2007). Han y Kamber (2001) consideran que existen cuatro grandes grupos de técnicas para preprocesar los datos con vistas a eliminar estas características indeseables:

1) la limpieza de datos (data cleaning), la que puede ser aplicada para eliminar ruido y corregir inconsistencias en los datos,
2) la integración de datos (data integration), que se centra en mezclar los datos provenientes de fuentes múltiples hacia un escenario único y coherente,
3) la transformación de datos (data transformation), que abarca la aplicación de técnicas como la normalización para mejorar el desempeño de determinados grupos de algoritmos de minería, y
4) la reducción de datos (data reduction), que se centra en reducir el tamaño de la base de datos a través de la agregación o eliminación de rasgos redundantes, o a través del agrupamiento.

Famili, Shen, Weber, y Simoudis (1997), por su parte, consideran que las tareas de preprocesamiento de datos vienen dadas por todas las acciones llevadas a cabo con anterioridad al inicio del proceso real de análisis de datos. Esencialmente, esta tarea se define como una transformación T que convierte los datos iniciales X en un nuevo conjunto de datos Y, de manera que:

1) Y preserva la información “valiosa” presente en X.
2) Y elimina al menos uno de los problemas que están presentes en X, y
3) Y es más útil que X.

Asimismo, identifican diferentes problemas asociados a los datos y que hacen necesaria la aplicación de técnicas de preprocesamiento con vistas a mejorar el desempeño de las técnicas de minería. Entre estos problemas encuentran la presencia de datos corruptos y ruidosos, la ausencia de características representativas, la presencia de información irrelevante, y la presencia de datos incompatibles y de diferentes niveles de granularidad.

Específicamente en el escenario de los sistemas recomendadores, varios autores han planteado y presentado evidencias de que los datos asociados contienen un importante grado de inconsistencia (Amatriain, et al., 2011; Amatriain, Pujol, & Oliver, 2009). Estas

inconsistencias han sido específicamente catalogadas como ruido malicioso o como ruido natural, en dependencia de la causa que las origina (Amatriain, et al., 2011). Por una parte, el ruido malicioso es ocasionado por aquellas insertadas de forma premeditada por un grupo de usuarios que introducen perfiles falsos con vistas a manipular el sistema para promover o degradar cierto ítem (Mehta & Hofmann, 2008). Sin embargo, por otro lado se ha concluido que no todas las inconsistencias que suelen aparecer en los datos tienen carácter premeditado, sino que muchos perfiles de usuarios genuinos pueden tener datos inciertos debido a su comportamiento imperfecto y sin una mala intención previa (O'Mahony, et al., 2006).

En los últimos años se han realizado varios trabajos centrados en el tratamiento de inconsistencias en sistemas recomendadores, utilizando diferentes enfoques de preprocesamiento que caerían dentro de la categoría de la limpieza de datos (data cleaning). En las siguientes secciones se realiza un recorrido por los enfoques más representativos de procesamiento de ruido malicioso y de ruido natural.

1.4.1 Tratamiento de ruido malicioso

Las investigaciones en el área del tratamiento de ruido malicioso asumen que los usuarios tienen propósitos muy diversos cuando interactúan con los sistemas recomendadores, y que en algunos casos, estos propósitos pueden ser contrarios a los del proveedor del sistema y a los de la mayoría de los usuarios. De igual forma que en una herramienta clásica de búsqueda la meta de un usuario "atacante" es hacer que la página web que este desea promover parezca una buena respuesta ante cualquier consulta realizada, en el caso de la recomendación, y sobre todo en la recomendación basada en filtrado colaborativo (que es una de las más vulnerables a los ataques), el objetivo es hacer que determinado ítem se presente como una buena recomendación a un usuario en particular (o tal vez para todos los usuarios), independientemente de que este realmente lo sea. Alternativamente, un ataque puede centrarse en evitar que un producto sea recomendado, inclusive siendo este realmente una buena opción.

Partiendo de que la mayoría de los sistemas de filtrado colaborativo realizan las recomendaciones basándose puramente en la información de los perfiles de usuario, la mayoría de las estrategias de ataque se centran en la adición de nuevos perfiles que sesguen la salida del método de recomendación con vistas a producir el efecto deseado. La posibilidad de diseñar nuevos perfiles de usuarios para deliberadamente provocar este efecto, fue inicialmente

analizada por O'Mahony, Hurley, y Silvestre (2002). A partir de este trabajo, la mayoría de los estudios en dicha dirección pueden ubicarse en cuatro grupos fundamentales (Gunes, Kaleli, Bilge, & Polat, 2014).

-El primero de estos está enfocado en modelar nuevas estrategias de ataque,

-El segundo en definir algoritmos para la detección de ataques,

-El tercero en el análisis de la robustez de los métodos tradicionales de recomendación y la formulación de nuevos algoritmos robustos ante los ataques,

-Finalmente el último grupo se centra en el análisis costo-beneficio de la mitigación de los ataques, y en la discusión del costo asociado a la realización de los ataques.

Burke, O'Mahony, y Hurley (2011) formalmente definen un ataque de inyección de perfiles contra un sistema recomendador, como la introducción en el sistema, por parte del atacante, de un conjunto de perfiles. Específicamente, un perfil está conformado por un conjunto de pares (rating, ítem), y en todos los casos siempre existe un ítem objetivo i que el atacante está interesado en promover o degradar. En adición al ítem destino usualmente también existe un conjunto de ítems adicionales, denominados "de relleno", que igualmente juegan un papel importante. Considerando esto, Burke et al. realizan una descripción detallada de los diferentes tipos de ataque, entre los que se destacan el ataque aleatorio (Random Attack), el ataque basado en promedio (Average Attack), el Bandwagon attack, el Reverse Bandwagon attack, el Popular attack, y la Probe attack strategy. El anexo 1 detalla las especificidades de cada una de estas estrategias de ataque.

Tomando en consideración la posible diversidad de estrategias de ataque, la detección de perfiles maliciosos se considera de vital importancia para prevenir los efectos negativos de estos y para proteger a los ítems usualmente utilizados como objetivos. Con este fin se han formulado varias técnicas agrupadas en cuatro categorías fundamentales: métodos estadísticos, métodos de agrupamiento, métodos de clasificación, y métodos basados en la reducción de los datos (Gunes, et al., 2014).

Entre los trabajos más relevantes basados en procesamiento estadístico está el desarrollado por Hurley, Cheng y Zhang (2009), donde se utiliza la teoría estadística de detección de Neyman-Pearson a través del desarrollo de una prueba de hipótesis binaria para discriminar entre perfiles genuinos y perfiles de ataque. Li y Luo (2011) proponen el empleo de modelos basados en redes bayesianas para determinar si un nuevo perfil es malicioso o auténtico. Más recientemente, Chung, Hsu y Huang (2013) han propuesto un método denominado Beta-

Protection, que se basa en la distribución beta para detectar y eliminar perfiles de usuarios maliciosos. En una dirección semejante, Zhang y Zhou (2014) proponen un método para la detección de ataques de inyección de perfiles, basado en la transformada Hilbert-Huang y las máquinas de soporte vectorial.

Por otro lado, se han utilizado diferentes técnicas de clasificación supervisada en la detección de varios esquemas de ataque. Específicamente, Burke, Mobasher, Williams y Bhaumik (2006a, 2006b) proponen un enfoque de clasificación para la detección de usuarios maliciosos basándose en atributos derivados de cada perfil individual, tomando como referencia el trabajo previamente hecho por Chirita, Nejdl, y Zamfir (2005). Entre estos atributos están la desviación de los ratings con respecto a la media consensuada (RDMA) (ecuación (1. 28)), el grado de similitud con los vecinos más cercanos (DegSim) (ecuación (1. 29)), la desviación pesada con respecto a la media consensuada (WDMA) (ecuación (1. 30)), el nivel de consenso pesado (WDA) (ecuación (1. 31)), y la variación de la longitud del perfil (lengthVar) (ecuación (1. 32)). En las ecuaciones, U representa el universo de usuarios u en la base de datos, $r_{u,i}$ el rating del usuario u sobre el ítem i, n_u la cantidad de ratings pertenecientes al usuario u, l_i la cantidad de ratings asociados al ítem i, $W_{u,v}$ el grado de semejanza entre los usuarios u y v calculado a través del coeficiente de correlación de Pearson, $\bar{r}_i$ el rating promedio para el ítem i, y finalmente $\bar{n}_u$ la cantidad de ratings promedio por usuario.

$$RDMA_u = \frac{\sum_{i=0}^{n_u} \frac{\left| r_{u,i} - \bar{r}_i \right|}{l_i}}{n_u} \quad (1.\ 28)$$

$$DegSim_u = \frac{\sum_{v \in neighbors(u)} W_{u,v}}{k} \quad (1.\ 29)$$

$$WDMA_u = \frac{\sum_{i=0}^{n_u} \frac{\left| r_{u,i} - \bar{r}_i \right|}{l_i^{\,2}}}{n_u} \quad (1.\ 30)$$

$$WDA_u = \sum_{i=0}^{n_u} \frac{\left| r_{u,i} - \bar{r}_i \right|}{l_i} \quad (1.\ 31)$$

$$LengthVar_u = \frac{n_u - \bar{n}_u}{\sum_{u \in U} \left(n_u - \bar{n}_u \right)^2} \quad (1.\ 32)$$

En adición a estos atributos genéricos, Burke et al. (2006a) proponen otros que son específicos del modelo de ruido malicioso que se desee procesar. En este caso, la detección basada en estos atributos se centra en el descubrimiento de particiones del perfil en cuestión, tratando de maximizar la posible similitud con respecto a un modelo específico de ataque.

Por otra parte, Mobasher, Burke, Bhaumik, y Williams (2007) plantean que si se desea ejecutar un ataque sobre un ítem específico, hay una mayor posibilidad de que exista un conjunto de perfiles inyectados asociados a este, con respecto a que exista un único perfil. A raíz de esto, introducen un nuevo grupo de atributos, denominados atributos de detección inter-perfil, con el fin de descubrir este tipo de ataques. Dichos atributos se centran en medir el grado con el que las particiones de un perfil dado se enfocan en los ítems que son comunes a las particiones asociadas a otros perfiles.

La efectividad de todos estos atributos es medida en un escenario de clasificación supervisada en el que previamente tanto en el conjunto de entrenamiento como en el de prueba se inyectan de forma manual perfiles maliciosos, y en el que utilizando los datos del conjunto de entrenamiento y un clasificador específico, se intenta identificar los perfiles maliciosos del conjunto de prueba. Entre los clasificadores más comúnmente empleados en este escenario están el kNN, el C4.5 y las máquinas de soporte vectorial (SVM) (Burke, et al., 2006a, 2006b; Mobasher, et al., 2007; Williams, Mobasher, & Burke, 2007). Alternativamente, Mobasher et al. (2007) proponen un modelo híbrido de detección combinando técnicas estadísticas con clasificación, a través de atributos específicos del modelo. Por otra parte, He, Wang, y Liu (2010) se apoyan en la teoría de los conjuntos aproximados para llevar a cabo un enfoque de clasificación que etiqueta cada perfil de usuario como ataque o como auténtico.

Varios enfoques basados en agrupamiento también se han utilizado para la detección de perfiles maliciosos en sistemas recomendadores. O'Mahony, Hurley, y Silvestre (2003) utilizan este enfoque para detectar perfiles maliciosos que se centran en degradar la popularidad de ciertos ítems. Específicamente proponen agrupar periódicamente los usuarios de la base de datos y chequear si los centros de los grupos cambian significativamente. En este caso, los perfiles extremos que afectan los centros se consideran maliciosos. Mehta y Nejdl (2009) y Bhaumik, Mobasher y Burke (2011) utilizan enfoques de agrupamiento con este mismo propósito, considerando además que la aparición de grupos relativamente pequeños puede ser indicador de la presencia de usuarios maliciosos.

Finalmente, también se ha desarrollado otro grupo de trabajos basándose en técnicas como el análisis de componentes principales (PCA) (Mehta, 2007; Mehta & Nejdl, 2009), o la descomposición en valores singulares (SVD) (Zhang, Ouyang, Ford, & Makedon, 2006).

1.4.2 Tratamiento de ruido natural

La sección anterior mostró la existencia de una importante cantidad de investigaciones centradas en el tratamiento de ruido malicioso en sistemas recomendadores. Estos trabajos se enfocan en la detección de perfiles de usuario ruidosos insertados intencionalmente con vistas a alterar el resultado de la recomendación. Sin embargo, algunos autores han insistido en que esta no es la única fuente de inconsistencias en los datos de los sistemas recomendadores.

Específicamente en los escenarios basados en filtrado colaborativo, Amatriain, Pujol y Oliver (2009) mostraron que los usuarios tienden a tener un comportamiento inconsistente cuando evalúan los ítems, provocando esto un efecto negativo en el funcionamiento del sistema recomendador. Así, desarrollan un conjunto de experimentos para obtener el grado de inconsistencia de los usuarios al evaluar ítems, la magnitud de error en la predicción debido a estas inconsistencias, y los factores que las ocasionan. Considerando las evaluaciones hechas por los usuarios en diferentes momentos de tiempo, los autores llegan a la conclusión de que la predicción puede variar considerablemente, demostrándose así que el ruido natural afecta con notoriedad la calidad de las recomendaciones. Específicamente, utilizando los datos capturados en diferentes instantes de tiempo en la generación de las predicciones, se obtuvo como resultado que el error cuadrático medio varió en un rango entre 0,557 y 0,8156.

En esta misma dirección, Ekstrand et al. (2010) por su parte, han considerado que el proceso de expresar preferencias como ratings no está libre de errores, que los ratings provistos por los usuarios pueden contener ruido, y que la detección y compensación de este tiene potencial como para devenir en la obtención de mejores sistemas recomendadores. En adición, Konstan y Riedl (2012) han planteado que uno de los retos a tener en cuenta en las investigaciones en sistemas recomendadores es que los datos en los que se basan las recomendaciones pueden ser ruidosos, corruptos, o simplemente erróneos. Recientemente (Bellogín, Said, & de Vries, 2014; Said, Jain, Narr, & Plumbaum, 2012; Said et al., 2012) han considerado que los ratings de los usuarios son inherentemente ruidosos, y que ningún rating debe verse como una verdad absoluta, y por otro lado Kluver, Nguyen, Ekstrand, Sen y Riedl (2012) han afirmado que estas inconsistencias afectan el poder predictivo de los sistemas recomendadores.

De esta forma, se concluye que el tratamiento del ruido natural es un elemento clave en los sistemas recomendadores, y que tener en cuenta estas inconsistencias en el diseño de algoritmos de filtrado colaborativo implicaría la obtención de recomendaciones más eficaces. En las siguientes secciones se presentarán los aportes más relevantes realizados en esta línea de investigación.

- Consistencia de cada rating con respecto al resto de las preferencias

O'Mahony et al. (2006) relacionan la aparición de ruido natural con la forma en que los sistemas recomendadores colectan o infieren las preferencias. Muchos de estos sistemas operan sobre esquemas explícitos de ratings, en los que al usuario se le solicita expresar su grado de preferencia sobre ítems que ha visto o comprado. Partiendo de que toda actividad humana es propensa al error, afirman que en este proceso por naturaleza deben introducirse inconsistencias en los datos.

En el mencionado trabajo se define como problema el de determinar si un posible rating $r_{u,i}$ contiene ruido natural o es libre de ruido. Con vistas a darle solución, se adopta el enfoque de determinar el grado de consistencia de un rating asociado a un par usuario-ítem en particular, con respecto a la predicción realizada para el mismo par a través de un algoritmo de recomendación G. Para realizar esta predicción, se asume la existencia de un conjunto de entrenamiento T. Específicamente, se señala que la selección de los usuarios que formarán parte de este conjunto de entrenamiento se debe realizar de forma manual por parte del administrador del sistema, con vistas a garantizar que sean usuarios genuinos.

Utilizando G y T, se define la consistencia c de un rating $r_{u,i}$ como el error absoluto entre el rating real y el rating predicho, de la forma:

$$c(G,T)_{u,i} = \frac{\left|r_{u,i} - p_{u,i}\right|}{r_{\max} - r_{\min}} \qquad (1.\ 33)$$

donde $p_{u,i}$ es el rating predicho para el par usuario-ítem (u, i) y r_{min} y r_{max} son los valores de rating mínimo y máximo posible. El término de normalización se inserta para facilitar la comparación entre sistemas que utilicen diferentes escalas de valoración. Tras esta definición, un rating se considera ruidoso y por tanto descartado del proceso de generación de recomendaciones, si verifica:

$$c(G,T)_{u,i} > \delta \tag{1. 34}$$

donde δ es un valor de umbral predefinido.

Este enfoque asume que la precisión de cualquier algoritmo de recomendación G garantiza que la diferencia entre el rating predicho y el rating real para cualquier par usuario-ítem siempre está por debajo del umbral δ, y que cualquier diferencia que exceda este umbral implica que el rating real correspondiente sea ruidoso.

-Eliminación de ruido a través de la reevaluación de ítems

Amatriain et al. (2009) presentan un enfoque para mejorar la eficacia de los sistemas recomendadores a través de la reducción del ruido natural siguiendo un procedimiento dependiente del usuario. Con vistas a comprender el impacto cuantitativo del ruido natural, inicialmente los autores analizan la respuesta de algoritmos tradicionales de recomendación ante este, utilizando para ello datos obtenidos de la evaluación online realizada por un grupo de usuarios sobre el mismo conjunto de películas en tres instantes de tiempo diferentes. Como resultado se obtuvo que el error cuadrático medio de las predicciones realizadas varió considerablemente en cada uno de los instantes de tiempo, tomando como referencia los algoritmos de recomendación basados en similitud usuario-usuario e ítem-ítem, y un método que se apoya en la reducción de dimensión.

Como núcleo del trabajo, se presenta un algoritmo que recibe como entrada un conjunto de t+1 ratings que un usuario ha dado a un ítem específico (el rating original más t re-ratings), y devuelve un único rating r sin ruido asociado. Este rating final se obtiene a través de la eliminación recursiva de todo par de ratings (a, b) cuya diferencia sea mayor que un valor γ, y a través del reemplazo de todo par cuya diferencia sea menor o igual que γ, por el promedio de los dos ratings.

En adición, se analiza la variante de depurar únicamente un grupo de ratings seleccionados y no todos los ratings como inicialmente se plantea. En esta dirección, se examina la relación entre el por ciento de ratings depurados y el error de la predicción. Específicamente, se discuten las alternativas de seleccionar aleatoriamente los ratings a procesar, los ratings extremos, los ratings ubicados en el centro de la escala de puntuaciones, y sólo los ratings asociados a usuarios inconsistentes. Siguiendo esta última estrategia, se logra una mejora de alrededor del

5 % del error cuadrático medio, reevaluando menos del 10 % de los ítems asociados al 30 % de los usuarios del experimento hecho.

-Corrección de ratings basada en las preferencias del usuario

En los últimos años, Pham y Jung han propuesto un método que utiliza información adicional a los ratings de los usuarios, con vistas a detectar y corregir preferencias inconsistentes en los sistemas recomendadores (Hwang, et al., 2011; Pham & Jung, 2013). En este enfoque, se propone determinar si los ratings son consistentes con respecto a las preferencias de los usuarios, las que son representadas por un conjunto de valores de atributos dominantes.

Estos trabajos parten de un modelo usuario-ítem representado por un cuádruplo (U, I, A, R) donde U es el conjunto de usuarios, I es el conjunto de ítems, A es el conjunto de atributos de los ítems, y R es un conjunto de ratings de los usuarios sobre los ítems. En el procedimiento que se propone, inicialmente se determinan los atributos que son dominantes para cada uno de los ítems, y a la vez las preferencias de cada uno de los usuarios en término de estos atributos. Utilizando esta información, se define la cobertura de las preferencias de un usuario sobre un ítem específico, como el grado de solapamiento entre dichas preferencias y los atributos dominantes correspondientes a este ítem. A su vez, esta cobertura se utiliza para determinar un conjunto de ítems $Item_{pref}(u)$ que corresponden a las preferencias de los usuarios, y un conjunto $Item_{unpref}(u)$ de los que no pertenecen a las preferencias. Tras esto, un rating se marca como incorrecto si el ítem asociado pertenece a las preferencias del usuario actual, pero su valor está por debajo de un umbral predefinido y del valor medio de los ratings para ese usuario.

Estos ratings detectados como incorrectos son corregidos. Para esto, se formaliza un criterio para seleccionar un conjunto de usuarios expertos para cada uno de los posibles usuarios, teniendo en cuenta el solapamiento de preferencias y la cantidad de ratings asociados. Usando estos expertos, se proponen tres criterios de corrección de ratings. El primero se basa en sustituir el rating actual por un promedio pesado de los ratings, sobre el mismo ítem, de los expertos más parecidos al usuario actual. El segundo considera un promedio pesado de los ratings dados por el mismo usuario a ítems semejantes que pertenecen al modelo de preferencia del usuario, y el tercero calcula la media de los valores obtenidos por los dos enfoques previos.

Como una continuidad a este trabajo, Pham et al. en (Pham, Jung, & Nguyen, 2012, 2013) proponen nuevas estrategias para integrar la información de múltiples expertos con vistas a

mejorar el proceso de corrección. Específicamente, proponen cuatro nuevas soluciones para realizar el cálculo de los nuevos valores de preferencias, denominadas respectivamente Mejor Correspondencia (Best Matching), Valoración Mayoritaria (Majority Rating), Basado en Pesos (Weighting), y Consenso Máximo (Maximal Consensus).

-Detección de usuarios ruidosos pero no maliciosos

Tomando como base los numerosos trabajos asociados a la detección de ruido malicioso en sistemas recomendadores, Li et al. (2013) han propuesto la detección de usuarios ruidosos pero no maliciosos. Basándose en la asunción de que los ratings provistos por un mismo usuario sobre ítems fuertemente correlacionados deben tener valores similares, se propone un método para la detección de usuarios ruidosos pero no maliciosos a través de la captura y acumulación de "auto-contradicciones" asociadas a cada uno de los usuarios. Precisamente, estas "auto-contradicciones" se definen como los casos en los que un usuario proporciona valores muy diferentes de preferencia para ítems altamente correlacionados.

La propuesta se basa inicialmente en conformar para cada uno de los usuarios un grafo de correlaciones G entre ítems, partiendo de sus ratings conocidos y_L. Posterior a esto, se realiza la estimación en términos de y_L de los ratings desconocidos para el usuario correspondiente, a través de la propagación en G de los ratings en y_L. En un tercer momento, a cada rating en y_L se le agrega un parámetro desconocido para representar el ruido asociado correspondiente, posibilitando que tanto los ratings conocidos como los desconocidos puedan ser representados en términos de estos parámetros agregados. Así, basándose en el ya mencionado supuesto de que para un usuario los valores genuinos de las preferencias sobre ítems que son vecinos en el grafo G deben tener valores cercanos de ratings, se propone el cálculo de los parámetros desconocidos a través de la minimización de una función de costo que combina la diferencia entre los ratings (conocidos y desconocidos), y los valores de correlación entre los ítems. Finalmente, el grado de inconsistencia del usuario activo se calcula promediando los valores obtenidos para los parámetros desconocidos asociados a los ratings pertenecientes a dicho perfil. Específicamente, el cálculo de los valores que minimizan la función de costo se modela como un problema de optimización cuadrática con restricciones.

Tras identificar los usuarios más ruidosos y con vistas a elevar la eficacia de la recomendación, se propone excluir los ratings proporcionados por estos, de los datos que se utilizan como base para la generación de las recomendaciones.

La efectividad de este procedimiento fue verificada experimentalmente partiendo de la selección aleatoria de un grupo de usuarios, e insertándose ruido natural en una fracción de los ratings de cada uno, a través de la modificación también aleatoria de los valores de estos.

1.4.3 Insuficiencias de los métodos para el tratamiento de ruido natural en sistemas recomendadores de filtrado colaborativo

Las secciones anteriores mostraron que a pesar de haberse desarrollado una gran cantidad de trabajos centrados en la manipulación de inconsistencias en los datos de los sistemas recomendadores, la mayoría de estos se enfocan en el tratamiento de ruido malicioso. En contraste, también se pudo apreciar que en los últimos años varios autores se han centrado en las inconsistencias asociadas al comportamiento imperfecto de los usuarios, las que son denominadas como ruido natural.

La tabla 1.2 muestra un sumario de los trabajos presentados que se centran en el problema del ruido natural. Tomando en consideración su principio de funcionamiento, estos han sido clasificados en dos taxonomías fundamentales que se basan en las fuentes de información empleadas, y en el tratamiento que se le da a los datos detectados como inconsistentes.

Tabla 1.2 Trabajos centrados en el procesamiento de ruido natural

	Dependen de información adicional más allá de los ratings	**No dependen de información adicional**
Eliminan los datos ruidosos		O'Mahony et al. (2006) Li et al. (2013)
Corrigen los datos ruidosos	Amatriain et al. (2009) Pham y Jung (2013) Pham et al. (2012, 2013)	

Por una parte, el trabajo presentado por Amatriain et al. (2009) se centra en la corrección de datos ruidosos, pero depende de que el usuario exprese varias veces su preferencia sobre el mismo ítem, lo que resulta en la práctica difícil de lograr en un sistema recomendador en tiempo real. En esta misma dirección, Pham y Jung han presentado alternativas para la corrección de ratings, pero que dependen también de información adicional, la que en este caso viene dada por la presencia de atributos asociados a los ítems. Por otro lado, O'Mahony et al. (2006) y Li et al. (2013) han propuesto métodos para el procesamiento del ruido natural que no dependen de información adicional, pero que implican pérdida de información, puesto que se basan en la eliminación de ratings en el caso de O'Mahony et al. (2006); e incluso en la eliminación de

perfiles completos de usuarios, en Li et al. (2013). Resulta evidente la existencia de una brecha importante respecto a la disponibilidad de métodos que sean capaces de corregir los datos ruidosos, sin necesitar información adicional más allá de los valores de preferencia.

Finalmente, Li et al. (2013) también han demostrado que existen varios factores que impiden la utilización de métodos centrados en el procesamiento de ruido malicioso, para cubrir esta brecha asociada al ruido natural. Entre estos, resalta el carácter no intencional de este tipo de ruido, eliminando así la alternativa de entrenar un modelo de clasificación efectivo para descubrir las inconsistencias, debido a que en este caso es muy difícil encontrar rasgos identificativos. Además, el carácter diverso de las inconsistencias imposibilita la aplicación de enfoques no supervisados como el agrupamiento con vistas a identificarlas. Esto reafirma la necesidad de desarrollar trabajos centrados en este problema en particular.

A raíz de todo lo anterior, en la presente tesis se concibe un nuevo enfoque centrado específicamente en el preprocesamiento de ruido natural, que no depende de información adicional, y que se basa en la corrección de la información detectada como inconsistente.

Conclusiones del capítulo

La revisión de la literatura permitió arribar a las siguientes conclusiones:

-Los sistemas recomendadores se han convertido en herramientas imprescindibles para los usuarios, en la obtención de aquella información, en el contexto digital, que mejor se corresponda con sus intereses y preferencias.

-Los métodos de recomendación de filtrado colaborativo gozan de gran popularidad, al estar entre los de mejor desempeño dependiendo sólo de los valores de preferencia de los usuarios.

-El funcionamiento de los sistemas recomendadores puede verse notablemente afectado por la presencia de inconsistencias en las preferencias de los usuarios.

-Las preferencias inconsistentes en sistemas recomendadores se clasifican en dos grandes grupos: ruido malicioso, conformado por perfiles de usuario insertados deliberadamente con vistas a sesgar el funcionamiento del sistema, y ruido natural, conformados por preferencias asociadas a perfiles de usuarios genuinos que pueden tener datos inciertos debido a su comportamiento imperfecto.

-Existe una importante insuficiencia de trabajos centrados en el tratamiento de ruido natural, que no dependan de información adicional y que no eliminen las preferencias de los usuarios. Hacia esta dirección va encaminada la presente tesis.

2. PREPROCESAMIENTO DE DATOS PARA SISTEMAS RECOMENDADORES DE FILTRADO COLABORATIVO, INDEPENDIENTE DE INFORMACIÓN ADICIONAL

Introducción

En el presente capítulo se propone un método para el procesamiento de ruido natural en sistemas recomendadores de filtrado colaborativo. La sección 2.1 propone un esquema general de este método, mientras que la sección 2.2 lo desarrolla en un escenario tradicional de filtrado colaborativo con ratings explícitos. La propuesta tiene la peculiaridad de que es independiente de cualquier otra información adicional más allá de los propios ratings. Como parte de este desarrollo, se realiza un estudio experimental con vistas a evaluar su efecto sobre diferentes algoritmos de recomendación. En adición, se estudian otros criterios de interés como el nivel de intrusión del método y sus tiempos de ejecución. Finalmente, se termina con las conclusiones parciales.

Los resultados obtenidos en este capítulo han sido previamente publicados en Yera Toledo, Caballero Mota, y Martínez (2015).

2.1 Un método general para el procesamiento de ratings ruidosos basado en contradicciones

El objetivo principal de la presente investigación es el de proponer un enfoque para el procesamiento de preferencias ruidosas en sistemas recomendadores de filtrado colaborativo, específicamente preferencias afectadas por ruido natural. A su vez, se desea que este enfoque no dependa de información adicional más allá de las propias preferencias de los usuarios. Con este fin se parte del concepto de *contradicción* como una de las formas principales de caracterizar las inconsistencias. Refiriéndose a las nociones de lógica clásica, Nguyen (Nguyen, 2007) indica que en un contexto de gestión del conocimiento inconsistente, una contradicción

se puede definir como una incompatibilidad entre dos o más proposiciones, apareciendo cuando dichas proposiciones dan lugar a conclusiones lógicamente opuestas.

Varios de los trabajos de tratamiento de ruido natural analizados en el capítulo anterior se basan en el descubrimiento de contradicciones. Entre estos, es meritorio citar los desarrollados por Pham y Jung (Pham & Jung, 2013; Pham, et al., 2012, 2013), donde las contradicciones se buscan entre los valores de los atributos asociados a un determinado ítem preferido por el usuario, y los atributos globales preferidos por este.

A raíz del objetivo principal de la tesis, se requiere un enfoque basado en contradicciones en el que sólo se utilicen los datos de las categorías principales de la recomendación asociadas al escenario básico y mostradas en la tabla 1.1 (usuarios, ítems y ratings). Un requerimiento adicional es que el procesamiento se haga a nivel de ratings y no a nivel de usuarios o ítems, considerando que la meta final es la corrección de estos con vistas a llenar la brecha planteada en la tabla 1.2.

La definición de esta propuesta requiere, por tanto, inicialmente una caracterización de estas tres categorías. En el caso de los usuarios, varios autores han mostrado que es posible su caracterización basándose exclusivamente en sus ratings. Cabe citar a Hu y Pu (2013) y Cantador, Fernández-Tobías, Bellogín, Kosinski, y Stillwell (2013), que han mostrado recientemente que la personalidad de los usuarios está relacionada con los ratings que estos proveen, y que por tanto es viable realizar una clasificación del usuario acorde a sus ratings. A su vez, Jeong, Kim, Park, y Kwak (2013) recientemente han presentado un trabajo en el que se hace uso de los factores de predisposición de los usuarios, obtenidos a través de cuestionarios, con vistas a caracterizarlos y así enfrentar los típicos problemas de escalabilidad y dispersión de los datos en los sistemas recomendadores.

En el caso de los ítems, varios enfoques de recomendación han considerado, de un modo más o menos implícito, clasificaciones semejantes. Lemire y Maclachlan (2005) consideran como parte de su método de recomendación bipolar slope one, la existencia de ítems preferidos y no preferidos por parte de los usuarios. A su vez Zhang y Zeng (2012), utilizan en su trabajo clasificaciones como ítems “hot” (ítems con muchos ratings), e ítems “niche” (ítems con pocos ratings). Por otra parte, varios trabajos como (Vargas, Castells, & Vallet, 2012) y (Yeh & Cheng, 2015) emplean deliberadamente el término “popular” para referirse a los ítems.

Finalmente, en el caso de los ratings, el hecho de estar usualmente representados por un valor numérico o lógico hace directamente posible su clasificación en determinadas categorías.

Considerando todo lo anterior, la figura 2.1 propone el esquema de un método general basado en contradicciones para el procesamiento de ratings ruidosos. Este inicialmente utilizaría los datos disponibles (ratings) para realizar una clasificación de los perfiles de los usuarios, los perfiles de los ítems, y de los propios ratings, Tras esto, dichas clasificaciones se analizan en busca de contradicciones, y a partir de estas se identifican los ratings posiblemente ruidosos. Finalmente, se propone aplicar a estos ratings una estrategia de corrección, con vistas a obtener una base de datos carente de ruido natural o al menos con este mitigado. En el resto de la tesis serán presentados enfoques para el tratamiento de ruido natural, siguiendo este esquema.

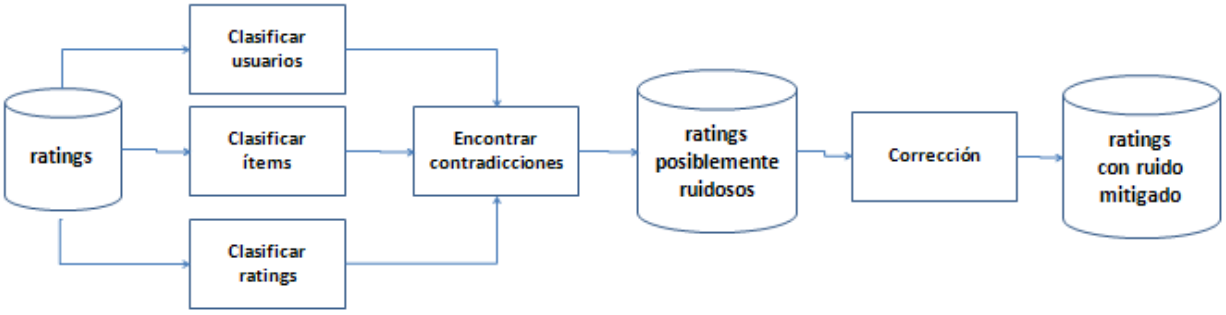


Figura 2.1. Método general para el procesamiento de ratings ruidosos

2.2 Preprocesamiento de ratings ruidosos en un escenario de filtrado colaborativo

En la presente sección se toman como base las ideas planteadas en el epígrafe 2.1 para concebir un método de preprocesamiento en un escenario clásico de filtrado colaborativo, similar al presentado en la tabla 1.1, donde los ratings están enmarcados en un rango numérico dado. La propuesta está conformada por una etapa de detección (sección 2.2.1) y una etapa de corrección (sección 2.2.2). En adición, se realiza un análisis detallado de las alternativas para la inicialización de los parámetros del método (sección 2.2.3), se presenta un ejemplo ilustrativo del funcionamiento del mismo (sección 2.2.4), el análisis teórico de su costo computacional (sección 2.2.5), y la evaluación experimental (sección 2.2.6).

2.2.1 Detección de ratings posiblemente ruidosos

Con vistas a detectar ratings afectados por ruido natural, acorde a las etapas del método presentado con anterioridad, inicialmente se lleva a cabo un proceso de clasificación para *ratings*, *usuarios*, e *ítems*. Partiéndose de lo planteado por Amatriain et al. (2009) sobre la presencia de ratings extremos (tanto bajos como altos) y de ratings medios, se propone clasificar cada rating en tres clases diferentes acordes a su valor:

Clases para los ratings:

1) *Débil*: si $r(u,i) < \kappa$.
2) *Medio*: si $\kappa \leq r(u,i) < \nu$.
3) *Fuerte*: cuando $r(u,i) \geq \nu$.

Esta clasificación depende de un umbral κ (*kappa*) débil-medio y un umbral ν (*nu*) medio-fuerte, debiendo satisfacer ambos la condición $\kappa < \nu$. Los valores de estos parámetros podrían basarse en información global, o ser dependientes del usuario (pasando entonces a llamarse κ_u y ν_u), o dependientes del ítem (siendo entonces κ_i y ν_i). En la sección 2.2.3 serán analizadas con mayor detenimiento estas propuestas de inicialización.

Por otro lado, partiendo de lo planteado en la sección 2.1 sobre lo sugerido por varios autores sobre las tendencias que pueden tener usuarios e ítems cuando respectivamente dan o reciben ratings (Cantador, et al., 2013; Hu & Pu, 2013; Jeong, et al., 2013; Lemire & Maclachlan, 2005; Zhang & Zeng, 2012), se propone identificar a los usuarios e ítems respectivamente acorde a cuatro tendencias. En cada caso, las tres primeras están asociadas a cada una de las tres clases de rating, mientras que la cuarta está concebida para aquellos usuarios e ítems cuyos ratings no tienden a inclinarse hacia ninguna clase en específico. Estas tendencias serían:

Tendencias de los usuarios:

1) Usuario positivo: el usuario tiende a evaluar todos los ítems positivamente.
2) Usuario promedio: el usuario suele proveer valoraciones promedio.
3) Usuario negativo: el usuario tiende a dar valoraciones bajas.
4) Usuario variable: las valoraciones del usuario oscilan entre las categorías previas, sin poder clasificarlo en alguna de ellas en específico.

Tendencias de los ítems:

1) Ítem preferido: el ítem tiende a ser preferido por todos los usuarios.
2) Ítem preferido en promedio: el ítem tiende a ser medianamente preferido.
3) Ítem no preferido: el ítem es no preferido por la mayoría de los usuarios.
4) Ítem dudoso: existen opiniones contradictorias con respecto a su preferencia.

Con vistas a facilitar la definición formal para cada tendencia de usuario y de ítem, se propone agrupar los ratings considerando las mencionadas clases y los umbrales dependientes de los usuarios y los ítems.

Siendo U e I el conjunto de usuarios e ítems, se propone agrupar las preferencias para cada usuario u, en los siguientes conjuntos W_u, A_u y S_u

1) Conjunto de ratings débiles provistos por el usuario *u,* W_u:
   $W_u=\{r(u,i) \mid \forall i \in I \text{ donde } r(u,i) < \kappa_u\}$

2) Conjunto de ratings medios provistos por el usuario *u,* A_u:
   $A_u=\{r(u,i) \mid \forall i \in I \text{ donde } \kappa_u \leq r(u,i) < v_u\}$

3) Conjunto de ratings fuertes provistos por el usuario *u,* A_u:
   $S_u=\{r(u,i) \mid \forall i \in I \text{ donde } r(u,i) \geq v_u\}$

Del otro lado, para cada ítem i, las preferencias son agrupadas en los conjuntos W_i, A_i y S_i:

1) Conjunto de ratings débiles asignados al ítem *i,* W_i:
   $W_i= \{r(u,i) \mid \forall u \in U \text{ donde } r(u,i) < \kappa_i\}$

2) Conjunto de ratings medios asignados al ítem *i,* A_i:
   $A_i=\{r(u,i) \mid \forall u \in U \text{ donde } \kappa_i \leq r(u,i) < v_i\}$

3) Conjunto de ratings fuertes asignados al ítem *i,* S_i:
   $S_i=\{r(u,i) \mid \forall u \in U \text{ donde } r(u,i) \geq v_i\}$

Considerando estos conjuntos y las clases de los ratings, formalmente se definen nuevas clases asociadas a cada tendencia de usuario e ítem previamente mencionada. En el caso de los usuarios, sería posible etiquetarlos como:

- *Benevolente:* usuarios que pertenecen al grupo positivo.
- *Promedio*: aquellos que pertenecen al grupo promedio.

- *Crítico*: aquellos que pertenecen al grupo negativo.
- *Variable*: usuarios dubitativos.

De una forma similar se define una clasificación para los ítems, etiquetándolos como:

- *Fuertemente preferido*: ítems que pertenecen al grupo preferido.
- *Medianamente preferido:* ítems que pertenecen al grupo preferido en promedio.
- *Débilmente preferido:* ítems que pertenecen al grupo no preferido.
- *Variablemente preferido:* aquellos que pertenecen al grupo dudoso.

Estas clasificaciones se formalizan a través del uso de las cardinalidades de los conjuntos W_u, A_u, S_u (para los usuarios), o W_i, A_i, S_i (para los ítems). Cada usuario o ítem se clasifica dependiendo de la cardinalidad de cada conjunto correspondiente, con respecto a la cardinalidad combinada de los otros dos conjuntos (tabla 2.1). En la tabla 2.1, $|X|$ representa la cardinalidad del conjunto X.

Tabla 2.1. Clases de usuario y de ítem

Clases de usuario	
Usuario crítico	$\|W_u\| \geq \|A_u\| + \|S_u\|$
Usuario promedio	$\|A_u\| \geq \|W_u\| + \|S_u\|$
Usuario benevolente	$\|S_u\| \geq \|W_u\| + \|A_u\|$
Usuario variable	No satisface ninguna de las otras tres condiciones
Clases de ítem	
Ítem débilmente preferido	$\|W_i\| \geq \|A_i\| + \|S_i\|$
Ítem medianamente preferido	$\|A_i\| \geq \|W_i\| + \|S_i\|$
Ítem fuertemente preferido	$\|S_i\| \geq \|W_i\| + \|A_i\|$
Ítem variablemente preferido	No satisface ninguna de las otras tres condiciones

Una vez que cada rating, usuario e ítem ha sido clasificado, esta información se usa para encontrar ratings ruidosos analizando la presencia de contradicciones entre las clases. Con este propósito, se definen tres grupos de clases homólogas respectivamente asociadas a usuarios, ítems y ratings (tabla 2.2). En estos grupos, es importante notar que para cada caso son

consideradas como homólogas las clases de usuario e ítem cuya definición (tabla 2.1) tiene como conjunto de mayor cardinalidad al conformado por la misma clase de rating, siendo ambas clases homólogas con la mencionada clase de rating. Así, la clase de usuario *crítico* tiene como homóloga a la clase de ítem *débilmente preferido*, pues en ambos casos los usuarios o ítems se clasifican de tal manera si la cardinalidad del conjunto de ratings *débiles* excede a la cardinalidad combinada de los otros dos conjuntos. Este mismo razonamiento se aplica para las dos restantes clases de rating.

Tabla 2.2. Clases homólogas

	Clases de usuario	Clases de ítem	Clases de rating
Grupo 1	*Crítico*	*Débilmente preferido*	*Débil*
Grupo 2	*Promedio*	*Medianamente preferido*	*Medio*
Grupo 3	*Benevolente*	*Fuertemente preferido*	*Fuerte*

Específicamente, el proceso de detección asume que para un rating r(u,i) de la base de datos, si las clases asociadas con el correspondiente usuario u y el ítem i pertenecen al mismo grupo de la tabla 2.2, entonces el rating r(u,i) también debe pertenecer a dicho grupo. De no verificarse esto, pues entonces r(u,i) podría ser un rating ruidoso, se etiquetaría como posible ruido, y su transformación podría mitigar el ruido natural global asociado con la base de datos.

Esta estrategia no es aplicable para los ratings de los usuarios *variables*, ni para aquellos asociados a ítems *variablemente preferidos*, puesto que para ninguno de los dos casos existe una clase de rating homóloga.

Nota 1: Es importante resaltar los beneficios que trae el hecho de que la clasificación considere la presencia de una clase "variable" para los usuarios y los ítems. Esta se encuentra concebida para cubrir los casos en que estos puedan tener diferentes comportamientos a lo largo del tiempo en el momento de expresar sus preferencias. En tales escenarios, no tiene sentido la búsqueda de ratings con ruido natural.

El algoritmo 2.1 resume el enfoque de detección presentado. Con respecto a este, es importante agregar que se concibe cubriendo la aparición de casos extremos en los que un mismo perfil (de usuario o ítem) se pueda clasificar al mismo tiempo en dos clases distintas, diferentes de las variables. Con respecto a los parámetros de entrada del algoritmo, en la sección 2.2.3 se presentan varias alternativas para su inicialización. Posteriormente, en la

sección de experimentación se realiza un estudio del desempeño de cada una de estas alternativas.

Entrada: r = {r(u, i)} – conjunto de ratings disponibles, κ_u, v_u, κ_i, v_i, κ, v, – umbrales de clasificación **Salida:** possible_noise = {r(u, i)} – conjunto de ratings posiblemente ruidosos	
01	W_u={}, W_i={}, A_u={}, A_i={}, S_u={}, S_i={}
02	possible_noise={}
03	**para cada** rating r(u,i)
04	**si** r(u,i) < κ_u
05	Añadir r(u,i) al conjunto W_u
06	**si no**, si r(u,i) >= κ_u y r(u,i) < v_u
07	Añadir r(u,i) al conjunto A_u
08	**si no**
09	Añadir r(u,i) al conjunto S_u
10	**si** r(u,i) < κ_i
11	Añadir r(u,i) al conjunto W_i
12	**si no**, si r(u,i) >= κ_i y r(u,i) < v_i
13	Añadir r(u,i) al conjunto A_i
14	**si no**
15	Añadir r(u,i) al conjunto S_i
16	**fin**
17	**para cada** usuario u e ítem i
18	Clasificarlo usando los conjuntos asociados, acorde a la definición en la tabla 2.1.
19	**fin**
20	**para cada** rating r(u,i)
21	**si** u es crítico, i es débilmente preferido, y r(u,i) >= κ
22	Añadir r(u,i) al conjunto possible_noise
23	**continuar**
24	**si** u es promedio, i es medianamente preferido, y (r(u,i) < κ o r(u,i)>= v)
25	Añadir r(u,i) al conjunto possible_noise
26	**continuar**
27	**si** u es benevolente, i es fuertemente preferido, y r(u,i) < v
28	Añadir r(u,i) al conjunto possible_noise
29	**continuar**
30	**Fin**

Algoritmo 2.1. Detección de ratings posiblemente ruidosos

2.2.2 Proceso de corrección de ruido

Una vez que el método ha detectado los posibles ratings ruidosos, la siguiente fase manipula estos ratings. La literatura recoge diferentes alternativas para realizar esto, que van desde el

descarte (O'Mahony, et al., 2006), hasta su corrección (Pham & Jung, 2013). La presente propuesta decide adoptar la segunda variante, tomando en cuenta que varios autores han sugerido que es preferible corregir la información anómala presente en los datos (en caso de ser posible) en vez de eliminarla, evitando así una posible pérdida de información (Zhu & Wu, 2004). Por tanto, se propone una estrategia que modifica cada posible rating ruidoso r(u,i), a través del cálculo de un nuevo rating $r^*(u,i)$ para el correspondiente usuario e ítem, a través de un algoritmo de filtrado colaborativo tradicional tomando como base los ratings disponibles (Ekstrand, et al., 2010; Ricci, et al., 2011). En caso de cumplirse la condición $|r(u,i) - r^*(u,i)| > \delta$ (siendo δ un umbral), entonces r(u,i) se reemplaza por $r^*(u,i)$. De lo contrario, r(u,i) mantiene su valor original.

	Entrada: r = {r(u, i)} – conjunto de ratings disponibles, δ – umbral de diferencia **Salida: r* =** {r(u, i)} – conjunto de ratings disponibles corregidos
01	poss_noise = possible_noise_detection()
02	**para cada** rating r(u,i) en poss_noise
03	Predecir un nuevo rating r*(u,i) para el usuario u y el ítem i, a través del filtrado colaborativo usuario-usuario basado en memoria con coeficiente de correlación de Pearson, y usando r como conjunto de entrenamiento
04	**si** (abs(r*(u, i) - r(u, i)) > δ)
05	Reemplazar r(u, i) por r*(u, i) en el conjunto original de ratings r
06	**fin**

Algoritmo 2.2.Corrección de ratings ruidosos

El método de predicción seleccionado para este proceso de corrección es un enfoque de filtrado colaborativo usuario-usuario con el coeficiente de correlación de Pearson como medida de similitud y k=60 como la cantidad de vecinos más cercanos (Resnick, et al., 1994). Este método fue pionero en el campo del filtrado colaborativo, y no necesita para su funcionamiento de la construcción de conocimiento intermedio. Esta última característica permite aliviar el posible costo computacional de la propuesta. Por otro lado, en la sección experimental se justificará el valor de k utilizado en esta predicción.

El algoritmo 2.2 presenta un esbozo del proceso de corrección. Inicialmente, este invoca al algoritmo 2.1 para luego iterar sobre los ratings posiblemente ruidosos, haciendo la corrección basándose en la predicción del nuevo rating usando como entrenamiento todo el conjunto de ratings disponibles.

En la siguiente sección se presentarán varias alternativas para la inicialización del parámetro δ.

2.2.3 Sobre los valores de los parámetros

La propuesta introducida previamente depende de tres grupos de parámetros: los umbrales débil-medio (κ_u, κ_i, and κ), los umbrales medio-fuerte (v_u, v_i, and v), y el umbral de diferencia (δ). Estos parámetros son altamente dependientes del dominio, por lo que resulta muy complicado predeterminar sus valores óptimos. Sin embargo, es posible definir una estrategia para asignarles valores iniciales buenos.

Con este propósito serán evaluados dos enfoques de inicialización. El primero de estos sigue una perspectiva global, mientras que el segundo trata de obtener unos valores que estén de cierta forma adaptados a los correspondientes datos.

- Perspectiva global

A raíz del hecho de que los ratings originalmente son valores ordinales en una escala, y que se asumieron tres posibles clases para los ratings, se propone que los valores para todos los umbrales κ y v (κ_u, v_u κ_i, v_i, κ, v) sean seleccionados de una forma que aproximadamente dividan el rango de ratings en un número de particiones igual a la cantidad de clases identificadas, siendo las particiones aproximadamente de igual tamaño. En este caso serían tres particiones.

Las ecuaciones (2. 1) y (2. 2) presentan la forma de calcular los valores para todos los parámetros κ y v, siguiendo este criterio. Se considera el uso de *round(n)* como función para determinar el entero más cercano a n, y en adición se hace uso de los valores extremos posibles de los ratings.

$$\kappa = \kappa_u = \kappa_i = \min R + round(\frac{1}{3} * (\max R - \min R)) \quad (2.\ 1)$$

$$\nu = \nu_u = \nu_i = \max R - round(\frac{1}{3} * (\max R - \min R)) \quad (2.\ 2)$$

Por otro lado, igualmente basándose en el hecho de que los valores de los ratings son representados en una escala, se sugiere asignar el valor del paso mínimo en esta escala al umbral de diferencia δ. Con esto, se asume que si la diferencia entre el rating nuevo y el antiguo excede la diferencia mínima entre dos valores consecutivos, entonces se considera lo

suficientemente grande como para realizar el reemplazo. De lo contrario, si esta diferencia no excede la diferencia mínima entre dos ratings consecutivos de la base de datos, esto se interpreta como que la discrepancia entre el rating real y el predicho usando la información del vecindario es menor que la discrepancia entre dos ratings diferentes de la base de datos original. En este caso se considera preferible mantener el valor original con vistas a evitar un comportamiento invasivo del método.

Esta alternativa será referenciada más adelante como el enfoque global-pv *(*global parameters' values*)* para la inicialización de los valores de los parámetros. El anexo 2.A muestra gráficamente un esbozo del método de tratamiento de ruido natural presentado en este capítulo, en el que el cálculo de los parámetros se hace siguiendo esta alternativa global.

-Adaptando los valores de los parámetros

En esta alternativa, los valores se adaptan acorde a la distribución de los ratings para los correspondientes usuarios e ítems. Específicamente, se sigue el principio de que es posible que muchos de los ratings para ciertos usuarios e ítems puedan estar localizados en una sección específica del rango de ratings, y que inclusive en este caso deberían estar asociados a diferentes clases. En adición, se considera el caso de que los valores de los parámetros se asocien a la distribución de los ratings de manera global, sin tomar en cuenta los ratings de cada usuario o ítem en particular. Así, se proponen tres variantes de inicialización: 1) basada en el usuario (user-based-pv), 2) basada en el ítem (item-based-pv), y 3) basada en los datos (rating-based-pv).

1) Inicialización basada en el usuario (user-based-pv): Para modelar esta alternativa se toma como referencia la media x' y la desviación p' de los ratings asociados a cada usuario u e ítem i, denominándose respectivamente como x_u', p_u', x_i', y p_i'. Estas magnitudes son utilizadas para definir los umbrales κ_u, v_u, κ_i, v_i tal y como se muestra en las ecuaciones (2. 3) - (2. **6**).

$$\kappa_u = x_u' - p_u' \tag{2. 3}$$

$$v_u = x_u' + p_u' \tag{2. 4}$$

$$\kappa_i = x_i' - p_i' \tag{2. 5}$$

$$v_i = x_i' + p_i' \tag{2. 6}$$

Para el cálculo de los valores de los parámetros κ y v (usados en las líneas 21-29 del algoritmo 2.1) se presenta la disyuntiva de hacerlo tomando como referencia a los usuarios (ecuaciones 2.3 y 2.4), o a los ítems (ecuaciones 2.5 y 2.6). En este caso, se toma como referencia a los usuarios, haciéndose ($\kappa = \kappa_u$ and $v = v_u$).

Finalmente, en relación con el umbral de diferencia (δ), se propone usar las mencionadas magnitudes estadísticas, considerándose $\delta = p_u'$.

El anexo 2.B muestra un esbozo del método de corrección siguiendo esta alternativa de inicialización. En esta figura, se muestra claramente cómo para la clasificación de los usuarios se usan los parámetros κ_u y v_u, para la de los ítems los parámetros κ_i y v_i, y para la de los ratings (tercer cuadro de la fase de clasificación en la figura), los parámetros κ_u y v_u.

2) Inicialización basada en el ítem (item-based-pv): En la inicialización basada en el ítem se sigue un procedimiento muy similar al anterior, con la diferencia de que en la clasificación de los ratings para la detección de las contradicciones (líneas 21-29 del algoritmo 2.1), los parámetros κ y v se inicializan tomándose como referencia a los ítems y no a los usuarios, haciéndose $\kappa = \kappa_i$ y $v = v_i$. De igual manera, el cálculo del umbral de diferencia δ se realiza basándose en la desviación del ítem y no del usuario correspondiente, asignándose $\delta = p_i'$. El anexo 2.C muestra un esbozo de la propuesta con esta alternativa de inicialización, pudiéndose apreciar que los parámetros para la clasificación de los usuarios y los ítems se inicializan de forma semejante a user-based-pv, siendo diferentes la inicialización para la clasificación de los ratings, la que se hace con los valores de κ_i y v_i.

3) Inicialización basada en los datos (rating-based-pv): La alternativa basada en los datos pretende inicializar los parámetros desde una perspectiva más global con respecto a user-based-pv e item-based-pv, pero menos global que global-pv. En este caso, se propone calcular los parámetros usando la media x_{data}' y la desviación p_{data}' de todas las preferencias disponibles, y no las del usuario o ítem correspondiente (ecuaciones 2.7, 2.8 y 2.9).

$$\kappa = \kappa_u = \kappa_i = x_{data}' - p_{data}' \quad (2.7)$$

$$v = v_u = v_i = x_{data}' + p_{data}' \quad (2.8)$$

$$\delta = p_{data}' \quad (2.9)$$

El anexo 2.D muestra un esbozo de esta alternativa, observándose cómo a diferencia de user-based-pv e item-based-pv, en este caso todos los parámetros son inicializados acorde al promedio y la desviación de todos los ratings de la base de datos.

Nota 2: Se desea destacar que en la primera parte del estudio experimental, se realiza una comparación (cuyos resultados se muestran posteriormente en las tablas 2.6 y 2.7) del desempeño del método propuesto siguiendo estas cuatro alternativas de inicialización de parámetros (global-pv, user-based-pv, item-based-pv y rating-based-pv). Las tres últimas columnas de estas dos tablas sorpresivamente muestran que las estrategias más sofisticadas (user-based-pv, item-based-pv y rating-based-pv) globalmente no son superiores a la más simple (global-pv). Por tanto, será global-pv la que se tome como referencia para el resto de las evaluaciones. A pesar de hacerse referencia nuevamente a estas dos tablas en la sección de experimentación, se considera importante resaltar este resultado en la presente sección de inicialización de los parámetros.

2.2.4 Ejemplo ilustrativo

En esta sección se muestra un escenario simplificado para exponer con claridad cómo la propuesta puede detectar y corregir ratings afectados por ruido natural en un sistema recomendador de filtrado colaborativo.

La tabla 2.3 muestra una matriz de ratings en el rango [1, 5], con cuatro usuarios y cuatro ítems. Sobre estos datos y con propósitos demostrativos, inicialmente se aplica el algoritmo 2.1 con la alternativa global-pv para la inicialización de los parámetros, siendo $\kappa=2$, $v=4$, y $\delta=1$. Así, se calculan los conjuntos W_u, A_u, S_u, W_i, A_i, y S_i, y se hace uso de sus cardinalidades para realizar la clasificación. La tabla 2.4 muestra los resultados de este paso, en el que se han obtenido tres usuarios benevolentes y uno variable; y tres ítems fuertemente preferidos y uno medianamente preferido.

Tabla 2.3. Escenario de uso para la propuesta

	i1	i2	i3	i4
u1	5	4	3	4
u2	5	4	1	4
u3	5		3	1
u4	2	5	5	5

El paso final en el algoritmo 2.1 es entonces aplicado para encontrar las contradicciones. Como resultado, la exploración descubre al rating r_{u4i1} como posible ruido, debido a que aunque el usuario asociado u4 es benevolente y el ítem i1 es fuertemente preferido, dicho rating no es fuerte.

Finalmente, el algoritmo 2.2 verifica la necesidad de corregir los ratings detectados como posible ruido. Inicialmente, este predice un nuevo valor de rating para el usuario u4 y el ítem i1, usando la base de datos completa con la técnica de filtrado colaborativo basado en vecindario anteriormente especificada (en este caso con k=2). Tras esto, el nuevo valor obtenido es r_{u4i1}=3,05, y dado el hecho de que la diferencia entre este y el valor antiguo es mayor que uno, entonces se lleva a cabo el reemplazo del rating en la matriz original.

Tabla 2.4. Clasificación de los usuarios y los ítems, considerando los ratings en la tabla 2.3.

Clasificación de los usuarios				
	$\|W_u\|$	$\|A_u\|$	$\|S_u\|$	Clase
u1	0	1	3	Benevolente
u2	1	0	3	Benevolente
u3	1	1	1	Variable
u4	0	1	3	Benevolente
Clasificación de los ítems				
	$\|W_i\|$	$\|A_i\|$	$\|S_i\|$	Clase
i1	0	1	3	Fuertemente preferido
i2	0	0	3	Fuertemente preferido
i3	1	2	1	Medianamente preferido
i4	1	0	3	Fuertemente preferido

Este escenario es un ejemplo ilustrativo de cómo el enfoque propuesto lleva a cabo la corrección de ratings para sistemas recomendadores de filtrado colaborativo. En la sección se 2.2.6 se muestra, a través de la experimentación, cómo estas transformaciones tienen un impacto positivo en el mejoramiento de la eficacia de la recomendación.

2.2.5 Análisis del costo computacional del método propuesto

Esta sección presenta el análisis del costo computacional del método propuesto. Existen dos estudios posibles a hacer para evaluar este costo: uno que proporciona una medida teórica (a priori), que consiste en obtener una función que acote el tiempo de ejecución del algoritmo; y otra que ofrece una medida real (a posteriori), consistente en medir el tiempo de ejecución del algoritmo en una computadora concreta (Aho, Hopcrof, & Ullman, 1988). En esta sección se

realiza el primero de estos estudios, mientras que el segundo se hará como parte de la sección 2.2.6.

La tabla 2.5 muestra tres etapas fundamentales en las que se divide la propuesta, para estudiar cada una por separado. La primera etapa se centra en encontrar la información necesaria para poder clasificar a los usuarios e ítems, que viene dada por hallar los conjuntos W_u, A_u y S_u para cada usuario, y W_i, A_i y S_i para cada ítem. Para ello, se necesita recorrer el conjunto completo de ratings. Asumiendo que la matriz de ratings es de u usuarios por i ítems, y que en cada iteración sólo se realizan operaciones elementales, el costo de esta etapa en el peor caso (que es aquel en el que la matriz estuviera completamente llena), tendría un orden de O(u*i). En función del número de ratings r, este costo sería de O(r). (En este análisis es válido comentar que a nivel de implementación las líneas 17-19 pueden ser omitidas, puesto que en la etapa siguiente la clasificación de los usuarios y los ítems vuelve a ser realizada. Al estar cada clasificación compuesta por operaciones elementales acorde a la tabla 2.1, su cota superior es O(1) y por tanto no hace falta precalcularla con anterioridad. Por esta razón estas líneas no se incluyeron en el análisis que se acaba de realizar. No obstante, su inclusión no alteraría la cota superior O(u*i) obtenida)

En la segunda etapa, se tiene que recorrer nuevamente la matriz de ratings, y para cada uno determinar si su clase entra en contradicción con la clase del usuario e ítem correspondiente. Siendo elementales las operaciones a realizar en cada ciclo, el costo de esta etapa también tendría una cota superior de O(u*i) o O(r), en función de los usuarios y los ítems, o los ratings.

Tabla 2.5. Análisis del costo computacional de la propuesta.

No.	Etapa		Costo en función del número de usuarios u e ítems i	Costo en función del número de ratings r
1	Detección de ratings posiblemente ruidosos	Clasificación de usuarios e ítems (líneas 03-19, Algoritmo 2.1)	O(u*i)	O(r)
2		Detección de contradicciones entre las clasificaciones de usuario, ítem y	O(u*i)	O(r)

		rating (líneas 20-30)		
3	Corrección de ratings posiblemente ruidosos (líneas 02-06, Algoritmo 2.2)		O((u*i/2)*costo de método para predecir nuevos ratings)	O((r/2)*costo de método para predecir nuevos ratings)

En la tercera etapa, se requiere realizar la predicción de un nuevo rating por cada rating detectado como posible ruido. Partiendo de la definición de las clases de usuario e ítem, para que un usuario sea clasificado en las clases crítico, promedio o benevolente, este debe tener al menos el 50% de los ratings pertenecientes a las clases baja, media o fuerte respectivamente. Al cumplirse, para todo rating posiblemente ruidoso, que su clase no es homóloga con la del usuario correspondiente, esto implica que un usuario siempre tendrá a lo sumo el 50% de sus ratings ruidosos, y que por tanto el ciclo de la etapa de corrección se ejecutará a lo sumo u*i/2 veces, o r/2 en función de los ratings.

En cada una de estas iteraciones se realiza la invocación, generalmente costosa, de un método de filtrado colaborativo tradicional. Partiendo del objetivo general de la presente tesis que es el de mejorar la eficacia de la recomendación, a la hora de elegir el método de predicción no se tuvo en cuenta como elemento prioritario el costo computacional de este. Así, se eligió un método de filtrado colaborativo basado en usuario, pionero pero efectivo, con un costo aproximadamente de O(u+i) acorde a Linden et al. (2003), que a su vez es considerado como un método de costo intermedio (Lee, Sun, & Lebanon, 2012a). Con este, el costo final de la tercera etapa sería O((u*i/2)*(u+i))), siendo este a su vez la cota superior de la propuesta completa, aplicando la regla de suma de cotas superiores (O(u*i)+ O(u*i)+ O((u*i/2)*(u+i)))= O((u*i/2)*(u+i)))). No obstante, en caso de necesitar dársele una mayor prioridad a la eficiencia del método, sería posible usar predictores de menor costo computacional, tales como los métodos slope one (Lemire & Maclachlan, 2005), centrados específicamente en predecir eficientemente los ratings desconocidos.

Más allá del resultado del presente análisis, en la sección 2.2.6 se evaluará el tiempo de ejecución de la propuesta en las diferentes bases de datos usadas en la experimentación, mostrándose que esta es capaz de procesar estos datos en un período de tiempo relativamente pequeño.

2.2.6 Experimentos y resultados

Esta sección presenta el impacto de la propuesta, aplicada como técnica de preprocesamiento, en la eficacia de la recomendación asociada con varios de los enfoques de filtrado colaborativo revisados en el capítulo 1. Adicionalmente, la evaluación incluye una comparación con trabajos semejantes también referidos en dicho capítulo. Finalmente, se examinan dos cuestiones críticas asociadas con este tipo de propuestas: el estudio del nivel de intrusión del método, y el estudio de su tiempo de ejecución.

-Bases de datos y protocolo experimental

El presente estudio se apoya en tres bases de datos bien conocidas en las investigaciones en sistemas recomendadores. La primera es Movielens, que contiene 100000 ratings de películas, provistos por 943 usuarios sobre 1682 ítems, donde cada rating yace en el intervalo [1, 5]. En segundo lugar se utilizará MovieTweeting (Dooms, De Pessemier, & Martens, 2013), compuesta por alrededor de 140000 ratings provistos por 21018 usuarios sobre 12569 ítems en noviembre del 2013, y en este caso los ratings están en el intervalo [0, 10]. En tercer lugar, Netflix Tiny, disponible en el sitio web de la plataforma PREA (Lee, Sun, & Lebanon, 2012b), con 4427 usuarios, 1000 ítems, y 56136 ratings en el intervalo [1,5].

Con vistas a preparar los datos para los experimentos, se seleccionan 900 usuarios en el caso de Movielens, y todos los usuarios con más de 20 ratings en el caso de MovieTweeting, y más de 10 en el caso de Netflix. Estos usuarios se usan para construir los conjuntos de entrenamiento y prueba, tal y como se indicó en la sección 1.3. En esta preparación, es válido señalar que en el caso de MovieTweeting y Netflix se hizo la selección de manera que se garantizara la obtención de conjuntos de datos que no tuvieran una dispersión extrema, y que por tanto pudieran utilizarse de forma eficaz con enfoques tradicionales de recomendación como los revisados en el capítulo 1. Para estos casos extremos existen enfoques específicos (Rodríguez, Espinilla, Sánchez, & Martínez-López, 2010; Son, 2015), que se van más allá del alcance de esta tesis. Esta forma de seleccionar el subconjunto de los datos a usar en la experimentación ha sido ampliamente usada en la literatura de sistemas recomendadores, con vistas a lograr el mencionado propósito (Bhagat, Weinsberg, Ioannidis, & Taft, 2014; Bilge & Polat, 2012; Choi & Suh, 2013; Di Noia, Ostuni, Rosati, Tomeo, & Di Sciascio, 2014; Lin, et al., 2002; Liu, Chen, Xiong, Ding, & Chen, 2012).

Específicamente, para la evaluación del efecto del proceso de corrección se compara el desempeño (en la predicción de los ratings del conjunto de prueba) del algoritmo de filtrado colaborativo correspondiente usando el conjunto de entrenamiento transformado, con respecto al desempeño asociado al conjunto de entrenamiento original. La figura 2.2 detalla este protocolo de experimentación, mostrándose cómo al conjunto de entrenamiento se le aplica el preprocesamiento (considerando la variante de no realizar ningún tipo de preprocesamiento), para obtener el conjunto de entrenamiento transformado. Este nuevo conjunto se utiliza para entrenar cinco algoritmos de filtrado colaborativo clásicos analizados en las secciones 1.2.1 y 1.2.2 (vecinos más cercanos basado en usuarios, vecinos más cercanos basados en ítems, slope one pesado, bipolar slope one, y basado en factorización de matrices), cuyos desempeños permitirán evaluar la efectividad del preprocesamiento propuesto.

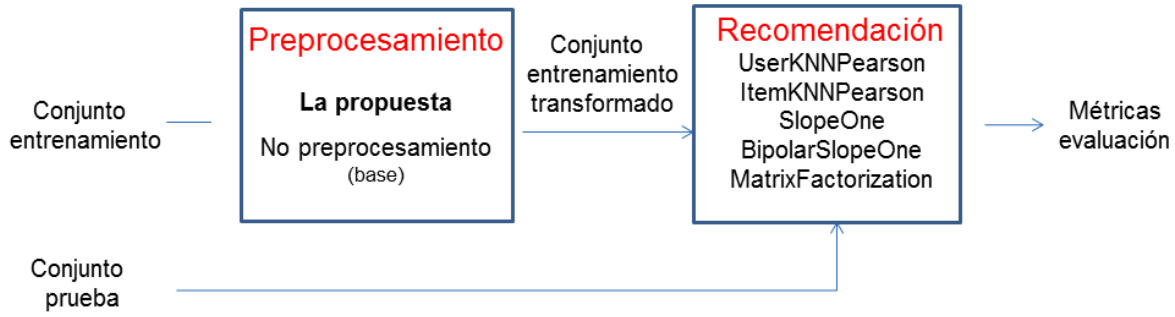


Figura 2.2. Protocolo de experimentación

Este desempeño se calcula a través de las métricas MAE y F1. Específicamente, los valores de F1 se obtienen considerando como relevantes a aquellas preferencias que pertenecen al cuarto superior de la escala de ratings, siendo esta la proporción que frecuentemente se tiende a utilizar para discriminar en los ratings entre preferencia y no preferencia (Bobadilla, Ortega, Hernando, & Bernal, 2012; Huete, Fernández-Luna, de Campos, & Rueda-Morales, 2012). Por tanto, se usa un umbral de relevancia = 4 para Movielens y Netflix (en una escala [1, 5]), y un umbral de relevancia = 7.5 para MovieTweeting (en una escala [0, 10]).

Específicamente, el protocolo descrito se ejecuta sobre 10 pares de conjuntos de entrenamiento-prueba, promediándose finalmente los valores de desempeño obtenidos. Es importante resaltar que en la literatura no se identificó un consenso con respecto al número de pares a utilizar (Gunawardana & Shani, 2009; Shani & Gunawardana, 2013).

-Configuración de los parámetros

Con vistas a medir el comportamiento de los métodos de filtrado colaborativo revisados en el capítulo 1, con y sin la corrección de ratings provista por la presente propuesta, se utilizan las implementaciones básicas de filtrado colaborativo proporcionadas por la plataforma MyMediaLite v1.0 (Gantner, Rendle, Freudenthaler, & Schmidt-Thieme, 2011). Acorde a Gantner et al. (2011), esta plataforma implementa los cinco métodos detallados en el capítulo 1. En el caso del método basado en reducción de dimensión, el incorporado a MyMediaLite permite la especificación de dos parámetros de regularización (λ) para los vectores y para los sesgos.

Así, el presente escenario experimental cuenta con dos grupos de parámetros: 1) aquellos asociados con los métodos tradicionales de filtrado colaborativo, y 2) aquellos asociados a la propuesta en sí. Los criterios para inicializar estos parámetros serán detallados a continuación:

1) En el primer caso, y asumiendo que la meta es determinar el efecto de la corrección en la eficacia de la recomendación, globalmente se fijan algunos valores en el método basado en factorización matricial (reducción de dimensión), basándose en los valores por defecto propuestos en MyMediaLite v1.0 (Gantner, et al., 2011). Estos parámetros son el valor de regularización para los vectores (λ_1=0.015), el valor de regularización de los sesgos (λ_2=0.015), y el factor de aprendizaje (α=0.01). Se considera que la asignación de diferentes valores a estos parámetros no afecta directamente la evaluación de los métodos de corrección.

   Para el caso de otros parámetros como el número de factores asociados al modelo (num_factors) y el número de iteraciones usado para obtener este (num_iters), sus mejores valores fueron predefinidos empíricamente para cada base de datos con vistas a garantizar el desempeño óptimo del método. Como resultado, en los tres casos se decidió utilizar num_factors=5 y num_iters=10.

   En relación con los métodos de filtrado colaborativo basados en vecindario, siempre se seleccionó k=60 como la cantidad de vecinos más cercanos a utilizar. Este valor tiende a ser usado para evaluar este tipo de métodos en diversos escenarios (Baltrunas & Ricci, 2008; Gantner, et al., 2011), aunque Cacheda et al. (2011) mostraron que para un dominio y datos similares a los usados en la presente tesis, los valores de k en el rango [50, 100] dan lugar a comportamientos similares.

2) Con respecto a los parámetros para la propuesta en sí, todas las inicializaciones fueron hechas según lo detallado en la sección 2.2.3. Para el enfoque global-pv, en Movielens y Netflix se asigna el valor κ=2 para el umbral débil-medio, y v=4 para el umbral medio-fuerte; y para MovieTweeting se asignan los valores κ=3 y v=7. En el caso de los umbrales de diferencia entre el rating original y el predicho, en los tres escenarios se usa la diferencia mínima entre los ratings originales en sus escalas, asignándose δ=1. En consonancia con el párrafo anterior, en el método de predicción de la etapa de corrección se usa k=60.

   Finalmente, los enfoques user-based-pv, item-based-pv, y rating-based-pv inicializan sus parámetros de forma personalizada, también según lo especificado en la referida sección.

-Selección de trabajos semejantes para comparar

En el capítulo 1 se presentaron algunos trabajos previos enfocados en manejar el ruido natural en escenarios de filtrado colaborativo. Con vistas a realizar una comparación con la presente propuesta, se seleccionan los métodos presentados por O'Mahony et al. (2006) y Li et al. (2013), considerando que el resto depende de información adicional que actualmente no se encuentra disponible como parte de las bases de datos empleadas en la experimentación. Adicionalmente, se propone también realizar una comparación con tres métodos que se enfocan en la eliminación de ruido malicioso (Burke, et al., 2006a; Chirita, et al., 2005), para verificar si estos también podrían ser competitivos mejorando la eficacia de la recomendación en un escenario donde se tenga en cuenta la posible presencia de ruido natural. Trabajos anteriores como Li et al. (2013) han realizado la comparación con estos últimos métodos, siguiendo el mismo propósito.

Con vistas a llevar a cabo la comparación, los cinco métodos se clasifican en dos nuevos grupos considerando la forma en la que el ruido se elimina de la base de datos:

i) O'Mahony et al. (2006) se enfoca en la detección y eliminación de ruido natural. Este método se basa en la eliminación de ratings ruidosos en el conjunto de entrenamiento, y por tanto se evalúa de una forma similar al método presentado.
ii) Los restantes cuatro métodos, propuestos por Li et al. (2013) (NNMU), Chirita et al. (2005) y Burke et al. (2006a) (RDMA, WDMA y WDA respectivamente) se basan en la eliminación de perfiles completos de usuarios, lo que realizan tras la obtención de un listado de usuarios ordenado por su nivel de ruido. Específicamente, estos enfoques seleccionan los z usuarios más ruidosos, eliminan sus ratings del conjunto de

entrenamiento, y evalúan la eficacia a través de la predicción del conjunto de prueba original tomando como base el resto del conjunto de entrenamiento. Por consiguiente, las predicciones para los ratings del conjunto de prueba que están asociados a usuarios eliminados del conjunto de entrenamiento, se hacen asignando el sesgo global para los ratings, referido por Mazurowski (2013) citando a Koren et al. (2009).

-Resultado de la evaluación

La presente sección muestra los resultados de la aplicación de la propuesta en los cinco escenarios de filtrado colaborativo presentados en el capítulo 1, así como una comparación con los trabajos semejantes recientemente mencionados.

1) El efecto de la propuesta en los métodos de filtrado colaborativo

Las tablas 2.6 y 2.7 muestran el desempeño de la propuesta acorde a las alternativas para inicializar los parámetros (global-pv, user-based-pv, item-based-pv y rating-based-pv). Para todos los casos excepto uno, al menos una de estas alternativas mejora la eficacia del algoritmo base. Se considera importante resaltar que inclusive en el caso del método basado en reducción de dimensión (que implícitamente ya se centra en eliminar algunas inconsistencias en los datos (Ekstrand, et al., 2010)), la propuesta introduce una mejora en el desempeño.

Tabla 2.6. Desempeño de la propuesta con respecto al algoritmo base acorde a la métrica MAE.

Filtrado colaborativo usuario-usuario					
	Baseline	global-pv	user-based-pv	item-based-pv	rating-based-pv
MAE Movielens	0,7647	0,7632	0,7633	**0,7628**	0.7657
MAE MovieTweeting	1,2117	1,1971	1,1942	**1,1939**	1.1990
MAE Netflix	0,7811	0,7768	0,7757	0,7766	**0.7748**
Filtrado colaborativo ítem-ítem					
	Baseline	global-pv	user-based-pv	item-based-pv	rating-based-pv
MAE Movielens	0,7706	**0,7674**	0,7685	0,7680	0.7720
MAE MovieTweeting	1,2032	**1,1834**	1,1880	1,1864	1.1888
MAE Netflix	0,7918	0,7874	**0,7866**	0,7879	0.7869
Filtrado colaborativo slope one					

	Baseline	global-pv	user-based-pv	item-based-pv	rating-based-pv
MAE Movielens	0,7771	0,7755	0,7737	**0,7732**	0.7763
MAE MovieTweeting	1,3372	1,3027	1,3003	**1,2959**	1.3009
MAE Netflix	0,8085	0,8048	0,8048	0,8058	**0.8035**
Filtrado colaborativo bipolar slope one					
	Baseline	global-pv	user-based-pv	item-based-pv	rating-based-pv
MAE Movielens	0,7837	0,7839	0,7826	**0,7822**	0.7842
MAE MovieTweeting	1,3058	1,2982	1,2923	**1,2892**	1.2920
MAE Netflix	0,7985	0,7967	0,7970	0,7982	**0.7957**
Filtrado colaborativo basado en factorización de matrices					
	Baseline	global-pv	user-based-pv	item-based-pv	rating-based-pv
MAE Movielens	0,7629	**0,7611**	0,7630	0,7630	0.7655
MAE MovieTweeting	1,1815	1,1697	1,1721	**1,1681**	1.1716
MAE Netflix	0,7656	**0,7629**	0,7645	0,7648	0.7655

Tabla 2.7. Desempeño de la propuesta con respecto al algoritmo base acorde a la métrica F1.

Filtrado colaborativo usuario-usuario					
	Baseline	global-pv	user-based-pv	item-based-pv	rating-based-pv
F1 Movielens	0,4601	**0,5098**	0,4532	0,4567	0.4551
F1 MovieTweeting	0,5103	**0,5367**	0,5082	0,5119	0.5091
F1 Netflix	0,3684	**0,3915**	0,3659	0,3648	0.3637
Filtrado colaborativo ítem-ítem					
	Baseline	global-pv	user-based-pv	item-based-pv	rating-based-pv
F1 Movielens	0,4630	**0,5039**	0,4596	0,4645	0.4625
F1 MovieTweeting	0,4985	**0,5219**	0,4964	0,4978	0.4957
F1 Netflix	0,4078	**0,4194**	0,4060	0,4045	0.3637

Filtrado colaborativo slope one					
	Baseline	global-pv	user-based-pv	item-based-pv	rating-based-pv
F1 Movielens	0,4477	**0,4904**	0,4491	0,4528	0.4504
F1 MovieTweeting	0,4895	**0,5046**	0,4826	0,4861	0.4876
F1 Netflix	0,3911	**0,4074**	0,3917	0.3922	0.3910
Filtrado colaborativo bipolar slope one					
	Baseline	global-pv	user-based-pv	item-based-pv	rating-based-pv
F1 Movielens	0,5635	**0,5640**	0,5502	0,5561	0.5347
F1 MovieTweeting	0,4819	**0,4831**	0,4648	0,4702	0.4717
F1 Netflix	**0,4741**	0,4720	0,4624	0,4631	0.4549
Filtrado colaborativo basado en factorización de matrices					
	Baseline	global-pv	user-based-pv	item-based-pv	rating-based-pv
F1 Movielens	0,4370	**0,5014**	0,4390	0,4419	0.4448
F1 MovieTweeting	0,5206	**0,5370**	0,5134	0,5181	0.5187
F1 Netflix	0,2981	**0,3293**	0,3020	0,3002	0.3056

Es notorio resaltar que los resultados sugieren que el uso de un enfoque de corrección más sofisticado (como user-based-pv, item-based-pv o rating-based-pv), no necesariamente implica la obtención de un mejor desempeño. En el caso particular del MAE, los mejores resultados tienden a estar compartidos entre global-pv e item-based-pv, mientras que para el caso de F1, global-pv obtiene para todos los casos el mejor resultado.

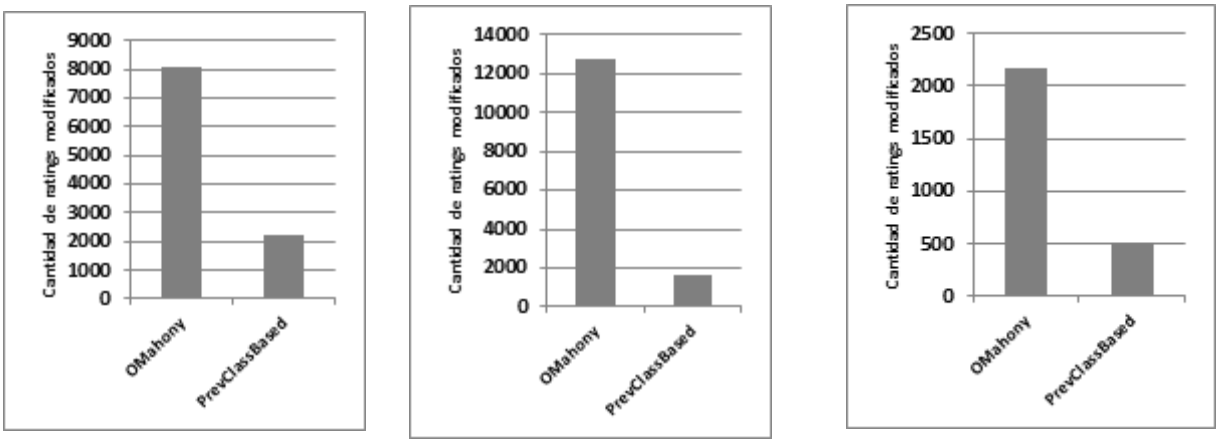
A raíz de estos resultados, global-pv será la opción tomada como referencia para el resto del presente estudio experimental, pasando a ser referenciada a partir de ahora como “PrevClassBased”. Es válido agregar que con respecto a esta opción, se realizaron evaluaciones adicionales con diferentes valores de κ y v, probándose que la inicialización propuesta es la óptima o está muy cercana a la óptima para la mayoría de los casos.

Tomando como base únicamente la alternativa global-pv, las tablas 1, 2 y 3 del anexo 3 presentan los valores promedio de la cantidad de usuarios e ítems en cada base de datos, por cada clase definida. Estos promedios se calculan a partir de los valores correspondientes a

cada una de las particiones generadas para la experimentación. En todos los casos, se observa una importante cantidad de usuarios y de ítems pertenecientes a las clases benevolente y fuertemente preferido, y en adición pocos usuarios con un comportamiento crítico. Además, los resultados también indican que el umbral δ juega un rol importante para determinar los ratings que finalmente serán modificados, considerando que varios ratings etiquetados como posible ruido finalmente no son corregidos.

2) Comparación con otros enfoques de tratamiento de ruido en filtrado colaborativo

Inicialmente se realizó la comparación entre PrevClassBased (global-pv) y el enfoque propuesto en (O'Mahony, et al., 2006). Para este último enfoque, el parámetro δ fue inicializado con el valor δ=1 en todas las bases de datos. En adición, como grupo de usuarios *genuinos* requeridos, se seleccionó el conjunto completo de usuarios. Como resultado, PrevClassBased supera en 22 casos de un total de 30, al enfoque de O'Mahony et al. De los restantes ocho casos en los que PrevClassBased tuvo un desempeño peor, seis estuvieron asociados a las bases de datos MovieTweeting o Netflix. Esto pudiera sugerir que para un conjunto de datos con un alto grado de dispersión, en algunos casos pudiera ser preferible utilizar un método basado en la eliminación de ratings ruidosos (como O'Mahony et al.) y no uno basado en la corrección como el presentado en esta tesis, debido a la dificultad de generar nuevas predicciones sobre datos con estas características. Las tablas 1-5 del anexo 4 presentan los resultados de esta comparación entre PrevClassBased (global-pv) y el enfoque propuesto en (O'Mahony, et al., 2006).



(a) (b) (c)

Figura 2.3 Cantidad de ratings transformados en cada caso. (a) Movielens. (b) MovieTweeting. (c) Netflix

Por otra parte, la figura 2.3 muestra la comparación entre la cantidad de usuarios que cada uno de estos dos métodos requirió modificar, y resulta claro que PrevClassBased necesita hacer un número mucho menor de manipulaciones para mejorar la eficacia de la recomendación.

En un segundo momento, se realiza la comparación entre PrevClassBased y los métodos basados en la eliminación de perfiles completos de usuarios. En este caso, cada método fue evaluado con el parámetro z, que indica el número de perfiles a eliminar, en el rango [10,60] con paso 10, registrándose el mejor resultado que es el empleado para la comparación con PrevClassBased.

Las figuras 1-3 del anexo 5 muestran los resultados de este segundo momento de evaluación, donde cada método se representa en las barras con el formato *método-z* (RDMA-60, WDA-10, por citar algunos ejemplos), donde z es el valor asociado al mejor desempeño en cada caso. Los resultados indican que los métodos enfocados en ruido malicioso (RDMA, WDMA, WDA) tienen un comportamiento incluso peor que el método usado como base para la comparación en la mayoría de los casos, y que el desempeño de estos tiende a empeorar para mayores valores de z (mayor pérdida de información). Específicamente, son capaces de superar el método base para la métrica MAE con el método user-user en la base de datos MovieTweeting, y para la métrica F1 en Movielens con el método user-user, y en Netflix con ítem-ítem, slope one, y con el método basado en factorización de matrices. Por otro lado, únicamente superan a la presente propuesta para el caso de la métrica F1 con el predictor bipolar slope one en la base de datos Netflix.

Por otro lado, si bien NNMU supera el desempeño del método base en un número mayor de ocasiones, únicamente supera el método propuesto para el caso del predictor bipolar slope one, con la métrica F1, en las bases de datos MovieTweeting y Netflix. En el resto de los casos, PrevClassBased también supera notablemente este enfoque.

Así, como conclusiones de los resultados que se presentan en las figuras 1-3 del anexo 5, se destaca que 1) el mejor desempeño de los cuatro métodos contra los que se comparó (RDMA, WDMA, RDA y NNMU) fue obtenido con la métrica F1, aunque este siguió siendo inferior al de PrevClassBased para la mayoría de los casos, 2) la base de datos sobre la que PrevClassBased tuvo peor desempeño fue Netflix, resultado semejante al de la comparación realizada contra O'Mahony et al. y discutida previamente en esta misma sección, y 3) no se detectó que los datos preprocesados con la presente propuesta tuvieran algún efecto negativo,

en específico, sobre alguno de los cinco métodos tradicionales de filtrado colaborativo usados para generar las recomendaciones.

Por tanto, los resultados presentados prueban que el método propuesto tiene un desempeño superior con respecto a otros presentes en el estado del arte. Sin embargo, con vistas a tener una visión más clara de esta superioridad, se decide aplicar una prueba estadística sobre los datos presentados. Específicamente, en este escenario se decide aplicar un procedimiento popular propuesto en (García, Fernández, Luengo, & Herrera, 2010) para realizar comparaciones múltiples entre algoritmos en problemas de inteligencia computacional.

Con este fin, inicialmente se hace uso del *test* de *Friedman* (Friedman, 1937) para determinar si la diferencia entre los rankings de cada algoritmo a comparar en cada uno de los diferentes escenarios es significativa. En este *test*, el rechazo de la hipótesis nula indica que existe una diferencia significativa con respecto a los algoritmos en cuestión. En tal caso, al rechazarse, se aplica entonces el *test* de *Holm* (Holm, 1979), utilizando como algoritmo de control aquel de mejor desempeño acorde a Friedman. El *test* de *Holm* compara entonces este algoritmo con cada uno de los restantes, verificando en cada caso si las diferencias son significativas.

El anexo 6 muestra los resultados de la aplicación de estas pruebas tomando como base los resultados recién presentados. Específicamente, se consideró la comparación de seis algoritmos de preprocesamiento de datos en sistemas recomendadores (RDMA, WDMA, WDA, Li et al. (NNMU), O'Mahony et al., y el método propuesto), más el caso base que no considera preprocesamiento alguno. Tal y como se mostró a la largo de esta sección, estas siete alternativas de preprocesamiento fueron evaluadas sobre 15 escenarios diferentes (tres bases de datos X cinco métodos de recomendación). Finalmente, las comparaciones fueron hechas utilizando tanto la métrica MAE como la F1.

Las tablas 1 y 4 del anexo 6 presentan los resultados del test de Friedman, mostrando que para ambas métricas el ranking promedio del método propuesto en el presente capítulo supera ampliamente al resto de los métodos.

Las tablas 2 y 5, por su parte, muestran los resultados del procedimiento de Holm para $\alpha=0.05$, considerando la propuesta como método de control. La comparación entre los valores de p y de Holm en cada caso (las dos últimas columnas), muestran que p siempre yace por debajo del valor de Holm, rechazándose la hipótesis de igualdad y por tanto concluyendo que la superioridad del método propuesto es estadísticamente significativa. La única excepción a esto

aparece en la comparación entre la propuesta y el método de O'Mahony et al. para la métrica F1. En este caso la hipótesis nula se acepta.

Finalmente, las tablas 3 y 6 muestran los resultados de este mismo test pero considerando α=0.10. En este caso la hipótesis nula siempre es rechazada, resultando el método propuesto estadísticamente superior a las restantes alternativas para todas las comparaciones.

-Sobre el grado de intrusión del método, y su tiempo de ejecución.

Eventualmente, esta sección discute dos elementos importantes con vistas a determinar la viabilidad de la aplicación de PrevClassBased en escenarios reales de recomendación. El primero hace referencia a cuán intrusivo resulta el método en relación con la modificación de los perfiles de usuario y de ítem, mientras que el segundo está relacionado con su tiempo de ejecución.

-Nivel de intrusión

Con vistas a evaluar el nivel de intrusión de la propuesta, se ha computado la cantidad de ratings finalmente corregidos por usuario y por ítem para cada caso, resultando las cantidades notablemente bajas. Específicamente para Movielens, el 85,33% de los usuarios y el 87,15% de los ítems requirieron la modificación de menos del 10% de sus ratings. Por otro lado, para MovieTweeting se calcularon los mismos porcentajes y los resultados fueron el 80,30% para los usuarios y el 93,15% para los ítems, mientras que para Netflix fueron respectivamente de 87,71% y de 96,26%. Estos valores permiten concluir que la propuesta solamente necesita modificar una pequeña porción de los perfiles de los usuarios y los ítems, constituyendo esto una característica deseable con vistas a evitar un comportamiento altamente intrusivo.

-Tiempo de ejecución

En relación con el tiempo de ejecución, se considera que la propuesta puede implementarse en dos escenarios diferentes:

1. *Como paso de preprocesamiento ejecutado sobre la base de datos del sistema recomendador para transformar los ratings ruidosos:* en este escenario el costo computacional no afecta la generación de recomendaciones, puesto que el paso de preprocesamiento puede hacerse en un segundo plano (offline)

2. *Como paso previo antes de cada predicción, en donde la transformación de los ratings sería temporalmente hecha sólo para la predicción actual, y por tanto los datos originales no serían modificados.* La implementación en este escenario podría verse afectada por el costo computacional de la propuesta. Tal y como se mostró previamente en esta misma sección 2.2.6, la etapa de corrección de los ratings detectados como posible ruido es la más costosa, puesto que demanda realizar la predicción de un nuevo rating por cada posible ruido encontrado. El método de predicción utilizado (filtrado colaborativo usuario-usuario) requiere para cada predicción, del cálculo de similitudes entre el usuario actual y aquellos usuarios que también evaluaron el ítem cuyo rating tuvo que ser modificado, siendo este cálculo específicamente la tarea costosa que pudiera hacer lenta la ejecución del método.

A raíz de lo anterior, se hace necesario evaluar el tiempo de ejecución real del método propuesto. Esta evaluación podría ofrecer una conclusión inicial sobre si al menos con un volumen de datos similar al de las bases de datos empleadas, se podría implementar la propuesta en los dos mencionados escenarios, o sólo en el primero.

Así, se computó el tiempo de ejecución real de cada paso del método propuesto (Anexo 7), y adicionalmente se le comparó con el método propuesto por O'Mahony et al. (2006). Con este fin, se usó una plataforma de hardware compuesta por un procesador Intel Pentium (R) Dual Core 3.06 GHz. No se realizó la comparación con Li et al. (2013) debido a que este método toma, con notoriedad, un mayor tiempo de ejecución. El análisis de los resultados en las referidas tablas muestra que la ejecución del paso de detección de ratings posiblemente ruidosos es instantánea. A su vez, reafirma que dicho paso posibilita la ejecución más rápida del paso de corrección, debido a la pequeña cantidad de ratings que finalmente se requiere corregir (ver figura 2.3). Por otro lado, tal y como se había planteado, el cálculo de las similitudes usuario-usuario resultó ser la tarea más costosa. No obstante, se desea resaltar que en el presente trabajo se empleó una implementación básica de este procedimiento, cuya única particularidad es la de usar estructuras de datos complementarias para acelerar el recorrido de la matriz de usuarios e ítems. Más allá del escenario experimental, varios autores han expuesto diferentes alternativas para acelerar dicho cálculo, con vistas a reducir su tiempo de ejecución (Schelter, Boden, & Markl, 2012; Zhou & Zhang, 2015).

Resumiendo, a pesar de que el resultado obtenido en la sección dedicada al análisis teórico del costo computacional de la propuesta indicó que el tiempo de ejecución real del método iba a aumentar con el incremento del número de usuarios e ítems, los resultados de la presente

sección sugieren que esto en principio no constituye un problema para volúmenes de datos similares a los empleados. Se desea recalcar además que este volumen coincide con el usado en este tipo de evaluaciones a escala experimental (Alexandridis, Siolas, & Stafylopatis, 2012; Cacheda, et al., 2011; Lin, et al., 2002; Wang, Lee, Kim, Youngjoon, & Lee, 2015).

Con esto, se ratifica el hecho de que el paso de detección de ratings posiblemente ruidosos facilita el uso de la propuesta también en el segundo de los dos escenarios prácticos antes mencionados (el más crítico de los dos). En contraste, el trabajo realizado por O'Mahony et al. (2006) consume un tiempo muy valioso en este escenario, tal y como se muestra en el Anexo 7, al corregir una mayor cantidad de ratings.

-Análisis complementario de los resultados

Esta sección hace una discusión complementaria sobre los resultados experimentales. Primeramente, se centra en resumir la sección de comparación y justificar algunos resultados inesperados, específicamente relacionados con los valores de F1. En segundo lugar, se dan razones para, más allá de los meros valores de desempeño, justificar la viabilidad de la corrección de ratings ruidosos usando únicamente ratings.

-Sobre algunos resultados inesperados

Los resultados obtenidos en la sección anterior mostraron que, de forma general, los métodos para manipular el ruido malicioso tuvieron un desempeño peor que el algoritmo base. Por el contrario, los métodos centrados en la manipulación de ruido natural ligeramente lo mejoran, siendo la excepción el método propuesto que lo mejora con notoriedad.

Como resultado inesperado, resalta para una buena cantidad de casos la importante mejora relacionada con los valores obtenidos por la propuesta para F1, en relación con aquellos obtenidos por los métodos basados en la eliminación de perfiles completos de usuarios. Con vistas a encontrar una explicación para esto, se realizó un análisis suplementario que agrupó, para cada base de datos, los ratings transformados acorde a su valor original y a su valor final después del proceso de corrección (redondeado al entero más cercano). Los resultados de este análisis se presentan en el anexo 8, e indican que la mayoría de las correcciones tienden a incrementar los valores iniciales de los ratings. Partiendo de que la transformación de algunos ratings hacia valores más altos debe implicar también que las predicciones que toman a estos como base tiendan a ser más elevadas; este hecho combinado con la conclusión

empíricamente obtenida de que los usuarios tienden a dar votaciones altas (Amatriain, Pujol, & Oliver, 2009) (y que por tanto el conjunto de prueba, compuesto por ratings, satisface esta conclusión), de cierta forma justifica la notable mejora en el F1, tomando en consideración la definición de esta métrica.

Adicionalmente, también se desea resaltar que el anexo 8 muestra que los ratings ubicados cerca del centro de la escala de ratings fueron más propensos a ser corregidos, en relación con los ratings extremos. Esta conclusión fue también obtenida por Amatriain, Pujol y Oliver (2009), mostrándose así que el presente trabajo está alineado con investigaciones semejantes orientadas en la misma dirección.

-Sobre la corrección de ratings sin información adicional

La presente propuesta supera a los trabajos previos en el sentido de que realiza el proceso de corrección apoyándose en conocimiento global extraído directamente de los ratings, en vez de depender de información adicional como los atributos de los usuarios y los ítems. En este sentido, el trabajo se desarrolla bajo el principio de que de estos datos se puede extraer información tan o más valiosa que la mencionada información suplementaria. En esta sección se presentan tres razones adicionales para justificar la viabilidad de la corrección de las inconsistencias usando sólo los ratings, sugiriéndose que la información adicional disponible no siempre es una fuente confiable para ser usada de manera sistemática en esta tarea.

1. Park, Pennock, Madani, Good, & DeCoste (2006) han apuntado que considerando la tarea de generar perfiles de usuario artificiales (filterbots) para aliviar el problema del arranque en frío, los perfiles generados a través del uso de información fundamentalmente basada en ratings tienden a obtener resultados mejores con respecto a los obtenidos por perfiles creados usando enfoques más dependientes de los atributos. Así, concluyen que estos últimos perfiles son más propensos a provocar un sobreajuste de los datos.
2. En segundo lugar, se realizó un análisis de los datos para determinar el impacto de la información adicional en la distribución de ratings, y en consecuencia su posible impacto en el procesamiento de ruido natural. Específicamente, se intentó determinar si la presencia de algunos atributos podría influir en el hecho de que los usuarios amaran u odiaran el correspondiente ítem. Con este fin, se utilizó la información relacionada al género de las películas (también disponible como parte de la base de datos Movielens). Para cada género, se calculó la razón entre la cantidad de filmes preferidos (valores de rating de cuatro o

cinco), y de no preferidos, considerando cada usuario de forma independiente. Los resultados obtenidos mostraron una razón promedio mucho más baja que la esperada en principio, donde los géneros posiblemente más populares (Acción, Ciencia Ficción, Comedia, Aventura) estuvieron entre las categorías de menor valor (siempre por debajo de tres). Este resultado indica que las preferencias y no preferencias asociadas a filmes del mismo género tienden a estar balanceadas. Así, sugiere que el grado de preferencia sobre cierto género (como típica información adicional), no constituye un atributo apropiado para caracterizar a los usuarios, y por tanto no representa información confiable para ser utilizada en tareas como la corrección de ruido natural. Se considera que este hecho es también aplicable a otros atributos distintos del género. En un estudio reciente, Zhao, Niu, y Chen (2013) han obtenido un resultado semejante a este, con respecto a los valores de los ratings asociados a ítems con cierto atributo en común.

3. Más allá del punto anterior, se piensa que existen determinados contextos "ocultos" que podrían representar la clave para el tratamiento de ruido natural; sin embargo, se considera que su descubrimiento es una tarea compleja. No obstante, es posible afirmar que el enfoque de corrección presentado implícitamente toma en cuenta dichos contextos. En este caso, el método de filtrado colaborativo empleado para el cálculo de los nuevos valores de rating, solamente considera información de los usuarios ubicados en el vecindario correspondiente. Estos usuarios tendrían preferencias similares a las del usuario actual, y por esta razón se considera que representan la mejor fuente de información para finalmente determinar si un rating posiblemente ruidoso lo es finalmente, más allá que cualquier otra dimensión adicional que pueda ser usada con este propósito. Se piensa que los resultados presentados, indicando que una dimensión adicional como el género de los filmes preferidos tiene un bajo poder discriminante para identificar los usuarios, refuerza este hecho.

Considerando el escenario de uso presentado, en este caso la corrección del rating para el usuario u4 estuvo influenciada por un vecindario compuesto por usuarios que realmente no son similares al usuario actual. Sin embargo, se desea destacar que este ejemplo es sólo un escenario simplificado explícitamente concebido para mostrar la aplicación del método. Se considera que para un escenario real de mayores dimensiones, cada usuario con frecuencia estaría altamente correlacionado con su vecindario, evitándose así la corrección basada en perfiles muy diferentes, que pertenecerían a diferentes categorías en un posible contexto oculto.

Conclusiones del capítulo

En el capítulo se ha propuesto un método de preprocesamiento de datos para sistemas recomendadores de filtrado colaborativo. Específicamente, este se centra en el tratamiento de ruido natural en los ratings, basándose en el descubrimiento de contradicciones entre los correspondientes perfiles de usuario y de ítem, y el propio valor del rating. Se arribó a las siguientes conclusiones:

-El uso de un enfoque basado en contradicciones entre usuarios, ítems y ratings en un sistema recomendador de filtrado colaborativo, permite la detección y corrección de ruido natural sin depender de información adicional.

-El procesamiento de ruido natural en los datos del sistema recomendador de filtrado colaborativo garantiza una mayor eficacia en la generación de recomendaciones obtenidas a través de métodos tradicionales de recomendación, tanto por parte de los basados en vecindario, como por parte de los basados en modelos.

-La propuesta de este trabajo presenta un desempeño superior con respecto a otros métodos centrados en el procesamiento de ruido natural, superando a su vez a métodos que se enfocan en el ruido malicioso.

-El grado de intrusión de la propuesta es mínimo, necesitando transformar sólo una pequeña parte de los perfiles de los usuarios como parte del proceso de corrección.

-A pesar del que el cálculo del costo computacional teórico mostró que el tiempo de ejecución debe incrementarse con el aumento del volumen de los datos, los resultados experimentales en la práctica también mostraron que la propuesta es factible de utilizar en escenarios con un volumen de datos similar al usado en la experimentación, al tener un tiempo de ejecución inferior al de los restantes trabajos enfocados en tratamiento de ruido natural.

3. UN RECOMENDADOR PARA JUECES EN LÍNEA DE PROGRAMACIÓN, BASADO EN FILTRADO COLABORATIVO Y TÉCNICAS DE PREPROCESAMIENTO

Introducción

El presente capítulo se centra en la construcción de un sistema recomendador para jueces en línea de programación, basado en filtrado colaborativo y técnicas de preprocesamiento de ruido natural. La sección 3.1 presenta una introducción a los jueces en línea de programación, y hace referencia a la sobrecarga de información asociada a estos sistemas. La sección 3.2 presenta una visión global de los jueces en línea a través de la descripción de sus funcionalidades principales, y de un análisis de los trabajos asociados a estos. A su vez, la sección 3.3 presenta una breve revisión sobre sistemas recomendadores en el ámbito de la enseñanza de la programación. En la parte principal del capítulo, las secciones 3.4, 3.5 y 3.7 presentan métodos basados en filtrado colaborativo para sugerir problemas a resolver en un juez en línea, empleando diferentes niveles de información. En adición, la sección 3.6 hace uso del esquema propuesto en el capítulo anterior para concebir un esquema de detección y corrección de ruido natural, con vista a mejorar el desempeño del sistema recomendador propuesto. La sección 3.8 presenta la evaluación experimental de estos métodos, y la sección 3.9 muestra la integración de las propuestas a través de la creación de un sistema recomendador de problemas acoplado a un juez en línea. Finalmente, se termina con las conclusiones parciales.

El grueso de los resultados obtenidos en este capítulo han sido publicados en (Yera Toledo & Caballero Mota, 2014).

3.1 Jueces en línea y sobrecarga de información

Los jueces en línea de programación son aplicaciones que contienen una gran cantidad de ejercicios de programación a resolver, y que se centran en automatizar el proceso de

compilación y evaluación de soluciones a estos ejercicios. En los últimos años se ha detectado un incremento en el desarrollo y empleo de estas herramientas (Fernández Alemán, 2011; Kurnia, et al., 2001; Llana, Martin-Martin, Pareja-Flores, & Velázquez-Iturbide, 2014; Petit, Giménez, & Roura, 2012; Revilla, Manzoor, & Liu, 2008), destacándose entre las más populares el Archivo en Tiempo Real de la ACM-ICPC (https://icpcarchive.ecs.baylor.edu/), el Juez en Línea de la Universidad de Valladolid, (http://uva.onlinejudge.org/), el Juez en Línea de la Universidad de Pekín (http://poj.org), el Juez en Línea Sphere (http://spoj.com), el Juez en Línea de la Universidad Estatal de Los Urales (http://acm.timus.ru/), y el Juez en Línea de la Universidad Tecnológica de Harbin (http://acm.hit.edu.cn/hoj/). En el área caribeña, en los últimos años ha venido emergiendo con mucha fuerza el Juez en Línea Caribeño (http://coj.uci.cu), empleado para dar soporte al aprendizaje basado en la programación competitiva (Revilla, et al., 2008; Ribeiro & Guerreiro, 2008).

Actualmente existe una tendencia global a extender el uso de estas aplicaciones a escenarios diferentes del competitivo, enfocándolo en la ejercitación de las habilidades básicas, y provocando así un incremento importante en su popularidad. Una muestra reciente de esto es el proyecto EduJudge (Verdú et al., 2012), cuyo propósito fundamental es el de integrar el Juez en Línea de la Universidad de Valladolid en un ambiente efectivo de e-learning para estudiantes de cursos de pregrado y postgrado en computación. En los últimos años, en la Universidad de las Ciencias Informáticas, en Cuba, también se han desarrollado varias experiencias exitosas enfocadas en la integración de los jueces en línea en este tipo de escenarios docentes tradicionales (Altuna Castillo & Guibert Estrada, 2013; Junco Vázquez, Altuna Castillo, & Soria Ramirez, 2009).

Al ser generalmente aplicaciones web, los jueces en línea también están afectados por la sobrecarga de información asociada al desarrollo de las tecnologías de la información y las comunicaciones. Por sólo citar un ejemplo, en estos momentos (enero 2015) el Juez en Línea Caribeño COJ cuenta con cerca de 20000 usuarios, más de 3000 problemas, y cerca de 750000 intentos de solución. Esta cantidad de información provoca que los usuarios en la mayoría de los casos no sean capaces de seleccionar el problema más apropiado a resolver.

Básicamente, los jueces en línea presentan un listado secuencial de problemas de programación que el usuario puede seleccionar e intentar resolver. Así, el usuario de mejor desempeño es aquel que ha resuelto una mayor cantidad de problemas, independientemente de la categoría y el nivel de complejidad de los mismos. Por tanto, en este escenario puntual, el

objetivo principal es resolver el mayor número posible de problemas y, de ser posible, hacerlo de forma tal que los problemas resueltos estén acorde a las capacidades y aptitudes del usuario en cuestión.

Tomando en cuenta lo anterior, se considera que la incorporación de un sistema recomendador en este espacio sería de gran valor con vistas a alcanzar el mencionado objetivo. En este contexto, un sistema recomendador de filtrado colaborativo, semejante a los presentados en los capítulos anteriores, se centraría en buscar aquellos usuarios con capacidades y aptitudes semejantes a las del usuario activo, y posteriormente recomendar a este aquellos problemas que ya han sido resueltos por los usuarios semejantes. Esta estrategia se presupone como acertada para ayudar al usuario a alcanzar su meta principal en este tipo de sistemas, y a la vez enfrentar la mencionada sobrecarga de información.

En el presente capítulo se desarrolla un sistema recomendador para jueces en línea de programación, centrado en los objetivos antes mencionados. La propuesta está orientada en dos direcciones fundamentales: un método basado en filtrado colaborativo que usa los datos específicos de este escenario; y un método de preprocesamiento de los datos que toma como referencia las ideas previamente presentadas, y cuya eficacia fue verificada en un escenario clásico de recomendación en el capítulo anterior.

3.2 Visión global de los jueces en línea

Los jueces en línea de programación usualmente almacenan un listado de problemas a resolver. Una vez que el usuario accede a la aplicación, realiza la selección de un ejercicio a resolver, y escribe un programa en forma de solución para este. El sistema le proporciona una pantalla para enviar la solución, la que una vez sometida, es compilada y evaluada a través de un conjunto de datos predefinidos con este fin. Si la salida proporcionada por la solución coincide con la esperada, el problema se considera resuelto, o *aceptado* en la terminología usada por el juez en línea.

En caso contrario, el problema se considera como no resuelto. En este caso, el fallo se clasifica de acuerdo a la causa de la no solución. Las categorías principales en las que este puede ser catalogado, acorde a Skiena y Revilla (2006), son:

- *Respuesta incorrecta:* Cuando el programa retorna una respuesta incorrecta ante alguno de los juegos de datos secretos del problema.

- *Error de compilación:* Cuando el compilador no logra compilar el código sometido.
- *Error en tiempo de ejecución:* Cuando el programa falla durante la ejecución debido a algún error inesperado.
- *Error de presentación:* Cuando la salida del programa es correcta, pero no está presentada en el formato especificado.
- *Tiempo límite excedido:* Cuando el programa toma más tiempo que el establecido para procesar todas las entradas.

La aplicación informa inmediatamente al usuario y almacena el resultado de la evaluación. Finalmente, los usuarios se ordenan tomando en consideración la cantidad de problemas aceptados.

En los últimos años, en adición a la creciente presencia en la web de estas aplicaciones, también se han realizado varias investigaciones alrededor de su empleo para evaluar tareas de programación, resaltándose su importancia por encima de las alternativas tradicionalmente empleadas (Liu, 2013). Kurnia et al. (2001) formalmente presentaron a los jueces en línea como herramientas para evaluar de forma automática los códigos fuentes desarrollados por los estudiantes como parte de la resolución de problemas de programación previamente propuestos. Adicionalmente, hicieron un análisis del efecto que causa la implantación de este tipo de herramientas en una institución académica. Leal y Silva presentaron a Mooshak (Leal & Silva, 2003), un sistema que actúa como plataforma de gestión de competencias de programación, y a la vez como juez en línea; y posteriormente Georgouli y Guerreiro (2011) mostraron la integración de este sistema con una plataforma tradicional de enseñanza en línea tipo Learning Management System (LMS). Kosowski, Małafiejski, y Noiński (2007) formalmente describieron el juez en línea y sistema de competencias SPOJ, aplicado de forma exitosa en los estudiantes de la Universidad Tecnológica de Gdańsk. De la Fuente, Pardo y Delgado-Kloos (2013) presentan un sistema que automáticamente evalúa las actividades sometidas por los estudiantes, y gradualmente desbloquea las siguientes basándose en los resultados recibidos. Zhu y Chen (2013) han propuesto recientemente un enfoque basado en vínculos para integrar la información asociada a diferentes jueces en línea de programación en un sistema único a nivel local. Finalmente, Llana et al. (2014) proponen *FLOP*, un sistema de evaluación automática de ejercicios de programación centrado en el usuario, donde se da prioridad a las facilidades de administración, siendo así su propuesta superior, en esta dirección, a otras aplicaciones semejantes.

La revisión de la literatura sugiere la inexistencia de trabajos en el área de los jueces en línea, relacionados con la tarea de sugerir problemas a resolver a los estudiantes, basándose en su propio comportamiento previo y en el comportamiento de usuarios semejantes. Aunque jueces en línea tradicionales como el de la Universidad de Valladolid han presentado servicios del tipo "siguiente-a-resolver" (next-to-solve), que proponen problemas a resolver basándose en la cantidad de soluciones aceptadas, no se han podido detectar trabajos que vayan mucho más allá de esto.

En la práctica, siguiendo los conceptos del aprendizaje competitivo (Regueras et al., 2009), la experiencia muestra que el objetivo final de los usuarios en este tipo de aplicaciones es el de resolver la mayor cantidad de problemas. Sin embargo, en la parte baja del ranking de los usuarios siempre se observa un importante grupo que, a pesar de tener posibilidades de alcanzar una mejor posición en el listado, tienen muy pocos problemas resueltos y muchos fallos. Una de las causas de esto se considera asociada a que muchas veces se carece de un buen criterio para escoger qué problema resolver, acorde al estado actual del usuario. Esta es la principal motivación para el desarrollo de un sistema recomendador de apoyo a estas aplicaciones.

3.3 Sistemas recomendadores en e-learning

Los escenarios de e-learning siempre han sido una de las principales áreas de aplicación de los sistemas recomendadores, habiéndose publicado recientemente varias revisiones de los trabajos más significativos en esta temática, entre las que están (Ali, Ghani, & Abd Latiff, 2015; Chughtai, Selama, & Ghani, 2013; Manouselis, Drachsler, Vuorikari, Hummel, & Koper, 2011; Rafaeli, Dan-Gur, & Barak, 2005; Romero & Ventura, 2010; Verbert et al., 2012). En estas revisiones se citan alrededor de 300 trabajos de sistemas de recomendación en escenarios de e-learning. Al hacerse difícil referenciar este volumen de artículos, en la presente sección sólo se hará referencia a los trabajos que resultan de interés con respecto al contexto presentado en la sección 3.2.

Zaïane (2002) define un sistema recomendador en un escenario de e-learning, como un agente que "inteligentemente" recomienda acciones a un estudiante, basándose en las acciones de otros estudiantes. Específicamente, propone el uso de la minería de reglas de asociación para descubrir estas acciones, y posteriormente emplearlas en la recomendación de ítems

desconocidos. No obstante, a pesar de la cantidad de investigaciones en esta área, sólo existe un reducido grupo de estas que se centra en la recomendación de tareas de programación.

En este sentido, De Oliveira, Marques Ciarelli, & Oliveira (2013) desarrollan un sistema recomendador como soporte a las prácticas de programación, a través de la recomendación de "clases" de actividades. Esta tarea se trata como un enfoque de clasificación multietiqueta, donde el perfil de cada estudiante se asocia con una o más de las mencionadas clases. Como ejemplo de las clases de actividades consideradas pueden citarse "operadores aritméticos", "tipos de datos", "estructura if", "estructura while", entre otras.

Hsiao, Sosnovsky, & Brusilovsky (2010), por su parte, presentaron JavaGuide, un sistema centrado en guiar a los estudiantes a través de las preguntas apropiadas en un curso de programación en Java; y en adición investigaron el efecto de este sistema en un grupo de personas en tiempo real. En el mismo, cada pregunta está compuesta por un pequeño fragmento de código, sobre el que el estudiante debe introducir el valor final de una variable indicada, o lo que será impreso hacia la salida estándar.

Klašnja-Milićević, Vesin, Ivanović, y Budimac (2011) y Vesin, Klašnja-Milićević, Ivanović, y Budimac (2013) describen el módulo de recomendación de un sistema tutorial de programación, que automáticamente se adapta a los niveles de conocimiento del estudiante. El sistema es capaz de reconocer los estilos de aprendizaje de los usuarios, utilizando para esto varias ontologías de dominio. El objetivo principal de esta propuesta es descubrir secuencias de acceso a las actividades, con vistas a utilizar técnicas clásicas de filtrado colaborativo para recomendar vínculos relevantes y acciones durante el proceso de aprendizaje. Este sistema posteriormente se extiende a través de un uso más amplio de ontologías y reglas de adaptación (Vesin, Ivanović, Klašnja-Milićević, & Budimac, 2012).

Hsieh, Lee y Su (2013) proponen un sistema de aprendizaje remedial para asistir a los estudiantes, tras una evaluación en línea en asignaturas de programación. Con este fin, utilizan las categorías asociadas a las actividades para sugerir rutas de aprendizaje alternativas con vistas a reforzar los temas con mayores dificultades.

Finalmente, Ruiz-Iniesta, Jimenez-Diaz, & Gómez-Albarrán (2014) presentan una estrategia basada en el conocimiento para recomendar recursos educativos en un curso de programación, con el objetivo de proveer un acceso personalizado a estos. La propuesta depende de una

descripción de los recursos basada en estándares de metadatos enriquecidos por información semántica basada en una ontología, e información contextual del usuario.

La revisión realizada permite concluir que la mayoría de las propuestas, que pueden ser clasificadas como sistemas recomendadores híbridos, están concebidas para escenarios muy específicos y tienen una alta dependencia de información adicional más allá de la mera interacción de los usuarios sobre los ítems. Con respecto a esto, en el presente problema, descrito en la sección anterior, únicamente se dispone del historial de cada usuario en el juez en línea. Así, de forma general resulta sumamente complicado adaptar la mayoría de los trabajos previos para darle solución al mismo. Por tanto, se hace necesario desarrollar una nueva propuesta a través de un punto de vista más general en el área de investigación de los sistemas recomendadores. Esta propuesta será comparada exclusivamente con el trabajo presentado por Zaïane (2002), el que sí se considera que es de un propósito más general y puede ser adaptado al contexto presentado.

3.4 Un método para la recomendación de problemas a resolver en el juez en línea

La información básica requerida para la generación de recomendaciones está esencialmente contenida en la base de datos del juez en línea. Específicamente, se cuenta con un conjunto de sumisiones, cada una conformada por una tripleta <usuario, problema, veredicto>, que representa un intento de resolver el problema correspondiente, por el usuario correspondiente, dando como resultado el veredicto correspondiente (aceptado, respuesta incorrecta, tiempo límite excedido, error en tiempo de ejecución, o error de compilación). A los efectos de la presente propuesta, esta información será representada en términos de una función **D: (u, i)→ (resuelto, n)**[7], que recibirá como entrada un usuario u y un problema i, y devolverá el número de veces n que dicho u ha intentado solucionar el mencionado problema i, y si finalmente lo ha resuelto o no.

Tomando en cuenta los datos disponibles así como el objetivo inicial de construir un sistema recomendador que posibilite al usuario dar solución a la mayor cantidad de problemas posibles basándose en el desempeño de sus pares, tal y como se mencionó al principio del capítulo, se opta por concebir un enfoque de recomendación basado en filtrado colaborativo. Trabajos como (Wang & Yang, 2012) (Winoto, Tang, & McCalla, 2012) y (Zheng, Ma, & Li, 2013) han

[7] A lo largo del capítulo, se utilizarán indistintamente las variables *i* y *p*, para hacer referencia a los problemas.

presentado previamente resultados positivos en el empleo de esta técnica en escenarios de e-learning.

La figura 3.1 muestra el esquema general de un sistema de filtrado colaborativo para un juez en línea de programación. Este esquema está compuesto por cuatro etapas fundamentales, que a la vez coinciden con las cuatro etapas generales de un sistema de filtrado colaborativo basado en vecindario (recopilación de datos, cálculo de vecindario, predicción y recomendación), presentadas en el capítulo 1.

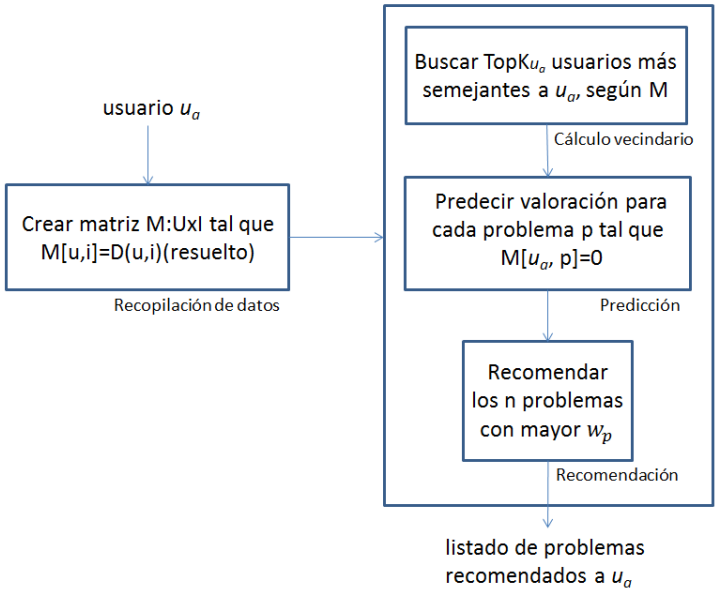


Figura 3.1. Esquema general del sistema recomendador para jueces en línea de programación.

A continuación se analizarán en detalle cada una de las cuatro etapas fundamentales mostradas en la figura 3.1.

Crear matriz de usuarios-problemas: En este caso, se requiere la transformación de los datos disponibles en un formato que facilite la generación de recomendaciones basadas en filtrado colaborativo. En esta dirección, se opta por una matriz *booleana* donde cada fila representa un perfil específico, y cada columna un problema. Cada posición recibe un valor 1, si el correspondiente usuario ha resuelto el correspondiente problema, y 0 de lo contrario. La tabla 3.1 muestra como ejemplo una hipotética matriz usuario-problema. En este caso "lgvallejo" tendría resueltos los problemas Problema_1, Problema_3 y Problema_4, "jelara" tendría resuelto los problemas Problema_3 y Problema_4, y así sucesivamente.

Tabla 3.1. Matriz de usuarios/problemas

		problemas			
		Problema_1	Problema_2	Problema_3	Problema_4
Usuarios	lgvallejo	1	0	1	1
	jelara	0	0	1	1
	ejaltuna	1	1	0	0
	yclemente	1	0	0	1

Cálculo del vecindario del usuario actual: Considerando la ya mencionada meta principal de los usuarios en los jueces en línea, la aplicación de un enfoque de recomendación usuario-usuario podría verse como la tarea de "recomendar al usuario activo, aquellos problemas resueltos por el resto de los usuarios que tienen un comportamiento similar atendiendo a los problemas resueltos y fallidos". Para dar cumplimiento a este propósito, inicialmente se debe obtener el vecindario del usuario actual, basándose en una medida de similitud.

Tal y como se mostró en el capítulo 1, el coeficiente de correlación de Pearson es utilizado frecuentemente como medida de similitud en entornos con ratings numéricos. Sin embargo, en este escenario la matriz usuarios-problemas está compuesta por categorías (resuelto y no resuelto), y por tanto no es apropiada la aplicación de esta medida. Para estos escenarios, Amatriain et al. (2011) han valorado la aplicación de funciones alternativas como el Simple Matching o el coeficiente de Jaccard. Específicamente, el coeficiente de Jaccard mide el grado de solapamiento entre dos conjuntos A y B, y se define como $|A \cap B|/|A \cup B|$ (Manning, Raghavan, & Schütze, 2008). Así, se decide tomar esta medida como función de similitud entre dos usuarios en el contexto presentado.

Una comparación de dos usuarios usando esta función sería como sigue. Tomando como ejemplo "lgvallejo" y "ejaltuna" de la tabla 3.1, al menos uno de ellos ha resuelto los problemas (Problema_1, Problema_2, Problema_3 y Problema_4), pero tienen únicamente el mismo resultado en común para Problema_1, por lo que su valor de similitud es 1/4.

Predicción: Una vez que se obtiene el listado de los k vecinos más cercanos al usuario actual u_a, esta información se usa para calcular una valoración w_p de la idoneidad de la recomendación de cada uno de los problemas p no resueltos por este. Esta valoración viene dada por el número de vecinos que sí han logrado resolver el correspondiente problema, tal y como se muestra en la ecuación (3. 1).

$$w_p = \sum_{v \in TopK_{u\_a}} M[v, p] \qquad (3.\ 1)$$

Recomendación: Finalmente, la etapa de recomendación viene dada por recomendar los n problemas con mayores valores de w_p.

El algoritmo 3.1 presenta un esbozo del método para sugerir problemas a resolver. Este recibe como entrada el usuario activo u_a, la matriz de usuarios-problemas M, y la cantidad de recomendaciones deseadas. En la primera parte (líneas 01-05), se calculan las similitudes entre el usuario activo y el resto de los usuarios, se almacenan en una matriz s, y se ordenan descendentemente acorde a los valores de similitud. Tras esto, se determinan los k perfiles más similares (línea 06), y se valora cada problema no resuelto por el usuario activo como la cantidad de perfiles más similares que sí lo han resuelto (líneas 07-11). Finalmente, se retornan en calidad de recomendación los n problemas con mayor valoración.

	Input: u_a, M, cantidadRecomendaciones **Output:** listado de problemas recomendados
01	**para cada** usuario u diferente de u_a
02	*s*[u, u_a]= Similitud(u, u_a)
03	*vecinos*.adicionar(u)
04	**fin para**
05	**ordenar** descendentemente *vecinos***,** según los valores *s*[u, u_a]
06	*topKVecinos* = *vecinos*.subLista(0, k)
07	w_p[]={0}
08	**para cada** usuario u en *topKVecinos*
09	**para cada** problema p
10	**si(**M[u,p]== 1 y M[u_a,p]!= 1)
11	w_p[p]++
12	**fin para**
13	**fin para**
14	**devolver** lista de los n problemas con mayores ponderaciones

Algoritmo 3.1. Filtrado colaborativo usuario-usuario adaptado al escenario del juez en línea.

El marco de trabajo presentado no incorpora información específica del juez en línea, actualmente disponible. En este caso sólo considera que un usuario u ha resuelto exitosamente un problema p (M[u, p]=1), o que no lo ha resuelto (M[u, p]=0). A raíz de esto, en la siguiente sección se presentará un nuevo método centrado en aprovechar los datos asociados a los fallos de los usuarios (problemas no resueltos), con vistas a mejorar el desempeño del método de recomendación. De esta forma, se pretende enriquecer los perfiles de manera que estos

reflejen de una mejor forma las capacidades y aptitudes de los usuarios, permitiéndose por tanto hacer una mejor selección del vecindario a utilizar para generar las recomendaciones.

3.5 Recomendación basada en la matriz extendida de usuarios-problemas

Los métodos de filtrado colaborativo basados en usuario se centran en encontrar perfiles con intereses y preferencias similares a las del usuario actual, y utilizar esta información para la generación de recomendaciones. En relación con el problema que se aborda en el presente capítulo, en la sección anterior se presentó un método de filtrado colaborativo en el que se buscan usuarios con conocimientos y aptitudes semejantes al usuario actual, y donde esta semejanza viene dada por los problemas resueltos y no resueltos.

Sin embargo, se considera que una semejanza entre usuarios, basada en conocimientos y aptitudes, va más allá de la similitud en cuanto a problemas resueltos con éxito y problemas no resueltos. Así, tomando en cuenta que un usuario puede haber resuelto un problema con facilidad, o haber necesitado muchos intentos fallidos previos a su solución exitosa, se asume la existencia de tres tipos de interacciones usuario-problema, en contraposición con las dos presentadas en la sección anterior. Estas interacciones son:

1. El usuario nunca ha tratado de resolver el problema.
2. El usuario resolvió el problema, necesitando a lo sumo un intento previo fallido.
3. El usuario resolvió el problema, necesitando como mínimo dos intentos previos fallidos.

Se considera que en adición a establecerse dos categorías atendiendo a los problemas resueltos, también es factible establecer dos nuevas categorías para hacer referencia a los problemas cuya solución ha sido intentada pero en los que este intento no ha sido exitoso.

Así, estas categorías permitirían representar dos nuevos escenarios diferentes. El primero estaría asociado a aquellos problemas que el usuario ha intentado resolver, ha fallado, pero no ha sido insistente con el intento; quedando así la posibilidad de que en futuro lo pueda resolver si continúa probando con nuevas soluciones. El segundo, por otro lado, vendría dado por aquellos problemas a los que insistentemente se ha intentado dar solución, pero que a pesar de haberse tratado varias veces, se ha fallado siempre.

Por tanto, de manera general en esta sección se asume la posible existencia de cinco diferentes interacciones usuario-problema. Estas interacciones son:

0. El usuario nunca ha tratado de resolver el problema.
1. El usuario no ha resuelto el problema, y tiene como mínimo dos fallos sobre este.
2. El usuario no ha resuelto el problema, y tiene a lo sumo un intento fallido sobre este.
3. El usuario resolvió el problema, necesitando como mínimo dos intentos previos fallidos.
4. El usuario resolvió el problema, necesitando a lo sumo un intento previo fallido.

Estas asunciones son utilizadas para construir una matriz extendida M* que contendría perfiles de usuario con una mayor granularidad con respecto a M, y que sería tomada como base para el cálculo de los vecindarios correspondientes. La figura 3.2 muestra el esquema general del sistema, con esta extensión añadida. A continuación se analizará en detalle cada una de estas fases, haciendo hincapié en las particularidades con respecto al método anterior.

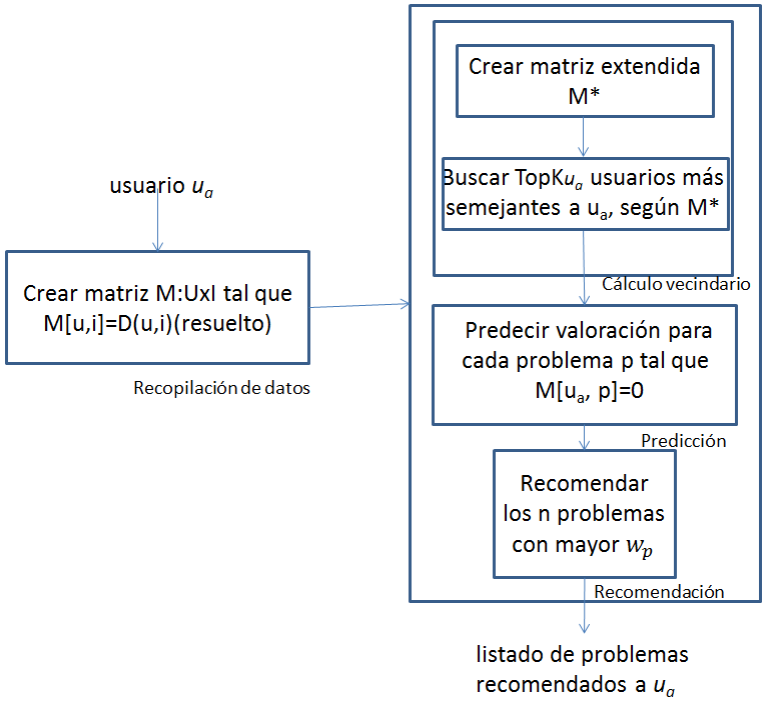


Figura 3.2. Esquema general del sistema recomendador para jueces en línea de programación, basado en la matriz extendida

Crear matriz de usuarios-problemas: Esta propuesta continúa partiendo de la matriz de usuarios-problemas obtenida de una forma igual a la de la sección anterior.

Cálculo del vecindario del usuario actual: El empleo de la matriz M* para el cálculo de la similitud entre usuarios constituye la principal novedad de esta propuesta. Las expresiones (3. 2) y (3. 3) muestran la forma de obtener M*, la que estaría conformada por valores en el rango

[0, 4], que se corresponden respectivamente con cada una de las cinco posibles interacciones usuario-problema mencionadas al inicio de esta sección.

El hecho de disponer de una matriz enriquecida evidentemente permite una caracterización más detallada de los perfiles de los usuarios, y por tanto una mayor efectividad para encontrar los más semejantes a un usuario dado. No obstante, el hecho de no ser M* una matriz binaria implica que no se pueda aplicar directamente el coeficiente de Jaccard, concebido exclusivamente para datos binarios, y empleado en la sección anterior. Asimismo, se descarta el uso del coeficiente extendido de Jaccard (Cha, 2007), por no haber sido considerado previamente en el contexto de los sistemas recomendadores (Amatriain, et al., 2011), posiblemente por estar cercanamente relacionado, según (Cha, 2007), con otra función que sí ha sido ampliamente usada en este contexto, que es la similitud basada en el coseno.

$$M[u,i]=0 \;\Rightarrow\; M^{*}[u,i]=\begin{cases} 0, si\ D(u,i)(n)=0\ (\text{ninguna propuesta de solución}) \\ 1, D(u,i)(n)\geq 2\ (\text{no resuelto, muchos fallos}) \\ 2, si\ D(u,i)(n)>0\ y\ D(u,i)(n)<2\ (\text{no resuelto, pocos fallos}) \end{cases} \quad (3.\ 2)$$

$$M[u,i]=1 \;\Rightarrow\; M^{*}[u,i]=\begin{cases} 3, si\ D(u,i)(n)\geq 2\ (\text{resuelto con muchos fallos previos}) \\ 4, si\ D(u,i)(n)<2\ (\text{resuelto con pocos fallos previos}) \end{cases} \quad (3.\ 3)$$

A raíz de esto, se decide usar una función de semejanza distinta del coeficiente de Jaccard. En este sentido, en la sección de experimentación se analizará el desempeño de varias funciones de semejanza posibles a utilizar en la comparación de estos perfiles. Específicamente, serán evaluadas dos funciones presentadas en el capítulo 1, que son el coeficiente de correlación de Pearson y la similitud basada en el coseno. En adición, será evaluado el desempeño del Simple Matching Coefficient (SMC), previamente mencionado en el capítulo 1, y que es considerado como la medida típica para datos nominales (Dunn & Everitt, 1982).
Dados dos objetos A y B, cada uno con n atributos binarios, SMC se define como:

$$SMC=\frac{número\ de\ atributos\ iguales}{número\ de\ atributos}=\frac{S_{00}+S_{11}}{S_{00}+S_{01}+S_{10}+S_{11}} \quad (3.\ 4)$$

donde:

S_{00} representa el número de atributos donde A y B tienen ambos valor 0.
S_{01} representa el número de atributos donde el atributo de A tiene valor 0 y el de B tiene valor 1.

S_{10} representa el número de atributos donde el atributo de A tiene valor 1 y el de B tiene valor 0. S_{11} representa el número de atributos donde A y B tienen ambos valor 1.

Esta función es fácilmente extensible para objetos cuyos atributos pueden tomar más de dos estados, como es el caso de las filas de la matriz extendida de usuarios e ítems. Para este caso, los atributos podrían tomar hasta cinco estados, los que son obtenidos a partir de las ecuaciones (3. 2) y (3. 3). La ecuación (3. 5) muestra la forma que tomaría la medida SMC en este escenario.

$$SMC = \frac{\sum_{i=0}^{4} S_{ii}}{\sum_{i=0}^{4}\sum_{j=0}^{4} S_{ij}} \qquad (3.\ 5)$$

Sin embargo, el problema actual presenta un caso particular, que es el de la cantidad S_{00}. Contrario a los demás valores de la matriz M* que están asociados a alguna acción específica del usuario sobre el problema y tal como lo muestra la ecuación (3. 2), el valor 0 es indicador de ausencia de acción. Así, resulta claro que el hecho de que dos usuarios que compartan un gran número de problemas sobre los que no han tenido ninguna acción, no debe ser indicador de semejanza. A raíz de esto, se propone excluir este término del cálculo de la similitud, quedando la función tal y como se muestra en la ecuación (3. 6).

$$SMC = \frac{\sum_{i=1}^{4} S_{ii}}{\sum_{i=0}^{4}\sum_{j=0}^{4} S_{ij} - S_{00}} \qquad (3.\ 6)$$

Finalmente, en este método las dos restantes etapas del proceso de recomendación (predicción y recomendación) se llevan a cabo de manera similar a la del método anterior.

3.6 Método basado en el procesamiento de ruido natural en las interacciones de los usuarios.

En el capítulo 2 se presentó un enfoque general para preprocesar preferencias en un sistema recomendador de filtrado colaborativo, centrado en el tratamiento de ruido natural. Este método se apoya en el empleo del propio conjunto de ratings para llevar a cabo la clasificación de los

usuarios e ítems, y finalmente identifica como posibles inconsistencias a aquellos ratings cuyos valores sean contradictorios con respecto al usuario e ítem correspondiente.

Las inconsistencias de tipo ruido natural en sistemas recomendadores de filtrado colaborativo son usualmente provocadas por usuarios genuinos debido a su comportamiento imperfecto. En el escenario particular de los jueces en línea de programación, este comportamiento imperfecto se considera asociado a aquellos casos en los que un usuario no resuelve un problema de la forma esperada, tomando en consideración sus conocimientos y aptitudes, y las características del problema en cuestión. Estos casos pueden ser provocados por factores muy diversos, tales como que el usuario haya copiado la solución de otro usuario, o que haya tenido dificultades inesperadas en solucionar un problema que debería haber resuelto de forma inmediata.

Actualmente está empezando a cobrar cierto auge el desarrollo de trabajos de preprocesamiento de datos para escenarios de e-learning (Romero, Romero, & Ventura, 2014). Sin embargo, son pocas las investigaciones centradas en la limpieza de los datos, categoría asociada a la propuesta presentada en el capítulo anterior. En esta dirección, Romero, Ventura, y De Bra (2004) presentan un sistema donde se detectan y eliminan períodos de tiempo largo entre dos acciones llevadas a cabo por el mismo estudiante, y en adición se eliminan datos incompletos tales como capítulos visitados de forma incompleta, y actividades y pruebas no finalizadas. García, Romero, Ventura y De Castro (2009) mostraron que a los estudiantes usualmente se les registraba mucho tiempo dentro de un mismo ejercicio o concepto, probando que esto se debía a que abandonaban la computadora sin ejecutar previamente el comando salir. Para mitigar este hecho, proponen establecer un umbral de tiempo máximo, considerando ruidosos todos los valores que sobrepasen dicho umbral. Todos estos valores detectados son reemplazados por el mencionado umbral. Finalmente, Aher y Lobo en (Aher & Lobo, 2012) y posteriormente en (Aher & Lobo, 2013) se apoyan en la minería de reglas de asociación para filtrar datos que satisfagan determinado patrón. En este caso, si los resultados tienen una regla con todas sus conclusiones NO (que indica que el estudiante no está interesado en el curso), estos cursos son entonces dejados de tener en cuenta posteriormente en el proceso de minería. Así, se eliminan los cursos con pocos estudiantes y los estudiantes con pocos cursos. Más allá de estas referencias, no se ha podido identificar en la literatura la presencia de otros trabajos de este tipo concebidos específicamente para sistemas recomendadores en e-learning.

Asumiendo que en las secciones 3.4 y 3.5 se presentó un método basado en filtrado colaborativo para sugerir problemas a resolver en un juez en línea de programación, en la presente sección se propone un método de preprocesamiento de datos para este escenario, que toma como base el enfoque presentado en el capítulo 2 y las ideas planteadas al principio de la presente sección para caracterizar el ruido natural. Tal y como se muestra en la figura 3.3, este método actuaría sobre la matriz extendida M* inmediatamente antes del cálculo de las similitudes entre los diferentes usuarios. En este caso se procederá a explicar únicamente esta etapa, considerando que el resto de los pasos coinciden con los del método de la sección 3.5.

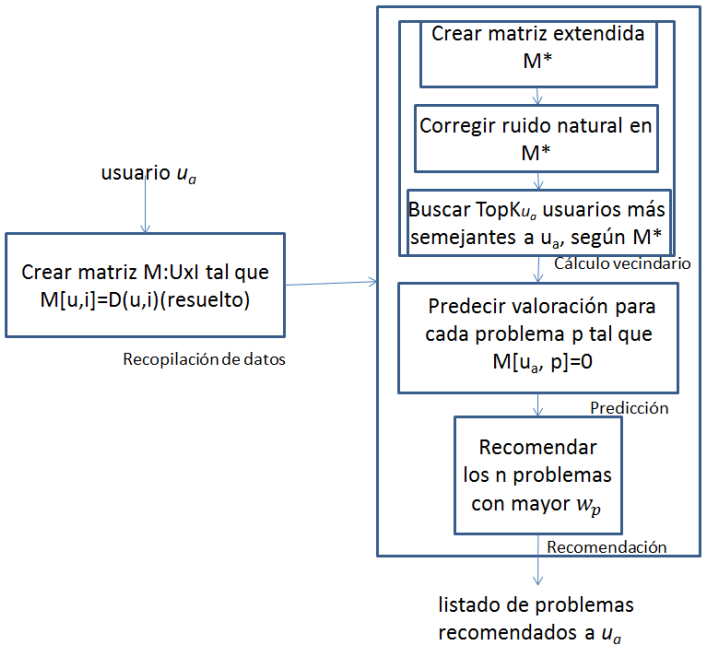


Figura 3.3. Esquema general del sistema recomendador para jueces en línea de programación, basado en la matriz extendida con corrección de ruido natural.

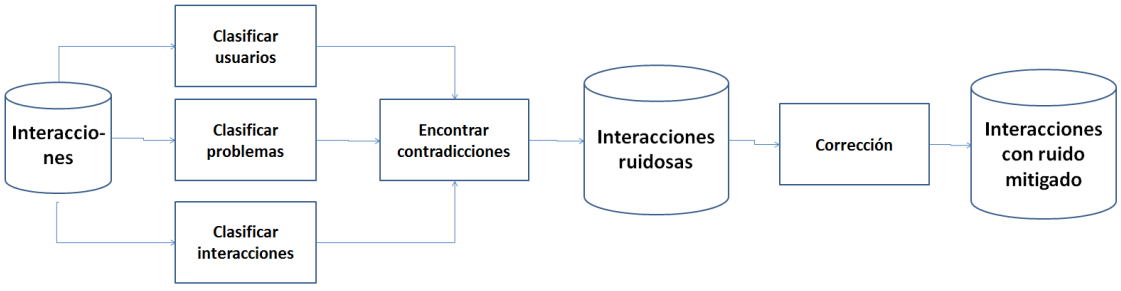


Figura 3.4. Procesamiento de ruido natural en el juez en línea de programación

La figura 3.4 muestra el proceso de corrección de ruido natural en este escenario, tomando como punto de partida el esquema general presentado en la figura 2.1 en el capítulo anterior. Así, inicialmente se realiza una clasificación de los usuarios, problemas, e interacciones, para luego encontrar contradicciones entre estas clasificaciones. Estas contradicciones son usadas para detectar interacciones ruidosas, las que son corregidas con vistas a mitigar el efecto del ruido encontrado.

Con este fin, se requiere inicialmente concebir una posible clasificación para los usuarios y problemas presentes en el actual escenario. Se propone identificar los usuarios acorde a las siguientes tres tendencias:

Tendencias de los usuarios:

1) Usuario impreciso: el usuario tiende a tener muchos intentos fallidos de solución antes de finalmente resolver el problema.
2) Usuario certero: el usuario tiende a solucionar de manera rápida la mayor parte de los problemas que finalmente logra resolver.
3) Usuario variable: el comportamiento del usuario oscila entre las dos categorías previas.

Por otra parte, se propone identificar a los problemas igualmente acorde a tres tendencias fundamentales:

1) Problema complejo: el problema en el que muchos usuarios necesitan muchos intentos fallidos previos, antes de finalmente lograr resolverlo.
2) Problema sencillo: el problema en el que la mayoría de los usuarios que lo intentan resolver, lo hacen necesitando pocos intentos previos fallidos.
3) Problema variable: el problema que oscila entre las dos categorías anteriores.

Con vistas a facilitar la definición formal de cada una de las categorías de los usuarios, se propone agrupar los problemas resueltos por el usuario u en dos conjuntos $C1_u$ y $C2_u$, que se corresponden con las categorías presentadas en la ecuación 3.3 de la sección 3.5.

1) Conjunto de problemas resueltos necesitando muchos intentos por parte del usuario u, $C1_u$: $C1_u=\{p|\ M^*[u,p]=3\}$
2) Conjunto de problemas resueltos necesitando pocos intentos por parte del usuario u, $C2_u$: $C2_u=\{p|\ M^*[u,p]=4\}$

A su vez, se definen dos conjuntos $C1_p$ y $C2_p$ para agrupar todos los usuarios que han resuelto cierto problema p.

1) Conjunto de usuarios que han resuelto el problema p necesitando muchos intentos, $C1_p$: $C1_p=\{u|\ M^*[u,p]=3\}$
2) Conjunto de usuarios que han resuelto el problema p necesitando pocos intentos, $C2_p$: $C2_p=\{u|\ M^*[u,p]=4\}$

La tabla 3.2 define formalmente las tendencias asociadas a los usuarios y problemas, en término de las cardinalidades de los conjuntos presentados. Específicamente, se define el usuario certero como aquel cuya cantidad de problemas resueltos necesitando pocos envíos, al menos duplique la cantidad de aquellos resueltos necesitando muchos envíos. El usuario impreciso se define como el caso contrario. De forma similar, un problema se define como sencillo si el número de usuarios que lo han resuelto con pocos envíos al menos duplica al número de resueltos con muchos envíos, mientras que se categoriza como complejo en el caso opuesto. Finalmente, se han definido los perfiles de usuario y problema variable para aquellos casos en los que no se cumpla ninguna de las condiciones anteriores.

Tabla 3.2. Clases de usuario y de problema

Clases de usuario	
Usuario certero	$\|C2_u\| \geq 2\|C1_u\|$
Usuario impreciso	$\|C1_u\| \geq 2\|C2_u\|$
Usuario variable	No satisface ninguna de las otras dos condiciones
Clases de problema	
Problema sencillo	$\|C2_p\| \geq 2\|C1_p\|$
Problema complejo	$\|C1_p\| \geq 2\|C2_p\|$
Problema variable	No satisface ninguna de las otras dos condiciones

Una vez que cada usuario y problema ha sido clasificado, esta información de conjunto con las categorías de interacción definidas en la ecuación 3.3 de la sección 3.5, se utiliza para detectar interacciones “ruidosas”. Con este propósito se parte de considerar dos grupos de categorías homólogas respectivamente asociadas a usuarios, problemas, e interacciones (tabla 3.3). El principal criterio para identificar las categorías homólogas es que estas estén definidas sobre el

mismo tipo de interacción, siendo las clases certero y sencillo definidas sobre el tipo de interacción M*[u,p]=4, y las clases impreciso y complejo sobre el tipo M*[u,p]=3.

Más allá de esto, la literatura relacionada con el desempeño de los usuarios en los jueces en línea ha sugerido la existencia de estas clases de usuarios y de problemas, y la asociación entre las mismas. Skiena y Revilla (2006) han afirmado que usualmente existen diferentes niveles en los estudiantes, desde los estudiantes de nivel introductorio, hasta los estudiantes con nivel apropiado para competencias internacionales. Por otro lado se ha planteado que existen varios niveles de problemas, clasificándose en categorías tales como básico, moderado, avanzado, entre otras (Arefin, 2006; Skiena & Revilla, 2006). En adición, Skiena y Revilla (2006) afirman que el grado de facilidad con el que se resuelvan los ejercicios dependerá del nivel de los estudiantes, esperándose por tanto que los estudiantes aventajados no necesiten muchos intentos fallidos para resolver cierto grupo de problemas (posiblemente los más sencillos), mientras que los usuarios más novatos, por el contrario, sí los necesiten para cierto grupo de problemas (posiblemente los más complejos). Estos razonamientos entran en correspondencia con las asunciones hechas para la conformación de las clases homólogas, y de cierta forma sirven de justificación para la definición de las mismas.

Tabla 3.3. Clases homólogas

	Clases de usuario	**Clases de problema**	**Tipo de interacción**
Grupo 1	*Certero*	*Sencillo*	M*[u,p]=4
Grupo 2	*Impreciso*	*Complejo*	M*[u,p]=3

Así, se propone un proceso de detección de interacciones “ruidosas” que asume que para cierta interacción en la que se haya resuelto un problema determinado, el hecho de que el correspondiente usuario y problema pertenezcan respectivamente a las clases impreciso y complejo, implica que el valor finalmente asociado a dicha interacción deberá también ser el homólogo con dichas clases (M*[u,p]=3). De lo contrario, si M*[u,p]=4, este debe considerarse ruidoso, y en este caso se propone directamente corregirlo a través del reemplazo por el valor 3.

Un procedimiento semejante sería realizado para el caso en que, habiéndose resuelto el problema, el usuario y el problema pertenecieran a las clases certero y sencillo, y M*[u,p]=3. En este caso, el reemplazo se haría de la forma M*[u,p]=4.

Las ecuaciones (3. 7) y (3. 8) resumen esta transformación de la matriz M*, centrada en la reducción del ruido natural presente.

$$M^*[u,p]=3 \;\Rightarrow\; M^*[u,p]=\begin{cases}4, si\ |C2_u|\geq 2|C1_u|\ \ y\ \ |C2_p|\geq 2|C1_p| \\ 3, de\,lo\,contrario\end{cases} \quad (3.\ 7)$$

$$M^*[u,p]=4 \;\Rightarrow\; M^*[u,p]=\begin{cases}3, si\ |C1_u|\geq 2|C2_u|\ \ y\ \ |C1_p|\geq 2|C2_p| \\ 4, de\,lo\,contrario\end{cases} \quad (3.\ 8)$$

Se desea recalcar que el ruido natural se ha detectado trabajando sobre las dos categorías asociadas al éxito en la solución de problemas. En este caso resulta clara la contradicción entre ambas: una de ellas está asociada a soluciones rápidas de problemas, y la otra está asociada a soluciones demoradas. Sin embargo, para el caso de las categorías asociadas a la no solución de un problema una posible contradicción no queda reflejada de forma tan clara, puesto que el intento de solución de un problema p por un usuario u podría estar en la categoría $M^*[u, p]=2$, y posteriormente pasar a la categoría $M^*[u, p]=1$ si aumenta la cantidad de intentos fallidos. A raíz de esto se decidió no buscar ruido natural en este último escenario.

3.7 Método basado en la recomendación exclusiva de problemas desconocidos por el usuario

En esta sección se propone una extensión del método anterior en la que únicamente se le recomienda al usuario final aquellos problemas que nunca ha tratado de resolver, descartándose aquellos cuya solución ha sido intentada, pero sobre los que se ha fallado.

La particularidad de este método está asociada a la etapa de predicción, siendo las restantes etapas iguales al método de la sección anterior. En este caso, en vez de predecirse una valoración para todos los problemas p tal que $M[u_a, p]=0$, se predice una valoración y por tanto se considera como candidato a ser recomendado, a todo problema que verifique $M^*[u_a, p]=0$. Esto es, que la cantidad de intentos fallidos de solución del problema sea 0, y que no se haya resuelto.

Este enfoque de recomendación toma en consideración que el usuario final podría desear que el sistema recomendador le sugiera problemas desconocidos, en vez de recomendarle problemas que ya ha intentado resolver pero que ha fallado.

En la siguiente sección se llevará a cabo un estudio experimental para evaluar el desempeño de los métodos presentados.

3.8 Experimentación y resultados

Con vistas a evaluar las propuestas, en esta tesis se desarrollarán experimentos offline usando una base de datos obtenida del Juez en Línea Caribeño (http://coj.uci.cu). Esta está compuesta por 1910 usuarios y 575 problemas, donde cada usuario tiene resuelto como promedio alrededor de 42 problemas. Estos datos fueron representados a través de tripletas de la forma (usuario, problema, veredicto), obteniéndose un total de 151667 tripletas. Hasta donde se conoce, este trabajo está entre los pioneros en el desarrollo de una evaluación offline a través de una base de datos extraída de un escenario de recomendación en e-learning. Previos al mismo, en la literatura únicamente se ha podido identificar los trabajos de Manouselis (Manouselis, Kyrgiazos, & Stoitsis, 2014; Manouselis, Vuorikari, & Van Assche, 2010), mientras que por otro lado Verbert, Manouselis, Drachsler, y Duval (2012) han planteado la necesidad de continuar desarrollando investigaciones en esta dirección.

El marco de trabajo experimental fue preparado acorde a las sugerencias de Gunawardana y Shani (2009), también empleadas en el capítulo anterior. En este caso, para construir el conjunto de entrenamiento, para cada usuario se seleccionó aleatoriamente un conjunto de problemas resueltos por él, mientras que los restantes problemas resueltos fueron usados para construir el conjunto de prueba. La propuesta también recibe como entrada la información asociada a los fallos para cada par usuario-problema, por lo que esta se almacenó independientemente.

Precision y Recall son dos métricas de evaluación usualmente empleadas en la tarea de recomendar buenos ítems (Gunawardana & Shani, 2009), por lo que consecuentemente serán aplicadas en este escenario para evaluar los métodos propuestos. La sección 1.3 brevemente explicó el empleo de estas métricas en un escenario tradicional de recomendación. No obstante, a continuación se presenta la forma de aplicarlas en el presente contexto de recomendación.

Considerando la clasificación del posible resultado de recomendar un problema a un usuario (tabla 3.4), precision se define como la razón entre los problemas recomendados y a la vez resueltos, y los problemas recomendados. Del otro lado, recall se define como la razón entre los

mismos problemas recomendados y resueltos, y el total de resueltos (ecuaciones (3. 9) y (3. 10)).

Tabla 3.4. Posible resultado de la recomendación, considerando los problemas realmente resueltos.

	Recomendado	**No recomendado**
Resuelto	Verdadero-positivo(tp)	Falso-negativo (fn)
No resuelto	Falso-positivo (fp)	Verdadero-negativo (tn)

$$precision = \frac{\#tp}{\#tp + \#\ fp} \tag{3. 9}$$

$$recall = \frac{\#tp}{\#tp + \#\ fn} \tag{3. 10}$$

Como criterio para combinar precision y recall, se hará uso específicamente de la métrica F1, la que ya fue referenciada también en el capítulo 1 (ecuación (3. 11)).

$$F1 = \frac{2 * precision * recall}{precision + recall} \tag{3. 11}$$

En los experimentos, para cada usuario se obtiene el listado de ítems a recomendar, usando el conjunto de entrenamiento, el correspondiente método a evaluar, y los datos de los fallos en el caso que corresponda. Usando este listado y el listado de problemas en el conjunto de prueba (que representan los problemas resueltos), el valor de F1 se puede calcular fácilmente. Una vez computados estos para cada usuario, la eficacia final se obtiene a través de un F1 promedio considerando todos los usuarios. Para todos los experimentos se asume un valor k=80 para la cantidad de vecinos más cercanos, habiéndose obtenido este tras varias evaluaciones previas con diferentes valores de k.

Considerando las diferentes alternativas propuestas en las secciones anteriores para recomendar problemas a resolver, en esta sección se realizará la comparación de los siguientes enfoques:

-La recomendación de problemas basada en la matriz binaria de usuarios-problemas (sección 3.4), el que será nombrado *binario* en lo adelante.

-La recomendación de problemas basada en la matriz extendida de usuarios-problemas (sección 3.5), el que será nombrado *extendido* en lo adelante.

-La recomendación basada en la matriz extendida combinada con el método de detección de ruido natural propuesto en la sección 3.6, el que será nombrado como *extendido+nn*.

-El método *extendido+nn*, recomendando únicamente aquellos problemas que el usuario nunca ha tratado de solucionar. Esta alternativa será nombrada como *extendido+nn+exclusivo* a partir de ahora.

Previo a esta comparación principal, se evalúan tres posibles funciones de similitud para comparar dos perfiles de usuario en el método basado en la matriz extendida, para luego emplear la función de mejor desempeño en el resto de los métodos que dependen de esta matriz. Estas funciones de similitud son el Simple Matching Coefficient (SMC), el coeficiente de correlación de Pearson, y la similitud basada en el coseno.

La tabla 3.5 muestra el desempeño del método *extendido* con estas tres medidas de similitud, variando el número de recomendaciones en el intervalo [5, 40] con paso 5. Resulta evidente la diferencia entre el desempeño del SMC y el de las otras dos medidas. El hecho de que la matriz M* se obtenga a partir de la matriz binaria M, sugiere que los datos de los perfiles de usuario en M* tienden a ser de naturaleza nominal, más que de naturaleza ordinal. Por otra parte, SMC se concibe específicamente para estos escenarios, lo que hace pensar que sea esta la razón por la que su desempeño fue el mejor, en contraparte con Pearson y coseno que suelen utilizarse en otro tipo de escenarios.

Tabla 3.5. Valores de F1 con el desempeño de *extendido* con tres medidas diferentes de similitud

n	***5***	***10***	***15***	***20***	***25***	***30***	***35***	***40***
extendido+cosine	0,2230	0,2973	0,3294	0,3455	0,3495	0,3531	0,3492	0,3436
extendido+pearson	0,2305	0,3191	0,3573	0,3799	0,3905	0,3935	0,3939	0,3908
extendido+smc	**0,2689**	**0,4002**	**0,4639**	**0,4973**	**0,5122**	**0,5160**	**0,5156**	**0,5121**

En otra dirección, la tabla 3.6 presenta los resultados de la comparación entre los cuatro enfoques mencionados más arriba, variando el número de problemas recomendados de manera semejante a la tabla 3.5. Se muestra que los valores de F1 se mejoran con la incorporación de la información sobre los fallos y las demoras al proceso de recomendación, y que en adición, la búsqueda y procesamiento de inconsistencias en este escenario provoca en la mayoría de los casos una mejora en el desempeño del método que no considera este aspecto.

Por otro lado, se obtuvo un resultado negativo por parte del método *extendido+nn+exclusivo*. Tal y como se explicó en la sección 3.7, este método se centra en recomendar únicamente aquellos problemas que el usuario nunca ha tratado de resolver, o sea, sólo aquellos que verifiquen $M^*[u_a,p]=0$. Sin embargo, el resultado que se obtiene sugiere que es importante no dejar de considerar la variante de poder recomendar aquellos problemas con fallos previos, ya que el notable contraste entre los valores de F1 de este método con respecto a los restantes, indica que gran parte de estos problemas sí son finalmente resueltos con éxito.

La tabla 3.6 muestra que para más de la mitad de los posibles valores analizados variando el número de recomendaciones, la incorporación del método de preprocesamiento de ruido natural (*extendido+nn*) implica una mejora en el desempeño, u obtiene un resultado equivalente al del método sin preprocesamiento (*extendido*). Sin embargo, con vistas a obtener una conclusión definitiva sobre la mejora asociada al proceso de corrección presentado, se decide aplicar una prueba estadística en este escenario. Siguiendo las sugerencias de Gunawardana y Shani (2009), se decide aplicar la prueba de Wilcoxon para comparar *extendido+nn* con *extendido*.

Tabla 3.6. Valores de F1, modificando la longitud del listado de recomendaciones

n	***5***	***10***	***15***	***20***	***25***	***30***	***35***	***40***
binario	0,2668	0,3931	0,4538	0,4853	0,4992	0,5053	0,5032	0,4989
extendido	**0,2689**	0,4002	0,4639	**0,4973**	**0,5122**	0,5160	0,5156	**0,5121**
extendido+nn	0,2682	**0,4024**	**0,4661**	0,4972	**0,5122**	**0,5192**	**0,5168**	0,5107
extendido+nn +exclusivo	0,2487	0,3528	0,3983	0,4189	0,4242	0,4247	0,4202	0,4147

Con este fin, para los dos métodos se obtuvieron los valores de F1 variando el número de recomendaciones en el intervalo [1, 40], en este caso con paso 1 para poder tener mayores

evidencias del impacto del ruido natural en este escenario, con respecto a las evidencias presentadas en la tabla 3.6, obtenidas con paso 5. Estos valores de F1 fueron usados posteriormente para la comparación por pares considerando la hipótesis nula. Como resultado de esta evaluación, se obtuvo que se rechaza esta hipótesis con $p<0.05$, lo que ratifica que la diferencia entre ambos enfoques es estadísticamente significativa. La tabla 3.7 muestra un resumen de los resultados de esta comparación, especificando la cantidad de casos en los que *extendido+nn* gana, empata, o pierde con *extendido*. Los resultados muestran una evidente superioridad del método basado en el preprocesamiento de ruido natural.

Tabla 3.7. Resultados de la aplicación del test de Wilcoxon para comparar *extendido+nn* con *extendido*.

	Gana	**Pierde**	**Empata**	**Sig. asintót.**
extendido+nn vs *extendido*	33	5	2	0,000

Finalmente, se decide comparar los resultados obtenidos contra un enfoque de minería de reglas de asociación en sistemas recomendadores de e-learning, propuesto por Zaïane y previamente referenciado (Zaïane, 2002). Las reglas fueron obtenidas a través del método propuesto por Lin et al. (2002), donde se descubren reglas (para la recomendación) con exactamente un ítem en el consecuente, tomando en este escenario la forma <problema1, resuelto> y <problema2, resuelto> → <problema3, resuelto>. De forma semejante a Lin et al., en el presente caso estas reglas fueron descubiertas, para cada problema, siguiendo un enfoque Apriori (Agrawal & Srikant, 1994), verificando una confianza mínima, y con un soporte adaptativo que permite que finalmente se obtenga un conjunto cuyo tamaño yazca en un intervalo predefinido. Para generar las recomendaciones, también de forma semejante a Lin et al., para cada usuario se ponderó cada problema no resuelto a través de los siguientes dos pasos: 1) se seleccionaron las reglas (con el problema actual en el consecuente) cuyos antecedentes satisfacen al usuario actual, y 2) se ponderó el problema como la suma de los productos de la confianza y el soporte de las reglas seleccionadas. Finalmente, se presentaron en calidad de recomendaciones al usuario, aquellos problemas con mayor ponderación.

Con vistas a ajustar los parámetros de este método (el posible intervalo para el tamaño del conjunto de reglas, y la confianza mínima) se realizaron varias ejecuciones previas. Por tanto, es importante aclarar que este ajuste de la propuesta de Lin et al. (2002) es únicamente válido para el presente contexto de recomendación. Como resultado, los mejores valores fueron obtenidos para el intervalo [10, 50], y para un valor de confianza mínimo de 0.3. Con esta

configuración, se comparó este enfoque con *extendido+nn*, que fue de forma general el de mejor desempeño en la experimentación anterior, y también contra el método inicialmente presentado (*binario*). La tabla 3.8 presenta los resultados de esta comparación en términos de F1, variando n en el rango [5, 40], siendo este el mismo rango usado en la comparación de la tabla 3.6. Para todos los casos, los valores para el enfoque basado en reglas estuvieron notablemente por debajo de *extendido+nn*, y también por debajo de *binario* en la mayoría de estos. Esta conclusión ratifica el hecho de que los trabajos encontrados en la revisión del estado del arte no son apropiados para la generación de recomendaciones en el escenario de los jueces en línea de programación.

Tabla 3.8. Comparación del método basado en reglas contra los enfoques presentados. Valores de F1

N	***5***	***10***	***15***	***20***	***25***	***30***	***35***	***40***
binario	0,2668	0,3931	0,4538	0,4853	0,4992	0,5053	0,5032	0,4989
extendido+nn	**0,2682**	**0,4024**	**0,4661**	**0,4972**	**0,5122**	**0,5192**	**0,5168**	**0,5107**
basado en reglas	0,2645	0,3806	0,4414	0,4710	0,4893	0,4983	0,4994	0,4992

3.9 Un sistema recomendador para un juez en línea de programación

Los resultados experimentales previamente presentados en este capítulo, sugieren que los métodos propuestos son factibles de aplicar en la construcción de un sistema recomendador de problemas a resolver en un juez en línea de programación. Con este fin, se tomó como punto de partida el núcleo de un juez en línea desarrollado en la Universidad de las Ciencias Informáticas (Junco Vázquez, et al., 2009), para incorporar el enfoque de recomendación presentado.

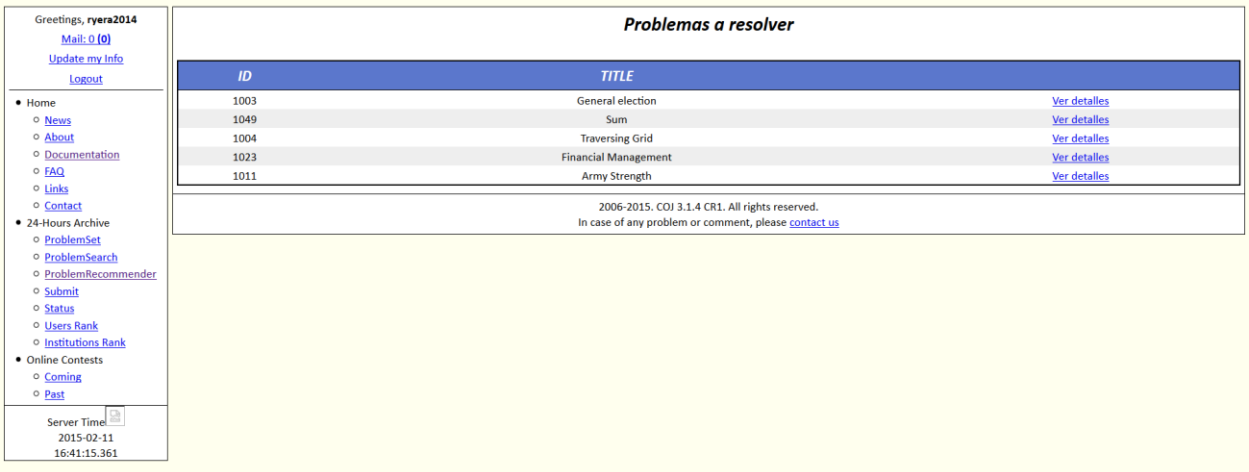


Figura 3.5. Pantalla de presentación de recomendaciones

En este escenario, el funcionamiento del sistema recomendador estaría conformado por tres etapas. La primera está asociada con enriquecer la matriz inicialmente binaria de usuarios-problemas, utilizando la técnica presentada en la sección 3.5. En segundo lugar, corregir las interacciones inconsistentes a través del método presentado en la sección 3.6; y finalmente generar el listado de recomendaciones, según el método presentado en la sección 3.4. Todos estos pasos son en principio transparentes de cara al usuario final, mostrándosele únicamente el listado de recomendaciones generadas, tal y como se presenta en la figura 3.5 a través de la interfaz correspondiente.

Conclusiones del capítulo

En el presente capítulo se utilizó el método general para preprocesamiento de datos en sistemas recomendadores propuesto en el capítulo 2, en la construcción de un sistema para la recomendación de problemas en un juez en línea de programación. El desarrollo del capítulo permitió arribar a las siguientes conclusiones:

-Los jueces en línea de programación son escenarios con una gran sobrecarga de información debido al número de problemas de programación que estos contienen.

-La introducción en este contexto de un enfoque de recomendación que sugiera problemas basándose en el desempeño de los usuarios con conocimientos y aptitudes similares a las del usuario activo, permite reducir eficazmente el efecto negativo de esta sobrecarga de información.

-La incorporación de información específica del juez en línea al enfoque de recomendación creado, implica la obtención de una mayor eficacia en la generación de recomendaciones.

-A su vez, la aplicación de un método de corrección de ruido natural basado en el propuesto en el capítulo 2, también implica una mejora en el desempeño del enfoque de recomendación concebido.

-Finalmente, el enfoque de recomendación concebido presenta un desempeño superior al de un método general de recomendación, identificado en la literatura, y creado específicamente para escenarios de e-learning.

A raíz de lo anterior, el enfoque desarrollado fue utilizado en la construcción de un módulo en calidad de sistema recomendador de problemas a resolver, acoplado a un juez en línea de programación.

CONCLUSIONES

Como resultado de la presente investigación se obtuvo un método general para el tratamiento de información inconsistente, asociada a los usuarios, en sistemas recomendadores de filtrado colaborativo. Tomando como base los objetivos propuestos al inicio de la investigación, se arribó a las siguientes conclusiones.

-El método de preprocesamiento concebido es capaz de eliminar inconsistencias presentes en los ratings de sistemas recomendadores de filtrado colaborativo, al basarse en una caracterización global de los usuarios y los ítems, y procesar como inconsistentes aquellos ratings que no se correspondan con dichas caracterizaciones. Según la literatura consultada, el presente es el único trabajo identificado que corrige ratings inconsistentes sin necesitar información externa tal como atributos de los ítems o información demográfica de los usuarios.

-El método propuesto impacta positivamente en el desempeño de métodos tradicionales de filtrado colaborativo. Esta efectividad se evaluó en tres bases de datos y cinco algoritmos tradicionales de recomendación, arrojando que en más del 95% de los casos la aplicación de la propuesta implicaba una mejora en la generación de recomendaciones. Esta mejora fue ratificada con la aplicación de pruebas estadísticas.

-El método de preprocesamiento obtenido es igualmente eficaz en escenarios más específicos de recomendación como el de un juez en línea de programación. Esto se verificó con la construcción de un enfoque de recomendación para este escenario, basado en filtrado colaborativo básico y el método de preprocesamiento propuesto; y en el que se verificó que la incorporación del preprocesamiento mejora las recomendaciones generadas.

-La construcción de un módulo en calidad de sistema recomendador de problemas a resolver, acoplado a un juez en línea de programación y basado en el enfoque de recomendación concebido, facilita el acceso por parte de los usuarios a los problemas con mayores posibilidades de ser resueltos. Los resultados experimentales obtenidos sirven de sostén a esta afirmación.

RECOMENDACIONES

Entre las posibles direcciones para dar continuidad a la investigación se plantean las siguientes:

-Incorporar la manipulación de la incertidumbre al clasificar los perfiles de usuarios e ítems. La presente propuesta se centra en una clasificación rígida para ambos casos. Sin embargo, la incorporación de técnicas basadas en lógica difusa podría permitir realizar una caracterización más precisa, dando lugar a la obtención de mejores resultados generales.

-Incorporar la dimensión tiempo como un dato a tener en cuenta como parte del preprocesamiento de ratings. Resulta claro el hecho de que las preferencias cambian con el tiempo, por lo que el empleo de esta dimensión podría implicar una detección más precisa de los ratings inconsistentes.

-Proponer métodos que tomen como base la propuesta actual para procesar ratings inconsistentes en escenarios de recomendación distintos del tradicional filtrado colaborativo, tales como los escenarios de recomendación grupal, de recomendación basada en el contenido, o de recomendación basada en el conocimiento. Aunque muchas de las ideas presentadas pueden ser extrapoladas a estos contextos, siempre se requerirá una formalización y evaluación acorde a sus particularidades.

Para el caso particular del recomendador para el juez en línea de programación, varias ideas también se pueden desarrollar para mejorar la propuesta. Específicamente se propone el desarrollo de ideas centradas en mejorar las recomendaciones a través de los datos disponibles, descartándose el empleo de información adicional asociada a los usuarios y problemas, tal como atributos de los problemas, información demográfica, y otras categorías. Las posibles ideas a considerar podrían girar alrededor de los siguientes hechos:

-El trabajo actual sólo considera el éxito o el fracaso en la solución de los problemas, pero no explora las categorías de fracaso (respuesta incorrecta, error en tiempo de ejecución, tiempo límite excedido, entre otras). El procesamiento de esta información podría permitir la extracción de nuevo conocimiento que podría ser útil para la generación de las recomendaciones, y para la detección de comportamientos inconsistentes.

-El juez en línea también asocia etiquetas de tiempo para cada interacción, por lo que el uso conveniente de estas podría mejorar la eficacia de la recomendación.

REFERENCIAS BIBLIOGRÁFICAS

Abbasi, A., Javari, A., Jalili, M., & Rabiee, H. R. (2014). *Enhancing precision of Markov-based recommenders using location information.* In Advances in Computing, Communications and Informatics (ICACCI, 2014) International Conference on, pp. 188-193, Greater Noida, India.

Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering, 17*(6), 734-749.

Agrawal, R., & Srikant, R. (1994). *Fast Algorithms for Mining Association Rules.* In Proceedings of the 20th VLDB Conference, pp. 487-499, Santiago, Chile.

Aher, S. B., & Lobo, L. (2012). Data preparation strategy in e-learning system using association rule algorithm. *International Journal of Computer Applications, Foundation of Computer Science, New York, USA, 41*(3), 35-40.

Aher, S. B., & Lobo, L. (2013). Combination of machine learning algorithms for recommendation of courses in e-learning system based on historical data. *Knowledge-Based Systems, 51*, 1-14.

Aho, A., Hopcrof, J., & Ullman, J. (1988). *Data Structures and Algorithms*. New York, USA: Addison-Wesley.

Alam, S., Dobbie, G., Koh, Y. S., & Riddle, P. (2014). Web usage mining based recommender systems using implicit heterogeneous data: A Particle Swarm Optimization based clustering approach. *Web Intelligence and Agent Systems, 12*(4), 389-409.

Alexandridis, G., Siolas, G., & Stafylopatis, A. (2012). Applying k-separability to collaborative recommender systems. *International Journal on Artificial Intelligence Tools, 21*(1), 1-30.

Ali, S. M., Ghani, I., & Abd Latiff, M. S. (2015). Interaction-based Collaborative Recommendation: A Personalized Learning Environment (PLE) Perspective. *KSII Transactions on Internet and Information Systems, 9*(1), 446-465.

Altuna Castillo, E. J., & Guibert Estrada, L. (2013). Generación de pistas durante el aprendizaje de la programación para concursos usando el análisis estático y dinámico de las soluciones. *Ingeniare. Revista chilena de ingeniería, 21*(2), 205-217.

Amatriain, X., Jaimes, A., Oliver, N., & Pujol, J. M. (2011). Data Mining Methods for Recommender Systems. In F. Ricci, L. Rokach, B. Shapira & P. Kantor (Eds.), *Recommender Systems Handbook* (pp. 39-72). New York: Springer.

Amatriain, X., Pujol, J. M., & Oliver, N. (2009). *I like it... i like it not: Evaluating user ratings noise in recommender systems.* In Proceedings of the 17th International Conference on User Modeling, Adaptation and Personalization (UMAP), pp. 247-258, Trento, Italy.

Amatriain, X., Pujol, J. M., Tintarev, N., & Oliver, N. (2009). *Rate it again: Increasing recommendation accuracy by user re-rating.* In Proceedings of 3rd ACM International Conference on Recommender Systems, pp. 173–180, New York, USA.

Anderson, C. (2004). The long tail. *Wired, 12*(10), 170-177.

Arefin, A. S. (2006). *Art of Programming Contest*. Bangladesh: Gyankosh Prokashoni.

Balabanovic, M. (1998). *Learning to surf: multiagent systems for adaptive web page recommendation.* Ph.D. Thesis, Stanford University.

Baltrunas, L., & Ricci, F. (2008). *Locally adaptive neighborhood selection for collaborative filtering recommendations.* In Proceedings of the 5th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, pp. 22-31, Hannover, Germany.

Bellogín, A., Cantador, I., Díez, F., Castells, P., & Chavarriaga, E. (2013). An empirical comparison of social, collaborative filtering, and hybrid recommenders. *ACM Transactions on Intelligent Systems and Technology (TIST), 4*(1), 14.

Bellogín, A., Said, A., & de Vries, A. P. (2014). The Magic Barrier of Recommender Systems–No Magic, Just Ratings. *User Modeling, Adaptation, and Personalization* (pp. 25-36): Springer.

Bernardes, D., Diaby, M., Fournier, R., FogelmanSoulié, F., & Viennet, E. (2015). A Social Formalism and Survey for Recommender Systems. *ACM SIGKDD Explorations Newsletter, 16*(2), 20-37.

Bhagat, S., Weinsberg, U., Ioannidis, S., & Taft, N. (2014). *Recommending with an agenda: Active learning of private attributes using matrix factorization.* In Proceedings of the 8th ACM conference on recommender systems, pp. 65-72, Foster City, Silicon Valley, CA, USA.

Bhaumik, R., Mobasher, B., & Burke, R. D. (2011). *A clustering approach to unsupervised attack detection in collaborative recommender systems.* In Proceedings of the 7th IEEE international conference on data mining, pp. 181-187, Las Vegas, NV, USA.

Bilge, A., & Polat, H. (2012). An improved privacy-preserving DWT-based collaborative filtering scheme. *Expert Systems with Applications, 39*(3), 3841-3854.

Billsus, D., & Pazzani, M. (1998). *Learning Collaborative Information Filters.* In Proceedings of International Conference on Machine Learning (ICML), pp. 46-54, Madison, Wisconsin, USA.

Bobadilla, J., Ortega, F., Hernando, A., & Bernal, J. (2012). A collaborative filtering approach to mitigate the new user cold start problem. *Knowledge-Based Systems, 26*, 225-238.

Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-Based Systems, 46*, 109-132.

Breese, J., Heckerman, D., & Kadie, C. (1998). *Empirical analysis of predictive algorithms for collaborative filtering.* In Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI '98), pp. 43–52, San Francisco, CA, USA.

Burke, R. (2002). Hybrid recommender systems:survey and experiments. *User Modeling and User-Adapted Interaction, 12*(4), 331-370.

Burke, R., Mobasher, B., Williams, C. A., & Bhaumik, R. (2006a). *Classification features for attack detection in collaborative recommender systems.* In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 542-547, Philadelphia, PA, USA.

Burke, R., Mobasher, B., Williams, C. A., & Bhaumik, R. (2006b). *Detecting profile injection attacks in collaborative recommender systems.* In Proceedings of the 8th IEEE conference on e-commerce technology, pp. 23-30, San Francisco, CA, USA.

Burke, R., O'Mahony, M. P., & Hurley, N. J. (2011). Robust Collaborative Recommendation. In F. Ricci, L. Rokach, B. Shapira & P. Kantor (Eds.), *Recommender Systems Handbook* (pp. 805–835). New York: Springer.

Cacheda, F., Carneiro, V., Fernández, D., & Formoso, V. (2011). Comparison of Collaborative Filtering Algorithms: Limitations of Current Techniques and Proposals for Scalable, High-Performance Recommender Systems. *ACM Transactions on the Web, 5*(1), 1-33.

Cantador, I., Bellogín, A., & Vallet, D. (2010). *Content-based recommendation in social tagging systems.* In Proceedings of the 4th ACM Conference on Recommender systems, pp. 237-240, Barcelona, Spain.

Cantador, I., Castells, P., & Bellogín, A. (2011). An enhanced semantic layer for hybrid recommender systems: Application to news recommendation. *International Journal on Semantic Web and Information Systems, 7*(1), 44-78.

Cantador, I., Fernández-Tobías, I., Bellogín, A., Kosinski, M., & Stillwell, D. (2013). *Relating Personality Types with User Preferences in Multiple Entertainment Domains.* In Proceedings of 1st Workshop on Emotions and Personality in Personalized Services in conjunction with 21st conference on User Modeling, Adaptation, and Personalization (EMPIRE at UMAP), pp. 13-28, Rome, Italy.

Cappella, J. N., Yang, S., & Lee, S. (2015). Constructing Recommendation Systems for Effective Health Messages Using Content, Collaborative, and Hybrid Algorithms. *The ANNALS of the American Academy of Political and Social Science, 659*(1), 290-306.

Contreras, D., Salamó, M., & Pascual, J. (2015). A Web-Based Environment to Support Online and Collaborative Group Recommendation Scenarios. *Applied Artificial Intelligence, 29*(5), 480-499.

Cha, S.-H. (2007). Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions. *International Journal of Mathematical Models and Methods in Applied Sciences, 4*(1), 300-307.

Chen, W., Niu, Z., Zhao, X., & Li, Y. (2014). A hybrid recommendation algorithm adapted in e-learning environments. *World Wide Web, 17*(2), 271-284.

Chirita, P. A., Nejdl, W., & Zamfir, C. (2005). *Preventing shilling attacks in online recommender systems.* In Proceeding of the 7th ACM international workshop on web information and data management, pp. 67-74, Bremen, Germany.

Choi, K., & Suh, Y. (2013). A new similarity function for selecting neighbors for each target item in collaborative filtering. *Knowledge-Based Systems, 37*, 146-153.

Choi, K., Yoo, D., Kim, G., & Suh, Y. (2012). A hybrid online-product recommendation system: Combining implicit rating-based collaborative filtering and sequential pattern analysis. *Electronic Commerce Research and Applications, 11*(4), 309-317.

Chughtai, W. M., Selama, A. B., & Ghani, I. (2013). Short systematic review on e-learning recommender systems. *Journal of Theoretical & Applied Information Technology, 57*(2), 139-148.

Chung, C.-Y., Hsu, P.-Y., & Huang, S.-H. (2013). βP: A novel approach to filter out malicious rating profiles from recommender systems. *Decision Support Systems, 55*(1), 314-325.

De-La-Fuente-Valentín, L., Pardo, A., & Delgado Kloos, C. (2013). Addressing drop-out and sustained effort issues with large practical groups using an automated delivery and assessment system. *Computers & Education, 61*, 33-42.

De Oliveira, M. G., Marques Ciarelli, P., & Oliveira, E. (2013). Recommendation of programming activities by multi-label classification for a formative assessment of students. *Expert Systems with Applications, 40*(16), 6641-6651.

Deshpande, M., & Karypis, G. (2004). Item-based top-N recommendation algorithms. *ACM Transactions on Information Systems, 22*(1), 143-177.

Desrosier, C., & Karypis, G. (2011). A Comprehensive Survey of Neighborhood-based Recommendation Methods. In F. Ricci, L. Rokach, B. Shapira & P. Kantor (Eds.), *Recommender Systems Handbook* (pp. 107-145). New York: Springer.

Di Noia, T., Ostuni, V. C., Rosati, J., Tomeo, P., & Di Sciascio, E. (2014). *An analysis of users' propensity toward diversity in recommendations.* In Proceedings of the 8th ACM Conference on Recommender systems, pp. 285-288, Foster City, Silicon Valley, CA.

Dooms, S., De Pessemier, T., & Martens, L. (2013). *MovieTweetings: a Movie Rating Dataset Collected From Twitter.* In Workshop on Crowdsourcing and Human Computation for Recommender Systems (CrowdRec at RecSys), pp. 43-44, Hong Kong.

Dunn, G., & Everitt, B. S. (1982). *An introduction to mathematical taxonomy*. New York, USA: Cambridge University Press.

Ekstrand, M. (2014). *Towards Recommender Engineering: Tools and Experiments for Identifying Recommender Differences.* Ph.D. Thesis, University of Minnesota.

Ekstrand, M., Riedl, J., & Konstan, J. A. (2010). Collaborative Filtering Recommender Systems. *Foundations and Trends in Human–Computer Interaction, 4*(2), 81-173.

Famili, F., Shen, W.-M., Weber, R., & Simoudis, E. (1997). Data pre-processing and intelligent data analysis. *Intelligent Data Analysis, 1*(1), 3-23.

Fernández Alemán, J. L. (2011). Automated assessment in a programming tools course. *IEEE Transactions on Education, 54*(4), 576-581.

Formoso López, V. (2013). *Técnicas eficientes para la recomendación de productos basadas en filtrado colaborativo.* Ph.D. Thesis, Universidad de A Coruña, A Coruña.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association, 32*(200), 675-701.

Gantner, Z., Rendle, S., Freudenthaler, C., & Schmidt-Thieme, L. (2011). *MyMediaLite: A free recommender system library.* In Proceedings of the 5th ACM conference on Recommender systems, pp. 305-308, Chicago, Illinois, USA.

García, E., Romero, C., Ventura, S., & De Castro, C. (2009). An architecture for making recommendations to courseware authors using association rule mining and collaborative filtering. *User Modeling and User-Adapted Interaction, 19*(1-2), 99-132.

García, S., Fernández, A., Luengo, J., & Herrera, F. (2010). Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences, 180*(10), 2044-2064.

Georgouli, K., & Guerreiro, P. (2011). Integrating an Automatic Judge into an Open Source LMS. *International Journal on E-Learning, 10*(1), 27-42.

Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM, 35*(12), 61-70.

Gunawardana, A., & Shani, G. (2009). A Survey of Accuracy Evaluation Metrics of Recommendation Tasks. *Journal of Machine Learning Research, 10*, 2935-2962.

Gunes, I., Kaleli, C., Bilge, A., & Polat, H. (2014). Shilling attacks against recommender systems: a comprehensive survey. *Artificial Intelligence Review, 42*(4), 767-799.

Guo, S. (2011). *Bayesian Recommender Systems: Models and Algorithms.* Ph.D. Thesis, The Australian National University.

Han, J., & Kamber, M. (2001). *Data mining: Concepts and Techniques*. San Francisco: Morgan Kaufmann.

He, F., Wang, X., & Liu, B. (2010). *Attack detection by rough set theory in recommendation system.* In Proceedings of the IEEE international conference on granular computing, pp. 692-695, San Jose, CA, USA.

Herlocker, J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS), 22*(1), 5-53.

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics, 6*, 65-70.

Howe, A. E., & Forbes, R. D. (2008). *Re-considering neighborhood-based collaborative filtering parameters in the context of new data.* In Proceeding of the 17th ACM conference on Information and knowledge management., pp. 1481–1482, New York, NY, USA.

Hsiao, I.-H., Sosnovsky, S., & Brusilovsky, P. (2010). Guiding students to the right questions: adaptive navigation support in an E-Learning system for Java programming. *Journal of Computer Assisted Learning, 26*(4), 270-283.

Hsieh, T.-C., Lee, M.-C., & Su, C.-Y. (2013). Designing and implementing a personalized remedial learning system for enhancing the programming learning. *Educational Technology & Society, 16*(4), 32-46.

Hu, R., & Pu, P. (2013). *Exploring Relations between Personality and User Rating Behaviors.* In Proceedings of 1st Workshop on Emotions and Personality in Personalized Services in conjunction with 21st conference on User Modeling, Adaptation, and Personalization (EMPIRE at UMAP), pp. 28-40, Rome, Italy.

Hu, Y. (2014). *A model-based music recommendation system for individual users and implicit user groups.* Ph.D. Thesis, University of Miami.

Huete, J. F., Fernández-Luna, J. M., de Campos, L. M., & Rueda-Morales, M. A. (2012). Using past-prediction accuracy in recommender systems. *Information Sciences, 199*, 78-92.

Hurley, N. J., Cheng, Z., & Zhang, M. (2009). *Statistical attack detection.* In Proceedings of the 3rd ACM Internacional Conference on Recommender Systems, pp. 149-156, New York, USA.

Hwang, D., Pham, X. H., & Jung, J. J. (2011). *Preference-based user rate correction process for interactive recommendation systems.* In Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services, pp. 412-415, Ho Chi Minh City, Vietnam.

Javari, A., & Jalili, M. (2015). A probabilistic model to resolve diversity–accuracy challenge of recommendation systems. *Knowledge and Information Systems, 44*(3), 609-627.

Jeong, W.-H., Kim, S.-J., Park, D.-S., & Kwak, J. (2013). Performance Improvement of a Movie Recommendation System based on Personal Propensity and Secure Collaborative Filtering. *Journal of Information Processing Systems, 9*(1), 157-172.

Junco Vázquez, T. O., Altuna Castillo, E. J., & Soria Ramirez, J. A. (2009). *El Jurado Online, una nueva forma de ejercitación y evaluación en las asignaturas de programación.* In XIII Congreso Internacional de Informática en la Educación, pp. 1-10, La Habana, Cuba.

Kiran, R. U., & Kitsuregawa, M. (2013). *An improved neighborhood-restricted association rule-based recommender system.* In Proceedings of the Twenty-Fourth Australasian Database Conference, pp. 43-50, Adelaide, Australia.

Klašnja-Milićević, A., Vesin, B., Ivanović, M., & Budimac, Z. (2011). E-Learning personalization based on hybrid recommendation strategy and learning style identification. *Computers & Education, 56*(3), 885-899.

Kluver, D., Nguyen, T. T., Ekstrand, M., Sen, S., & Riedl, J. (2012). *How many bits per rating?* In Proceedings of the 6th ACM conference on Recommender systems, pp. 99-106, Dublin, Ireland.

Knijnenburg, B. P., Willemsen, M. C., Gantner, Z., Soncu, H., & Newell, C. (2012). Explaining the user experience of recommender systems. *User Modeling and User-Adapted Interaction, 22*(4-5), 441-504.

Kohavi, R., Deng, A., Longbotham, R., & Xu, Y. (2014). *Seven rules of thumb for web site experimenters.* In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 1857-1866, New York, USA.

Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R., & Riedl, J. (1997). GroupLens: applying collaborative filtering to usenet news. *Communications of the ACM, 40*(3), 77-87.

Konstan, J. A., & Riedl, J. (2012). Recommender systems: from algorithms to user experience. *User Modeling and User-Adapted Interaction, 22*, 101-123.

Koren, Y. (2010). Factor in the Neighbors: Scalable and Accurate Collaborative Filtering. *ACM Transactions on Knowledge Discovery from Data, 4*, 1-4.

Koren, Y., Bell, R. M., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *IEEE Computer, 42*(8), 30-37.

Kosowski, A., Małafiejski, M., & Noiński, T. (2007). *Application of an online judge & contester system in academic tuition.* In Proceedings of 6th International Conference on Web Based Learning, pp. 343-354, Edinburgh, UK.

Kurnia, A., Lim, A., & Cheang, B. (2001). Online judge. *Computers & Education, 36*(4), 299-315.

Leal, J. P., & Silva, F. (2003). Mooshak: a Web-based multi-site programming contest system. *Software: Practice and Experience, 33*(6), 567-581.

Lee, J., Sun, M., & Lebanon, G. (2012a). A Comparative Study of Collaborative Filtering Algorithms. *arXiv:1205.3193v1*.

Lee, J., Sun, M., & Lebanon, G. (2012b). Prea: Personalized recommendation algorithms toolkit. *The Journal of Machine Learning Research, 13*(1), 2699-2703.

Lemire, D., & Maclachlan, A. (2005). *Slope one predictors for online rating-based collaborative filtering.* In Proceedings of the SIAM Data Mining Conference (SDM '05), pp. 471-475, Newport Beach, California, USA,.

Li, B., Chen, L., Xingquan, Z., & Chengqi, Z. (2013). Noisy but non-malicious user detection in social recommender systems. *World Wide Web, 16*(5-6), 677-699.

Li, C., & Luo, Z. (2011). *Detection of shilling attacks in collaborative filtering recommender systems.* In Proceedings of the International Conference of SoftComputing and Pattern Recognition, pp. 190-193, Dalian, China.

Li, M., Yuan, M., & Xu, Y. (2015). An approach to task-oriented knowledge recommendation based on multi-granularity fuzzy linguistic method. *Kybernetes, 44*(3), 460-474.

Lin, W., Alvarez, S., & Ruiz, C. (2002). Efficient adaptive-support association rule mining for recommender systems. *Data Mining and Knowledge Discovery, 6*, 83-105.

Linden, G., Smith, B., & York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing, 7*(1), 76-80.

Liu, Q., Chen, E., Xiong, H., Ding, C. H., & Chen, J. (2012). Enhancing collaborative filtering by user interest expansion via personalized ranking. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 42*(1), 218-233.

Liu, X. (2013). A new automated grading approach for computer programming. *Computer Applications in Engineering Education, 21*(3), 484-490.

Lops, P., de Gemmis, M., & Semeraro, G. (2011). Content-based recommender systems: state of the art and trends In F. Ricci, L. Rokach, B. Shapira & P. Kantor (Eds.), *Recommender Systems Handbook* (pp. 73-100). New York: Springer.

Lu, J., Wu, D., Mao, M., Wang, W., & Zhang, G. (2015). Recommender system application developments: A survey. *Decision Support Systems, 74*, 12-32.

Lü, L., Medo, M., Yeung, C. H., Zhang, Y.-C., Zhang, Z.-K., & Zhou, T. (2012). Recommender systems. *Physics Reports, 519*(1), 1-49.

Lucas, J. P., Laurent, A., Moreno, M. N., & Teisseire, M. (2012). A fuzzy associative classification approach for recommender systems. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 20*(04), 579-617.

Llana, L., Martin-Martin, E., Pareja-Flores, C., & Velázquez-Iturbide, J. Á. (2014). FLOP: A User-Friendly System for Automated Program Assessment. *Journal of Universal Computer Science, 20*(9), 1304-1326.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. Cambridge, UK: Cambridge University Press

Manouselis, N., Drachsler, H., Vuorikari, R., Hummel, H., & Koper, R. (2011). Recommender systems in technology enhanced learning. In F. Ricci, L. Rokach, B. Shapira & P. Kantor (Eds.), *Recommender Systems Handbook* (pp. 387-415). New York: Springer.

Manouselis, N., Kyrgiazos, G., & Stoitsis, G. (2014). Exploratory study of multi-criteria recommendation algorithms over technology enhanced learning datasets. *Journal of e-Learning and Knowledge Society, 10*(1), 33-49.

Manouselis, N., Vuorikari, R., & Van Assche, F. (2010). Collaborative recommendation of e-learning resources: an experimental investigation. *Journal of Computer Assisted Learning, 26*(4), 227-242.

Martinez, L., Rodriguez, R. M., & Espinilla, M. (2009). *Reja: A georeferenced hybrid recommender system for restaurants.* In Proceedings of IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technologies, pp. 187-190, Milano, Italy.

Mazurowski, M. A. (2013). Estimating confidence of individual rating predictions in collaborative filtering recommender systems. *Expert Systems with Applications, 40*(10), 3847-3857.

Mehta, B. (2007). *Unsupervised shilling detection for collaborative filtering.* In Proceedings of the 22nd international conference on artificial intelligence, pp. 1402–1407, Vancouver, BC, Canada.

Mehta, B., & Hofmann, T. (2008). A survey of attack-resistant collaborative filtering algorithms. *Bulletin of the Technical Committee on Data Engineering, 31*(2), 14-22.

Mehta, B., & Nejdl, W. (2009). Unsupervised strategies for shilling detection and robust collaborative filtering. *User Modeling and User-Adapted Interaction, 19*(1-2), 65–97.

Miyahara, K., & Pazzani, M. J. (2002). Improvement of collaborative filtering with the simple Bayesian classifier. *IPSJ Journal, 43*(11), 3429-3437.

Mobasher, B., Burke, R. D., Bhaumik, R., & Williams, C. A. (2007). Towards trustworthy recommender systems: an analysis of attack models and algorithm robustness. *ACM Transactions on Internet Technology, 7*(4), 23-60.

Nakamura, A., & Abe, N. (1998). *Collaborative filtering using weighted majority prediction algorithms.* In Proceedings of 15th International Conference on Machine Learning pp. 395–403, San Francisco, CA, USA.

Nguyen, N. T. (2007). *Advanced methods for inconsistent knowledge management*. London: Springer.

O'Mahony, M. P., Hurley, N. J., & Silvestre, G. C. M. (2002). Promoting recommendations: An attack on collaborative filtering. *Lecture Notes on Computer Sciences, 2453*, 494-503.

O'Mahony, M. P., Hurley, N. J., & Silvestre, G. C. M. (2003). Collaborative filtering-safe and sound. *Lecture Notes on Computer Science, 2871*, 506-510.

O'Mahony, M. P., Hurley, N. J., & Silvestre, G. C. M. (2006). *Detecting noise in recommender system databases.* In Proceedings of the 11th international conference on Intelligent user interfaces, pp. 109-115, Sydney, Australia.

Park, S.-T., Pennock, D., Madani, O., Good, N., & DeCoste, D. (2006). *Naïve filterbots for robust cold-start recommendations.* In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 699-705, New York, NY, USA.

Parra, D. (2013). *User controllability in a hybrid recommender systems.* Ph.D. Thesis, University of Pittsburgh.

Pazzani, M. J. (1999). A framework for collaborative, content-based, and demographic filtering. *Artificial Intelligence Review, 13*(5-6), 393-408.

Pazzani, M. J., & Billsus, D. (2007). Content-Based Recommendation Systems. In P. Brusilovsky, A. Kobsa & W. Nejdl (Eds.), *The adaptive web.* (pp. 325–341). Berlin-Heidelberg: Springer.

Petit, J., Giménez, O., & Roura, S. (2012). *Jutge.org: an educational programming judge.* In Proceedings of the 43rd ACM technical symposium on Computer Science Education, pp. 445-450, Raleigh, NC, USA.

Pham, H. X., & Jung, J. J. (2013). Preference-based user rating correction process for interactive recommendation systems. *Multimedia Tools and Applications, 65*(1), 119-132.

Pham, H. X., Jung, J. J., & Nguyen, N. T. (2012). Integrating multiple experts for correction process in interactive recommendation systems *Computational Collective Intelligence. Technologies and Applications* (pp. 31-40): Springer.

Pham, H. X., Jung, J. J., & Nguyen, N. T. (2013). Integrating Multiple Experts for Correction Process in Interactive Recommendation Systems. *Journal of Universal Computer Science, 19*(4), 581-599.

Pham, M. C., Cao, Y., Klamma, R. K., & Jarke, M. (2011). A clustering approach for collaborative filtering recommendation using social network analysis. *Journal of Universal Computer Science, 17*(4), 583-604.

Pilászy, I., & Tikk, D. (2009). *Recommending new movies: even a few ratings are more valuable than metadata.* In Proceedings of the 3rd ACM conference on Recommender Systems, pp. 93-100, New York, NY, USA.

Porcel, C., Tejeda-Lorente, A., Martínez, M. A., & Herrera-Viedma, E. (2012). A hybrid recommender system for the selective dissemination of research resources in a technology transfer office. *Information Sciences, 184*(1), 1-19.

Rafaeli, S., Dan-Gur, Y., & Barak, M. (2005). Social recommender systems: recommendations in support of e-learning. *International Journal of Distance Education Technologies, 3*(2), 30-47.

Rapečka, A., & Dzemyda, G. (2015). A New Recommendation Model for the User Clustering-Based Recommendation System. *Information Technology and Control, 44*(1), 54-63.

Regueras, L. M., Verdú, E., Muñoz, M. F., Pérez, M. A., de Castro, J. P., & Verdú, M. J. (2009). Effects of competitive e-learning tools on higher education students: a case study. *IEEE Transactions on Education, 52*(2), 279-285.

Resnick, P., Iacovu, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994). *GroupLens: An Open Architecture for Collaborative Filtering of Netnews.* In Proceedings of ACM Conference on Computer Supported Cooperative Work, pp. 175-186, Chapel Hill, North Carolina, USA.

Revilla, M. A., Manzoor, S., & Liu, R. (2008). Competitive learning in informatics: The UVa online judge experience. *Olympiads in Informatics, 2*, 131-148.

Ribeiro, P., & Guerreiro, P. (2008). Early introduction of competitive programming. *Olympiads in Informatics, 2*, 149-162.

Ricci, F., Rokach, L., Shapira, B., & Kantor, P. (2011). *Recommender Systems Handbook*. New York: Springer.

Rodríguez, R. M., Espinilla, M., Sánchez, P. J., & Martínez-López, L. (2010). Using linguistic incomplete preference relations to cold start recommendations. *Internet Research, 20*(3), 296-315.

Romero, C., Romero, J. R., & Ventura, S. (2014). A Survey on Pre-Processing Educational Data. In A. Peña-Ayala (Ed.), *Educational Data Mining* (pp. 29-64). Switzerland: Springer International Publishing.

Romero, C., & Ventura, S. (2010). Educational data mining: a review of the state of the art. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 40*(6), 601-618.

Romero, C., Ventura, S., & De Bra, P. (2004). Knowledge discovery with genetic programming for providing feedback to courseware authors. *User Modeling and User-Adapted Interaction, 14*(5), 425-464.

Ruiz-Iniesta, A., Jimenez-Diaz, G., & Gómez-Albarrán, M. (2014). A Semantically Enriched Context-Aware OER Recommendation Strategy and Its Application to a Computer Science OER Repository. *IEEE Transactions on Education, 57*(4), 255-260.

Said, A., Jain, B. J., Narr, S., & Plumbaum, T. (2012). *Users and Noise: The Magic Barrier of Recommender Systems.* In Proceedings of the 21st International Conference on User Modeling, Adaptation and Personalization (UMAP '12), pp. 237-248, Montreal, Canada.

Said, A., Jain, B. J., Narr, S., Plumbaum, T., Albayrak, S., & Scheel, C. (2012). *Estimating the Magic Barrier of Recommender Systems: A User Study.* In Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1061-1062, Portland, OR, USA.

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2000a). *Analysis of recommendation algorithms for e-commerce.* In Proceedings of the 2nd ACM conference on Electronic commerce, pp. 158-167, Minneapolis, Minnesota, USA.

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2000b). *Application of dimensionality reduction in recommender systems. A case study.* In Proceedings of Web Mining for E-Commerce Workshop at 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1-12, Boston, Massachusetts, USA.

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2002). *Recommender systems for large-scale E-commerce: scalable neighborhood formation using clustering.* In Proceedings of the 5th International Conference on Computer and Information Technology (ICCIT '02), pp. 1-6.

Sarwar, B., Karypis, G., Konstan, J. A., & Riedl, J. (2001). *Item-based collaborative filtering recommendation algorithms.* In Proceedings of the 10th International Conference on World Wide Web (WWW '01), pp. 285-295, Hong Kong.

Schafer, J. B., Frankowski, D., Herlocker, J., & Sen, S. (2007). Collaborative filtering recommender systems. In P. Brusilovsky, A. Kobsa & W. Nejdl (Eds.), *The adaptive web* (pp. 291–324). Berlin-Heidelberg: Springer.

Schelter, S., Boden, C., & Markl, V. (2012). *Scalable similarity-based neighborhood methods with mapreduce.* In Proceedings of the 6th ACM Conference on Recommender Systems, pp. 163-170, Dublin, Ireland.

Schwartz, B. (2004a). *The paradox of choice: Why less is more*. New York, USA: ECCO.

Schwartz, B. (2004b). The tyranny of choice. *Scientific American, 290*, 70-75.

Shani, G., & Gunawardana, A. (2011). Evaluating recommendation systems. In F. Ricci, L. Rokach, B. Shapira & P. Kantor (Eds.), *Recommender Systems Handbook* (pp. 257-297). New York: Springer.

Shani, G., & Gunawardana, A. (2013). Tutorial on application-oriented evaluation of recommendation systems. *AI Communications, 26*(2), 225-236.

Shani, G., Heckerman, D., & Brafman, R. I. (2005). An MDP-based recommender system. *Journal of Machine Learning Research, 6*, 1265-1295.

Skiena, S. S., & Revilla, M. A. (2006). *Programming challenges: The programming contest training manual*. New York: Springer.

Son, L. H. (2015). HU-FCF++: A novel hybrid method for the new user cold-start problem in recommender systems. *Engineering Applications of Artificial Intelligence, 41*, 207-222.

Su, X., Greiner, R., Khoshgoftaar, T. M., & Zhu, X. (2007). *Hybrid collaborative filtering algorithms using a mixture of experts.* In Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI '07), pp. 645-649, Silicon Valley, Calif, USA.

Su, X., & Khoshgoftaar, T. M. (2009). A Survey of Collaborative Filtering Techniques. *Advances in Artificial Intelligence, 2009*, 1-19.

Tavakol, M., & Brefeld, U. (2014). *Factored MDPs for detecting topics of user sessions.* In Proceedings of the 8th ACM Conference on Recommender systems, pp. 33-40, Foster City, Silicon Valley, CA.

Tkalčič, M., Burnik, U., & Košir, A. (2010). Using affective parameters in a content-based recommender system for images. *User Modeling and User-Adapted Interaction, 20*(4), 279-311.

Tyagi, S., & Bharadwaj, K. K. (2013). Enhancing collaborative filtering recommendations by utilizing multi-objective particle swarm optimization embedded association rule mining. *Swarm and Evolutionary Computation, 13*, 1-12.

Vargas, S., Castells, P., & Vallet, D. (2012). *Explicit relevance models in intent-oriented information retrieval diversification.* In Proceedings of the 35th international ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 75-84, Portland, OR, USA.

Verbert, K., Manouselis, N., Drachsler, H., & Duval, E. (2012). Dataset-Driven Research to Support Learning and Knowledge Analytics. *Educational Technology & Society, 15*(3), 133-148.

Verbert, K., Manouselis, N., Ochoa, X., Wolpers, M., Drachsler, H., Bosnic, I., et al. (2012). Context-aware recommender systems for learning: a survey and future challenges. *IEEE Transactions on Learning Technologies, 5*(4), 318-335.

Verdú, E., Regueras, L. M., Verdú, M. J., Leal, J. P., de Castro, J. P., & Queirós, R. (2012). A distributed system for learning programming on-line. *Computers & Education, 58*(1), 1-10.

Vesin, B., Ivanović, M., Klašnja-Milićević, A., & Budimac, Z. (2012). Protus 2.0: Ontology-based semantic recommendation in programming tutoring system. *Expert Systems with Applications, 39*(15), 12229-12246.

Vesin, B., Klašnja-Milićević, A., Ivanović, M., & Budimac, Z. (2013). Applying Recommender Systems and Adaptive Hypermedia for e-Learning Personalization. *Computing and Informatics, 32*(3), 629-659.

Vozalis, M. G., & Margaritis, K. G. (2007). Using SVD and demographic data for the enhancement of generalized collaborative filtering. *Information Sciences, 177*, 3017-3037.

Vucetic, S., & Obradovic, Z. (2005). Collaborative filtering using a regression-based approach. *Knowledge and Information Systems, 7*(1), 1-22.

Wang, P.-Y., & Yang, H.-C. (2012). Using collaborative filtering to support college students' use of online forum for English learning. *Computers & Education, 59*(2), 628-637.

Wang, W.-S., Lee, H.-J., Kim, S.-W., Youngjoon, W., & Lee, M.-S. (2015). Efficient recommendation methods using category experts for a large dataset. *Information Fusion, in press*, doi: 10.1016/j.inffus.2015.1007.1005.

Williams, C. A., Mobasher, B., & Burke, R. D. (2007). Defending recommender systems: detection of profile injection attacks. *Service Oriented Computing and Applications, 1*(3), 157–170.

Winoto, P., Tang, T. Y., & McCalla, G. I. (2012). Contexts in a paper recommendation system with collaborative filtering. *The International Review of Research in Open and Distance Learning, 13*(5), 56-75.

Wu, I., & Niu, Y. F. (2015). Effects of anchoring process under preference stabilities for interactive movie recommendations. *Journal of the Association for Information Science and Technology, 66*(8), 1673-1695.

Xin, X. (2011). *Effective Fusion-based Approaches for Recommender Systems.* Ph.D. Thesis, The Chinese University of Hong Kong.

Yeh, D.-Y., & Cheng, C.-H. (2015). Recommendation system for popular tourist attractions in Taiwan using Delphi panel and repertory grid techniques. *Tourism Management, 46*, 164-176.

Yera Toledo, R., & Caballero Mota, Y. (2014). An e-Learning Collaborative Filtering Approach to Suggest Problems to Solve in Programming Online Judges. *International Journal of Distance Education Technologies, 12*(2), 51-65.

Yera Toledo, R., Caballero Mota, Y., & Martínez, L. (2015). Correcting noisy ratings in collaborative recommender systems. *Knowledge-Based Systems, 76*, 96-108.

Zaïane, O. R. (2002). *Building a recommender agent for e-learning systems.* In Proceedings of International Conference on Computers in Education, pp. 55-59, Auckland, New Zealand.

Zhang, C.-J., & Zeng, A. (2012). Behavior patterns of online users and the effect on information filtering. *Physica A: Statistical Mechanics and its Applications, 391*(4), 1822-1830.

Zhang, F., & Zhou, Q. (2014). HHT–SVM: An online method for detecting profile injection attacks in collaborative recommender systems. *Knowledge-Based Systems, 65*, 96-105.

Zhang, S., Ouyang, Y., Ford, J., & Makedon, F. (2006). *Analysis of a low-dimensional linear model under recommendation attacks.* In Proceedings of the 29th ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 517-524, Seattle, WA, USA.

Zhao, X., Niu, Z., & Chen, W. (2013). Interest before liking: Two-step recommendation approaches. *Knowledge-Based Systems, 48*, 46-56.

Zheng, X.-w., Ma, H.-w., & Li, Y. (2013). An Instructional and Collaborative Learning System with Content Recommendation. *International Journal of Distance Education Technologies, 11*(3), 109-121.

Zhong, J., & Li, X. (2010). Unified collaborative filtering model based on combination of latent features. *Expert Systems with Applications, 37*, 5666-5672.

Zhou, W., & Zhang, H. (2015). Correlation range query for effective recommendations. *World Wide Web, 18*(3), 709-729.

Zhu, G., & Chen, Y. (2013). Knowledge-based Links for Automatic Interaction with Programming Online Judges. *Journal of Software, 8*(5), 1209-1218.

Zhu, X., & Wu, X. (2004). Class Noise vs. Attribute Noise: A Quantitative Study of Their Impacts. *Artificial Intelligence Review, 22*, 177–210.

PRODUCCIÓN CIENTÍFICA DEL AUTOR SOBRE EL TEMA DE LA TESIS

Publicaciones en revistas, libros, y memorias de eventos:

1. Yera Toledo, R., Caballero Mota, Y., Martinez, L. "Correcting noisy ratings in collaborative recommender systems," **Knowledge-Based Systems**, vol. 76, pp. 96-108, 2015. ISSN 0950-7051. Indizada en: Science Citation Index (Thomson Reuters Web of Science), Scopus. **Factor de impacto: 3.058.**
2. Yera Toledo, R., Caballero Mota, Y. "An e-Learning Collaborative Filtering Approach to Suggest Problems to Solve in Programming Online Judges," **International Journal of Distance Education Technologies**, vol. 12, pp. 51-65, 2014. ISSN 1539-3100. Indizada en: Scopus, ACM Digital Library, Ei Compendex, INSPEC, LISA, DBLP, EBSCOhost's Computer Science Index.
3. Yera Toledo, R., Caballero Mota, Y., García, M. "A Regularity-Based Preprocessing Method for Collaborative Recommender Systems," **Journal of Information Processing Systems**, vol. 9, pp. 435-460, 2013. ISSN 1976-913X. Indizada en: Scopus, Compendex, DBLP.
4. Yera Toledo, R., Martinez, L., Caballero Mota, Y. "Managing natural noise in collaborative recommender systems," **Proceedings of the 2013 Joint IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS)**, Edmonton, Canada, 2013, pp. 872-877. ISBN 978-1-4799-0347-4. Indizada en: Scopus, Compendex, IEEE Xplore.
5. Yera Toledo, R., Caballero Mota, Y., Martinez, L., “Corrigiendo preferencias inconsistentes en sistemas recomendadores de filtrado colaborativo”, Proceedings of 2nd International Conference on Computer Science and Informatics (CICCI 2013), La Habana, Cuba, 2013. ISBN 978-959-7213-02-4.
6. Yera Toledo, R y otros, “RECJUDGE: un sistema de filtrado colaborativo de apoyo a las asignaturas de programación”, Memorias del 2do Congreso Internacional “TICs y educación: relato de experiencias”, Toluca, México, 2012, ISBN 978-607-422-344-6, pp. 337-351.
7. Yera Toledo, R., Caballero Mota, Y., Martinez, L., “Un enfoque para la corrección de ruido natural en sistemas recomendadores de filtrado colaborativo”, Memorias del X Congreso Nacional de Reconocimiento de Patrones (RECPAT 2012), Camagüey, Cuba, 2012. ISBN 978-959-16-2065-1.

8. Yera Toledo, R., Caballero Mota, Y., “Corrigiendo preferencias en un sistema de filtrado colaborativo para aumentar la eficacia de la recomendación”, publicado en Memorias de la X Conferencia Internacional UNICA 2012, Ciego de Ávila, Cuba. ISBN 978-959-16-2046-0.
9. Yera Toledo, R y otros, "RECJUGDE: Un sistema recomendador de apoyo a las asignaturas de programación," Memorias XIV Congreso Internacional de Informática Educativa, (InforEdu 2011), La Habana, Cuba, 2011. ISBN 978-959-7213-01-7.
10. Yera Toledo, R y otros, “Herramienta de filtrado colaborativo como complemento a un jurado online de programación” Memorias V Conferencia Científica de la Universidad de las Ciencias Informáticas, UCIENCIA 2010, La Habana, Cuba. ISBN 978-959-286-011-7

Participación en eventos nacionales e internacionales:

1. "Managing natural noise in collaborative recommender systems”. Joint IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS), Edmonton, Canada, 2013.
2. “Corrigiendo preferencias inconsistentes en sistemas recomendadores de filtrado colaborativo”, 2nd International Conference on Computer Science and Informatics (CICCI 2013), La Habana, Cuba, 2013.
3. “RECJUDGE: un sistema de filtrado colaborativo de apoyo a las asignaturas de programación”, 2do Congreso Internacional “TICs y educación: relato de experiencias”, Toluca, México, 2012.
4. “Un enfoque para la corrección de ruido natural en sistemas recomendadores de filtrado colaborativo”, X Congreso Nacional de Reconocimiento de Patrones (RECPAT 2012), Camagüey, Cuba, 2012.
5. “Corrigiendo preferencias en un sistema de filtrado colaborativo para aumentar la eficacia de la recomendación”, X Conferencia Internacional UNICA 2012, Ciego de Ávila, Cuba, 2012.
6. “RECJUGDE: Un sistema recomendador de apoyo a las asignaturas de programación," XIV Congreso Internacional de Informática Educativa, (InforEdu 2011). La Habana, Cuba, 2011.
7. “Herramienta de filtrado colaborativo como complemento a un jurado online de programación” Memorias V Conferencia Científica de la Universidad de las Ciencias Informáticas (UCIENCIA 2010), La Habana, Cuba, 2010.

Premios y Reconocimientos obtenidos:

1. **Premio CITMA Provincial**: Métodos de preprocesamiento de datos para sistemas recomendadores de filtrado colaborativo, con impacto en un escenario de e-learning. 2013.
2. **Premio del Rector de la Universidad de las Ciencias Informáticas,** en la categoría de Resultado que refleja el avance científico, tecnológico e innovativo de mayor transcendencia y originalidad: “Juez en Línea Caribeño”, 2013
3. **Premio del Rector de la Universidad de las Ciencias Informáticas,** en la categoría de Resultado ya aplicado más útil a la educación superior: “Juez en Línea Caribeño”, 2013

Tesis tutoradas o en proceso de tutoría:

1. Un método basado en información contextual para la recomendación de problemas de programación en un juez en línea. 2015-2017 (planificado). Néstor Corredera Jiménez. Universidad de Ciego de Ávila. (Maestría)
2. Versión 2.0 del módulo de recomendación de problemas, para el Juez en Línea Caribeño. 2014-2015. Eiquel Osorio Ramírez y Randy Espino Lobaina. Universidad de las Ciencias Informáticas. (Grado)
3. Evaluación experimental de métodos clásicos de filtrado colaborativo basados en vecindario. 2013-2014. Oscar García Martín, Universidad de Ciego de Ávila. (Grado)
4. Evaluación de métodos de corrección de inconsistencias a través del tiempo, en sistemas recomendadores de filtrado colaborativo. 2013-2014. Alain Álvarez Cardoso. Universidad de Ciego de Ávila. (Grado)

ANEXOS

Anexo 1. Estrategias de ataque para la introducción de ruido malicioso

-**Ataque aleatorio (Random Attack).** Se apoya en la introducción de perfiles utilizando ratings aleatoriamente asignados a los ítems de relleno, y un rating predefinido para el ítem destino. En el caso que se desee promover este, se asignaría el valor máximo posible, y en el caso que se desee degradar, se asignaría el mínimo.

-Ataque basado en promedio (Average Attack). Semejante al Random Attack, pero con la diferencia de que el rating de cada ítem de relleno no es generado aleatoriamente, sino que se le asigna su valor promedio calculado a partir de todos los usuarios que lo han evaluado en la base de datos correspondiente.

-Bandwagon attack. Asocia el ítem objetivo con un grupo pequeño de ítems positivamente evaluados. En esta estrategia, los perfiles inyectados están compuestos por el ítem destino y por un pequeño grupo de ítems que usualmente reciben mucha atención, tales como libros muy vendidos, películas muy preferidas, y que por tanto son muy votados. Así, los perfiles se generan asignándole el valor máximo de rating al ítem objetivo, el valor máximo de rating para los ítems altamente preferidos, y un valor aleatorio para los ítems de relleno. Estos perfiles tienen una alta probabilidad de ser similares a un alto número de usuarios, y por tanto introducen un sesgo importante en la recomendación de ítems para estos usuarios, respecto al ítem objetivo.

-Reverse bandwagon attack. Representa una variación del bandwagon attack, pero en el que los ítems seleccionados son aquellos que tienden a ser pobremente evaluados por muchos usuarios. En este caso, a estos ítems y al ítem objetivo se le asignan valores bajos de preferencia. La asociación del ítem objetivo a otros ítems no preferidos incrementa la posibilidad de que el sistema genere predicciones bajas para este.

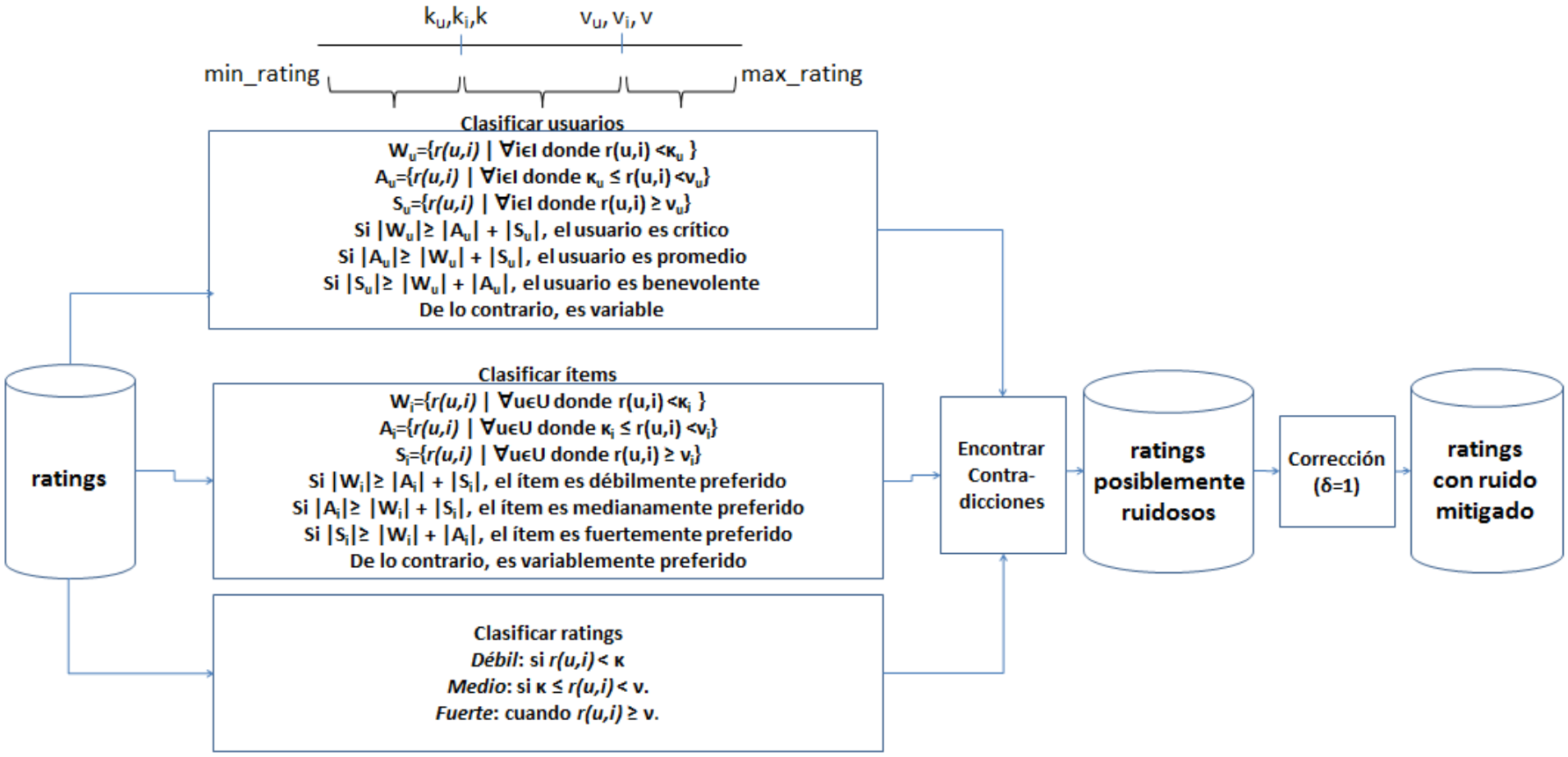
-Popular attack. Parte de asumir que el sistema recomendador asociado emplea el enfoque de filtrado colaborativo basado en el usuario para generar las recomendaciones. De manera semejante al bandwagon attack, este modelo inicialmente construye los nuevos perfiles a inyectar utilizando ítems populares del dominio asociado. Sin embargo, mientras que en el bandwagon attack a los ítems de relleno se le asignan valores aleatorios, en este caso se le asigna un mayor o menor valor de rating a cada uno tomando en consideración si su rating

promedio es mayor o menor. De esta forma, se garantiza que los perfiles generados estén correlacionados positivamente con algunos de los usuarios genuinos del sistema.

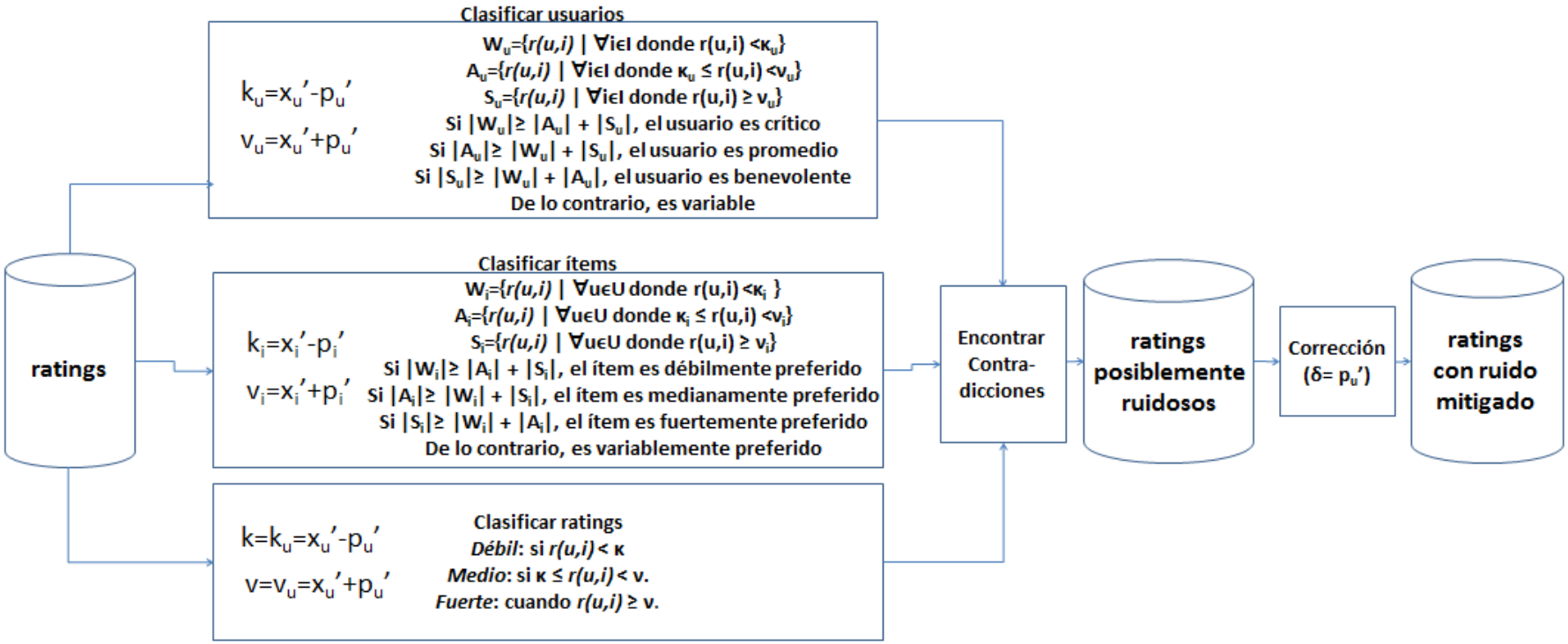
-Probe attack strategy. Semejante al popular attack, pero con la diferencia de que en este caso inicialmente se inserta un perfil "semilla" dentro del sistema para luego calcular los ratings de los ítems de relleno basándose en los valores generados para este perfil en particular. Esta estrategia garantiza que los perfiles generados estén fuertemente correlacionados con los usuarios cercanos al perfil "semilla".

Anexo 2. Presentación gráfica del método de procesamiento de ruido natural

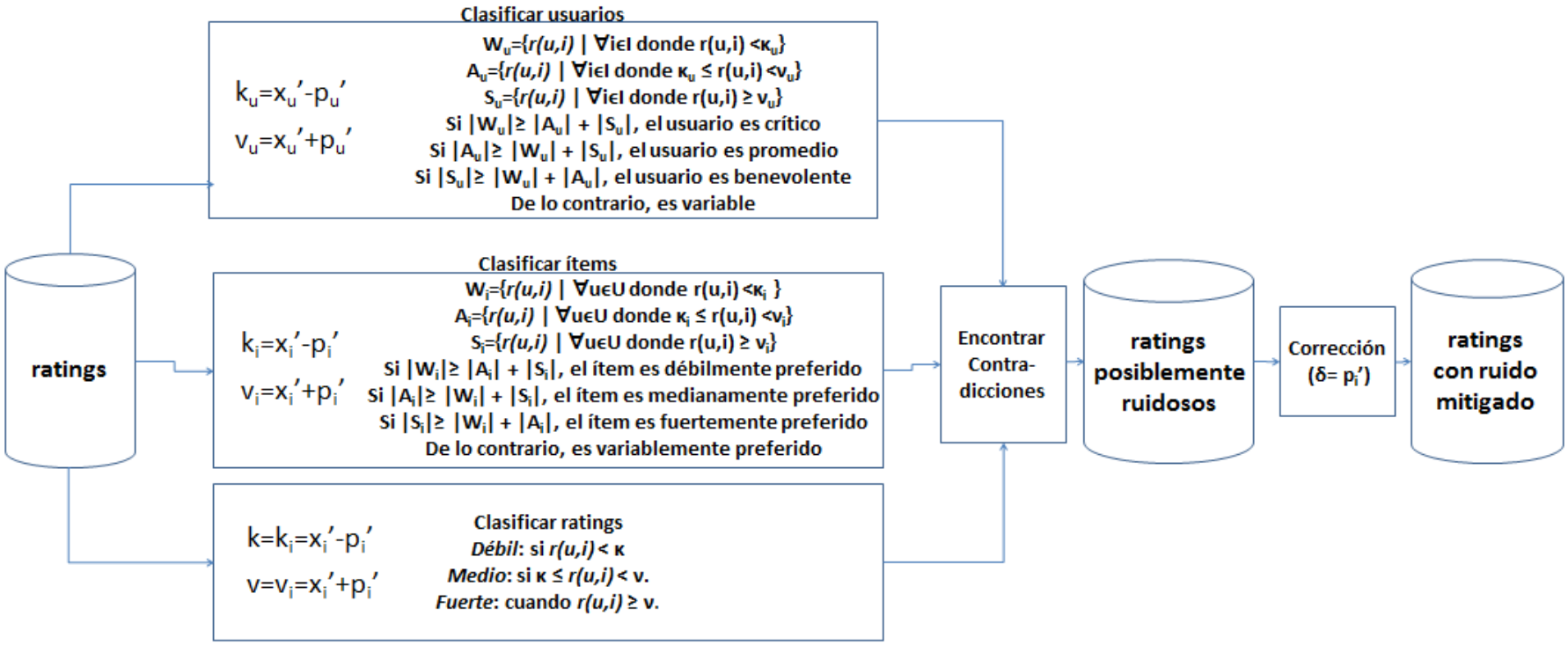
A. Perspectiva global de inicialización de los parámetros (global-pv)



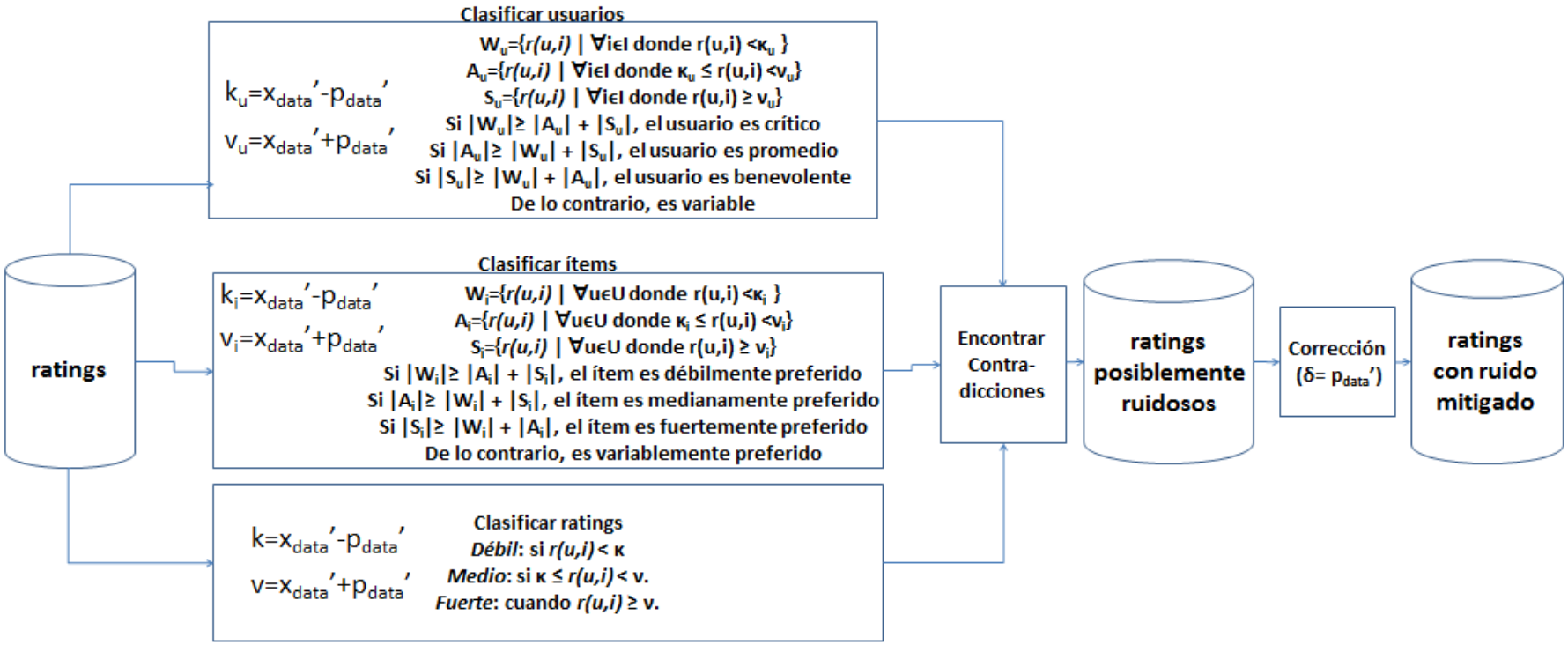
B. Perspectiva de inicialización basada en el usuario (user-based-pv)



C. Perspectiva de inicialización basada en el ítem (item-based-pv)



D. Perspectiva de inicialización basada en los datos (rating-based-pv)



Anexo 3. Cantidad de usuarios e ítems por clase, de ratings posiblemente ruidosos, y de ratings finalmente corregidos

Tabla 1. Resultados para Movielens

Cantidad de usuarios por clase			
críticos	promedio	benevolentes	variables
4	184	577	135
Cantidad de ítems por clase			
Débilmente preferidos	Medianamente preferidos	Fuertemente preferidos	Variablemente preferidos
84	568	592	334
Cantidad de ratings posiblemente ruidosos		Cantidad de ratings finalmente corregidos	
5404 (11,05 % del total de ratings)		2262 (4,62 %)	

Tabla 2. Resultados para MovieTweeting

Cantidad de usuarios por clase			
críticos	promedio	benevolentes	variables
1	179	1301	210
Cantidad de ítems por clase			
Débilmente preferidos	Medianamente preferidos	Fuertemente preferidos	Variablemente preferidos
338	1862	4324	720
Cantidad de ratings posiblemente ruidosos		Cantidad de ratings finalmente corregidos	
2761 (7.45 % del total de ratings)		1844 (4.97 %)	

Tabla 3. Resultados para Netflix

Cantidad de usuarios por clase			
críticos	promedio	benevolentes	variables
3	415	1022	317
Cantidad de ítems por clase			
Débilmente preferidos	Medianamente preferidos	Fuertemente preferidos	Variablemente preferidos
37	228	342	141
Cantidad de ratings posiblemente ruidosos		Cantidad de ratings finalmente corregidos	
2406 (10.03 % del total de ratings)		511 (2.13 %)	

Anexo 4. Comparación entre PrevClassBased y O'Mahony et al.

Tabla 1. Comparación entre PrevClassBased y O'Mahony et al., acorde al filtrado colaborativo user-user

	User-User CF + O'Mahony et al.	User-User CF + PrevClassBased
MAE Movielens	0,7662	**0,7632**
MAE MovieTweeting	**1,1965**	1,1971
MAE Netflix	**0,7746**	0,7768
F1 Movielens	0,5022	**0,5098**
F1 MovieTweeting	0,5146	**0,5367**
F1 Netflix	**0,4007**	0,3915

Tabla 2. Comparación entre PrevClassBased y O'Mahony et al., acorde al filtrado colaborativo item-item

	Item-Item CF + O'Mahony et al.	Item-Item CF + PrevClassBased
MAE Movielens	0,7749	**0,7674**
MAE MovieTweeting	1,2006	**1,1834**
MAE Netflix	0,7904	**0,7874**
F1 Movielens	**0,5044**	0,5039
F1 MovieTweeting	0,5055	**0,5219**
F1 Netflix	**0,4260**	0,4194

Tabla 3. Comparación entre PrevClassBased y O'Mahony et al., acorde al filtrado colaborativo slope one

	Slope one CF + O'Mahony et al.	Slope one CF + PrevClassBased
MAE Movielens	0,7800	**0,7755**
MAE MovieTweeting	1,3081	**1,3027**
MAE Netflix	0,8062	**0,8048**
F1 Movielens	**0,4939**	0,4904
F1 MovieTweeting	0,4338	**0,5046**
F1 Netflix	**0,4167**	0,4074

Tabla 4. Comparación entre PrevClassBased y O'Mahony et al., acorde al filtrado colaborativo bipolar slope one.

	Bipolar Slope one CF + O'Mahony et al.	Bipolar Slope one CF + PrevClassBased
MAE Movielens	0,7953	**0,7839**
MAE MovieTweeting	1,3243	**1,2982**
MAE Netflix	0,8077	**0,7967**
F1 Movielens	0,5527	**0,5640**
F1 MovieTweeting	0,3939	**0,4831**
F1 Netflix	0,4659	**0,4720**

Tabla 5. Comparación entre PrevClassBased y O'Mahony et al., acorde al filtrado colaborativo basado en factorización de matrices

	MF-based CF + O'Mahony et al.	MF-based CF + PrevClassBased
MAE Movielens	0,7691	**0,7611**
MAE MovieTweeting	1,1981	**1,1697**
MAE Netflix	0,7649	**0,7629**
F1 Movielens	0,5011	**0,5014**
F1 MovieTweeting	0,4941	**0,5370**
F1 Netflix	**0,3487**	0,3293

Anexo 5. Comparación de PrevClassBased contra los métodos basados en la eliminación del perfil

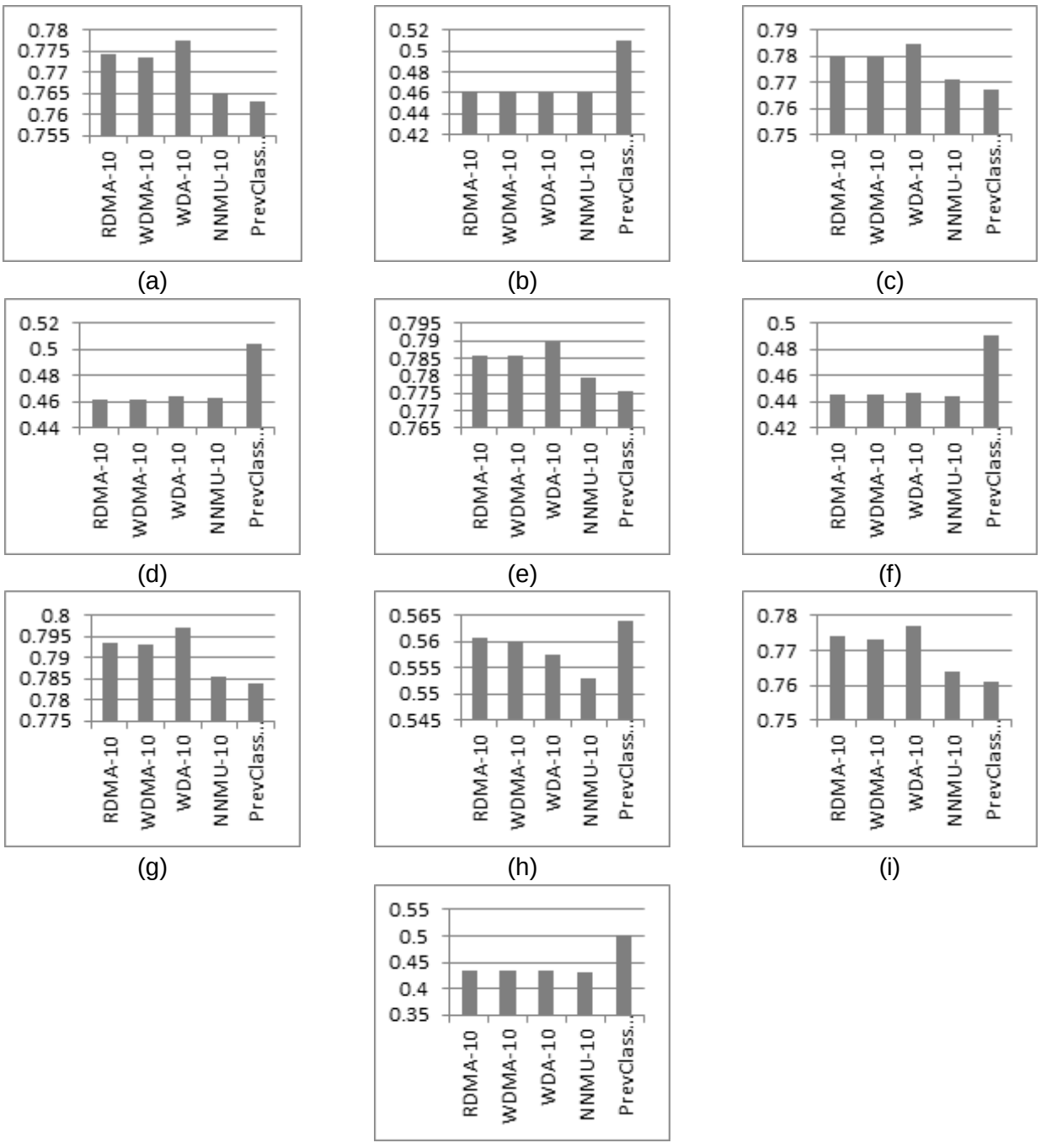


Figura 1. Comparación de PrevClassBased contra los métodos basados en la eliminación del perfil, en el dataset Movielens. (a) User-User CF + MAE (b) User-User CF + F1 (c) Item-Item CF + MAE (d) Item-Item CF + F1 (e) Slope one CF + MAE (f) Slope one CF + F1 (g) Bipolar Slope one CF + MAE (h) Bipolar Slope one CF + F1 (i) MF-based CF + MAE (j) MF-based CF + F1

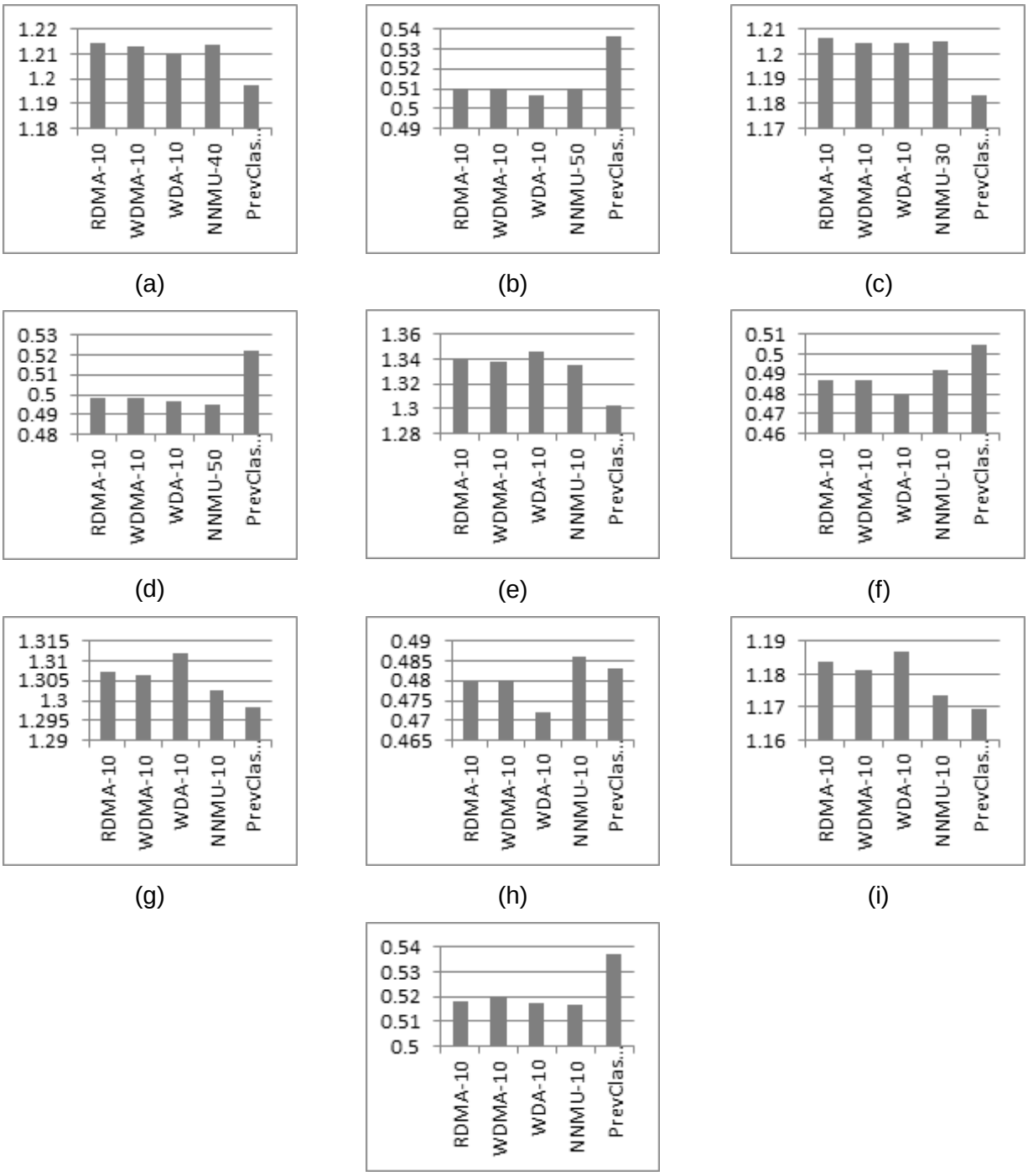


Figura 2. Comparación de PrevClassBased contra los métodos basados en la eliminación del perfil, en MovieTweeting. (a) User-User CF + MAE (b) User-User CF + F1 (c) Item-Item CF + MAE (d) Item-Item CF + F1 (e) Slope one CF + MAE (f) Slope one CF + F1 (g) Bipolar Slope one CF + MAE (h) Bipolar Slope one CF + F1 (i) MF-based CF + MAE (j) MF-based CF + F1

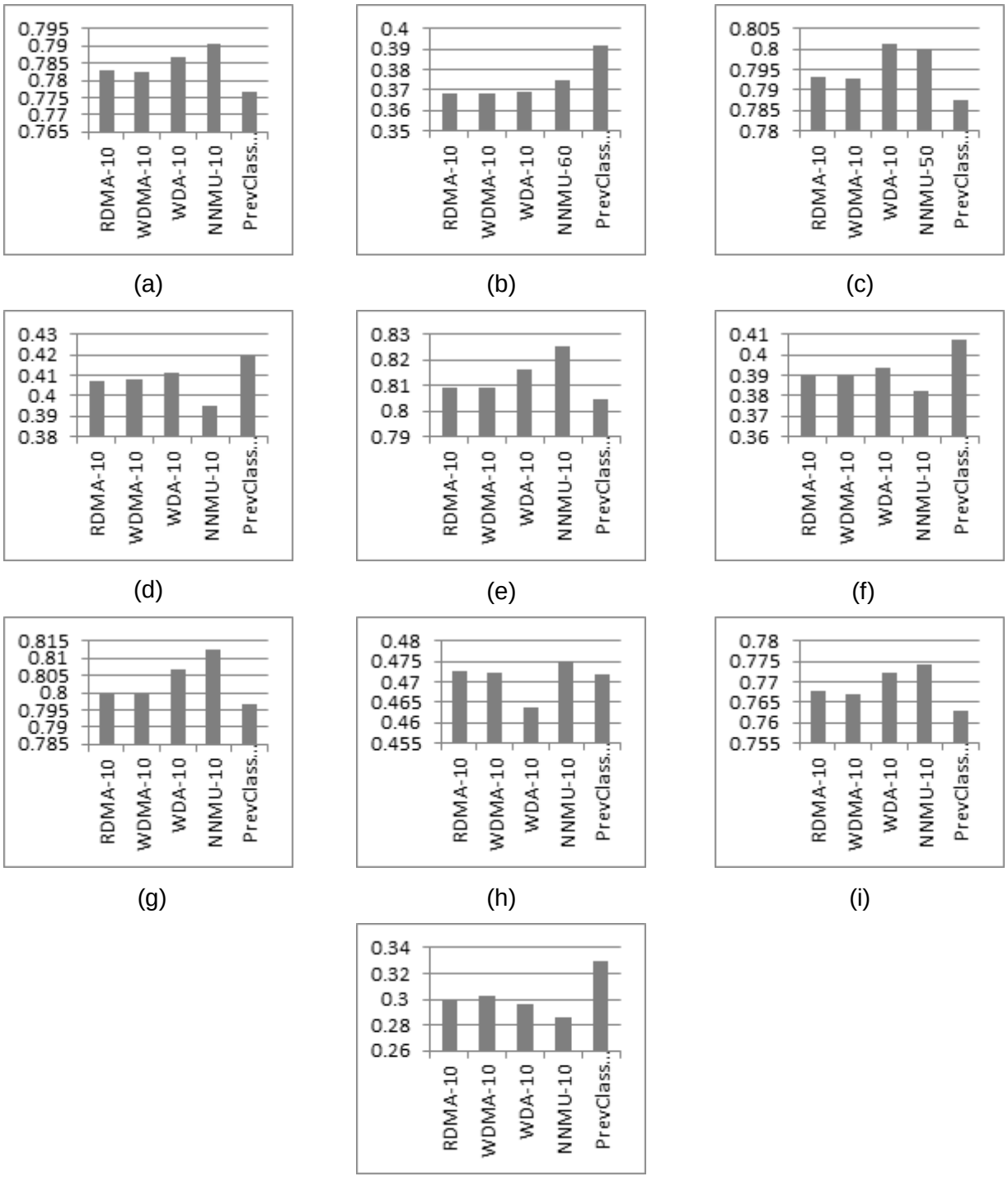


Figura 3. Comparación de PrevClassBased contra los métodos basados en la eliminación del perfil, en el dataset Netflix. (a) User-User CF + MAE (b) User-User CF + F1 (c) Item-Item CF + MAE (d) Item-Item CF + F1 (e) Slope one CF + MAE (f) Slope one CF + F1 (g) Bipolar Slope one CF + MAE (h) Bipolar Slope one CF + F1 (i) MF-based CF + MAE (j) MF-based CF + F1

Anexo 6. Análisis estadístico del método propuesto para la corrección del ruido natural

Tabla 1. Ranking promedio de los algoritmos para la medida MAE. Valor de p calculado por el test de Friedman: 4.34624597422939E-8

Algoritmo	**Ranking**
Base	3.4000000000000004
RDMA	5.300000000000001
WDMA	4.366666666666667
WDA	5.933333333333334
O'Mahony et al.	3.533333333333333
Li et al.	4.2666666666666675
PrevClassBased	1.2

Tabla 2. Tabla de Holm para α=0.05, para la medida MAE.

i	**Algoritmo**	**$z=(R_0-R_i)/SE$**	**p**	**Holm**
6	WDA	6.000595208572467	1.965955356215122E-9	0.008333333333333333
5	RDMA	5.1976986665803775	2.017708534293573E-7	0.01
4	WDMA	4.014482709960454	5.957631052456605E-5	0.0125
3	Li et al.	3.8877095717511767	1.011946052458122E-4	0.016666666666666666
2	O'Mahony et al.	2.9580398915498076	0.0030960205499590623	0.025
1	Base	2.7890090406041046	0.005286958915938801	0.05

Tabla 3. Tabla de Holm para α=0.10, para la medida MAE.

i	**Algoritmo**	**$z=(R_0-R_i)/SE$**	**p**	**Holm**
6	WDA	6.000595208572467	1.965955356215122E-9	0.016666666666666666
5	RDMA	5.1976986665803775	2.017708534293573E-7	0.02
4	WDMA	4.014482709960454	5.957631052456605E-5	0.025
3	Li et al.	3.8877095717511767	1.011946052458122E-4	0.03333333333333333
2	O'Mahony et al.	2.9580398915498076	0.0030960205499590623	0.05
1	Base	2.7890090406041046	0.005286958915938801	0.1

Tabla 4. Ranking promedio de los algoritmos para la métrica F1. Valor de p calculado por el test de Friedman: 6.494476057694598E-5

Algoritmo	**Ranking**
Base	3.6666666666666665
RDMA	4.7
WDMA	4.566666666666667
WDA	5.066666666666667
O'Mahony et al.	3.200000000000001
Li et al.	5.066666666666667
PrevClassBased	1.7333333333333332

Tabla 5. Tabla de Holm para α=0.05, para la medida F1.

i	Algoritmo	$z=(R_0-R_i)/SE$	p	Holm
6	Li et al.	4.225771273642583	2.381237684776802E-5	0.008333333333333333
5	WDA	4.225771273642583	2.381237684776802E-5	0.01
4	RDMA	3.7609364335418984	1.6927844756719902E-4	0.0125
3	WDMA	3.591905582596196	3.2826879084233633E-4	0.016666666666666666
2	Base	2.450947338712698	0.014248079672347133	0.025
1	O'Mahony et al.	1.859339360402738	0.06297905121445503	0.05

Tabla 6. Tabla de Holm para α=0.10, para la medida F1.

i	Algoritmo	$z=(R_0-R_i)/SE$	p	Holm
6	Li et al.	4.225771273642583	2.381237684776802E-5	0.016666666666666666
5	WDA	4.225771273642583	2.381237684776802E-5	0.02
4	RDMA	3.7609364335418984	1.6927844756719902E-4	0.025
3	WDMA	3.591905582596196	3.2826879084233633E-4	0.03333333333333333
2	Base	2.450947338712698	0.014248079672347133	0.05
1	O'Mahony et al.	1.859339360402738	0.06297905121445503	0.1

Anexo 7. Tiempo de respuesta (en milisegundos) del método de corrección propuesto.

Tabla 1. Resultado para la base de datos Movielens

Movielens	Detección de ratings posiblemente ruidosos	Cálculo de correlación	Corrección de ratings	Total
PrevClassBased	15	1406	1485	**2906**
O'Mahony et al. (2006)	-	1406	8047	**9453**

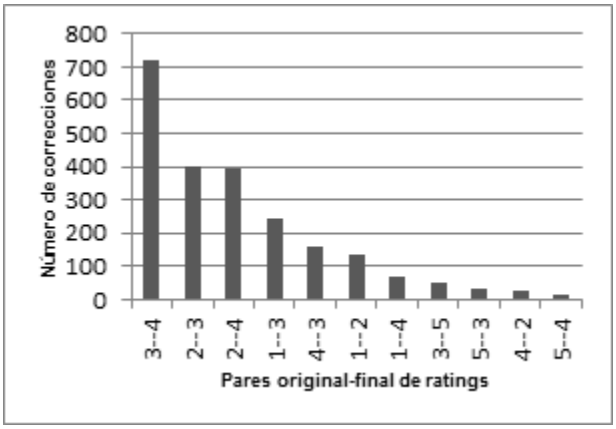
Tabla 2. Resultado para la base de datos MovieTweeting

MovieTweeting	Detección de ratings posiblemente ruidosos	Cálculo de correlación	Corrección de ratings	Total
PrevClassBased	16	2906	578	**3500**
O'Mahony et al. (2006)	-	2906	5625	**8531**

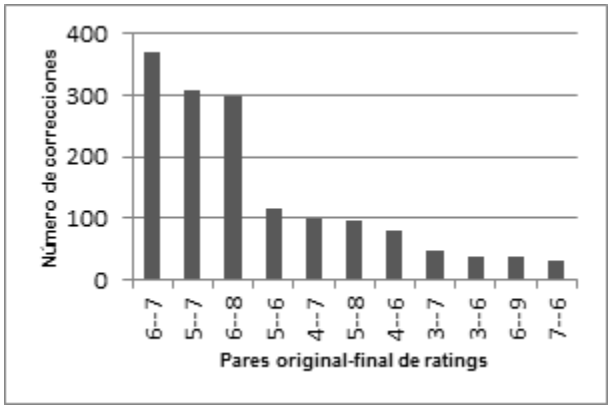
Tabla 3. Resultado para la base de datos Netflix

Netflix	Detección de ratings posiblemente ruidosos	Cálculo de correlación	Corrección de ratings	Total
PrevClassBased	31	3203	938	**4172**
O'Mahony et al. (2006)	-	3203	4906	**8109**

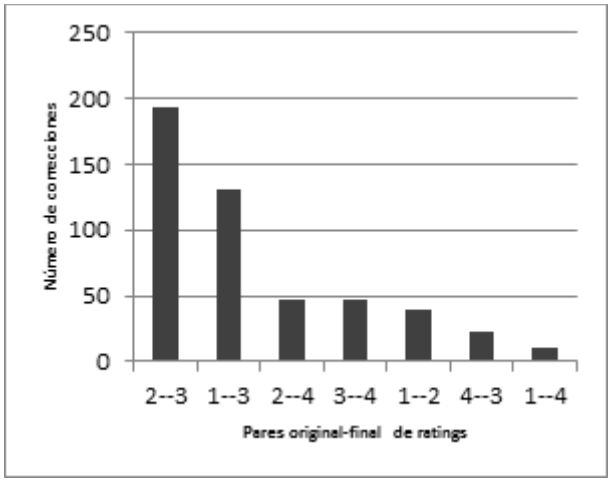
Anexo 8. Número de correcciones agrupadas por el valor original del rating (en primer lugar), y el valor final (en segundo lugar). Los grupos fueron ordenados descendentemente por su tamaño, mostrándose sólo los mayores. (a) Movielens. (b) MovieTweeting. (c) Netflix



(a)



(b)



(c)