UNIVERSIDADE DA CORUÑA

Técnicas eficientes para la recomendación de productos basadas en filtrado colaborativo

Tesis de doctorado

Vreixo Formoso López
Departamento de Tecnoloxías da Información e as Comunicacións
Universidade da Coruña

2013

Tesis de doctorado

Técnicas eficientes para la recomendación de productos basadas en filtrado colaborativo

Autor: Vreixo Formoso López

Director: Víctor M. Carneiro Díaz
Director: Fidel Cacheda Seijo

Departamento: Tecnoloxías da Información e as Comunicacións
Año: 2013

Resumen

Los sistemas de recomendación ayudan al usuario a seleccionar los productos más adecuados a sus necesidades, lo que explica su indiscutible éxito en dominios como el comercio electrónico. En concreto, la técnica de filtrado colaborativo, basada en las opiniones o preferencias de otros usuarios, es especialmente popular gracias a la calidad que presentan sus recomendaciones, fruto de los avances llevados a cabo en los últimos años. Sin embargo, aspectos como la eficiencia o escalabilidad, entre otros, no han recibido la misma atención y, en consecuencia, suponen una limitación que dificulta su expansión a nuevos dominios y aplicaciones con gran volumen de datos.

En esta tesis se abordan diversos problemas que afectan a los algoritmos de filtrado colaborativo, siguiendo una aproximación centrada en el desarrollo de técnicas fáciles de entender, y por tanto de implementar o adaptar a las necesidades concretas de cada entorno. En particular, hemos desarrollado algoritmos sencillos que presentan resultados comparables o incluso superiores a técnicas mucho más complejas. Por un lado, presentamos el algoritmo basado en tendencias, que utiliza un novedoso enfoque consistente en modelar las diferencias entre usuarios en lugar de sus similitudes. Por otro, una variante de los populares algoritmos basados en vecinos, especialmente diseñada para la recomendación de productos.

Estas técnicas, centradas en buscar usuarios con gustos similares, ofrecen muy buenos resultados en multitud de dominios a pesar de su sencillez. En esta tesis veremos como los principales problemas que afectan a su eficacia y rendimiento pueden ser abordados de forma satisfactoria, a partir de soluciones inspiradas en técnicas ampliamente utilizadas en el ámbito de la recuperación de información, tales como la expansión de consultas o la compresión y distribución de índices.

Resumo

Os sistemas de recomendación axudan aos usuarios a seleccionar produtos axeitados ás súas necesidades, sendo esta a principal razón do seu indiscutible éxito en dominios como o do comercio electrónico. En concreto, a técnica de filtrado colaborativo, baseada nas opinións ou preferencias doutros usuarios, é especialmente popular grazas á calidade que presentan as súas recomendacións, resultado dos avances realizados nos últimos anos. Porén, aspectos como a eficiencia ou escalabilidade, entre outros, non recibiron a mesma atención e, en consecuencia, supoñen unha limitación que dificulta a expansión das devanditas técnicas a novos dominios ou aplicacións con gran volume de datos.

Nesta tese abórdanse varios problemas que afectan aos algoritmos de filtrado colaborativo, seguindo unha aproximación centrada no desenvolvemento de técnicas fáciles de entender, e polo tanto de implementar ou adaptar ás necesidades concretas de cada situación. En particular, desenvolvemos algoritmos sinxelos que acadan resultados comparables ou incluso superiores a outras técnicas moito máis complexas. Por unha parte, presentamos o algoritmo baseado en tendencias, que utiliza unha nova aproximación, consistente en modelar as diferenzas entre usuarios en lugar das súas similitudes. Por otra parte, unha variante dos populares algoritmos baseados en veciños, especialmente deseñada para a recomendación de produtos.

Estas técnicas, centradas en procurar usuarios con gustos similares, ofrecen moi vos resultados en multitude de dominios, a pesar da súa simplicidade. Nesta tese veremos como os principais problemas que afectan á súa eficacia e rendemento poden abordarse de forma satisfactoria, a partir de solucións inspiradas en técnicas amplamente utilizadas no ámbito da recuperación de información, tales como a expansión de consultas ou a compresión e distribución de índices.

Abstract

Recommender systems are a popular technique used in fields such as e-commerce to help users to find the products they need. A particularly successful approach is collaborative filtering, that computes high-quality recommendations based on the preferences of other users with similar tastes or interests. In the past years, many advances to this technique have been developed. However, there is still a lot of work to do regarding the performance and scalability of most approaches. Nowadays, these key aspects are a serious drawback that stops the expansion of these techniques to new domains and applications with huge amounts of data.

In this thesis we address several problems of collaborative filtering algorithms. We focus on the development of techniques that can be easily implemented and adapted to the particular needs of each application. First, we have developed simple algorithms whose results are at least similar to those obtained by the complex state of the art techniques. We present the tendencies-based algorithm, which uses a novel approach focused on modeling the differences among users instead of their similarities. We also introduce a variation of the well know neighbor-based algorithms, specially designed for the recommendation task.

These techniques, focused on selecting users with similar tastes, present very good results in many domains despite their simplicity. In this thesis we show how the main issues regarding their quality and efficiency can be successfully addressed using techniques similar to those used in the field of Information Retrieval. In particular, we adapt several popular techniques such as query expansion, index compression and distributed approaches based on index partition.

Índice general

ÍNDICE GENERAL

Índice de figuras

ÍNDICE DE FIGURAS

ÍNDICE DE FIGURAS

Índice de tablas

ÍNDICE DE TABLAS

Parte I

Contextualización

Capítulo 1

Introducción

Los sistemas de recomendación han adquirido una gran popularidad en los últimos años, especialmente en contextos como el comercio electrónico. La comunidad científica no ha permanecido impasible ante este hecho, y ya desde mediados de los años noventa los sistemas de recomendación emergen como un área de investigación independiente. Sin embargo, en estas casi dos décadas la mayor parte de los esfuerzos se han dedicado a mejorar la precisión de los algoritmos, y otros aspectos no menos importantes se han visto relegados a un segundo plano.

Esta tesis aborda algunos de estos problemas, especialmente aquellos relacionados con la eficiencia y escalabilidad de los sistemas de recomendación. Dado el volumen de usuarios y productos que muchas aplicaciones deben manejar, hoy en día este problema cobra una importancia fundamental.

Este primer capítulo ofrece una visión general del contexto en que esta tesis se desarrolla, y de las principales contribuciones de la misma. En la Sección 1.1, se introducen los problemas y oportunidades que han motivado la realización de este trabajo. La Sección 1.2 presenta claramente los objetivos del mismo. A continuación, en la Sección 1.3, se describen las principales aportaciones realizadas, sobre las que se profundizará a lo largo de los sucesivos capítulos. Finalmente, la Sección 1.4 describe brevemente la estructura y contenido del presente documento.

1.1. Motivación

El desarrollo y popularización que las tecnologías de la información y particularmente Internet han sufrido en los últimos años es, probablemente, el hecho que más trascendencia ha tenido en la historia reciente de la humanidad. Más allá de una red global de comunicaciones, Internet ha sido el medio que ha permitido el desarrollo y

evolución de un gran número de servicios que se han convertido en parte de nuestras vidas, y sin los cuales difícilmente se podría entender el éxito de la red de redes. Hoy en día, utilizamos Internet para actividades tan diferentes como leer la prensa diaria, comunicarnos con nuestros familiares y amigos, compartir nuestras creaciones y opiniones, buscar información, o incluso ir de compras. Tal es ya su importancia, que las generaciones más jóvenes no conciben un mundo sin Internet, igual que muchos de nosotros no nos imaginamos un mundo sin electricidad, sin coches, o sin televisión.

La cantidad de información y servicios disponibles *online* es tan elevada, que apenas podríamos llegar a aprovechar una pequeña fracción de su potencial si no dispusiésemos de herramientas capaces de buscar y filtrar aquello que realmente necesitamos. Los sistemas de recuperación de información, popularmente conocidos como *buscadores*, fueron adquiriendo un protagonismo cada vez mayor a medida que la información disponible en la Red crecía más y más. Actualmente, los buscadores se han convertido en herramientas indispensables para buscar y localizar información en Internet.

Sin embargo, los usuarios no sabemos muchas veces como comunicar o expresar aquello que queremos buscar, o directamente no sabemos que es exactamente lo que necesitamos. Es posible, incluso, que ni siquiera conozcamos su existencia. Otras veces, simplemente nos vemos ahogados entre un gran volumen de información y multitud de opciones, y no tenemos ni el tiempo ni las ganas para buscar cual de ellas es la más adecuada.

Como escribía Adam Richardson en su ya célebre artículo *From the Information Age to the Recommendation Age* [Richardson, 2005], mientras hace unos años el principal problema a la hora de tomar una decisión era el poder contar con información suficiente, hoy en día es precisamente lo contrario: tenemos demasiada información. Tanta, que muchas veces cuesta encontrar aquella realmente valiosa o que nos ayude a decidir entre las numerosas opciones que tenemos a nuestra disposición.

En este contexto, los sistemas de recuperación de información tradicionales, si bien imprescindibles, no siempre son suficientes. En muchas situaciones necesitamos herramientas que se encarguen de buscar por nosotros, y directamente nos recomienden un conjunto limitado de opciones que respondan a nuestras necesidades. Este es, precisamente, el objetivo de los *sistemas de recomendación*. Estos sistemas mantienen un *perfil* con las preferencias de cada usuario, lo que les permite ser capaces de realizar *recomendaciones personalizadas*, adecuadas a los gustos o necesidades de cada individuo. Nos ayudan, pues, a tomar decisiones, de igual forma que lo hacen las recomendaciones de nuestros amigos, o la opinión de críticos o expertos reconocidos. Sin embargo, sólo los sistemas de recomendación automatizados son capaces de elegir entre los miles (o

millones) de opciones que se pueden llegar a presentar en muchos dominios [Bollen et al., 2010].

No es de extrañar, por tanto, el gran auge que estos sistemas han experimentado en los últimos años. Portales de comercio electrónico como Amazon [Linden et al., 2003] han sido los primeros y principales beneficiarios de esta nueva tecnología. Gracias a ellos, han visto como aumentaban sus beneficios a la par que lo hacía la satisfacción de sus clientes. En poco tiempo, los sistemas de recomendación se extendieron a multitud de aplicaciones y productos: películas, libros, canciones o grupos musicales, amigos en una red social, noticias, programas de televisión, ropa y complementos, etc.

En particular, la técnica de filtrado colaborativo [Goldberg et al., 1992] es una de las más populares, especialmente en ámbitos como el comercio electrónico. Su éxito reside en que, al contrario de otras técnicas como la recomendación basada en contenido, se basa enteramente en la opinión de los usuarios. Esto le permite ofrecer recomendaciones de altísima calidad, ya que al final los responsables de la recomendación son otras personas, que, por tanto, tienen una visión de la calidad de un producto que va mucho más allá de las características que puede observar una máquina (palabras en un artículo, píxeles en una imagen, etc.). El sistema únicamente se encarga de automatizar el proceso, permitiendo que millones de personas puedan recomendarse productos entre sí, sin ser tan siquiera conscientes de ello.

Casi dos décadas de investigación en sistemas de recomendación han dado lugar a infinidad de algoritmos y técnicas [Su y Khoshgoftaar, 2009], así como a cientos de publicaciones en congresos y revistas especializadas [Ricci et al., 2011a]. Sin embargo, no es menos cierto que la prioridad de la mayor parte de trabajos ha sido el mejorar la precisión de los algoritmos. En concreto, con la celebración del concurso organizado por Netflix [Bennett y Lanning, 2007], que premiaba con un millón de dólares al algoritmo de recomendación que mejor aproximase las preferencias de los usuarios, investigadores de todo el mundo se lanzaron a una carrera por ver quién conseguía el algoritmo más preciso. En parte gracias a ello, hoy en día disponemos de técnicas que son mucho más exactas, pero también bastante más complejas que los algoritmos tradicionales. Al buscar una mejora cada vez mayor de la precisión, muchas veces se ha olvidado que el propósito de un sistema de recomendación es el ser usado en aplicaciones reales, donde al final el éxito va a venir marcado por cuestiones como la satisfacción de los clientes o la rentabilidad para el negocio. La precisión es simplemente uno de los muchos problemas a abordar de cara a lograr ese objetivo.

Un buen sistema de recomendación deberá ser de utilidad a lo largo de toda la vida de una aplicación, contribuyendo a su éxito desde el principio. Cuando esta se lanza,

el sistema de recomendación debe ayudar a captar a los primeros clientes, ofreciendo recomendaciones de calidad a pesar de que la información que tendremos acerca de sus preferencias posiblemente sea limitada. En esta fase es especialmente importante ganar la confianza del usuario, explicando los motivos de la recomendación o recomendando de vez en cuando productos conocidos. Muchos algoritmos son tan complejos que es complicado saber el porqué de una recomendación en concreto, lo cual es un problema en esta etapa, donde el usuario todavía no confía en que el sistema le vaya a recomendar productos útiles.

A medida que los usuarios interactúan con la aplicación, conoceremos mejor sus gustos, y el sistema de recomendación deberá aprovecharlo para mejorar la calidad de las recomendaciones, contribuyendo así a fidelizar a los clientes. Por supuesto, la calidad de una recomendación no dependerá únicamente de la precisión del algoritmo. Cuestiones como la diversidad en las recomendaciones, el ofrecer al usuario recomendaciones novedosas e inesperadas, la integración de preferencias a corto y largo plazo, o incluso el garantizar la privacidad de las preferencias del usuario [Ricci et al., 2011a; Shani y Gunawardana, 2011], tienen un mayor impacto en el éxito del sistema que conseguir reducir el error en unas pocas centésimas.

Al mismo tiempo, si queremos que la aplicación crezca y se popularice, el sistema de recomendación no debe descuidar a los nuevos usuarios, ofreciendo buenas recomendaciones a pesar de que la información que tendremos sobre los mismos será muy reducida [Schein et al., 2002]. Técnicas que sólo se comportan bien cuando el sistema conoce perfectamente las preferencias de los usuarios no son suficientes para garantizar el éxito de una aplicación. Debemos emplear técnicas que ofrezcan buenas recomendaciones en multitud de casos.

Si todo va bien, la aplicación irá adquiriendo popularidad, y llegará un punto donde el número de usuarios empezará a crecer exponencialmente. En este momento, la escalabilidad del sistema adquiere una importancia fundamental, ya que debe ser capaz de absorber una carga cada vez mayor sin que se dispare el coste por usuario, pero también sin disminuir la calidad de las recomendaciones [Shani y Gunawardana, 2011]. Gracias a Internet, una aplicación exitosa puede llegar a contar con millones de usuarios poco tiempo después de ser lanzada. Los sistemas de recomendación deben ser lo suficientemente eficientes como para generar una recomendación en milésimas de segundo, así como ser capaces de proporcionar cientos o incluso miles de recomendaciones por segundo. Muchas de las técnicas que destacan por su gran precisión necesitan varios segundos, o incluso minutos, para calcular una recomendación. Otras se basan en complejos modelos que necesitan mucho tiempo para ser entrenados. A medida que

aumenta el número de usuarios, ya no es posible entrenar el modelo con frecuencia, y la precisión puede llegar a decaer de manera importante. La escalabilidad y rendimiento son factores claves para que un algoritmo pueda dar el salto desde el laboratorio hasta la nube.

Esta tesis está enfocada precisamente en el desarrollo de técnicas que puedan ser usadas en aplicaciones reales. Sin perder de vista la importancia de la precisión, nos hemos centrado en otros aspectos que son frecuentemente ignorados a pesar de su importancia de cara al éxito de un sistema de recomendación.

En primer lugar, hemos desarrollado algoritmos sencillos, fáciles de entender y que se comportan bien no sólo en un escenario concreto, sino en condiciones y dominios muy diferentes.

También hemos propuesto técnicas para abordar el problema de *cold-start*, es decir, el ofrecer recomendaciones de calidad a usuarios que empiezan a usar el sistema y de los que apenas se dispone de información.

Y finalmente, hemos dedicado gran parte de la tesis a abordar los problemas relacionados con la eficiencia y escalabilidad, tomando como referencia el mundo de la Recuperación de Información (RI), donde se han estado abordando este tipo de problemas durante años. En particular, hemos centrado buena parte de nuestros esfuerzos en los algoritmos basados en vecinos, y a su implementación mediante índices invertidos como los empleados habitualmente en RI. Este tipo de algoritmos son especialmente populares, y presentan importantes ventajas sobre otras técnicas, tal y como: su simplicidad, lo que hace que sean fáciles de entender e implementar; su comportamiento intuitivo, lo que facilita explicar el motivo de las recomendaciones, y su precisión y buen comportamiento en numerosos casos y situaciones [Desrosiers y Karypis, 2011].

Como veremos a lo largo de los próximos capítulos, todas las contribuciones de este trabajo están marcadas por un mismo patrón: su aplicabilidad al mundo real. Nos hemos centrado no sólo en resolver problemas presentes en las aplicaciones actuales y futuras, sino también en que nuestras soluciones puedan ser usadas fuera del laboratorio.

1.2. Objetivos

El principal objetivo de esta tesis es el diseño de **técnicas de recomendación eficientes y escalables**. Para que los usuarios de aplicaciones con gran volumen de datos puedan disfrutar de los beneficios de la recomendación, es necesario contar con algoritmos que sean capaces de procesar miles de millones de preferencias y ofrecer recomendaciones de calidad en unos pocos milisegundos, a la par que soportar la demanda

de millones de usuarios. Este tipo de técnicas no sólo son una necesidad en multitud de aplicaciones hoy en día, sino que posibilitarían la introducción de los sistemas de recomendación en nuevas aplicaciones y dominios.

El propósito de esta tesis es el desarrollo de técnicas que cumplan estos requisitos, pero sin renunciar a las recomendaciones de calidad a las que están acostumbrados los usuarios de los sistemas de recomendación actuales. En particular, los objetivos de esta tesis se pueden resumir en los siguientes puntos:

- El desarrollo de algoritmos de recomendación sencillos y eficientes. En particular, se estudiará la adaptación al ámbito del filtrado colaborativo de técnicas utilizadas en la RI. En el diseño de este tipo de sistemas siempre ha estado presente la necesidad de manejar grandes volúmenes de datos, lo que ha dado como resultado el desarrollo de un gran número de técnicas que permiten representar y procesar *terabytes* de información de forma eficiente. Se aplicarán técnicas similares a las propuestas en RI a algoritmos de recomendación reconocidos por sus buenos resultados en multitud de dominios, en particular los algoritmos basados en vecinos, lo que permitiría aprovechar las ventajas de ambos: buena calidad de las recomendaciones, y eficiencia y capacidad para manejar un gran volumen de datos.

- El estudio de las características de las técnicas de filtrado colaborativo actuales, identificando las causas de sus principales limitaciones y proponiendo soluciones a las mismas que cumplan con los requisitos indicados en el punto anterior, es decir, que sean eficientes computacionalmente. Se desarrollarán técnicas que se comporten de manera precisa incluso cuando la información disponible sea limitada. Por ejemplo, en caso de usuarios que han empezado a usar el sistema recientemente. Este tipo de técnicas son necesarias para garantizar el buen funcionamiento del sistema de recomendación desde los primeros días.

- El desarrollo de técnicas de recomendación que puedan ser explotadas en entornos distribuidos. La distribución de la información y su procesamiento es fundamental para conseguir una técnica de recomendación escalable. Se diseñarán algoritmos que puedan ser distribuidos entre diferentes máquinas con el objetivo de incrementar el número de recomendaciones que el sistema es capaz de procesar así como reducir el tiempo de respuesta.

1.3. Contribuciones

En esta tesis se presentan varias mejoras que afectan a diferentes aspectos relacionados con los sistemas de recomendación. Las principales contribuciones de este trabajo se pueden resumir en los siguientes puntos:

1. **La identificación y caracterización de diversos problemas que afectan a los sistemas de filtrado colaborativo actuales**. Hemos estudiado el comportamiento de gran parte de las técnicas actuales, e identificado numerosos problemas que pueden afectar a su éxito en sistemas reales.

2. **La propuesta de nuevas métricas y metodologías de evaluación** tanto para predicción como recomendación. Para la primera de ellas, hemos propuesto las métricas GIM y GPIM, que al centrarse en el error cometido en las predicciones que puedan tener un mayor impacto desde el punto del vista del usuario, ofrecen una evaluación más cercana a sus intereses. Para la recomendación, hemos propuesto una metodología de evaluación basada en la usada habitualmente en el ámbito de la recuperación de información. También hemos destacado la importancia de estudiar la evolución de los resultados de los algoritmos al variar la cantidad de información disponible.

3. **La propuesta de un nuevo algoritmo para predicción.** Presentamos una aproximación novedosa al problema de filtrado colaborativo: el algoritmo basado en tendencias, una nueva técnica basada en las diferencias entre usuarios en lugar de en sus similitudes, y que obtiene grandes resultados incluso cuando la información disponible es muy limitada. Además, es muy sencilla y eficiente, pudiendo ser usada en dominios con gran volumen de datos.

4. **La mejora de los algoritmos basados en vecinos en la recomendación.** Proponemos diversas modificaciones a los algoritmos basados en vecinos o *k-Nearest Neighbors* (kNN), especialmente diseñadas para la recomendación de listas de productos. Las principales aportaciones consisten en una función de similitud que sólo tiene en cuenta aquellos productos bien valorados por el usuario, así como una función de recomendación que prioriza aquellos productos que han sido bien puntuados por muchos vecinos. Ambas modificaciones permiten incrementar sensiblemente la precisión de las recomendaciones.

5. **El desarrollo de una nueva técnica para minimizar el problema de *cold-start***. Uno de los principales problemas que presentan los algoritmos de filtrado

colaborativo es el conocido como *cold-start* [Schein et al., 2002], que se produce cuando nuevos productos o usuarios son introducidos en el sistema. En este trabajo presentamos la técnica de *expansión del perfil*, que mejora significativamente la calidad de las recomendaciones en caso de usuarios que han empezado a usar el sistema recientemente, y cuyas preferencias, por tanto, son en gran parte desconocidas para el sistema. Una de las grandes ventajas de nuestra técnica es que no necesita más información que aquella ya conocida por el sistema: las preferencias de otros usuarios.

6. **El desarrollo de una implementación eficiente para algoritmos kNN.** Proponemos varias mejoras a las técnicas kNN tradicionales para recomendación de productos, basándonos en el modelo vectorial propuesto en el ámbito de la recuperación de información. En particular, proponemos una implementación basada en índices, extendiendo la propuesta por Cöster y Svensson [2002]. También estudiamos el impacto en eficiencia y precisión de la técnica de preselección de vecinos.

7. **El estudio de técnicas de compresión de la matriz de puntuaciones.** Los algoritmos de filtrado colaborativo almacenan las opiniones de los usuarios en una estructura que recibe el nombre de matriz de puntuaciones. A medida que el número de usuarios o productos manejados por el sistema aumenta, también lo hace el tamaño de dicha matriz. En las aplicaciones actuales, con millones de usuarios o elementos, puede llegar a suponer un problema importante. En este trabajo proponemos varias técnicas para reducir el tamaño de la matriz, basadas en la compresión de la información en ella almacenada. En primer lugar, demostramos como las técnicas de codificación que se usan habitualmente en dominios como el de recuperación de información pueden reducir de manera importante el tamaño de la matriz aparte de mejorar la eficiencia de los algoritmos. En segundo lugar, proponemos una técnica de reasignación de identificadores adaptada a filtrado colaborativo, que mejora significativamente los ratios de compresión alcanzados.

8. **La propuesta de una arquitectura de recomendación distribuida.** Hemos mostrado cómo los algoritmos kNN pueden ser implementados de forma distribuida mediante técnicas de partición de la matriz de puntuaciones, es decir, repartiendo las preferencias de los usuarios entre distintas máquinas. La principal ventaja de este tipo de implementaciones es que permiten reducir el tiempo de respuesta y al mismo tiempo incrementar el número de recomendaciones calculadas

por unidad de tiempo, siendo por tanto básicas para aumentar la eficiencia de los sistemas y facilitar su escalabilidad a medida que aumenta el número de usuarios, productos o preferencias. Hemos estudiado dos posibles estrategias: partición por usuarios y partición por productos, analizando las ventajas e inconvenientes de cada una de ellas.

9. **Un modelo de evaluación del rendimiento basado en la simulación.** La evaluación de la eficiencia y escalabilidad de un sistema de recomendación a gran escala supone un desafío importante, debido a la necesidad de realizar experimentos con multitud de configuraciones diferentes, y el coste de adquisición de la infraestructura necesaria (servidores, red, etc.), prohibitivo para la mayoría de grupos de investigación. Es por ello que en este trabajo proponemos el uso de la simulación, una alternativa mucho más sencilla, rápida y económica. A pesar de su popularidad en la evaluación del rendimiento en contextos como la recuperación de información, no somos conscientes de ningún trabajo que utilice estas técnicas en la evaluación de sistemas de recomendación. Por tanto, hemos diseñado un modelo de simulación adecuado para este tipo de sistemas, adaptando y mejorando el presentado por Cacheda et al. [2007] para la evaluación de sistemas de RI distribuidos.

1.3.1. Principales publicaciones

Gran parte de las contribuciones que acabamos de mencionar han sido publicadas en congresos y revistas de carácter científico. En particular, el desarrollo de esta tesis ha dado como resultado las siguientes publicaciones en revistas especializadas [1]:

- F. Cacheda, V. Carneiro, D. Fernández y V. Formoso. Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems. *ACM Transactions on the Web*, Vol. 5, Issue 1, Article 2, 33 págs. Febrero 2011. ISSN: 1559-1131.

- V. Formoso, D. Fernández, F. Cacheda y V. Carneiro. Using profile expansion techniques to alleviate the new user problem. *Information Processing & Management*. Disponible online 6 Septiembre 2012. ISSN 0306-4573.

- V. Formoso, D. Fernández, F. Cacheda y V. Carneiro. Using rating matrix compression techniques to speed up collaborative recommendations. *Information Re-*

[1] Hasta diciembre de 2011 en las publicaciones del grupo los autores se ordenaban alfabéticamente; en las publicaciones posteriores, por orden de contribución.

trieval. Publicado online 30 Octubre 2012. ISSN 1573-7659.

Y diversas publicaciones en congresos nacionales e internacionales:

- V. Formoso, F. Cacheda y V. Carneiro. Algorithms for Efficient Collaborative Filtering. *Efficiency Issues in Information Retrieval Workshop. European Conference on Information Retrieval, ECIR 2008*. Glasgow, United Kingdom, 30 March 2008.

- V. Formoso, F. Cacheda y V. Carneiro. Filtrado Colaborativo para Recuperación de Información. *VII Jornadas de Ingeniería Telemática, JITEL 2008*, pág. 285-292. Alcalá de Henares, 2008, ISBN: 978-84-612-5474-3.

- R. Baraglia, F. Cacheda, V. Carneiro, V. Formoso, R. Perego, F. Silvestri. Search Shortcuts Using Click-Through Data. En *Proceedings of Workshop on Web Search Click Data, WSCD09*. Barcelona, Spain. February 9, 2009.

- R. Baraglia, F. Cacheda, V. Carneiro, V. Formoso, R. Perego, F. Silvestri. Search Shortcuts: Driving Users Towards Their Goals. En *WWW 2009 Proceedings*, pág. 1073. Madrid 20-24 Abril 2009. ISBN: 978-1-60558-487-4.

- F. Cacheda, V. Carneiro, D. Fernández y V. Formoso. Search Shortcuts: recomendación de consultas en buscadores web. *VIII Jornadas de Ingeniería Telemática, JITEL 2009*, pág. 237-244. Cartagena, 2009. ISBN: 978-84-96997-27-1.

- R. Baraglia, F. Cacheda, V. Carneiro, D. Fernández, V. Formoso, R. Perego, F. Silvestri. Search Shortcuts: A New Approach to the Recommendation of Queries. *RecSys 2009 Proceedings*, pág 77-84. New York, USA, October 22-25, 2009. ISBN: 978-1-60558-435-5.

- F. Cacheda, V. Carneiro, D. Fernández y V. Formoso. Eficiencia y precisión de algoritmos de Filtrado Colaborativo: análisis y comparativa. *I Congreso Español de Recuperación de Información, CERI 2010*, Pág. 275-282. Madrid, 2010. ISBN: 978-84-693-2200-0.

- F. Cacheda, V. Carneiro, D. Fernández y V. Formoso. Improving K-Nearest Neighbors Algorithms: Practical Application of Dataset Analysis. *20th ACM Conference on Information and Knowledge Management, CIKM 2011*, pág 2253-2256. Glasgow, UK. 24-28 October 2011. ISBN: 978-1-4503-0717-8.

- V. Formoso, D. Fernández, F. Cacheda y V. Carneiro. Using Neighborhood Precomputation to Increase Recommendation Efficiency. En *Proceedings of the International Conference on Knowledge Discovery and Information Retrieval, KDIR 2012*, pág. 333-335, Barcelona, Spain, 4 - 7 October, 2012. SciTePress. ISBN 978-989-8565-29-7

Además, se ha presentado una solicitud de patente para el método de recomendación presentado en la Sección 4.2:

F. Cacheda, V. Carneiro, V. Formoso, D. Fernández, A. Freire. "Recommendation Method and System." U.S. Patent Application, 61/492,206, Jun. 1, 2011.

1.4. Estructura de la tesis

Los diez capítulos que componen esta tesis han sido agrupados en tres partes. La **primera parte** es una visión global a los sistemas de recomendación, y a la técnica de filtrado colaborativo en particular. En esta parte se presentan los conceptos básicos necesarios para entender el funcionamiento y características de estas técnicas, y se hace un repaso global al estado del arte.

Sin embargo, debido a que en esta tesis se han abordado mejoras que afectan a diversos aspectos de los sistemas de recomendación, hemos considerado que tratar en profundidad los detalles y trabajos relacionados con cada uno de los mismos en esta primera parte supondría enfrentar al lector de manera prematura a un gran número de detalles acerca de diferentes técnicas y problemas. Hemos preferido, por tanto, restringir las primeras páginas de este documento a una visión global de los sistemas de recomendación, dejando el estudio de la problemática más específica para capítulos posteriores. Es por ello que a lo largo de esta tesis se incluirá en muchos capítulos una sección destinada a presentar con mayor detenimiento la naturaleza de estos conceptos y problemas, y cómo han sido abordados en la literatura, para describir posteriormente la aproximación llevada a cabo en este trabajo.

El Capítulo 2 es una introducción a los sistemas de recomendación. Se describen sus principales características y utilidades, sus ventajas y los fundamentos de estas técnicas. Se presentan y describen las dos tareas principales de los sistemas de recomendación: la predicción y la recomendación, resaltando las diferencias entre ellas, y se introducen los distintos tipos de sistemas de recomendación existentes. Finalmente se presentan las métricas y metodologías de evaluación más populares.

En el Capítulo 3 profundizamos sobre la técnica de filtrado colaborativo, la cual concentrará nuestros esfuerzos en el resto de esta tesis. Describiremos en detalle su

funcionamiento e introduciremos la notación que se utilizará en el resto de este documento. Se hará un breve repaso al estado del arte, destacando las principales técnicas y algoritmos existentes, y se introducirán las principales limitaciones y problemas de las mismas.

En la **segunda parte** se presentarán un conjunto de mejoras relativas a la calidad de los sistemas de filtrado colaborativo, con el objetivo de conseguir algoritmos precisos pero sin perder de vista otros problemas como la eficiencia, escalabilidad o la recomendación a los nuevos usuarios del sistema. En esta parte nos centramos, por tanto, en el desarrollo de nuevas técnicas de recomendación que permitan conseguir recomendaciones de calidad minimizando el impacto de los problemas que afectan a las técnicas tradicionales.

En primer lugar, en el Capítulo 4, abordaremos la tarea de predicción. Presentaremos un novedoso algoritmo, el filtrado colaborativo basado en tendencias, que destaca por su sencillez, exactitud en la predicción y gran eficiencia. Se estudiará también la evaluación de esta tarea, presentando nuevas métricas, y analizaremos los principales problemas de las técnicas existentes y las características y ventajas de nuestra propuesta. Dado que gran parte de esta tesis se centrará en los algoritmos basados en vecinos, presentaremos también un estudio detallado de sus características, ventajas y limitaciones.

Posteriormente, en el Capítulo 5, nos centraremos en la otra tarea en la cual los sistemas de recomendación son comúnmente empleados: la recomendación. Destacaremos las limitaciones de los trabajos actuales, y propondremos una nueva metodología de evaluación basada en la utilizada para evaluar los sistemas de recuperación de información. Finalmente, optimizaremos los algoritmos basados en vecinos para mejorar su comportamiento en la recomendación, presentando diversas variantes que dan lugar a importantes mejoras.

En el Capítulo 6 abordaremos el problema de *cold-start*. Tras una introducción a la naturaleza del problema y a las soluciones más relevantes que podemos encontrar en la literatura, presentaremos una nueva técnica, la expansión del perfil, que permite reducir de manera importante su impacto sin necesidad de más información que aquella ya disponible acerca de las preferencias de los usuarios.

Finalmente, en la **tercera y última parte**, nos centraremos en la escalabilidad y eficiencia de los algoritmos de filtrado colaborativo. En particular, estudiaremos cómo los algoritmos basados en vecinos pueden beneficiarse de muchas de las técnicas desarrolladas para mejorar el rendimiento de sistemas de recuperación de información.

En el Capítulo 7 comenzaremos repasando los escasos trabajos que han aborda-

do este problema, en contraposición a ámbitos como la recuperación de información, donde la eficiencia y escalabilidad siempre ha ocupado un lugar fundamental. Propondremos una implementación eficiente de los algoritmos basados en vecinos para la tarea de recomendación, y describimos cómo puede implementarse utilizando índices. Finalmente, estudiaremos las ventajas de la técnica de preselección de vecinos, comprobando que incrementa de forma importante el rendimiento sin impactar negativamente en la calidad.

En el Capítulo 8 estudiaremos cómo técnicas de compresión populares en el mundo de la RI puede ser adaptadas con éxito para comprimir la matriz de puntuaciones en filtrado colaborativo. Presentaremos los fundamentos de estas técnicas y las ventajas de su uso, destacando las mejoras tanto en rendimiento de la recomendación como en el espacio necesario para almacenar los índices con el perfil de usuarios y productos. Finalmente, presentaremos una técnica de asignación de identificadores que permite alcanzar ratios de compresión aún más elevados.

Posteriormente, en el Capítulo 9 propondremos una arquitectura distribuida para algoritmos de filtrado colaborativo basados en vecinos. Inspirada en las técnicas de partición del índice utilizadas en RI, veremos las principales ventajas de estas técnicas y cómo pueden ayudar a reducir el tiempo de respuesta en un sistema de recomendación, al tiempo que permiten incrementar el número de recomendaciones que pueden ser procesadas por unidad de tiempo.

Por último, en el Capítulo 10 se resumirán las principales conclusiones de este trabajo, y se presentarán las líneas de investigación que serán desarrolladas en futuras investigaciones.

Capítulo 2

Sistemas de recomendación

En los últimos años, el protagonismo de los sistemas de recomendación no ha dejado de crecer, gracias en gran medida a su éxito en ámbitos como el comercio electrónico. Casos sobradamente conocidos como Amazon o Netflix, por ejemplo, son simplemente dos muestras entre cientos de empresas que han sabido aprovechar los beneficios de estos sistemas.

Su principal objetivo es el de ayudar a los usuarios a elegir los productos más adecuados a sus gustos o necesidades, bien sea ofreciendo una valoración de la utilidad de determinado producto para el usuario (predicción), bien recomendando directamente un conjunto de productos adecuados a sus preferencias (recomendación).

Una de sus principales ventajas es su capacidad para ofrecer recomendaciones personalizadas, lo que consiguen gracias a que mantienen un perfil independiente para cada usuario. En la literatura podemos encontrar una gran variedad de técnicas de recomendación, cada una de las cuales utiliza distintos tipos de información acerca de productos y usuarios, y distintos métodos para generar recomendaciones a partir de esta información. Entre ellas, la técnica de filtrado colaborativo, basada enteramente en las opiniones de los usuarios, destaca como una de las más útiles y populares.

En este capítulo se describen las características fundamentales de los sistemas de recomendación y las bases de su funcionamiento. En la Sección 2.1 se introducen los elementos que componen un sistema de recomendación, presentándose en la Sección 2.2 los principales beneficios de su uso. Posteriormente, en la Sección 2.3 hablaremos de las preferencias de los usuarios y cómo son usadas por estos sistemas. La Sección 2.4 abordará las principales tareas en que se usan, y en la Sección 2.5 se describirán los principales tipos de sistemas de recomendación. Finalmente, en la Sección 2.6 se presentarán los mecanismos usados para su evaluación.

2.1. ¿Que son los sistemas de recomendación?

Los sistemas de recomendación son herramientas cuyo principal propósito es el de ayudar a los *usuarios* a elegir aquellos *productos* más adecuados a sus *preferencias*. Mientras un buscador tradicional se limita a encontrar aquello que el usuario solicita, un sistema de recomendación asiste al usuario a tomar una decisión, sea esta la compra de un producto en un portal de comercio electrónico, el alquiler de una película en un videoclub, el agregar a un amigo en una red social, etc.

Aún a riesgo de ser excesivamente simplistas, podemos decir que la tarea de un sistema de recomendación consiste en adivinar qué productos[1] son adecuados para el usuario. Naturalmente, el significado del término adecuado depende de a quién vayamos a recomendar, ya que cada individuo no sólo tendrá distintos gustos o intereses, sino que también valorará de forma muy diferente aspectos como la calidad, el precio, la disponibilidad, etc. En definitiva, el que un producto sea adecuado o no vendrá dado por las *preferencias* de cada usuario. Los sistemas de recomendación ofrecen por tanto recomendaciones personalizadas en función de un *perfil* que representa los intereses de cada individuo.

Figura 2.1: Visión general de un sistema de recomendación.

La Figura 2.1 muestra la relación entre productos, usuarios y preferencias, y representa los aspectos más importante en el funcionamiento de un sistema de recomenda-

[1]Si bien en esta tesis usaremos el término *producto* para denotar el elemento a recomendar, debemos destacar que no nos estamos refiriendo únicamente a un producto en el sentido comercial del término, sino a cualquier objeto susceptible de ser recomendado.

ción. En general, podemos distinguir tres fases principales:

1. En primer lugar, la captura de las preferencias. Las preferencias reflejan el gusto o interés de cada usuario, y se adquieren a partir de su interacción con el sistema. La naturaleza de la interacción, y cómo las preferencias se derivan de la misma, es un factor que varía entre distintos sistemas. En general, se distingue entre sistemas con preferencias explícitas, en los cuales el usuario valora conscientemente los productos (por ejemplo, otorgándoles una puntuación en una determinada escala), e implícitas, en los que las preferencias se extraen a partir del uso que el usuario hace del sistema: los productos que compra, las páginas que visita, etc. En la Sección 2.3 analizaremos la naturaleza y características de las principales formas de capturar las preferencias.

2. En segundo lugar, la extracción de conocimiento, en la que el sistema interpreta la información de la que dispone para poder predecir los gustos del usuario. Si bien esta fase varía significativamente entre las distintas técnicas, la mayoría de ellas construyen un modelo más o menos complejo para representar este conocimiento, cuyos parámetros se entrenan a partir de los datos existentes. Entre estos datos destacan por su importancia las preferencias, pero también se pueden emplear fuentes de información adicional, como características de los productos, relaciones entre usuarios, etc. Las fuentes de información empleadas nos permiten distinguir entre distintos tipos de sistemas de recomendación, tal y como veremos en la Sección 2.5.

3. Finalmente, el conocimiento adquirido en la fase anterior se emplea para aconsejar o ayudar a los usuarios en la elección. Normalmente, esto se consigue gracias a la recomendación de una lista de productos adecuados a sus necesidades. Otras veces, en lugar de recomendar productos, el sistema se limita a predecir la utilidad de un producto previamente escogido por el usuario. En la Sección 2.4 se discutirá la diferencia entre ambas formas de recomendación.

2.2. Beneficios de los sistemas de recomendación

El comercio electrónico ha sido, sin duda, uno de los ámbitos en que los sistemas de recomendación han tenido más éxito, gracias a los importantes beneficios que ofrece tanto a las empresas como a sus clientes. Hoy en día, ya nadie duda de que los sistemas de recomendación han supuesto una revolución en la forma de hacer negocios en Internet [Anderson, 2006].

Desde el punto de vista del usuario, el sistema de recomendación es fundamentalmente un servicio que sirve de ayuda a la hora de buscar un producto adecuado y finalmente tomar una decisión relativa a su adquisición, es decir, ayudan al usuario a elegir. Y elegir no es ni mucho menos una tarea sencilla, especialmente teniendo en cuenta el exceso de información y opciones del que disponemos hoy en día. De hecho, para hacernos una idea de la importancia de un sistema de recomendación, sólo tenemos que considerar el esfuerzo necesario para llevarla a cabo en su ausencia. En su libro *The paradox of choice : why more is less*, Schwartz [2004] estudió los principales problemas que los usuarios nos encontramos hoy en día a la hora de elegir. Según él, una buena elección consta de los siguientes pasos:

1. Conocer y establecer nuestros 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 nuestros objetivos.
5. Escoger la opción ganadora.
6. Tener en cuenta las consecuencias de la elección para modificar los objetivos, la importancia asignada a cada uno de ellos, y la forma de evaluar futuras posibilidades.

Cuando el número de opciones aumenta, también lo hace el esfuerzo necesario para tomar una buena decisión. Un sistema de recomendación, sin embargo, automatiza la mayoría de pasos anteriores.

Para empezar, el establecer y evaluar nuestros objetivos es ya una tarea compleja. Por ejemplo, si quiero elegir "una película que me guste para ver hoy", podré realizar luego búsquedas por género, director o reparto, pero plasmar en una consulta objetiva un concepto totalmente subjetivo, como es el gusto particular de un individuo, no es nada fácil. Un sistema de recomendación, en cambio, puede extraer e interpretar cuáles son nuestros objetivos a partir de nuestras preferencias anteriores, ahorrándonos a nosotros el tiempo necesario para llevar a cabo esta tarea.

Además, en general las personas vamos a tener unos objetivos sesgados, basados en gran medida en nuestras experiencias anteriores, lo que nos puede llevar a ignorar determinados aspectos que luego descubriremos como importantes. Por ejemplo, en la compra de un piso podemos considerar aspectos como los metros cuadrados o el número

de habitaciones, pero olvidarnos de otros como la cercanía a una parada de autobús o metro. Otras veces, nuestro desconocimiento del dominio hace que directamente nos veamos incapaces de establecer nuestros objetivos. Por ejemplo, si no tengo ni idea de ordenadores, puede ser difícil trasladar mis necesidades a un conjunto de características técnicas, con lo que tendré o bien que buscar información que me ayude a adquirir ese conocimiento, o bien buscar el consejo de otras personas. En cambio, un sistema de recomendación puede tener en cuenta la relación entre nuestros gustos y los de otros usuarios, ofreciéndonos recomendaciones que no sólo están basadas en nuestras experiencias y conocimiento, sino en el de miles de personas.

Por tanto, el sistema también nos ayuda a buscar y seleccionar los productos más adecuados a nuestras necesidades. Automáticamente, el sistema valora las distintas opciones, y nos presenta un número limitado de productos ya adecuados a nuestros intereses. Finalmente, podremos reflejar nuestra opinión sobre la recomendación aportada, de forma que el sistema automáticamente la tenga en cuenta en futuras recomendaciones.

En definitiva, un sistema de recomendación ayuda al usuario a superar el exceso de información y opciones, adaptando las recomendaciones a sus preferencias. Para un usuario, por tanto, el sistema de recomendación supondrá tres grandes beneficios:

- Permite un importante **ahorro de tiempo**. En primer lugar, el sistema identifica los objetivos e intereses del usuario a partir de su interacción pasada o la relación entre esta y la de otros usuarios. No es necesario, por tanto, que el usuario establezca y valore sus objetivos, o que busque formas de comunicárselos al sistema, como sucede, por ejemplo, en las consultas que se hacen en un buscador. A medida que el usuario interactúa con el sistema, el conocimiento que este tendrá sobre sus gustos y preferencias será cada vez más completo, concretándose en mejores recomendaciones, pero también en una adaptación a los posibles cambios en los intereses del usuario. Además, el usuario no necesita buscar y evaluar los distintos productos que puedan ajustarse a sus necesidades, sino que el sistema lo hace automáticamente.

- Aumenta la **satisfacción** del usuario. Diversas causas pueden hacer que los usuarios nos acabemos arrepintiendo tras haber adquirido un producto: no haber considerado un determinado aspecto o característica que luego resulta ser más importante de lo que habíamos pensado, el que realmente el producto no cumpla con nuestras expectativas, o incluso que poco después nos encontremos con que había una opción mejor. En definitiva, nuestras limitaciones a la hora de seleccionar los objetivos y valorar las distintas opciones puede llevarnos a escoger la

opción equivocada. Sin embargo, al tener en cuenta no sólo nuestra experiencia previa, sino también la de otros usuarios y/u otras fuentes de información, el sistema podrá tomar una decisión más adecuada, lo que repercute en el grado de satisfacción del usuario. Otras veces, el arrepentimiento puede deberse a haber perdido una buena oportunidad, por ejemplo, si nos enteramos de una gran oferta cuando esta ya ha pasado. Un sistema de recomendación también puede estar pendiente de estas oportunidades, notificándonos de aquellos productos u ofertas que considera adecuadas a nuestros intereses.

- Permite encontrar **productos novedosos o inesperados**. En ausencia de un sistema de recomendación, el número de productos a considerar va a estar limitado a aquellos que conocemos, bien por nuestras experiencias pasadas o por las de nuestros amigos, bien por una campaña publicitaria, porque los hayamos visto en el escaparate de una tienda de nuestro barrio o ciudad, etc. Por el contrario, un sistema de recomendación considera muchos más factores, y puede acabar recomendándonos productos en los que nunca habíamos pensado, bien directamente por desconocimiento del producto, bien porque nunca pensamos que tal producto pudiese ser adecuado para nosotros. En general, este tipo de recomendaciones pueden ser una grata sorpresa y convertirse en uno de los factores principales para la adopción y uso del sistema de recomendación.

Por otra parte, la utilidad de un sistema de recomendación no se limita a los usuarios, sino que también es útil desde el punto de vista del comercio o empresa que lo utiliza para vender sus productos [Ricci et al., 2011a]:

- **Incrementa las ventas.** Los sistemas de recomendación ayudan al usuario a encontrar los productos que desean, por lo que las posibilidades de que adquieran un producto aumentan. Además, a la larga el usuario reconocerá el acierto del sistema de recomendación y se fiará cada vez más de sus sugerencias, por lo que dudará menos a la hora de adquirir un producto, y preferirá adquirir nuestros productos a los de la competencia.

- **Diversifica las ventas.** Al recomendar productos menos populares, el sistema de recomendación ayuda a vender productos poco conocidos, lo que a su vez permite al comercio ampliar su catálogo sin miedo a que muchos productos se queden olvidados en el almacén. Esto es especialmente útil en la venta a gran escala por Internet, y facilita un modelo de negocio basado en la diversidad de opciones en lugar de limitarse a productos muy populares y de fácil salida.

- **Aumenta la satisfacción del usuario**, lo cual no sólo es una ventaja para el usuario, sino también para la empresa, que verá como se reducen las devoluciones al tiempo que mejora su reputación y buena imagen.

- **Incrementa la fidelidad de los usuarios**. Las recomendaciones producidas por el sistema mejoran significativamente a medida que van conociendo al usuario. Cuando mayor sea su interacción con el sistema, mejores las recomendaciones. Esto ayuda a fidelizar a los clientes, que cada vez se van a ver más satisfechos y por tanto nos preferirán frente a otras opciones, al tiempo que será más reacio a empezar de cero con otro sistema.

- **Permite entender mejor los deseos del usuario.** El conocimiento que tiene el sistema sobre las preferencias del usuario puede usarse para otras tareas aparte de la recomendación, como por ejemplo mejorar la oferta de cierto tipo de productos, su disponibilidad, etc.

2.3. Preferencias implícitas y explícitas

Las preferencias de los usuarios son una de las principales fuentes de información usadas por los sistemas de recomendación, pues es a partir de ellas que el sistema obtiene información acerca de los gustos e intereses del usuario. Las preferencias pueden ser explícitas, en las que el usuario ofrece conscientemente su opinión sobre un producto, o implícitas, en las que esa opinión se extrae a partir de la interacción del usuario con el sistema.

En un sistema con preferencias explícitas, el usuario se encarga de valorar los distintos productos según su opinión. Generalmente, se utiliza un mecanismo basado en *puntuaciones*, en el que el usuario puntúa cada producto según su satisfacción con el mismo. A mayor número de puntos, mayor satisfacción. Por ejemplo, el sistema de recomendación de películas empleado por Netflix[1] utiliza una escala de 1 a 5 puntos, donde el usuario otorgaría una puntuación de 1 cuando una película no ha sido de su agrado, y 5 cuando le ha gustado mucho, permitiendo las puntuaciones intermedias (2, 3 y 4) reflejar distintos grados de satisfacción.

Por otra parte, en un sistema con preferencias implícitas, la opinión del usuario se infiere a partir del uso que hace de la aplicación. En la práctica se han usado una gran variedad de eventos como medida de la relevancia de un producto para el usuario [Nichols, 1998], cada uno de los cuales puede ser más o menos adecuado dependiendo

[1] http://www.netflix.com/

del dominio. Por ejemplo, los productos que el usuario adquiere o busca en un portal de comercio electrónico [Huang et al., 2007; Linden et al., 2003], los enlaces seguidos por un usuario en una página web [Das et al., 2007; Joachims et al., 2005; Jung et al., 2007], el tiempo invertido en la lectura de una noticia o artículo [Konstan et al., 1997], etc.

Entre ambos tipos de preferencias existe una importante diferencia, y es que las preferencias implícitas no reflejan directamente el grado de importancia para el usuario. Por ejemplo, que un usuario haya comprado un libro no implica que realmente le guste, y por tanto el sistema debe considerar este matiz al abordar el problema. En la literatura podemos encontrar numerosas técnicas diseñadas específicamente para sistemas con preferencias implícitas [Dou et al., 2007; Lee et al., 2008; Rendle et al., 2009; Wang et al., 2006b]. Recientemente se ha popularizado el término *One-Class Collaborative Filtering (OCCF)* [Pan et al., 2008] para denotar esta clase de sistemas.

Además, si bien la diferencia entre preferencias explícitas e implícitas es la más evidente, en la práctica el rango o escala de puntuaciones empleadas tiene también un gran impacto en el sistema, tanto en la cantidad de información que puede obtenerse de las preferencias, como en la forma en que el usuario podrá interactuar con el sistema [Sparling y Sen, 2011]. Precisamente, el elegir la escala de puntuaciones a emplear es uno de los problemas que nos podemos encontrar a la hora de diseñar un sistema basado en preferencias explícitas. El sistema más sencillo es el que sólo tiene una posible puntuación, que los usuarios otorgarán a aquellos productos que consideran más relevantes. Ejemplos de este modelo serían el "me gusta" de Facebook[1], o el "+1" usado por Google+[2]. Una extensión a este modelo sería permitir que el usuario también pueda expresar una opinión negativa sobre ciertos productos, añadiendo una segunda puntuación. Por ejemplo, este es el modelo usado por YouTube[3], donde el usuario puede escoger entre "me gusta" o "no me gusta". Finalmente, si queremos posibilitar que el usuario refleje con mayor precisión su opinión, deberemos permitir un mayor número de puntuaciones. Por ejemplo, las 5 disponibles en el ya mencionado Netflix. Normalmente se utiliza un sistema de puntuaciones discretas, en las que hay un número limitado de puntuaciones válidas, aunque en la literatura podemos encontrar aplicaciones que hacen uso de puntuaciones continuas, en las que el usuario otorga una puntuación arbitraria dentro de un cierto rango [Goldberg et al., 2001].

Cada una de las distintas alternativas tiene sus ventajas e inconvenientes, y en la práctica un tipo de preferencias puede funcionar bien en un dominio, pero ser poco

[1] http://www.facebook.com/
[2] https://plus.google.com/
[3] http://www.youtube.com/

adecuado para otro [Cosley et al., 2003]. Durante el diseño de un sistema de recomendación, se deberá llegar a un compromiso entre aspectos como la información que cada tipo de preferencias aporta, la forma en que estas van a ser capturadas y su disponibilidad, los posibles errores o imprecisiones, etc. En la Figura 2.2 se representan las principales características de cada tipo de preferencias.

Figura 2.2: Características de las diferentes formas de capturar las preferencias de los usuarios.

La mayor ventaja de las preferencias implícitas es su mayor disponibilidad, pues durante el uso normal de la aplicación el usuario lleva a cabo numerosas acciones a partir de las cuales se pueden extraer las preferencias. Es decir, el sistema puede obtener información sobre los gustos o intereses del usuario sin necesidad de solicitársela en ningún momento. Por el contrario, las preferencias explícitas necesitan que el usuario esté dispuesto a transmitir su opinión. En principio, esto no debería suponer un problema, ya que el usuario obtiene una recompensa de su participación: mejores recomendaciones. Sin embargo, el interés del usuario no será el mismo en una aplicación que usa habitualmente y de cuyas recomendaciones obtiene un gran beneficio, que en otra aplicación que usa muy de vez en cuando y donde las recomendaciones tienen menor importancia para él. Por tanto, es necesario considerar un cuidadoso diseño de la interfaz de usuario y del rango de puntuaciones a utilizar, alcanzando un compromiso entre la participación de los usuarios y el grado de información aportada por las puntuaciones que se adecúe a cada aplicación.

En general, el uso de puntuaciones unarias o binarias permite simplificar la interfaz de usuario, y facilitará que este transmita su opinión. Sin embargo, tienen el problema de que no permiten reflejar distintos grados de satisfacción, con lo que el usuario puede

acabar marcando como relevante todo tipo de productos, desde aquellos que realmente le gustan o interesan, hasta algunos que simplemente le hicieron gracia o le llamaron la atención en un momento dado. En el caso de preferencias unarias, por ejemplo, no es posible siquiera reflejar opiniones negativas, es decir, el sistema sólo tiene constancia de aquellos productos que han interesado al usuario. En este aspecto, la información que aportan es similar a la aportada por las preferencias implícitas, pero es importante no confundirlas con aquellas, pues en las puntuaciones unarias sí es el usuario el que conscientemente transmite esa información al sistema.

Por otra parte, el uso de diferentes niveles de puntuaciones permite reflejar con mayor exactitud el grado de importancia que un producto tiene para el usuario. Sin embargo, son en general mucho más sensibles a la variabilidad en las puntuaciones, tanto en las diferencias de apreciación entre individuos, como en las diferencias entre la puntuación que un mismo individuo da a un cierto producto en distintos momentos. Esta variabilidad [Amatriain et al., 2009; Hill et al., 1995] se debe a diferentes factores, pudiéndose distinguir entre cuatro tipos de variaciones:

- Apreciación y forma de puntuar. Cada individuo tiene su propia forma de puntuar, es decir, de relacionar una posible puntuación con el valor o utilidad real de un producto. Hay usuarios que acostumbran a dar puntuaciones positivas, utilizando puntuaciones negativas para productos realmente malos. Otros, sin embargo, reservan las puntuaciones más altas para los mejores productos, y suelen dar puntuaciones negativas [Herlocker et al., 1999; Resnick et al., 1994],

- Variación con el tiempo [Xiang y Yang, 2009]. Los intereses o gustos de una persona van a variar con el paso del tiempo, cambios que pueden deberse a la propia evolución de la persona (por ejemplo, no nos gustan las mismas películas de niños que de adultos), o bien relacionados con cambios de tendencia o modas de la sociedad. También puede cambiar nuestra forma de puntuar, y pasar de dar buenas puntuaciones a casi cualquier producto a ser mucho más exigentes.

- Variación con el contexto [Ellenberg, 2008]. Nuestra opinión sobre un producto puede variar según el contexto en que lo hemos disfrutado o nuestro estado de ánimo. Por ejemplo, si un usuario acaba de ver una buena película, a la que en consecuencia ha otorgado una buena puntuación, y justo después ve otra película todavía mejor, esta última posiblemente vaya a recibir una puntuación mayor que la que hubiese recibido en caso de que el usuario no hubiese visto la primera película. En este caso, el hecho de haber realizado una puntuación previa hace

que el usuario, consciente de que la segunda película es mejor, tienda a subir la puntuación de esta última.

- Error de precisión. En un sistema con un gran número de niveles, puede no ser sencillo asociar una puntuación a un producto. Por ejemplo, si tenemos que valorar un libro en un sistema con dos niveles, siempre otorgaremos la misma puntuación en función de si nos ha gustado o no, pero si tenemos 50 puntuaciones diferentes, es fácil hoy le demos una puntuación de 34 y otro día una de 33, aún cuando nuestra opinión sobre él siga siendo la misma.

En general, cuanto mayor sea el número de posibles puntuaciones podremos capturar más matices de los productos, pero también estaremos más expuestos a errores debidos a la variabilidad. Como veremos en el Capítulo 5, una aproximación que se ha tenido en cuenta en esta tesis es usar el sistema de escala pero sin preocuparse demasiado del valor concreto de cada puntuación.

2.4. Predicción y recomendación

Desde su aparición hace cerca de dos décadas, la forma en que un sistema de recomendación puede ayudar a los usuarios ha sido objeto de diferentes propuestas, tanto en la literatura como en sistemas comerciales. Una clasificación que ha adquirido cierta popularidad es la presentada por Herlocker et al. [2004], quienes distinguían hasta seis formas de recomendación diferentes:

1. *Annotation in context*, donde los productos visualizados por el usuario son anotados por el sistema con una puntuación que refleja el interés de los mismos de acuerdo a los gustos del usuario.

2. *Find Good Items*, donde el sistema sugiere una lista de productos al usuario.

3. *Find All Good Items*, similar a la anterior, pero donde el sistema sugiere todos los productos que considera adecuados según las preferencias del usuario.

4. *Recommend Sequence*, donde se recomiendan sucesivamente diferentes productos, siendo interesante para el usuario el conjunto o combinación de todos ellos.

5. *Just Browsing*, donde el objetivo del sistema es ayudar al usuario a navegar entre un conjunto de opciones o productos.

6. *Find Credible Recommender*, donde el objetivo principal es que el usuario vea que el sistema de recomendación funciona bien.

No todas ellas, sin embargo, han gozado del mismo grado de popularidad. Sin lugar a dudas, han sido las dos primeras, con sus variantes, las que más éxito han tenido, tanto por su amplio uso en diferentes dominios y aplicaciones, como por el interés despertado en la comunidad científica. Podemos decir, por tanto, que a pesar de que los sistemas de recomendación se pueden emplear en distintas tareas, en la práctica se usan principalmente para **predicción** y **recomendación**, tareas que corresponderían a grandes rasgos con *annotation in context* y *find good items*, respectivamente.

En la tarea de **predicción**, el usuario es el que escoge un determinado producto, quizás tras una búsqueda por contenido, o tras haber consultado un catálogo. El objetivo del sistema es predecir la puntuación que el usuario habría dado a dicho producto, es decir, la utilidad del producto para el usuario.

Precisamente, la principal aplicación de estos sistemas es ayudar a evaluar la calidad de un producto, ahorrando al usuario el tiempo necesario para evaluar las características de un cierto producto, cómo las mismas se ajustan a sus preferencias, etc. Sin embargo, el usuario todavía es responsable de localizar aquellos productos cuya puntuación desea conocer, por lo que el sistema de recomendación tiene en cierto modo un papel limitado en el proceso y no permite disfrutar de todas las ventajas que un sistema de recomendación podría aportar. Son, por así decirlo, una primera aproximación a la recomendación. En ciertos casos esto puede ser suficiente, por ejemplo si el número de opciones a considerar no es demasiado elevado, o en dominios donde para el usuario no supone un problema escoger por sí mismo el producto. Sin embargo, en general estos sistemas se utilizan como complemento a otros mecanismos de selección de productos (como buscadores, clasificación por categorías, etc.) o incluso a la tarea de recomendación. En particular, es bastante habitual complementar una lista de productos recomendados con una estimación de la utilidad de cada producto desde el punto de vista del usuario, es decir, el combinar predicción y recomendación.

En la tarea de **recomendación**, por otra parte, es el sistema el que se encarga de seleccionar los productos que pueden interesar al usuario, los cuales le serán mostrados en forma de una lista o conjunto de productos recomendados. El usuario, por tanto, ahorra el tiempo de buscar y seleccionar entre un gran número de opciones, y únicamente tendrá que escoger entre las presentadas por el sistema. La recomendación, por tanto, es en cierto modo una forma más avanzada de recomendación.

En la práctica, los productos recomendados se pueden presentar de dos formas: como una lista ordenada o como un conjunto de opciones sin un orden específico. En la literatura se ha preferido el método de lista ordenada en caso de sistemas basados en puntuaciones, y el conjunto en caso de sistemas basados en otro tipo de preferencias

(por ejemplo, implícitas o unarias). En parte esto se debe a que en el primer caso la recomendación se ha abordado muchas veces a partir de la predicción: el sistema se usa para predecir la utilidad de todos los productos, y estos se ordenan según la puntuación otorgada. Sin embargo, si bien es cierto que un sistema diseñado para predicción se puede usar también para recomendación, en la práctica es habitual que los resultados obtenidos sean mediocres [Cremonesi et al., 2010], aparte de que esta forma de recomendar no suele ser especialmente eficiente pues requiere de un gran número de cálculos que no son realmente necesarios.

De hecho, ambas tareas presentan diferentes peculiaridades y problemas, y generalmente es preferible abordarlas con técnicas diferentes. En estos casos, presentar los resultados de la recomendación como un conjunto no ordenado suele ser una opción más sencilla, sujeta a menos errores y, por otra parte, perfectamente válida desde el punto de vista del usuario, siempre y cuando, por supuesto, el número de productos recomendados no sea demasiado elevado.

2.5. Tipos de sistemas de recomendación

A lo largo de las dos últimas décadas se han desarrollado multitud de técnicas de recomendación diferentes, las cuales han hecho uso de distintos tipos de información sobre productos y usuarios. Según la información que utilizan, se puede distinguir entre seis tipos principales de sistemas de recomendación [Ricci et al., 2011a]:

- **Basados en contenido.** Estos sistemas recomiendan productos con contenido similar a aquellos que le habían gustado anteriormente al usuario [Mladenic, 1999]. De cada usuario se guarda un perfil que representa el contenido en que este está interesado. Por ejemplo, un sistema de recomendación de películas podría recomendar aquellas cuyo género, director, actores o actrices, etc. son similares a los de las películas que le gustan al usuario. La mayor parte de las técnicas basadas en contenido se han utilizado para la recomendación de productos cuyo contenido consiste en texto, como páginas web [Chen y Sycara, 1998; Lieberman, 1995], noticias [Ahn et al., 2007], libros [Mooney y Roy, 2000], artículos científicos [Bollacker et al., 1998], etc. En general, se utilizan técnicas relacionadas con la recuperación de información [Manning et al., 2008], como el modelo vectorial, métodos de ponderación como TF-IDF, modelos probabilísticos sencillos como Naïve Bayes, etc.

  Aunque se comportan relativamente bien con contenido basado en texto, presentan varios problemas [Shardanand y Maes, 1995], como las limitaciones a la

hora de interpretar cierto tipo de contenido (por ejemplo, contenido multimedia), su incapacidad para generar recomendaciones inesperadas (ya que los productos recomendados serán similares al perfil del usuario), o la falta de calidad de las recomendaciones (que dos artículos usen términos similares no quiere decir que los dos sean igual de buenos).

- **Filtrado colaborativo.** Los sistemas de filtrado colaborativo [Goldberg et al., 1992] se basan enteramente en las opiniones de otros usuarios. Por tanto, el perfil del usuario estará compuesto únicamente por sus preferencias. Los productos a recomendar se elegirán entre aquellos que han recibido una puntuación elevada por parte de otros usuarios con similares gustos o intereses, es decir, con un patrón de preferencias similar. En la literatura se pueden encontrar técnicas basadas en la correlación entre usuarios o productos [Shardanand y Maes, 1995], así como técnicas más complejas basadas en el aprendizaje automático [Marlin, 2004]. La gran ventaja del filtrado colaborativo es que, gracias a las preferencias, son las personas las que valoran la calidad de los productos a recomendar, lo que da lugar a muy buenos resultados. Debido a que esta tesis se centra en este tipo de sistemas de recomendación, en el Capítulo 3 se describirá con más detalle y se hará un repaso de los métodos más populares.

- **Basados en información demográfica.** Estos sistemas de recomendación agrupan a los usuarios teniendo en cuenta información demográfica, como edad, sexo, país o población, etc. Un usuario recibirá como recomendación productos que han gustado a usuarios con un perfil demográfico similar. Por ejemplo, Krulwich [1997] desarrollaron un sistema que dividía a los usuarios en 62 grupos diferentes según diversas características demográficas, elegidos a partir de encuestas, datos de subscripciones a revistas, etc. A pesar de la popularidad de este tipo de información en ámbitos como el *marketing*, lo cierto es que su uso en sistemas de recomendación es más bien escaso, en parte debido a que no consiguen el nivel de personalización que es posible obtener usando otras técnicas.

- **Basados en conocimiento sobre el dominio.** Estos sistemas utilizan información específica sobre el dominio, modelando aquellas características que determinan las preferencias de los usuarios. Generalmente, la interacción entre usuario y aplicación sigue un diálogo o conversación [Thompson et al., 2004] en el que el sistema pregunta al usuario sobre ciertos aspectos, y a partir de esta información se obtienen los productos que cumplen las características deseadas. Por ejemplo, en la recomendación de productos financieros [Felfernig et al., 2007], el

sistema preguntaría al usuario sobre su disposición a asumir riesgos, el periodo de inversión, etc. y recomendaría aquellos productos adecuados a sus respuestas. Alternativamente, la interacción puede seguir la idea de la crítica, en donde se presentan productos al usuario y este varía ciertas características (por ejemplo: solicita productos más baratos) hasta obtener el producto ideal [Burke, 2000].

Existen dos técnicas principales: la recomendación basada en casos [Lorenzi y Ricci, 2005; Mirzadeh et al., 2005], donde se almacenan las características de los productos y estas se comparan con las seleccionadas por los usuarios, recomendando los productos más similares, y la basada en restricciones [Felfernig y Burke, 2008], en donde en base a las preferencias de los usuarios se van descartando aquellos productos incompatibles con estas hasta que se tenga un número limitado de opciones. En general, este tipo de sistemas de recomendación son muy útiles en dominios donde es complicado adquirir un gran número de preferencias (por ejemplo, la compra de inmuebles). Por otra parte, en general no obtienen el conocimiento de forma automática, sino que necesitan de la ayuda de expertos, y además la interacción necesaria por parte del usuario es elevada, por lo que su uso se reduce principalmente a aplicaciones donde el beneficio de la recomendación es muy grande.

- **Basados en comunidades.** Este tipo de sistemas de recomendación se basa en las relaciones de confianza que existen entre usuarios. La idea es recomendar productos a un usuario a partir de las preferencias de sus amigos. Gracias al auge experimentado por las redes sociales en los últimos años, este tipo de sistemas ofrecen una forma sencilla de adquirir información sobre los usuarios, y de ahí que su importancia no haya dejado de crecer. De todas formas, la investigación en este campo todavía está en fases muy tempranas, aunque algunos artículos destacan los buenos resultados de estas técnicas, con resultados similares a otros métodos más tradicionales, e incluso mejores en ciertos contextos [Golbeck, 2006; Groh y Ehmig, 2007; Guy et al., 2009].

- **Híbridos.** Los sistemas híbridos combinan dos o más técnicas de las mencionadas anteriormente, con el objetivo de aprovechar las ventajas de cada una de ellas e intentar evitar o al menos minimizar el impacto de sus limitaciones. Por ejemplo, para minimizar el problema de *cold-start* [Schein et al., 2002] que sufren las técnicas basadas en filtrado colaborativo, se puede utilizar información basada en contenido [Melville et al., 2002].

  En la literatura podemos encontrar diferentes formas de combinar las distintas

técnicas [Burke, 2007]: (1) ponderar los resultados obtenidos por cada técnica para obtener la recomendación final [Pazzani, 1999]; (2) alternar entre los distintos métodos según cierto criterio que determine qué técnica obtendrá mejores resultados [Tran y Cohen, 2000]; (3) presentar al mismo tiempo productos recomendados por diferentes técnicas [Smith y Cotter, 2000]; (4) combinar las características derivadas de diferentes fuentes de información y usarlas conjuntamente [Basu et al., 1998]; (5) usar varias técnicas en cascada, donde los resultados de una técnica son posteriormente refinados por otra [Burke, 2007]; (6) usar una técnica para aumentar la información que usará otra, por ejemplo, el uso de un sistema basado en contenido para inferir preferencias luego usadas por un sistema de filtrado colaborativo [Good et al., 1999]; y finalmente (7) utilizar una técnica para derivar un cierto modelo posteriormente usado como entrada de una segunda aproximación [Balabanović y Shoham, 1997].

2.6. Evaluación de los sistemas de recomendación

La evaluación de un sistema de recomendación es una tarea compleja, pues la calidad de una recomendación va a depender de un gran número de factores, incluyendo la persona que la recibe, el dominio y objetivo del sistema, etc. Al contrario que otro tipo de sistemas, donde está relativamente claro qué parámetros se deben mejorar, la idoneidad de un sistema de recomendación depende muchas veces de llegar a un compromiso entre puntos de vista a menudo distantes.

Por ejemplo, ciertos usuarios preferirán sistemas muy precisos, que siempre les recomienden productos realmente adecuados, mientras a otros usuarios no les importará que el sistema se equivoque de vez en cuando a cambio de recomendaciones mejores o más variadas. Por supuesto, el coste de un error depende también del dominio: no es lo mismo un error en una recomendación de películas para alquilar que en una de fondos de inversión.

Además, los intereses de los distintos actores son en cierto modo contrapuestos: un cliente estará fundamentalmente interesado en que la recomendación le ayude a hacer una buena compra, mientras un vendedor tendrá como objetivo fundamental incrementar los beneficios. Al igual que en el caso de los clientes, la mejor forma de aumentar beneficios también dependerá del dominio: en ciertos casos lo ideal es que el sistema simplemente incremente las ventas, en otros el beneficio pasa por reducir las devoluciones, diversificar los productos vendidos, etc.

Si a esto unimos que muchas veces los algoritmos se evalúan en el laboratorio, antes

de ser utilizados en un dominio concreto, es de esperar que las métricas y formas de evaluación que podemos encontrar en la literatura sean de lo más diversas [Shani y Gunawardana, 2011].

2.6.1. Experimentos y conjuntos de datos

En cuanto al tipo de experimentos a realizar, en general podemos distinguir entre experimentos *offline*, en los cuales no intervienen usuarios reales, sino que se utilizan conjuntos de datos previamente almacenados, y experimentos *online*, donde sí se cuenta con usuarios que interactúan con el sistema.

La principal ventaja de la evaluación *offline* es la comodidad y coste, pues permiten comparar un gran número de técnicas y opciones sin salir del laboratorio. Otra ventaja que tienen es que los experimentos pueden repetirse en cualquier momento, lo que simplifica la comparación entre distintos algoritmos. La evaluación parte de un conjunto de datos previamente almacenado, que recoge información acerca de las preferencias u opiniones de un cierto número de usuarios sobre un conjunto de productos. Generalmente se utilizan conjuntos de datos seleccionados a partir de sistemas reales, que son públicos y por tanto pueden ser usados por distintos grupos de investigación, facilitando la comparación de resultados. En el ámbito de los sistemas de recomendación, y especialmente la técnica de filtrado colaborativo, son muy populares dos conjuntos de datos pertenecientes a sistemas de recomendación de películas: Movielens [Herlocker et al., 1999] y Netflix [Bennett y Lanning, 2007].

El principal problema de la evaluación *offline* es que sólo permite evaluar ciertos aspectos de las técnicas de recomendación. Además, la información disponible sobre cada usuario es limitada, lo que lleva a asumir que el comportamiento del usuario es el reflejado por los datos de que se dispone, sin posibilidad de considerar la influencia del sistema de recomendación, o la calidad de productos que el usuario no había valorado originalmente.

La alternativa es el uso de experimentos *online*, contando con usuarios reales que interactúan con el sistema. La evaluación puede ser llevada a cabo por usuarios contratados específicamente para realizarla, posiblemente en un entorno previo a la puesta en marcha del sistema, o totalmente *online*, estudiando el comportamiento de los usuarios reales de la aplicación.

2.6.2. Métricas de evaluación

Otro problema está en seleccionar qué aspectos del sistema se desean evaluar. Como hemos visto, las características que debe tener un sistema serán diferentes según el do-

minio o el punto de vista de cada actor implicado. En general, las métricas destinadas a evaluar la precisión del sistema de recomendación son las más empleadas en la literatura [Cacheda et al., 2011a], si bien algunos autores consideran que se deben evaluar otro tipo de aspectos más relacionados con la calidad subjetiva de las recomendaciones [McNee et al., 2006]. En la práctica, el uso de diferentes tipos de métricas permite hacerse una idea de los puntos fuertes (y debilidades) de una técnica de recomendación en particular, pero su adecuación a un dominio o aplicación concreta dependerá de los objetivos que predominen en la elección del algoritmo. Cada caso puede requerir una técnica diferente.

2.6.2.1. Precisión del sistema de recomendación

Las métricas más extendidas en la evaluación de sistemas de recomendación son aquellas que miden la precisión del mismo, es decir, la diferencia entre la recomendación ofrecida por el sistema y las preferencias reales del usuario. Según la tarea en la cual se va a usar el sistema de recomendación, predicción o recomendación, la precisión del sistema deberá ser evaluada de diferente forma. En la literatura podemos encontrar diferentes tipos de medidas de la precisión, que se pueden clasificar en tres tipos principales [Herlocker et al., 2004]:

- **Precisión en la predicción.** Estas métricas se utilizan en la tarea de predicción, y se basan en medir el error o diferencia entre la predicción hecha por el algoritmo y la puntuación real del usuario. Generalmente, el proceso de evaluación consiste en escoger un conjunto de evaluación formado por pares usuario-producto para los que se conoce la puntuación real hecha por el usuario. En un experimento *offline*, dicho conjunto estará formado por pares presentes en el conjunto de datos que se reservan para la evaluación, y por tanto no formarán parte del conjunto de entrenamiento. En una evaluación *online*, con usuarios reales, se solicitará a los usuarios que puntúen dichos productos. El sistema de recomendación deberá predecir la puntuación para cada uno de dichos pares, y esta se comparará con la puntuación real, midiendo el error cometido. Las medidas de error más populares son el *error absoluto medio* (MAE) y la *raíz del error cuadrático medio* (RMSE), las cuales serán descritas en el Capítulo 4.

- **Precisión en la clasificación.** Estas métricas, que se utilizan principalmente en la tarea de recomendación, evalúan si los productos recomendados por el algoritmo son realmente adecuados a las preferencias del usuario. Generalmente, en la evaluación se selecciona un conjunto de usuarios, y se obtiene una reco-

mendación para ellos utilizando el algoritmo. Luego se comparan los productos recomendados con aquellos que realmente son adecuados para cada usuario. Para tal fin, se pueden utilizar varias métricas, entre las que destacan la *precisión* y la exhaustividad o *recall*, que miden el ratio entre los productos relevantes que han sido recomendados, y, respectivamente, el total de productos recomendados o el total de productos relevantes. En el Capítulo 5 analizaremos en profundidad la evaluación de la clasificación y las principales métricas empleadas.

- **Precisión en la ordenación.** Este tipo de métricas evalúa el parecido entre el orden de los productos en la recomendación hecha por el sistema y la ordenación real que habría hecho el usuario. En muchos casos, este tipo de métricas es demasiado sensible, pues evalúa la capacidad del sistema para recomendar los mejores productos y en el orden en que los habría escogido el usuario, cuando en la práctica, en la mayoría de situaciones, es suficiente con recomendar al usuario una buena lista de productos, aún cuando esta no sea exactamente la que hubiese escogido el usuario. Métricas de este tipo son aquellas que miden la correlación entre la recomendación y la ordenación real (Pearson, ρ de Spearman, y τ de Kendall), la métrica *half-life utility* [Breese et al., 1998] o la *normalized distance-based performance measure* (NDPM) [Balabanović y Shoham, 1997].

2.6.2.2. Otro tipo de métricas

Si bien las métricas de precisión son las más populares, no es menos cierto que el hecho de tener un algoritmo muy preciso no implica que necesariamente vaya a ser útil para el usuario o comercio. Por ejemplo, un sistema que se limita a recomendar productos muy populares y generalmente bien valorados dará muy buenos resultados en términos de precisión, pero difícilmente podrá ser considerado un buen sistema de recomendación. En la práctica, es deseable no sólo valorar la precisión del sistema, sino también su utilidad [Herlocker et al., 2004; McNee et al., 2006], la cual involucra diferentes aspectos entre los que destacamos los siguientes[1]:

- **Cobertura o *coverage*.** Mide la capacidad del sistema para generar recomendaciones o predicciones. En el caso de la recomendación, se puede reportar la cobertura en el espacio de usuarios (es decir, el porcentaje de usuarios para los cuales el sistema es capaz de recomendar productos), y en el espacio de productos (es decir, el porcentaje de productos que el sistema es capaz de recomendar).

[1] Puede comprobarse como el ejemplo anterior de sistema que sólo recomienda productos populares no cumple la mayor parte de estas otras características

En el caso de la predicción, la cobertura indica el porcentaje de pares usuario-producto cuya puntuación es capaz de predecir el sistema. Lo ideal es tener una cobertura próxima al 100 %, pero muchas veces eso no va a ser posible debido a que no se dispone de suficiente información sobre usuarios o productos. En general es importante evaluar la cobertura de un sistema, pues algunas técnicas tienen muy buena precisión pero sólo cubren un número limitado de productos o usuarios, para los cuales se tiene mucha información. Dichos casos en general son más fáciles de tratar, y un sistema que se limitase a ofrecer este tipo de recomendaciones podría obtener unos resultados de precisión muy elevados que no se corresponderían realmente con la utilidad del sistema.

- **Recomendaciones novedosas o inesperadas.** Un sistema que únicamente recomienda productos ya conocidos por el usuario posiblemente no sea demasiado útil, por muy precisas que sean sus recomendaciones. La novedad de una recomendación [Konstan et al., 2006] mide el grado de familiaridad de un usuario con la recomendación del sistema, es decir, si el usuario ya conoce los productos recomendados. Una posible mejora a esta métrica es el considerar no sólo si un producto es o no conocido por el usuario, sino también si es inesperado. Por ejemplo, si un usuario puntúa bien películas de un cierto director, una nueva película del mismo director puede ser novedosa (el usuario aún no la ha visto), pero desde luego no es inesperada, y el usuario probablemente fuese a verla igualmente aunque el sistema no se la hubiese recomendado.

- **Confianza.** También es importante conseguir que el usuario confíe en los resultados del sistema. Errores en la recomendación pueden hacer que el usuario deje de fiarse de las recomendaciones del sistema, pero limitarse a recomendaciones más conservadoras, es decir, productos muy similares al perfil del usuario, limita el grado de novedad y utilidad del sistema. Para llegar a un compromiso entre ambos aspectos, el sistema puede incorporar a la recomendación información adicional que indique al usuario el riesgo de ciertas recomendaciones, o bien explicar al usuario el porqué de una determinada recomendación [Herlocker et al., 2000].

- **Diversidad.** Una recomendación formada por productos muy parecidos puede no ser la más adecuada Por ejemplo, recomendar todas las películas de una saga, por mucho que dichas películas sean del gusto del usuario, difícilmente puede considerarse una buena recomendación. La diversidad [Ziegler et al., 2005] mide precisamente la diferencia entre los productos recomendados. Lo ideal es un sistema que recomiende productos variados para que el usuario tenga varias op-

ciones para elegir. Es decir, un sistema de recomendación debe ayudar a limitar el número de opciones y evitar que el usuario tenga que valorar un gran número de ellas, pero no eliminarlas completamente.

- **Eficiencia y escalabilidad.** Un sistema que tarde horas en recomendar un producto, por muy buena que sea esta recomendación, será poco útil en la práctica, pues en muchos casos cuando reciba la recomendación el usuario ya habrá adquirido un producto de la competencia. En el mundo actual, donde gracias a Internet el usuario puede acceder a multitud de alternativas en cuestión de segundos, o en aplicaciones con gran dinamismo, un sistema de recomendación debe funcionar en tiempo real, es decir, debe ofrecer recomendaciones en unos pocos milisegundos. Además, las aplicaciones son usadas por cada vez más usuarios, los cuales interactúan cada vez más, generando más y más información. Hoy en día no es nada inusual encontrarnos con aplicaciones donde el número de usuarios y productos se cuenta por millones. Por tanto, la eficiencia y la escalabilidad de los algoritmos de recomendación es un factor clave para su uso en entornos reales.

Capítulo 3

Filtrado Colaborativo

Las técnicas de filtrado colaborativo son un método de recomendación especialmente popular, gracias en gran medida a los buenos resultados obtenidos en dominios como el comercio electrónico. Al estar basadas en las preferencias de otros usuarios con gustos similares, la calidad de las recomendaciones es particularmente elevada, pues al final son personas quienes evalúan cada producto.

El papel del algoritmo se centra en buscar las relaciones entre usuarios y productos, para lo cual se pueden utilizar diferentes técnicas. Generalmente, las aproximaciones propuestas en la literatura se pueden clasificar en técnicas heurísticas o basadas en memoria, y técnicas basadas en modelo. Las primeras procesan la matriz de puntuaciones en cada recomendación, buscando similitudes entre usuarios o productos, mientras las segundas definen un modelo que representa las relaciones entre usuarios y productos, el cual se entrena a partir de la matriz utilizando algoritmos de aprendizaje automático.

En este capítulo presentamos una introducción al filtrado colaborativo y a los métodos más populares. En la Sección 3.1 se describen los fundamentos de este mecanismo de recomendación, destacando en la Sección 3.2 sus principales ventajas e inconvenientes. A continuación, en la Sección 3.3, presentamos una definición formal del problema. Finalmente, la Sección 3.4 describe los fundamentos de las principales técnicas de filtrado colaborativo que pueden ser encontradas en la literatura.

3.1. Introducción

Las técnicas de filtrado colaborativo son posiblemente el método de recomendación más popular hoy en día, debido principalmente a sus numerosas ventajas y sus buenos resultados en multitud de dominios. Para elegir los productos a recomendar, estas técnicas utilizan las preferencias de otros usuarios.

Su éxito radica en que precisamente las personas utilizamos de forma natural un proceso de recomendación muy similar, transmitiendo nuestra opinión a otras personas con la esperanza de que nuestra experiencia les pueda ser útil. Por ejemplo, si la última película que hemos visto en el cine nos ha gustado especialmente, enseguida se la recomendaremos a nuestros amigos, y de igual forma, ellos nos recomendarán aquellos productos que nos pueden interesar y nos alertarán de aquellos que han supuesto una decepción. Sin embargo, por muy buenas que sean sus intenciones, no depositamos la misma confianza en las opiniones de todos ellos. Con el tiempo, nos damos de cuenta de que las recomendaciones de alguno de nuestros amigos son más útiles que las de otros. Es decir, tendemos a confiar más en las opiniones de personas con gustos o intereses similares a los nuestros.

Esta es, precisamente, la forma en que funcionan los sistemas de filtrado colaborativo. La idea es que si dos usuarios tienen un patrón de preferencias similar, se debe a que comparten gustos o intereses, y por tanto los productos que han sido de utilidad a uno de ellos pueden ser una buena recomendación para el otro. Por ejemplo, si una persona es fan de las películas de acción, dará una puntuación alta a este tipo de películas, y lo mismo harán otros usuarios con gustos similares. El sistema podrá detectar este tipo de similitudes en las preferencias de todos ellos, recomendando por tanto películas que han recibido una puntuación elevada por estos otros usuarios.

La principal diferencia entre un sistema de filtrado colaborativo y la recomendación *boca a boca* es que en lugar de restringirnos a nuestro círculo de amistades, el sistema puede tener un cuenta las opiniones de miles o incluso millones de personas, aprovechando por tanto el conocimiento y experiencias de todas ellas.

3.2. Principales ventajas y limitaciones

Las principales ventajas del filtrado colaborativo frente a otras técnicas de recomendación, como por ejemplo las basadas en contenido, derivan precisamente de estar basadas exclusivamente en las preferencias de los usuarios:

- **Independencia de usuarios y productos.** Desde el punto de vista del sistema, productos y usuarios son vistos simplemente como identificadores asociados a un conjunto de preferencias. En ningún momento, por tanto, se necesita conocer la naturaleza o características de los mismos, lo que permite que los sistemas de filtrado colaborativo puedan ser usados indistintamente en dominios diferentes. Para este tipo de sistemas, recomendar libros es lo mismo que recomendar películas. Otro tipo de técnicas, sin embargo, requieren de cierto conocimiento

sobre el dominio, lo que los hace exclusivos de aquel sistema para el que fueron diseñados, o que incluso no puedan ser usados en otros contextos. Por ejemplo, un sistema de recomendación basado en contenido podría recomendar libros analizando sus palabras o términos, pero difícilmente podría aplicarse a películas u otro contenido multimedia, pues la visión que una persona tiene del mismo es muy diferente del que puede llegar a tener una máquina (texturas, píxeles, etc).

- **Calidad evaluada por personas.** Esto nos lleva a la siguiente de las ventajas, y es que en un sistema de filtrado colaborativo la calidad de un producto está siempre evaluada por personas, pues para la recomendación sólo se tienen en cuenta las preferencias. En un sistema basado en contenido, por ejemplo, la recomendación implicará comparar el contenido de dos productos, pero es evidente que el hecho de que dos libros, por ejemplo, compartan un buen número de vocablos, no significa que su calidad vaya a ser la misma. En cambios, las preferencias reflejan la calidad del producto desde el punto de vista de un usuario, y esto permite que las recomendaciones ofrecidas por sistemas basados en filtrado colaborativo sean mucho más precisas.

- **Recomendación de productos sorprendentes o no esperados.** Al basarse enteramente en las preferencias, estos sistemas pueden recomendar productos inesperados, que no guardan relación aparente con aquellos que le gustan al usuario. Al no ser productos en principio parecidos, sea en términos de contenidos o de reglas del dominio, por ejemplo, otro tipo de sistemas de recomendación difícilmente podrían considerar tal producto una opción adecuada. Sin embargo, con filtrado colaborativo podría ser recomendado si, por ejemplo, le hubiese gustado a otro usuario cuyos gustos son similares a los nuestros. Como ya habíamos comentado, este tipo de recomendaciones son, en muchos casos, las más satisfactorias, pues el producto no se habría descubierto si no fuese gracias al sistema de recomendación.

Por otra parte, sin embargo, las técnicas de filtrado colaborativo también están sujetas a varias limitaciones, muchas de las cuales no están presentes en otro tipo de sistemas:

- **Escasez de puntuaciones.** Generalmente, el número de puntuaciones dadas por un usuario es muy bajo comparado con el número total de productos, lo cual es lógico pues un usuario apenas puntuará una parte de aquellos productos que conoce o ha experimentado, los cuales a su vez son sólo un pequeño subconjunto

de todos los disponibles en el sistema. Esto significa que la densidad de la matriz de puntuaciones va a ser muy baja[1], lo que complica la búsqueda de similitudes entre usuarios. Por ejemplo, es posible que dos usuarios sean similares pero al no haber puntuado los mismos productos, el sistema no sea capaz de averiguarlo. Al estar basados exclusivamente en las puntuaciones o preferencias, los sistemas de filtrado colaborativo son especialmente sensibles a este problema, conocido en la literatura como *sparsity problem* [Huang et al., 2004].

- ***Cold-start.*** Relacionado con el anterior, el problema de *cold-start*[Schein et al., 2002] se manifiesta con usuarios o productos recientemente introducidos en el sistema. En el caso de los usuarios, todavía no habrán puntuado demasiados productos, por lo que el sistema no podrá interpretar correctamente sus intereses dando lugar a recomendaciones poco apropiadas. En el caso de los productos, los mismos habrán recibido pocas puntuaciones y por tanto es más difícil que sean recomendados.

- **Ataques y manipulación de preferencias.** Los sistemas de filtrado colaborativo son susceptibles de sufrir ataques de *spam*, especialmente por parte de usuarios interesados en que se recomiende un determinado producto, y que introducen deliberadamente puntuaciones manipuladas para así engañar al sistema [Lam y Riedl, 2004; Mobasher et al., 2007]. Al estar enteramente basados en información introducida por los usuarios, los algoritmos de filtrado colaborativo son más sensibles a este problema que otro tipo de técnicas, si bien en la literatura podemos encontrar varias soluciones para mitigar el problema [Chirita et al., 2005; Sandvig et al., 2007].

- **Privacidad.** La captura y procesamiento de las preferencias de los usuarios por parte del sistema puede suponer un problema para los usuarios más preocupados por su privacidad, que ven como el sistema adquiere demasiada información sobre sus hábitos, gustos u opiniones. Es por ello que recientemente se han propuesto técnicas centradas en garantizar la privacidad de cada individuo [Aïmeur et al., 2008; Shokri et al., 2009].

- **Escalabilidad.** Las técnicas de filtrado colaborativo son especialmente sensibles a problemas de eficiencia y escalabilidad, en gran parte debido al coste de buscar similitudes entre los patrones de puntuaciones. Además, el número de preferencias es generalmente muy elevado en relación al número de usuarios y productos,

[1] Es decir, es una matriz muy dispersa, en la que desconocemos el valor de la mayor parte de celdas.

lo que dificulta la escalabilidad de la mayoría de técnicas. En general, técnicas demasiado complejas o que tienen en cuenta un número muy elevado de factores o características acabarán teniendo problemas de rendimiento a medida que aumenta el número de productos o usuarios en una aplicación.

3.3. Definición del problema

Un sistema de recomendación basado en filtrado colaborativo trabaja con un conjunto de usuarios, $\mathcal{U} = \{u_1, u_2, \ldots, u_m\}$, y otro de productos o elementos, $\mathcal{I} = \{i_1, i_2, \ldots, i_n\}$. Adicionalmente, el sistema guarda información sobre el perfil de los usuarios, es decir, las preferencias que estos han mostrado previamente mediante su interacción con el sistema, y que reflejan sus opiniones acerca de determinados productos. Normalmente, estas opiniones consisten en puntuaciones, pertenecientes a un conjunto $\mathcal{R}$. Sobra decir que $\mathcal{R}$ es un conjunto totalmente ordenado, y las puntuaciones reflejan el distinto grado de satisfacción del usuario con el producto. Generalmente, $\mathcal{R}$ es también un conjunto finito y por tanto puede ser representado como un subconjunto de los números naturales, es decir, $\mathcal{R} \subset \mathbb{N}$. Por ejemplo, el conjunto de puntuaciones válidas en un sistema podría ser $\mathcal{R} = \{1, 2, 3, 4, 5\}$, con 1 correspondiente a un mal producto, y 5 a un producto especialmente bueno.

Se puede, por tanto, definir la relación $r : \mathcal{U} \times \mathcal{I} \to \mathcal{R} \cup \varnothing$, la cual dado un usuario $u \in \mathcal{U}$ y un producto $i \in \mathcal{I}$, $r(u, i)$ representa la puntuación que el usuario u ha otorgado al producto i, siendo $r(u, i) = \varnothing$ en caso de que el usuario aún no haya puntuado dicho producto. Por simplicidad, denotaremos dicha relación como r_{ui}. La puntuación media de un usuario u se denotará como $\overline{r_{u.}}$, y la de un producto i, como $\overline{r_{.i}}$.

En general, el sistema almacena las distintas puntuaciones otorgadas en una estructura llamada la *matriz de puntuaciones*, $\mathcal{M} \in (\mathcal{R} \cup \varnothing)^{|\mathcal{U}| \times |\mathcal{I}|}$. Por definición, $\mathcal{M}_{ui} = r_{ui}$. Normalmente, los usuarios únicamente puntúan un pequeño subconjunto de los productos disponibles, por lo que la mayoría de celdas en la matriz tomarán el valor $\varnothing$. Se dice por tanto que la matriz es *dispersa*. Denotaremos como $\mathcal{I}^{(u)}$ el subconjunto de productos puntuados por un usuario u dado, es decir, $\mathcal{I}^{(u)} = \{i \in \mathcal{I} | \mathcal{M}_{ui} \neq \varnothing\}$, y como $\mathcal{U}^{(i)}$ al subconjunto de usuarios que han puntuado un determinado producto i. De igual forma, denotaremos como $\mathcal{I}^{(u,v)}$ el subconjunto de productos puntuados por ambos usuarios u y v, y como $\mathcal{U}^{(i,j)}$ al de usuarios que han puntuado tanto al producto i como al producto j. Es decir, $\mathcal{I}^{(u,v)} = \mathcal{I}^{(u)} \cap \mathcal{I}^{(v)}$ y $\mathcal{U}^{(i,j)} = \mathcal{U}^{(i)} \cap \mathcal{U}^{(j)}$

El *perfil de un usuario* estará compuesto por los elementos puntuados por dicho usuario, junto a la puntuación dada. Formalmente, para un usuario u, su perfil se

denota como $\mathcal{P}_u$ y se define de acuerdo a la Ecuación 3.1:

$$\mathcal{P}_u = \left\{ (i, r_{ui}) \, | i \in \mathcal{I}^{(u)} \right\} \tag{3.1}$$

Dependiendo de la tarea para la que va a ser usado, un sistema de recomendación va a venir definido por:

- Una **función de predicción**, $\hat{r} : \mathcal{U} \times \mathcal{I} \rightarrow \mathcal{R}$, la cual, dado un usuario y un producto, devuelve una puntuación que aproxima la utilidad que dicho producto tiene para el usuario.
- Una **función de recomendación**, $rec : \mathcal{U} \rightarrow \mathcal{I}^N$, la cual para un determinado usuario devolverá una lista de hasta N productos o elementos aún no puntuados por el usuario. Es decir, dado un usuario $u \in \mathcal{U}$, $rec(u) = \{i_1, i_2, \ldots, i_N | i \in \left(\mathcal{I} - \mathcal{I}^{(u)}\right)\}$. Dicha lista puede estar o no ordenada.

En ambos casos, el sistema de recomendación utilizará la información disponible en la matriz $\mathcal{M}$ para que el resultado sea adecuado a los intereses del usuario. Se denomina *usuario actual* al sujeto de la recomendación o predicción que el sistema calculará. En el resto de la tesis, se denotará con la letra a.

3.4. Principales técnicas de filtrado colaborativo

Desde que hace ya más de veinte años Goldberg et al. [1992] presentasen el uso del filtrado colaborativo en su sistema de correo electrónico *Tapestry*, se han desarrollado multitud de algoritmos y aproximaciones diferentes a este problema. Desde aquel primitivo sistema de consultas en que el usuario tenía un peso fundamental a la hora de elegir qué usuarios son más parecidos, se ha pasado al uso de avanzadas técnicas capaces de localizar relaciones ocultas entre millones de datos gracias al uso de algoritmos de aprendizaje automático [Marlin, 2004].

Tradicionalmente, en la literatura las técnicas de filtrado colaborativo se encuentran clasificadas en algoritmos basados en memoria y algoritmos basados en modelo, según la forma en que procesen la información disponible en la matriz de puntuaciones. Los algoritmos basados en memoria procesan la matriz cada vez que calculan una recomendación o predicción. Generalmente, usan medidas de similitud para encontrar usuarios o productos con un patrón de puntuaciones similar, y los utilizan para llevar a cabo la recomendación. Por otra parte, los algoritmos basados en modelo utilizan la información presente en la matriz para entrenar un modelo previamente especificado, el cual intenta reflejar características o relaciones entre productos y usuarios. Las

recomendaciones o predicciones se hacen directamente a partir de dicho modelo, sin que sea necesario volver a procesar la matriz continuamente. Al contrario, el modelo se entrena en un proceso *offline*, y sólo se vuelve a entrenar cuando los cambios en las puntuaciones presentes en la matriz requieren una actualización del mismo.

En las próximas páginas presentaremos las principales técnicas de filtrado colaborativo, describiendo alguno de los algoritmos más característicos de cada tipo de técnica.

3.4.1. Algoritmos basados en vecinos

Los algoritmos basados en vecinos son una de las técnicas más populares entre los sistemas de recomendación basados en filtrado colaborativo. Como su nombre indica, se centran en buscar los vecinos de usuarios o productos, es decir, otros usuarios o productos con preferencias similares, y a partir de los mismos generar las recomendaciones. En su funcionamiento se pueden distinguir tres fases principales, en cada una de las cuales se pueden aplicar diferentes técnicas [Herlocker et al., 2002]:

- Cálculo de la similitud entre los distintos usuarios (o productos).
- Selección del vecindario según la similitud previamente calculada.
- Cálculo de la predicción o recomendación a partir de las puntuaciones de los vecinos.

En la literatura podemos encontrar dos aproximaciones principales, según el vecindario esté formado por usuarios o productos similares: basados en usuarios [Resnick et al., 1994] y basados en productos [Sarwar et al., 2001].

3.4.1.1. Algoritmos basados en usuarios

Estos algoritmos, como su nombre indica, seleccionan el vecindario a partir de usuarios similares al usuario actual [Shardanand y Maes, 1995]. La similitud entre dos usuarios se calcula comparando sus perfiles, es decir, sus puntuaciones, tal como se refleja en la Figura 3.1. La idea del algoritmo es comparar las puntuaciones de aquellos productos que ambos usuarios han puntuado. Si dos usuarios tienen un perfil de puntuaciones similar en estos productos, se asume que el resto de puntuaciones también van a ser parecidas, y por tanto se puede recomendar a un usuario otros productos que han sido bien puntuados por sus vecinos.

Por tanto, tal y como se ha explicado anteriormente, la primera tarea de estos algoritmos es calcular la similitud entre el usuario activo y el resto de usuarios, a partir de una función de similitud $s : \mathcal{U} \times \mathcal{U} \rightarrow \mathbb{R}$ previamente definida. En la literatura

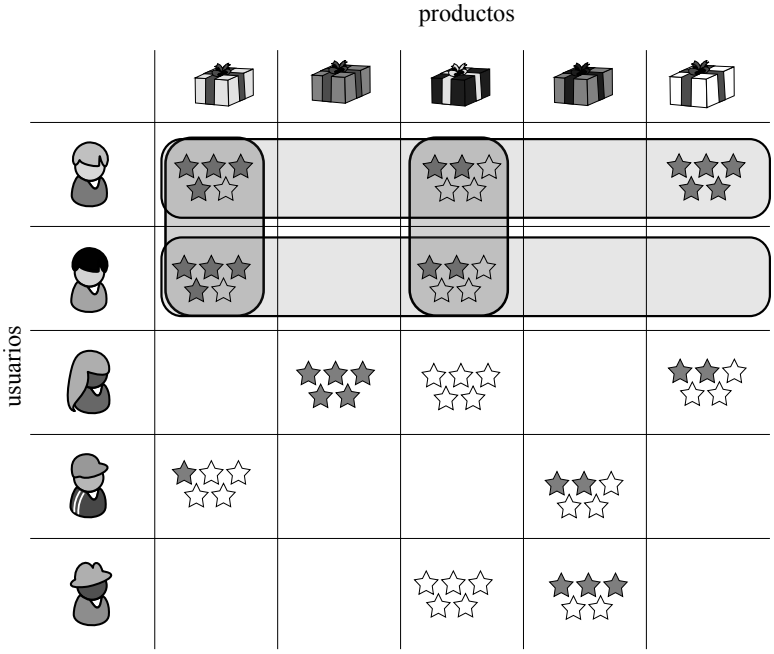


Figura 3.1: Cálculo de similitudes en algoritmos basados en usuario.

podemos encontrar distintas medidas de similitud, entre las que podemos destacar las siguientes:

- Coeficiente de correlación de Pearson. Es una de las primeras técnicas utilizadas en estos algoritmos [Resnick et al., 1994], y quizás también la más popular. Mide la similitud entre dos usuarios según la correlación que presenten sus puntuaciones, utilizando para ello el coeficiente de correlación de Pearson:

$$s(a,u) = \frac{\sum_{i \in \mathcal{I}(a,u)} (r_{ai} - \overline{r_{a.}})(r_{ui} - \overline{r_{u.}})}{\sqrt{\sum_{i \in \mathcal{I}(a,u)} (r_{ai} - \overline{r_{a.}})^2 \sum_{i \in \mathcal{I}(a,u)} (r_{ui} - \overline{r_{u.}})^2}} \tag{3.2}$$

  Este métrica toma valores entre 1 (correlación positiva) y -1 (correlación negativa), indicando el valor 0 que no existe correlación entre los usuarios.

- Coeficiente de Pearson forzado [Shardanand y Maes, 1995]. Es una variante del anterior, pero donde la media de cada usuario se fuerza a tomar un valor fijo, θ.

$$s(a,u) = \frac{\sum_{i \in \mathcal{I}(a,u)} (r_{ai} - \theta)(r_{ui} - \theta)}{\sqrt{\sum_{i \in \mathcal{I}(a,u)} (r_{ai} - \theta)^2 \sum_{i \in \mathcal{I}(a,u)} (r_{ui} - \theta)^2}} \tag{3.3}$$

La idea es que en lugar de usar la media real de cada usuario se use un umbral que diferencia entre buenos y malos productos. Dicho umbral es elegido dependiendo del dominio. Por ejemplo, en una escala del 1 al 5 se podría considerar que puntuaciones por debajo de 4 reflejan una opinión negativa, mientras que por encima de 4 reflejarían una opinión positiva, en cuyo caso se utilizaría $\theta = 4$.

- Coseno. Utilizando el coseno, se mide la distancia entre los vectores formados a partir del perfil del usuario [Breese et al., 1998]:

$$s(a,u) = \frac{\sum_{j \in \mathcal{J}(a,u)} r_{aj} r_{uj}}{\sqrt{\sum_{k \in \mathcal{J}(a)} r_{ak}^2} \sqrt{\sum_{k \in \mathcal{J}(u)} r_{uk}^2}} \tag{3.4}$$

  Un valor cercano a 1 indica similitud, mientras que uno próximo a 0 refleja todo lo contrario. Al contrario que el coeficiente de correlación de Pearson, esta medida no refleja similitud negativa.

- Coeficiente de Pearson ponderado [Herlocker et al., 1999]. Esta medida intenta capturar no sólo la similitud entre usuarios, sino también la confianza que se puede depositar en dicha similitud. Por ejemplo, si dos usuarios han puntuado muy pocos productos en común, es posible que simplemente por casualidad acaben siendo considerados muy similares. En general, a medida que este número aumenta, disminuyen las posibilidades de que una similitud elevada no refleje realmente la realidad. Es decir, aumenta la confianza en la medida. Por tanto, esta técnica tiene en cuenta el número de productos puntuados en común. Si este es mayor que un cierto umbral, θ, se usa el coeficiente de Pearson como medida de similitud, pero si es menor, el coeficiente de Pearson se multiplica por el cociente entre el número de productos y el umbral considerado. Es decir:

$$s(a,u) = \begin{cases} s_{pearson}(a,u) \frac{\left|\mathcal{J}(a,u)\right|}{50} & \left|\mathcal{J}(a,u)\right| < 50 \\ s_{pearson}(a,u) & otherwise \end{cases} \tag{3.5}$$

  El umbral se determina experimentalmente dependiendo del contexto, siendo $\theta = 50$ un valor ampliamente utilizado en la literatura.

- Diferencia cuadrática media (MSD) [Shardanand y Maes, 1995]. En primer lugar, se mide la diferencia cuadrática media entre las puntuaciones otorgadas a productos que ambos usuarios han valorado:

$$msd(a,u) = \frac{\sum_{i \in \mathcal{I}^{(a,u)}} (r_{ai} - v_{ri})^2}{|\mathcal{I}^{(a,u)}|} \tag{3.6}$$

posteriormente se descartan aquellas diferencias mayores a un cierto umbral L, y el resto se pondera según la siguiente fórmula:

$$s(a,u) = \frac{L - msd(a,u)}{L} \tag{3.7}$$

Una vez calculada la similitud entre usuarios, se seleccionan como vecinos aquellos más semejantes al actual. En la literatura encontramos dos técnicas principales para la selección del vecindario:

- Umbral de correlación [Shardanand y Maes, 1995]. Se seleccionan como vecinos todos los usuarios cuya similitud con el usuario actual supera un cierto umbral θ, es decir: $\mathcal{N}(a) = \{n \in \mathcal{U} | n \neq a \wedge s(n,a) > \theta\}$.

- Vecinos más similares [Resnick et al., 1994]. Se seleccionan los k usuarios más parecidos al usuario actual, es decir $\mathcal{N}(a) = \{n_1, \ldots, n_j | j \leq k\}$ cumpliendo que $\forall n : n \in \mathcal{U} \wedge \nexists x \in \mathcal{U} - \mathcal{N}(a) / s(a,x) > s(a,n)$. Esta esta la aproximación más popular, y motivo de que estos algoritmos sean popularmente conocidos como kNN (del inglés *k-Nearest Neighbors*).

En la tarea de predicción se suele seleccionar un vecindario diferente según el producto para el que se va a predecir la puntuación, $\mathcal{N}(u;i)$. El objetivo es asegurarse que los vecinos seleccionados hayan puntuado dicho producto y, por tanto, el algoritmo pueda generar dicha predicción. Esto repercute en un mejor *coverage* en la tarea de predicción, pero también en un mayor coste computacional.

En la literatura los algoritmos basados en usuario se han utilizado fundamentalmente para la tarea de predicción, habiéndose propuesto diferentes alternativas como función de predicción, todas ellas basadas en las puntuaciones otorgadas por los vecinos. Las más populares son:

- Media ponderada por similitud [Resnick et al., 1994]. La predicción de una puntuación se calcula a partir de la media de la puntuación de cada vecino, ponderada según su similitud con el usuario actual. Esta técnica asume que cuanto más semejante sea un vecino al usuario actual, más parecida será también su puntuación, y de ahí que se le dé más peso a la misma:

$$\hat{r}_{ai} = \frac{\sum_{u \in N(a;i)} (r_{ui} s(a,u))}{\sum_{u \in N(a;i)} s(a,u)} \tag{3.8}$$

- Media normalizada [Herlocker et al., 2002]. Es similar a la anterior, pero las puntuaciones se normalizan teniendo en cuenta la media y desviación típica de la distribución de puntuaciones de cada usuario. Como hemos discutido en la Sección 2.3, las puntuaciones están sujetas a variabilidad debida a diferentes causas, una de las cuales es la diferencia de criterio entre usuarios: algunos sólo dan puntuaciones altas a productos especialmente buenos, otros suelen dar puntuaciones altas salvo que el producto sea muy malo, etc. Esto hace que la distribución de puntuaciones sea diferente para cada usuario, por lo que esta técnica normaliza las puntuaciones de los vecinos y finalmente ajusta el resultado a la distribución esperada por el usuario actual:

$$\hat{r}_{ai} = \overline{r_{a.}} + \sigma_a \frac{\sum_{u \in N(a;i)} \left[\left(\frac{r_{ui} - \overline{r_{u.}}}{\sigma_u} \right) s(a,u) \right]}{\sum_{u \in N(a;i)} s(a,u)} \tag{3.9}$$

- Clasificación [Desrosiers y Karypis, 2011]. En lugar de calcular la predicción como una media de las puntuaciones de los vecinos, esta técnica lo que hace es devolver como predicción la puntuación más probable. Es decir, aquella que más vecinos han otorgado, pero teniendo en cuenta la similitud con cada vecino:

$$\hat{r}_{ai} = \arg\max_{r \in \mathcal{R}} \sum_{u \in N(a;i)} \delta(r_{ui} = r) s(a,u) \tag{3.10}$$

  donde $\delta(r_{ui} = r)$ devolverá 1 si $r_{ui} = r$ y 0 en otro caso. Alternativamente, se podrían normalizar las puntuaciones siguiendo una aproximación similar a la anterior.

Por otra parte, en la literatura apenas tenemos ejemplos de uso de algoritmos basados en usuarios en la tarea de recomendación. De hecho, no ha sido propuesta ninguna implementación de estas técnicas para dicha tarea, a excepción de aquella consistente en predecir la puntuación para todos los productos y devolver aquellos con una mayor puntuación [Wang et al., 2010]. El problema de esta aproximación es que es muy costosa. En el Capítulo 5 presentaremos una implementación especialmente diseñada para esta tarea, mucho más eficiente y que obtiene muy buenos resultados.

3.4.1.2. Algoritmos basados en productos

Estos algoritmos son similares a la aproximación basada en usuarios descrita en la sección anterior, pero en lugar de buscar usuarios similares al actual, se basan en buscar productos similares a los que este ha puntuado.

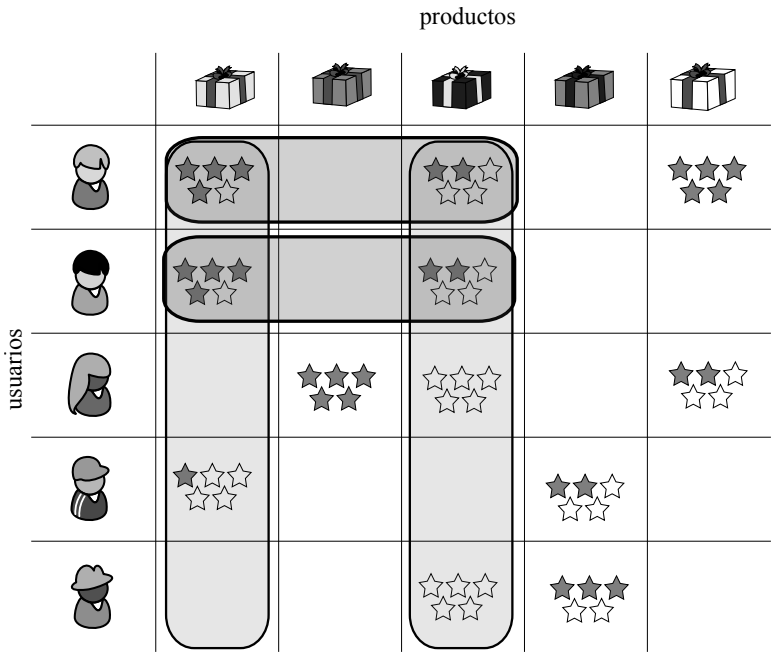


Figura 3.2: Cálculo de similitudes en algoritmos basados en productos.

Tal como se refleja en la Figura 3.2, se utiliza una función de similitud $s : \mathcal{I} \times \mathcal{I} \rightarrow \mathbb{R}$ para comparar las puntuaciones de cada par de productos. Las medidas de similitud más populares son:

- Coseno. Funciona exactamente igual que en el caso de algoritmos basados en usuarios, con la diferencia de que en este caso se comparan los perfiles de los productos, es decir, las puntuaciones que estos han recibido de los distintos usuarios.

- Coseno ajustado [Sarwar et al., 2001]. Similar a la anterior, pero las puntuaciones de cada usuario se normalizan respecto a su media, con el objetivo de minimizar el impacto de la variabilidad entre usuarios:

$$s(i,j) = \frac{\sum_{u \in \mathcal{U}} (r_{ui} - \overline{r_{u.}})(r_{uj} - \overline{r_{u.}})}{\sqrt{\sum_{u \in \mathcal{U}} (r_{ui} - \overline{r_{u.}})^2 \sum_{u \in \mathcal{U}} (r_{uj} - \overline{r_{u.}})^2}} \tag{3.11}$$

- Basada en probabilidad condicional [Deshpande y Karypis, 2004]. Se basa en estimar la probabilidad de que un usuario puntúe un producto sabiendo que ha puntuado el otro, y utilizar esta probabilidad condicional como medida de similitud:

$$s(i,j) = P(j|i) = \frac{|\mathcal{U}^{(j,i)}|}{|\mathcal{U}^i|} \tag{3.12}$$

En la tarea de predicción, el funcionamiento es similar al de los algoritmos basados en usuarios [Sarwar et al., 2001]: primero se seleccionan como vecinos los k productos más parecidos a aquel cuya puntuación se intenta predecir; posteriormente, la predicción se calcula como la media de la puntuación otorgada por el usuario actual a dichos vecinos, ponderada por su similitud.

Su uso en la tarea de recomendación fue propuesta por Deshpande y Karypis [2004]: en primer lugar se seleccionan los k productos más similares a cada uno de los que ha valorado el usuario; luego se suman las similitudes de cada vecino respecto a los diferentes productos en el perfil de usuario, y se recomiendan los N que hayan adquirido un valor más alto. Al no tener en cuenta la puntuación de cada usuario a la hora de recomendar, esta aproximación es adecuada únicamente para sistemas con puntuaciones unarias.

3.4.1.3. *Similarity fusion*

Este algoritmo es similar a los anteriores, basados en vecinos, pero en lugar de considerar únicamente las similitudes entre usuarios o productos, tiene en cuenta ambas, con el objetivo de aprovechar la mayor cantidad de información posible y obtener mejores resultados en condiciones de baja densidad [Wang et al., 2006a]. Para predecir una puntuación, se combina la puntuación dada por usuarios similares al actual (SUR), la puntuación dada por el usuario actual a productos semejantes (SIR), y, finalmente, la puntuación dada por usuarios semejantes a productos parecidos (SUIR). El peso dado a cada uno de los factores depende de dos parámetros, δ y λ, cuyos valores se determinan experimentalmente:

$$\begin{aligned}\hat{r}_{ai} &= \sum_{r\in\mathcal{R}} rP(r_{ai}=r|SUR,SIR,SUIR) \\ &= \left(\sum_{r\in\mathcal{R}} rP(r_{ai}=r|SUIR)\delta\right)+\left(\sum_{r\in\mathcal{R}} rP(r_{ai}=r|SUR)\delta(1-\lambda)\right) \\ &\quad +\left(\sum_{r\in\mathcal{R}} rP(r_{ai}=r|SIR)(1-\delta)(1-\lambda)\right)\end{aligned} \tag{3.13}$$

3.4.2. Algoritmos basados en regresión

Este tipo de algoritmos están orientados fundamentalmente a la tarea de predicción. Se basan en que la puntuación de un producto puede servir para estimar la de otro. Partiendo de esa premisa, entrenan un modelo de regresión para cada par de productos $i, j \in \mathcal{I}$, utilizando los datos disponibles en la matriz de puntuaciones. Este modelo, que denotaremos como $f_{ji} : \mathcal{R} \to \mathcal{R}$, puede verse como un experto capaz de estimar la puntuación del producto i a partir del producto j.

Normalmente un usuario habrá puntuado varios productos, por lo que las estimaciones obtenidas a partir de cada uno de ellos se combinan para obtener la predicción final. Es decir:

$$\hat{r}_{ai} = \sum_{j\in\mathcal{I}(a)}^{\star} f_{ji}(r_{aj}) \tag{3.14}$$

La función de agregación $\sum^{\star}$ es generalmente algún tipo de suma ponderada.

Generalmente, el uso de estos algoritmos para la tarea de recomendación requiere predecir una puntuación para todos los productos no puntuados por el usuario actual, y luego ordenarlos para devolver los que tengan una puntuación mayor.

3.4.2.1. Regresión lineal

Este algoritmo, propuesto por Vucetic y Obradovic [2000], utiliza un modelo de regresión lineal, es decir, asume que existe una relación lineal entre las puntuaciones de ciertos productos. Por tanto:

$$f_{ji}(r) = r\alpha_{ji} + \beta_{ji} \tag{3.15}$$

En la fase de entrenamiento, se estiman los parámetros del modelo, α_{ji} y β_{ji}, a partir de los datos de la matriz de puntuaciones. Para evitar emplear en la fase de predicción productos cuyas puntuaciones realmente no siguen una relación lineal con las

del producto a predecir, se tiene en cuenta el coeficiente de regresión, R^2, descartando aquellos que no superen un cierto umbral. Los restantes se combinan para obtener la predicción, utilizando una de las siguientes alternativas: media de la predicción obtenida por cada uno de los distintos expertos; media ponderada, dando a cada experto un peso calculado a partir de la matriz de covarianza, etc. Vucetic y Obradovic [2000]

3.4.2.2. *Slope One*

Los algoritmos *slope one* [Lemire y Maclachlan, 2005] parten de un modelo todavía más sencillo que el de regresión lineal, que asume que la pendiente de la recta de regresión es igual a 1. Por tanto,

$$f_{ji}(r) = r + b_{ji} \tag{3.16}$$

En una primera aproximación, la constante b se calcula como la diferencia media entre las puntuaciones de ambos productos. La predicción se haría según la siguiente ecuación:

$$\hat{r}_{ai} = \overline{r_{a.}} + \frac{1}{|I_{ai}|} \sum_{j \in I_{ai}} \sum_{u \in \mathcal{U}^{(ji)}} \frac{r_{ui} - r_{uj}}{|\mathcal{U}^{(ji)}|} \tag{3.17}$$

donde I_{ai} es el conjunto de productos puntuados por el usuario actual que han sido puntuados al menos por un usuario que también ha puntuado el producto i, es decir, $I_{ai} = \{j \in \mathcal{J}^{(a)} | j \neq i, |\mathcal{U}^{(ji)}| > 0\}$.

Una segunda aproximación consiste en ponderar la contribución de cada producto según el número de usuarios que tiene en común con el producto cuya puntuación se intenta predecir. Productos con más usuarios en común recibirán un peso mayor, pues la confianza que se puede depositar en la relación entre ambos es mayor. El algoritmo resultante, denominado *weighted slope one*, calcularía la predicción usando la siguiente ecuación:

$$\hat{r}_{ai} = \frac{\sum_{j \in \mathcal{J}^{(a)}} \left(\sum_{u \in \mathcal{U}^{(ji)}} \frac{r_{ui} - r_{uj}}{|\mathcal{U}^{(ji)}|} + r_{ai} \right) |\mathcal{U}^{(ji)}|}{\sum_{i \in \mathcal{J}^{(a)}} |\mathcal{U}^{(ji)}|} \tag{3.18}$$

Finalmente, una tercera variante, *bi-polar slope one*, tiene en cuenta la diferencia entre productos que el usuario ha puntuado positivamente y productos que ha puntuado negativamente.

3.4.3. Algoritmos de clasificación de la puntuación

Al igual que los anteriores, este tipo de algoritmos también se orientan a la tarea de predicción. Sin embargo, la abordan como un problema de clasificación, donde cada una de las posibles puntuaciones, $r \in \mathcal{R}$, es vista como una clase, y se intenta predecir a cuál de las posibles "clases" pertenece realmente la puntuación que el usuario actual habría dado a cierto producto. A partir de un determinado modelo, el algoritmo estima la probabilidad de pertenencia a cada clase, y la puntuación más probable es devuelta como predicción. Es decir:

$$\hat{r}_{ai} = \arg\max_{r \in \mathcal{R}} P(r_{ai} = r | \mathcal{P}_a) \tag{3.19}$$

Cada algoritmo utilizará un modelo diferente, el cual se entrena previamente a partir de la información en la matriz de puntuaciones.

3.4.3.1. *Naïve Bayes*

Una de las aproximaciones más sencillas es el uso de un clasificador Bayesiano [Breese et al., 1998] como el representado en la Figura 3.3, donde R_i representa la variable aleatoria "puntuación recibida por el producto $i \in \mathcal{I}$". Para cada producto se construye un modelo que asume que la puntuación que el usuario ha otorgado a los restantes productos depende únicamente de la puntuación otorgada a dicho producto. No se tienen en cuenta, por tanto, interacciones entre estos otros productos u otro tipo de factores.

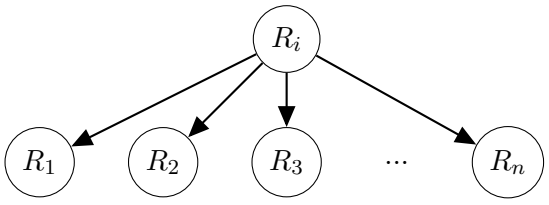


Figura 3.3: Clasificador Bayesiano para filtrado colaborativo.

Para predecir la puntuación $\hat{r}_{ai}$, el algoritmo sólo tiene que calcular la probabilidad de que el usuario otorgue cada una de las posibles puntuaciones $r \in \mathcal{R}$ y quedarse con la más probable. Usando el modelo anterior, esto se reduce al cálculo de la probabilidad de que $R_i = r$ condicionada por las distintas puntuaciones presentes en el perfil del usuario, es decir, calcular la probabilidad $P(R_i = r | R_1 = r_{a1}, R_2 = r_{a2}, \ldots)$. Aplicando el teorema de Bayes, dicha probabilidad se puede estimar a partir de las probabilidades

a priori calculadas sobre los datos existentes en la matriz. Dado que el modelo asume la independencia entre puntuaciones, la estimación de una predicción se reduce al cálculo de la siguiente fórmula:

$$\hat{r}_{ai} = \arg\max_{r \in \mathcal{R}} P(R_i = r) \prod_j P(R_j = r_{aj} | R_i = r) \tag{3.20}$$

donde las probabilidades a priori $P(R_i = r)$ y $P(R_j = r_{aj}|R_i = r)$ se calculan directamente a partir de los datos en la matriz:

$$P(R_i = r) = \frac{\left|\{u \in \mathcal{U}^{(i)} | r_{ui} = r\}\right|}{|\mathcal{U}^{(i)}|} \tag{3.21}$$

$$P(R_j = r_{aj} | R_i = r) = \frac{\left|\{u \in \mathcal{U}^{(i)} \cap \mathcal{U}^{(u)} | r_{ui} = r \wedge r_{uj} = r_{aj}\}\right|}{\left|\{u \in \mathcal{U}^{(i)} \cap \mathcal{U}^{(j)} | r_{ui} = r\}\right|} \tag{3.22}$$

Aunque las asunciones hechas por este modelo son obviamente incorrectas, en la práctica funciona bastante bien, y tiene la ventaja de que el entrenamiento del modelo es sencillo y se puede actualizar fácilmente a medida que los usuarios puntúan nuevos productos.

3.4.3.2. *Personality Diagnosis*

Esta técnica tiene en cuenta la posible variación de la puntuación de los usuarios dependiendo del contexto, es decir, que un mismo usuario puede otorgar puntuaciones diferentes sobre el mismo producto dependiendo del momento en que realice la puntuación. Por tanto, la personalidad real de un individuo vendrá determinada no sólo por su perfil de puntuaciones, sino también por dicha variación, que el algoritmo modela como ruido gaussiano [Pennock et al., 2000]. Es decir, $P(r_{ai} = x | r_{ai}^{true} = y) \simeq e^{-(x-y)^2/2\sigma^2}$, donde r_{ai}^{true} representa la utilidad real del producto para el usuario.

Finalmente, para estimar la predicción se utiliza un modelo probabilístico similar al anterior, que asume la independencia entre las puntuaciones de los productos.

3.4.4. Agrupamiento

Los algoritmos de agrupamiento o *clustering* consisten en clasificar usuarios y/o productos en grupos de acuerdo a sus puntuaciones. Estos algoritmos asumen que hay grupos de usuarios o productos con similares características y, por tanto, una vez que el usuario ha sido asignado a un grupo, la predicción o recomendación se obtiene a partir de las puntuaciones otorgadas por otros miembros del mismo grupo. Por ejemplo, como predicción se podría utilizar la puntuación media dentro del grupo.

Una de las técnicas de agrupamiento más sencillas es la de K-Means [Marlin, 2004], en la que se utiliza un proceso iterativo para minimizar la distancia de los usuarios con el perfil medio del grupo al que pertenecen. El perfil de cada grupo se inicializa de forma aleatoria. En cada iteración, los usuarios son asignados al grupo más parecido a su perfil, y finalmente se actualiza el perfil medio del grupo a partir de los usuarios a él asignados. En esta técnica generalmente sólo se agrupan los usuarios, y además cada usuario pertenece a un sólo grupo.

Otras técnicas utilizan modelos más completos, basados en *co-clustering* (en que se agrupan tanto usuarios como productos), o grupos superpuestos (*overlapping clustering*), en los cuales un usuario (o producto) pueden pertenecer a varios grupos [George y Merugu, 2005; Shafiei y Milios, 2006]. En general estas técnicas ofrecen un mejor resultado.

Alternativamente, los algoritmos de agrupamiento han sido utilizados en combinación con otro tipo de algoritmos, generalmente con dos objetivos: incrementar la densidad de la matriz de puntuaciones, aproximando puntuaciones desconocidas con la puntuación media del grupo al que pertenece al usuario, por ejemplo, o bien por motivos de eficiencia, sustituyendo a los usuarios, en ciertas fases del algoritmo, por el grupo al que pertenecen, reduciendo por tanto el volumen de datos a manejar.

Por ejemplo, el algoritmo *cluster-based smoothing* [Xue et al., 2005] combina un algoritmo basado en vecinos con una técnica de *clustering*, para lograr ambos objetivos, siguiendo un proceso que involucra 5 fases:

1. En un proceso *offline*, los usuarios se agrupan en k *clusters* usando el algoritmo K-means.

2. Durante la recomendación, en primer lugar se completa el perfil del usuario actual, aproximando las puntuaciones desconocidas con la media del grupo al que pertenece.

3. Luego, se buscan los grupos más parecidos al perfil completo calculado en la fase anterior.

4. Se seleccionan los vecinos, pero teniendo en cuenta sólo los usuarios que pertenecen a alguno de los grupos seleccionados.

5. Finalmente, se calcula la predicción a partir de los vecinos.

3.4.5. Modelos probabilísticos

De forma similar a los anteriores, esta técnica se basa en la existencia de "clases" que determinan las características de usuarios o productos. Utilizan un modelo probabilístico donde la pertenencia a una clase se modela como una variable latente $z \in Z = \{z_1, \ldots, z_k\}$. La clase a la que pertenezca un usuario y/o producto, según el modelo, determinará la distribución de sus puntuaciones. Al igual que en los algoritmos de clasificación de la puntuación, el modelo se utiliza para estimar la probabilidad de que un usuario puntúe a un producto de cierta forma, devolviendo como predicción la puntuación más probable.

Por otra parte, la probabilidad de pertenencia a una clase se estima a partir de los datos existentes en la matriz de puntuaciones. Generalmente, se utiliza el algoritmo *Expectation-Maximization* (EM) [Dempster et al., 1977], que permite obtener la estimación por máxima verosimilitud.

Entre las técnicas más populares podemos mencionar la de agrupamiento basado en un modelo bayesiano, el modelo de aspectos o los modelos mixtos, las cuales serán descritas a continuación.

3.4.5.1. Agrupamiento basado en un modelo bayesiano

Este tipo de algoritmos es similar al clasificador bayesiano mostrado en la Sección 3.4.3.1, pero sigue un enfoque ligeramente diferente: se asume que existen ciertas clases de usuarios, y que la pertenencia de un usuario a una clase determina su forma de puntuar productos [Breese et al., 1998].

Este modelo asume la independencia entre las puntuaciones de los distintos productos, por lo que, dado el perfil de un usuario, la probabilidad de pertenencia a una determinada clase se puede estimar usando la siguiente fórmula:

$$P(Z = z | R_1 = r_{a1}, R_2 = r_{a2}, \ldots) = P(Z = z) \prod_j P(R_j = r_{aj} | Z = z) \tag{3.23}$$

Por tanto, dado un usuario $a \in \mathcal{U}$, se puede estimar la probabilidad de que otorgue una determinada puntuación a los distintos productos de la siguiente forma:

$$P(R_1 = \hat{r}_1, R_2 = \hat{r}_2, \ldots | a) = \sum_z P(Z = z) \prod_j P(R_j = \hat{r}_j | z) \tag{3.24}$$

Es decir, el modelo selecciona la clase z a la que pertenece un usuario, y estima la puntuación para todos los productos partiendo de que el usuario pertenece a di-

cha clase. Las probabilidades $P(r|z)$ se estiman automáticamente durante la fase de entrenamiento, usando el algoritmo EM.

Tal como se muestra en Figura 3.4, este modelo es muy sencillo, y no permite que un usuario pertenezca a varias clases, únicamente tiene en cuenta la clasificación de usuarios, y no separa las preferencias reales de las puntuaciones presentes en el perfil.

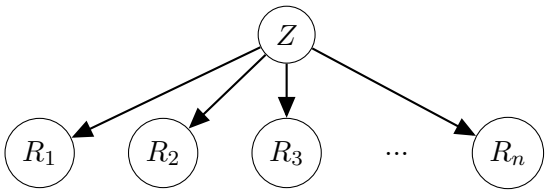
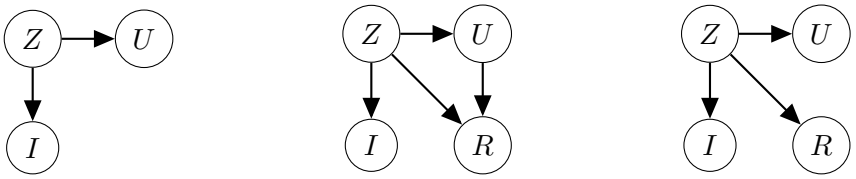


Figura 3.4: Agrupamiento basado en un modelo bayesiano.

3.4.5.2. Modelo de aspectos

Este algoritmo utiliza un modelo probabilístico basado en una variable latente $z \in Z = \{z_1, \ldots, z_k\}$, la cual determina las preferencias de los usuarios [Hofmann y Puzicha, 1999]. Originalmente, el modelo está pensado para sistemas con preferencias implícitas o unarias, en las que se conoce la interacción entre usuarios y productos pero no el grado de preferencia. Es decir, las preferencias se representan como pares (u, i), con $u \in \mathcal{U}$ y $i \in \mathcal{I}$.



(a) **(b)** **(c)**

Figura 3.5: Diferentes variantes del modelo de aspectos.

El modelo, tal y como se representa en la Figura 3.5a, asume que usuarios y productos son independientes y están condicionados por la variable latente z, por lo que el modelo probabilístico que refleja el posible interés de un usuario u en un producto i puede escribirse como:

$$P(u, i) = \sum_{z \in Z} P(z)P(u|z)P(i|z) \tag{3.25}$$

donde $P(u|z)$ y $P(y|z)$ siguen una distribución multinomial condicionada por la pertenencia a la clase z, y $P(z)$ es la probabilidad a priori de pertenencia a dicha clase. Normalmente este modelo se reescribe de la siguiente forma:

$$P(u,i) = P(u)\sum_{z\in Z} P(z|u)P(i|z) \tag{3.26}$$

expresión equivalente a la anterior pero que refleja mejor el propósito del algoritmo de recomendación.

El modelo anterior se puede extender de dos formas para incorporar el uso de puntuaciones. La primera de ellas, representada en la Figura 3.5b, considera que el usuario influye en la selección del producto a predecir, mientras en la segunda, Figura 3.5c, la selección del producto no es modelada. Al igual que en el caso anterior, el modelo se entrena utilizando un algoritmo EM.

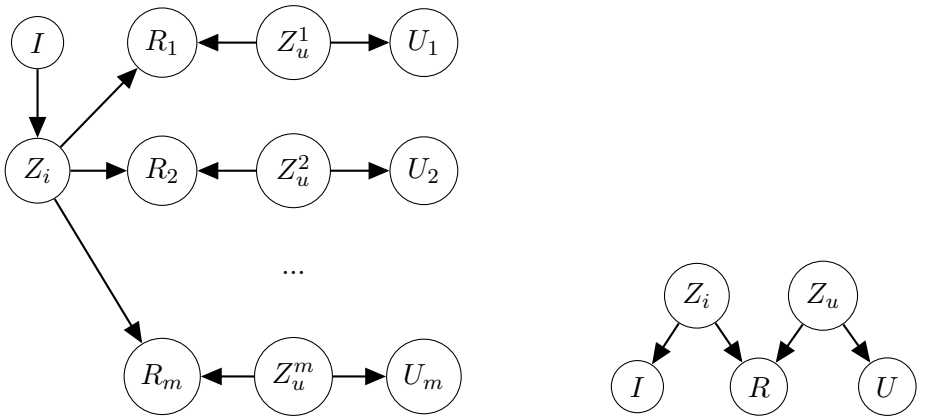
Al contrario que el algoritmo mostrado en la sección anterior, basado en un modelo bayesiano que sólo modela la relación entre la clase z y las puntuaciones, el modelo de aspectos modela también productos y usuarios. Además, la probabilidad de una puntuación se modela de forma independiente para cada producto, es decir, la probabilidad de cada tripleta de usuario, producto y puntuación ($P(u,i,r)$) podría estimarse usando una clase diferente. Sin embargo, considera únicamente una variable latente, es decir, el modelo, si bien clasifica tanto productos como usuarios, no permite clasificarlos de forma separada.

3.4.5.3. Modelos mixtos

Precisamente, los modelos mixtos permiten clasificar usuarios y productos por separado, definiendo dos variables latentes, Z_u y Z_i, para cada uno de ellos. Los algoritmos más populares son el *Joint Mixture Model (JMM)* [Jin et al., 2006] y el *Flexible Mixture Model (FMM)* [Si y Jin, 2003], representados, respectivamente, en las Figuras 3.6a y 3.6b. En el JMM, se asume que cada usuario pertenece a una única clase, mientras que en el FMM el usuario puede pertenecer a clases distintas en función del producto a puntuar.

3.4.6. Técnicas de reducción de la dimensionalidad de la matriz

En un sistema de filtrado colaborativo típico, la matriz de puntuaciones es muy grande y dispersa. Las técnicas de reducción de dimensionalidad tienen como objetivo transformar esta matriz en una de menores dimensiones, que refleje las características más relevantes a la hora de proceder con la recomendación. El principal problema de



(a) Joint Mixture Model **(b)** Flexible Mixture Model

Figura 3.6: Modelos mixtos para filtrado colaborativo.

dichas técnicas es seleccionar cuáles son las características más representativas. Una vez seleccionadas, la información se representaría con una matriz de reducidas dimensiones, en las que se representa el grado de afinidad de un usuario a cada una de las características, y el grado de pertenencia de un producto a las mismas. La idea es que dichas características representan atributos latentes en la matriz original, los cuales permiten expresar el comportamiento de los usuarios. Esto aceleraría el proceso de recomendación, ya que únicamente habría que considerar dichas características en lugar de todas las puntuaciones, al tiempo que minimizaría los problemas relacionados con la dispersión de la matriz o la presencia de datos anómalos.

3.4.6.1. *Singular Value Decomposition*

Este algoritmo se basa en utilizar la técnica de *Singular Value Decomposition* (SVD) para reducir la dimensionalidad de la matriz [Sarwar et al., 2000b]. Dicha técnica convierte la matriz de puntuaciones en tres matrices tales que $\mathcal{M} = U \cdot S \cdot V^T$, donde U y V serían dos matrices ortogonales y S una matriz diagonal de tamaño $r \times r$ (con r el rango de la matriz $\mathcal{M}$), formada por los valores singulares de la matriz de puntuaciones. Esta matriz se reduce descartando los valores más pequeños, para así obtener una matriz S_k, con $k < r$. La matriz reconstruida, $\mathcal{M}_k = U_k \cdot S_k \cdot V_k^T$, es la mejor aproximación de rango k de la matriz de puntuaciones original. Por tanto, las puntuaciones se aproximarían a partir de dicha matriz, usando la fórmula:

$$\hat{r}_{ai} = \overline{r_{a.}} + U_k \cdot \sqrt{S_k^T}(a) \cdot \sqrt{S_k} \cdot V_k^T(i) \tag{3.27}$$

3.4.6.2. SVD regularizado (RSVD) y variantes

El principal problema del algoritmo anterior es la gran cantidad de tiempo necesaria para factorizar la matriz. En el contexto del concurso organizado por Netflix, Funk [2006] propuso un nuevo método para aproximar dicha factorización, método que acabaría teniendo un gran éxito en el ámbito del filtrado colaborativo.

En este modelo, cada producto se representa por un conjunto de k características o aspectos, y cada usuario como un conjunto de valores indicando sus preferencias relativas a cada uno de dichos k aspectos. Es decir, productos y usuarios se representarían como vectores numéricos de longitud k. A partir de dichos vectores, la predicción de una puntuación se calcularía a partir del producto de ambos:

$$\hat{r}_{ai} = x_a^T y_i \tag{3.28}$$

Donde los vectores $x_a \in \mathbb{R}^k$ y $y_i \in \mathbb{R}^k$ representarían, respectivamente, la afinidad de cada usuario y producto a cada una de las k características. Los valores que tomarán los $m + n$ vectores necesarios se aproximarán a partir de los datos en la matriz de puntuaciones durante la fase de construcción del modelo, usando una variante de SVD que ignora las puntuaciones desconocidas y es mucho más rápida que la descomposición SVD tradicional. En concreto, se utiliza el método de descenso de gradiente para minimizar la diferencia entre la predicción hecha por el algoritmo y la puntuación real. En cada iteración del algoritmo los vectores se actualizarían de la siguiente forma:

$$\begin{aligned} e_{ai} &= r_{ai} - \hat{r}_{ai} \\ x_{ak} &+= l * (e_{ai} y_{ik} - \lambda x_{ak}) \\ y_{ik} &+= l * (e_{ai} x_{ak} - \lambda y_{ik}) \end{aligned}$$

Donde l sería el ritmo de aprendizaje deseado y λ la constante de regularización. El uso de regularización es necesario para evitar un exceso de ajuste del modelo a los datos de entrenamiento, lo que influiría negativamente en los resultados obtenidos con nuevos datos. Las distintas características se entrenarían una a una, siendo el número de características, k, un parámetro del modelo.

Este algoritmo ha sido mejorado posteriormente mediante la introducción de factores adicionales. Por ejemplo, Paterek [2007] propuso diversas modificaciones a dicho algoritmo, entre las que destacan:

- RSVD mejorado.

Añade un parámetro adicional a cada usuario, c_a, y otro a cada producto, d_i, los cuales representarían factores globales de cada uno de ellos. El modelo resultante sería:

$$\hat{r}_{ai} = c_a + d_i + x_a^T y_i \tag{3.29}$$

- NSVD2

  Este método reduciría el número de parámetros a estimar, modelando x_a como una función de los productos puntuados por el usuario, resultado el siguiente modelo:

$$\hat{r}_{ai} = c_a + d_i + \sum_k y_{ik} \sum_{j \in \mathcal{J}_a} y_{jk} \tag{3.30}$$

Posteriormente, Koren [2008] extendió el modelo para incluir información implícita. En su propuesta, el algoritmo SVD++, se tenía en cuenta que el hecho de que un usuario puntuase un producto implicaba cierto grado de relevancia, independientemente de la puntuación otorgada, dando lugar al siguiente modelo:

$$\hat{r}_{ai} = b_{ai} + y_i^T \left(x_a + |I_a|^{-\frac{1}{2}} \sum_{j \in I_a} w_j \right) \tag{3.31}$$

con w un vector de parámetros adicionales que ponderan la contribución de la información implícita.

3.4.7. Modelos integrados

En general, todas las aproximaciones presentadas tienen sus limitaciones, en gran medida relacionadas con aquella variabilidad en las preferencias de los usuarios que no es capaz de explicar el modelo utilizado. El diseño de modelos complejos y que incorporan multitud de factores permite aproximar de manera más precisa el comportamiento de los usuarios, dando como resultado recomendaciones de mayor calidad, al menos en relación al error cometido por el algoritmo. La contrapartida es que las técnicas pasan a ser más específicas y dependientes de un domino o conjunto de datos en particular, lo que no siempre es deseable.

No podemos obviar, sin embargo, que este tipo de modelos pueden llegar a ofrecer buenos resultados en dominios concretos, y que muchos autores han apostado por su diseño.

Entre ellos, podemos destacar el algoritmo presentado por Koren [2008], un complejo modelo que integra la técnica SVD++ descrita en la sección anterior con una variante de los algoritmos basados en vecinos. En el modelo resultante se tienen en cuenta efectos globales de productos y usuarios, efectos derivados de la interacción entre productos y usuarios, tanto implícitas como basadas en puntuaciones, y finalmente relaciones de similitud con los vecinos. Al igual que las técnicas anteriores, es entrenado utilizando el método de descenso de gradiente.

En la literatura se pueden encontrar modelos incluso más complejos, en gran parte diseñados para problemas muy concretos, como el concurso organizado por Netflix [Piotte y Chabbert, 2009; Töscher et al., 2009]. Como se ha comentado, estos algoritmos se caracterizan por manejar multitud de factores de diversa naturaleza.

Parte II

Mejorando las recomendaciones basadas en Filtrado Colaborativo

Capítulo 4

Técnicas de filtrado colaborativo en la tarea de predicción

La tarea de predicción es sin lugar a dudas la más popular dentro de la numerosa literatura dedicada a los sistemas de recomendación. En los últimos años, se han desarrollado cientos de técnicas enfocadas sobre todo a reducir el error en la predicción. Sin embargo, la exactitud de las predicciones no lo es todo, y muchas técnicas difícilmente pueden ser usadas en entornos reales debido a su extrema complejidad, bajo rendimiento y eficiencia, o sus malos resultados en condiciones ligeramente más realistas que las presentes en el entorno de laboratorio en que fueron desarrolladas.

En este capítulo se lleva a cabo una exhaustiva evaluación de un amplio abanico de técnicas de filtrado colaborativo, centrándonos en la tarea de predicción pero evaluando el rendimiento en diferentes casos y condiciones. Además, se presenta una nueva técnica de recomendación que, a pesar de su sencillez, obtiene resultados similares a técnicas mucho más complejas.

En la Sección 4.2 describimos nuestro nuevo algoritmo, una sencilla técnica que se diferencia de los algoritmos tradicionales en que se basa principalmente en las diferencias entre usuarios en lugar de en sus semejanzas, y que hemos denominado *filtrado colaborativo basado en tendencias*. Posteriormente, en la Sección 4.3 presentamos las métricas de evaluación de la predicción más populares, así como dos nuevas métricas con interesantes propiedades. Finalmente, las Secciones 4.4 y 4.5 describen en detalle los experimentos realizados y las principales conclusiones obtenidas. En la primera de ellas se estudia el comportamiento general de las diferentes técnicas, comparándolas con el algoritmo basado en tendencias y destacando las principales ventajas y limitaciones de las mismas. En la segunda, nos centramos en los algoritmos basados en vecinos, especialmente populares en el ámbito del filtrado colaborativo y con importantes ventajas

que se harán palpables en los próximos capítulos de esta tesis.

4.1. Introducción

La tarea de predicción ha sido tradicionalmente la que ha suscitado mayor interés entre la comunidad científica, en lo que a investigación en sistemas de recomendación se refiere. Impulsada en parte por concursos como el organizado por Netflix [Bennett y Lanning, 2007], la meta por ver qué algoritmo comete un menor error en la predicción ha estado presente a lo largo de la historia de los sistemas de recomendación, y continúa aún hoy en día a ocupar un porcentaje importante de los trabajos presentados en congresos y revistas especializadas.

Tal como hemos visto en el Capítulo 3, en la literatura podemos encontrarnos una gran variedad de técnicas, la mayor parte de ellas centradas en la tarea de predicción. Sin embargo, y a pesar de los importantes avances alcanzados especialmente en la mejora de la precisión en la predicción, no es oro todo lo que reluce y gran parte de los resultados obtenidos deben ser interpretados cuidadosamente, relativizando adecuadamente su importancia real.

En primer lugar, muchas de las técnicas propuestas han sido evaluadas únicamente en unas condiciones muy concretas, en un entorno de laboratorio, y no está claro cómo se comportarían realmente en situaciones más realistas. Escasean no sólo los trabajos dedicados a comparar diferentes técnicas entre sí, sino también estudios acerca del comportamiento de una técnica concreta en situaciones diferentes. Por ejemplo, un buen número de algoritmos han sido diseñados con el único objetivo de mejorar la precisión en el conjunto de evaluación usado en el concurso Netflix, y obtienen resultados mediocres al utilizarse en contextos diferentes. Es por ello que, a la hora de desplegar sistemas comerciales, técnicas más antiguas, pero con buenos resultados en multitud de situaciones, tales como los algoritmos basados en vecinos, siguen estando entre las primeras opciones.

Por otra parte, muchas de las técnicas presentan tasas de error realmente reseñables, pero gracias a complicados modelos y algoritmos, dependientes de multitud de parámetros. En estos casos, aparte del probable incremento del error en caso de ser usadas en entornos ligeramente diferentes, habría que considerar también su viabilidad en entornos reales, debido sobre todo a la ingente cantidad de recursos computacionales necesarios para su operación.

En definitiva, la utilidad real de una técnica no depende únicamente de lo pequeño que sea el error que comete, y de hecho aspectos como la eficiencia o buen comporta-

miento en diferentes entornos tienen tanta o más importancia.

Precisamente, la técnica presentada en este capítulo, el *filtrado colaborativo basado en tendencias*, ha sido desarrollada teniendo en cuenta precisamente estas otras cuestiones. La evaluación realizada incluye diferentes entornos y situaciones, y ha sido complementada con un estudio de diferentes técnicas presentes en la literatura, analizando las ventajas e inconvenientes de cada una de ellas. Tal como hemos comentado en la Sección 1.3.1, se ha presentado una solicitud de patente para este método.

4.2. Filtrado colaborativo basado en tendencias

Si observamos los algoritmos de filtrado colaborativo presentados hasta la fecha, vemos que la mayor parte de ellos, si no todos, se basan en buscar relaciones de similitud entre usuarios o productos. La idea es que si dos usuarios muestran un patrón de puntuaciones similar, probablemente también coincidan en las puntuaciones restantes. El principal problema es que encontrar dichas relaciones es realmente difícil en condiciones de baja densidad, es decir, en las que existen pocas puntuaciones, debido a que su cálculo necesita una mayor cantidad de información que aquella que tenemos disponible.

El algoritmo presentado en esta sección funciona de forma diferente. En lugar de buscar las relaciones entre individuos o productos, nos hemos basado en las diferencias entre ellos. Es decir, en las variaciones de cada usuario a la hora de puntuar un determinado producto.

Tal como habíamos comentado en la Sección 2.3, estas variaciones no sólo se deben a las lógicas diferencias de opiniones o gustos entre usuarios, sino también a otros aspectos como la apreciación o forma de puntuar de cada individuo. En concreto, hay usuarios que acostumbran a dar puntuaciones positivas, utilizando puntuaciones negativas para productos realmente malos. Otros, sin embargo, reservan las puntuaciones más altas para los mejores productos, y suelen dar puntuaciones negativas.

A pesar de que los sistemas basados en puntuaciones son los más sensibles a este problema, también se puede dar en sistemas basados en preferencias implícitas. Por ejemplo, el tiempo que un usuario tiene para interactuar con el sistema, o el dinero que tiene disponible para la compra de nuestros productos, introducirá variaciones entre usuarios a pesar de que puedan tener gustos o intereses similares.

Es decir, la puntuación de un usuario va a depender de varios factores, y no sólo de la calidad real del producto valorado. La existencia de estas variaciones ya se había observado previamente [Herlocker et al., 1999; Resnick et al., 1994], pero su incorpo-

ración a los algoritmos tenía un papel secundario, limitándose a una normalización previa de las puntuaciones para impedir que afectase negativamente al resultado final. Un ejemplo de ello es la normalización de las puntuaciones utilizada en los algoritmos basados en usuarios (ver Sección 3.4.1.1). En ese caso, las variaciones se eliminaban, para evitar que influyesen en el cálculo de las semejanzas entre usuarios o productos. Nuestro algoritmo, sin embargo, se basa precisamente en dichas variaciones. En nuestra opinión, estas son un indicador mucho más preciso de la calidad de un determinado producto y su utilidad para el usuario.

De hecho, en la literatura podemos encontrar trabajos que, en cierta medida, habían tenido en cuenta este hecho. Por ejemplo, Jin et al. [2003] habían observado la diferencia entre las puntuaciones de los usuarios y sus preferencias reales, lo cual quedó plasmado en su algoritmo al desacoplar ambas en un modelo probabilístico que introducía dos variables ocultas para tal fin. Más recientemente, G. Potter adoptó un enfoque psicológico basado en la economía conductual, relacionando estas variaciones con los factores sociales o emocionales que afectan a cada individuo [Ellenberg, 2008]. Por ejemplo, si un usuario acaba de ver una buena película, a la que en consecuencia ha otorgado una buena puntuación, y justo después ve otra película todavía mejor, esta última posiblemente vaya a recibir una puntuación mayor que la que hubiese recibido en caso de que el usuario no hubiese visto la primer película. En este caso, el hecho de haber realizado una puntuación previa hace que el usuario, consciente de que la segunda película es mejor, tienda a subir la puntuación de esta última.

Todas estas observaciones han sido plasmadas en nuestro algoritmo, pero lo hemos hecho de una forma sencilla y a la vez eficiente: hemos interpretado las variaciones como *tendencias* de usuarios y productos. En lugar de utilizar complejos cálculos, las tendencias son fáciles de calcular, y lo que es más importante, su cálculo necesita muchos menos datos que los necesarios para calcular las similitudes. Esto nos permite desarrollar un algoritmo muy preciso pero con unos requisitos de memoria y tiempo muy bajos.

El concepto de tendencia se refiere a si un usuario es propenso a puntuar positivamente los productos o, por contra, a hacerlo negativamente. Es importante no confundir esta tendencia con lo elevado o no de la puntuación media del usuario. Por ejemplo, un usuario que se limite a valorar productos que le han gustado va a tener una media elevada, pero podría suceder que sus puntuaciones sean inferiores a la media de cada producto. Es decir, el usuario tiende a valorar los productos negativamente, incluso a pesar de que su media es alta. Por tanto, definimos la tendencia de un usuario, $\tau_{u.}$, como la diferencia media de sus puntuaciones respecto a la media de los productos:

$$\tau_{u.} = \frac{\sum_{i \in \mathcal{I}(u)} (r_{ui} - \overline{r_{.i}})}{\left|\mathcal{I}(u)\right|} \tag{4.1}$$

También nos interesa capturar la tendencia de un producto, $\tau_{.i}$, es decir, si los usuarios lo consideran un producto especialmente bueno o especialmente malo. No se trata de ver si el producto está bien valorado (lo que podría hacerse comprobando si su puntuación media está por encima de la media global), sino de si destaca entre los productos valorados por un usuario. Parece lógico pensar que si un usuario valora un producto por encima de los demás es porque considera que es un buen producto, o al menos mejor que los otros que ha puntuado. Una vez más, lo que nos interesa son las puntuaciones relativas, es decir, la valoración respecto de la media del usuario, y no la media absoluta del producto. Calculamos este valor como:

$$\tau_{.i} = \frac{\sum_{u \in \mathcal{U}(i)} (r_{ui} - \overline{r_{u.}})}{\left|\mathcal{U}(i)\right|} \tag{4.2}$$

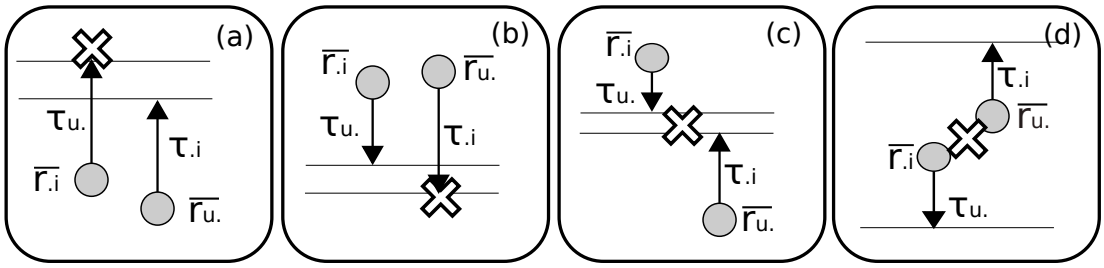


Figura 4.1: Posibles relaciones entre media (círculos) y tendencias (flechas).

Ambas tendencias toman valores que varían entre $-d$ y d, donde d es la diferencia entre la puntuación máxima y mínima permitida. El algoritmo tiene en cuenta tanto la media de usuario y producto como sus respectivas tendencias a la hora de calcular una predicción. Dependiendo de los valores de estas existen varios casos, que se reflejan en la Figura 4.1.

En el primer caso, Figura 4.1a, tanto el usuario como el producto tienen una tendencia positiva, es decir, el usuario tiende a valorar los productos por encima de la media de estos, y el producto tiende a ser valorado por encima de la media del usuario. Por tanto, parece una buena idea predecir una valoración por encima de la media de ambos. En concreto, usamos la fórmula:

$$\hat{r}_{ui} = \text{máx}(\overline{r_{u.}} + \tau_{.i}, \overline{r_{.i}} + \tau_{u.}) \tag{4.3}$$

donde el uso del máximo tiene la intención de dar una mejor valoración a este tipo de productos, cuya tendencia indica que son buenos.

El segundo caso, Figura 4.1b, es el contrario: tanto el usuario como el producto tienen tendencia negativa, es decir, el usuario suele valorar los productos por debajo de su media, y el producto tiende a ser valorado por debajo de la media del usuario. En este caso la predicción se calcula como:

$$\hat{r}_{ui} = \text{mín}(\overline{r_{u.}} + \tau_{.i}, \overline{r_{.i}} + \tau_{u.}) \tag{4.4}$$

donde el uso del mínimo tiene el objetivo de impedir que tal producto, cuya tendencia indica que es una mala recomendación, sea recomendado simplemente porque el usuario tiene una media muy alta.

El tercer caso, Figura 4.1c, se presenta cuando nos encontramos con un usuario "negativo" (su tendencia es valorar los productos por debajo de su media), y un buen producto (su tendencia es ser valorado por encima de la media del usuario) (o viceversa, usuario positivo y mal producto). Si las medias de ambos corroboran sus tendencias (es decir, usuario con media baja y producto con media alta), la predicción debería estar en un punto medio entre ambas, más próxima de una u otra según el valor de las distintas tendencias. La predicción se computa como:

$$\hat{r}_{ui} = \text{mín}(\text{máx}(\overline{r_{u.}}, (\overline{r_{.i}} + \tau_{u.})\beta + (\overline{r_{u.}} + \tau_{.i})(1-\beta)), \overline{v_{.i}}) \tag{4.5}$$

donde β es un parámetro que permite otorgar una mayor confianza en la media del usuario o en la del producto.

Finalmente, puede suceder que las medias no corroboren la tendencia (ver Figura 4.1d. Es decir, cuando un usuario con tendencia negativa valora un producto de media baja (con lo que la predicción debería ser mala), pero al mismo tiempo su media es alta y la tendencia del producto positiva (que por contra indicarían una buena valoración). En este último caso, la predicción se computa como

$$\hat{r}_{ui} = \overline{r_{.i}}\beta + \overline{r_{u.}}(1-\beta) \tag{4.6}$$

Como se observa, en los cuatro casos se utiliza una sencilla fórmula cuyo cómputo no va a depender del número de usuarios o productos del sistema. Obsérvese que el cálculo de media y tendencias sólo es necesario realizarlo la primera vez, pudiendo mantenerse almacenado. Su actualización, cada vez que tengamos nuevas puntuaciones, usuarios o productos, es muy sencilla y, una vez más, no depende del volumen de datos a tratar.

4.3. Evaluación de la predicción

4.3.1. Métricas y metodología de evaluación

Tradicionalmente, la tarea de predicción se ha evaluado utilizando métricas de precisión en la predicción, es decir, basadas en medir el error o diferencia entre la predicción hecha por el algoritmo y la puntuación real del usuario.

Generalmente, el proceso de evaluación consiste en escoger un conjunto de evaluación $\mathcal{T} = \{(u,i)|u \in \mathcal{U}, i \in \mathcal{I}\}$, formado por pares usuario-producto para los que se conoce la puntuación real hecha por el usuario. En un experimento *offline*, dicho conjunto estará formado por pares presentes en el conjunto de datos que se reservan para la evaluación, y por tanto no formarán parte del conjunto de entrenamiento.

Para la división de las puntuaciones entre conjuntos de evaluación y entrenamiento se pueden utilizar diferentes alternativas. Una de las más populares es la metodología *Given-N* [Breese et al., 1998], donde para cada usuario se seleccionan, aleatoriamente, un número fijo de puntuaciones que formarán parte del conjunto de entrenamiento, quedando las restantes disponibles para el conjunto de evaluación. Un valor bajo de N permite simular condiciones de baja densidad, es decir, donde el número de puntuaciones disponibles es muy bajo, características de entornos de *cold-start*.

Alternativamente, en lugar de escoger un número fijo de puntuaciones, se puede optar por seleccionar un determinado porcentaje como conjunto de entrenamiento. Nuevamente, porcentajes bajos permiten la evaluación en condiciones más adversas.

También es bastante habitual el separar ciertas puntuaciones, de entre aquellas en el conjunto de evaluación, para seleccionar valores más adecuados para los parámetros del algoritmo, usando validación cruzada.

Las medidas de error más populares son el *error absoluto medio* (MAE) y la *raíz del error cuadrático medio* (RMSE), calculadas respectivamente según las ecuaciones 4.7 y 4.8:

$$MAE = \frac{\sum_{(u,i)\in\mathcal{T}} |\hat{r}_{ui} - r_{ui}|}{|\mathcal{T}|} \tag{4.7}$$

$$RMSE = \sqrt{\frac{\sum_{(u,i)\in\mathcal{T}} (\hat{r}_{ui} - r_{ui})^2}{|\mathcal{T}|}} \tag{4.8}$$

La principal diferencia entre ambas es que RMSE otorga una mayor importancia a los errores más grandes, lo que es bastante razonable pues estos errores son los que probablemente tengan un mayor impacto en la percepción del usuario.

4.3.2. Métricas *GIM* y *GPIM*

Uno de los problemas de métricas tradicionales como MAE o RMSE es que al final el algoritmo es evaluado según su error medio, considerando todos los productos del conjunto de evaluación. En la práctica, sin embargo, no todos los errores tienen la misma importancia. Desde el punto de vista de un usuario, cometer un pequeño error en todas las predicciones es mucho menos grave que cometer un gran error, aunque sea en muy pocos productos. De hecho, los errores de predicción que realmente van a afectar al usuario son, sobre todo, aquellos en que un mal producto recibe una buena predicción, o al contrario, cuando un buen producto recibe una mala.

En esta tesis proponemos el uso de dos nuevas métricas, *Good Items MAE* (GIM) y *Good Predicted Items MAE* (GPIM), las cuales miden, respectivamente, el error absoluto medio de la predicción en los productos que son realmente buenos, y en aquellos que el sistema predice como buenos. Es decir:

$$GIM = \frac{\sum_{(u,i)\in\mathcal{T}_g} |\hat{r}_{ui} - r_{ui}|}{|\mathcal{T}_g|} \tag{4.9}$$

$$GPIM = \frac{\sum_{(u,i)\in\mathcal{T}_p} |\hat{r}_{ui} - r_{ui}|}{|\mathcal{T}_p|} \tag{4.10}$$

Con $\mathcal{T}_g = \{(u,i) \in \mathcal{T} | r_{ui} \geq \theta\}$, $\mathcal{T}_p = \{(u,i) \in \mathcal{T} | \hat{r}_{ui} \geq \theta\}$, y θ el umbral a partir del cual una puntuación es considerada buena, que por supuesto debe definirse antes de llevar a cabo la evaluación.

En lugar de tener en cuenta todos los productos en el conjunto de evaluación, estas métricas sólo tienen en cuenta el error en los productos relevantes. Por tanto, una de sus ventajas es que se valoran los errores de predicción que realmente tienen un impacto importante en la apreciación del usuario. En cierto modo, GPIM y GIM son el equivalente a precisión y *recall*, pero desde la perspectiva de la tarea de predicción.

Además, estas métricas tienen una ventaja adicional frente al uso de métricas de error tradicionales como MAE o RMSE, y es que ayudan a capturar la predisposición de un algoritmo a ofrecer puntuaciones demasiado optimistas o demasiado pesimistas, lo que puede ser un problema a la hora de implantar el sistema, a pesar de que el error medio no sea demasiado elevado. Por ejemplo, un sistema pesimista puede dar al usuario una impresión equivocada acerca de la calidad de un producto, haciendo que acabe desistiendo de su compra. Por contra, un sistema optimista puede llevar al usuario a pensar que el algoritmo está manipulado para aumentar las ventas, con lo que acabará desconfiando del sistema. El uso de las métricas propuestas permite detectar estas tendencias indeseables por parte del algoritmo: un algoritmo optimista presentará

muchos mejores resultados en GIM que en GPIM, mientras uno pesimista lo hará al contrario. La métricas tradicionales no permiten, sin embargo, detectar este problema.

Por supuesto, el predecir correctamente la puntuación en productos mal valorados sigue siendo necesario en ciertas aplicaciones, por lo que más que un sustituto de las métricas de error en la predicción tradicionales, *GIM* y *GPIM* son un complemento.

4.4. Experimentos y resultados

Dada la escasez de trabajos dedicados a comparar las distintas técnicas de filtrado colaborativo presentes en la literatura, el conjunto de experimentos que hemos realizado y presentamos en esta sección tiene el objetivo no sólo de evaluar el algoritmo basado en tendencias descrito en la Sección 4.2, sino también de evaluar el comportamiento de los algoritmos de filtrado colaborativo más populares en diferentes circunstancias.

En particular, y además de nuestro algoritmo basado en tendencias (TB), hemos considerado las siguientes técnicas:

- Basadas en usuario (UB), presentados en la Sección 3.4.1.1.
- Basadas en producto (IB, Sección 3.4.1.2).
- *Similarity fusion* (SF, Sección 3.4.1.3).
- Regresión lineal (RB, Sección 3.4.2.1).
- *Slope One* (SO, Sección 3.4.2.2).
- *Singular Value Decomposition* (SVD, Sección 3.4.6.1).
- Diferentes variantes de SVD regularizado (RSVD, RSVD2, NSDV2 y SVD++, Sección 3.4.6.2).
- *Personality Diagnosis* (PD, Sección 3.4.3.2).
- *Cluster-based soothing* (CBS, Sección 3.4.4).
- Modelo integrado kNN – SVD++ (IM, Sección 3.4.7)

Para la evaluación hemos utilizado dos de los conjuntos de datos más populares en sistemas de recomendación: Movielens [Herlocker et al., 1999] y Netflix [Bennett y Lanning, 2007]. A pesar de que el primero de ellos es relativamente pequeño y en la mayoría de trabajos actuales ha sido reemplazado por el Movielens 10M, de similares características pero mayor tamaño, es ideal para el propósito de esta sección. En

primer lugar, porque permite en parte comparar los resultados obtenidos en nuestros experimentos con los presentados en otros trabajos, pues muchas de las técnicas presentadas en la literatura han sido evaluadas sobre este conjunto de datos. En segundo lugar, porque tal y como hemos comentado anteriormente, muchas de estas técnicas no destacan precisamente por su eficiencia, y el evaluarlas en un conjunto de datos más grande alargaría injustificadamente el tiempo necesario para llevar a cabo nuestros experimentos.

Las técnicas más eficientes han sido evaluadas en el conjunto de datos Netflix, mucho más grande y utilizado en los trabajos más actuales. Aún así, para intentar evaluar el mayor número posible de algoritmos, en lugar de usar el conjunto de datos completo, hemos seleccionado aleatoriamente un 30 % de los usuarios y productos del mismo, obteniendo un conjunto de datos con en torno a los 10 millones de puntuaciones (frente a los 100 millones del conjunto original). Un subconjunto de los experimentos realizados ha sido ejecutado tanto en este conjunto reducido como en el original, obteniendo unos resultados linealmente relacionados ($R^2 = 99,91\,\%$), lo que indica que los resultados obtenidos en nuestros experimentos son extrapolables al conjunto Netflix completo.

Los experimentos se han realizado siguiendo la metodología introducida en la Sección 4.3, en la que dividimos el conjunto de datos en un subconjunto de evaluación y otro de entrenamiento. Con el objetivo de estudiar el comportamiento de las distintas técnicas en diferentes contextos, hemos utilizado dos aproximaciones para realizar dicha división.

En primer lugar, el uso de un porcentaje de las puntuaciones como conjunto de entrenamiento, reservando un 10 % de las puntuaciones como conjunto de evaluación. En concreto, hemos utilizado diferentes porcentajes, desde un 10 % hasta un 90 %, en intervalos del 10 %, lo que nos permite evaluar las técnicas en diferentes condiciones de densidad. Precisamente, para el estudio en condiciones más extremas, hemos empleado la técnicas *Given-N*, con N igual a 1, 2, 3, 4, 5, 7, 10, 12, 15, y 20 puntuaciones.

Cada experimento ha sido repetido 5 veces, con diferentes conjuntos de entrenamiento y evaluación, elegidos aleatoriamente, utilizándose el análisis de la varianza (ANOVA) para determinar si los resultados son estadísticamente significativos. Se ha empleado un nivel de significancia $\alpha = 0{,}05$.

4.4.1. Error en la predicción y falta de información

En primer lugar hemos estudiado el comportamiento de los diferentes algoritmos según la densidad de la información, es decir, variando el porcentaje de puntuaciones que forman parte del conjunto de entrenamiento. En concreto, hemos estudiado tanto

el error cometido en la predicción como el *coverage*. Los resultados en el conjunto de datos Movielens se muestran en las Figuras 4.2, 4.3 y 4.4, y en la Figura 4.5 se muestran los resultados en Netflix.

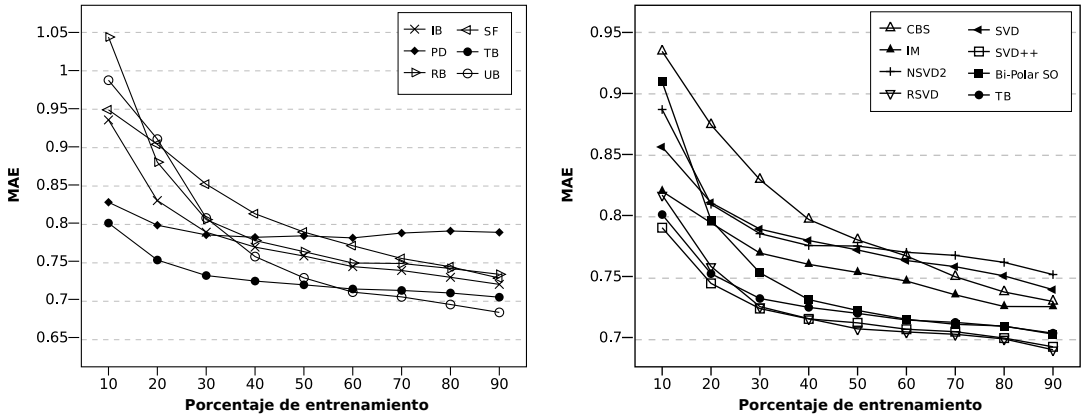


Figura 4.2: Error en la predicción (MAE) en Movielens según la densidad de la matriz. Obsérvese que los resultados de nuestro algoritmo basado en tendencias se muestran en ambas gráficas.

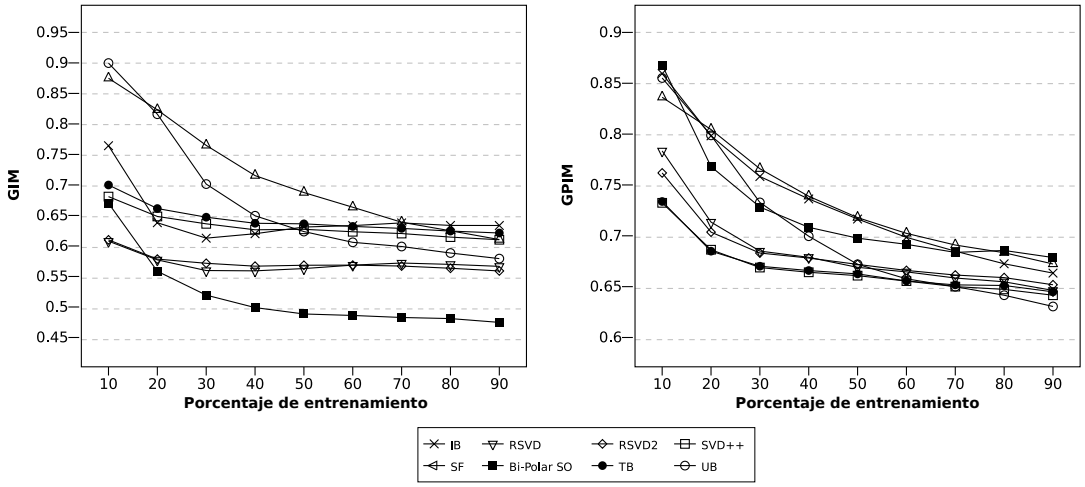


Figura 4.3: Evolución de GIM y GIPM en Movielens, según la densidad de la matriz.

Puede observarse como todos los algoritmos mejoran sus resultados a medida que aumenta el porcentaje de puntuaciones en el conjunto de entrenamiento. Este resultado es el esperado, y prueba que el error cometido por un algoritmo depende no tanto de la técnica en sí, sino de la cantidad de información de que dispone. De hecho, a medida que aumentamos la cantidad de información, puede observarse como en general también

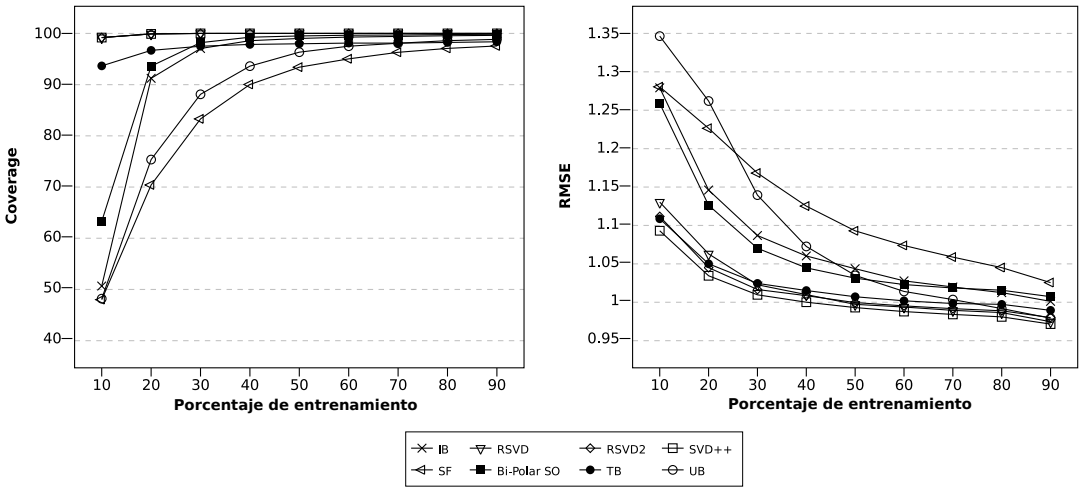


Figura 4.4: Evolución de RMSE y *coverage* en Movielens, según la densidad de la matriz.

se reducen las diferencias entre algoritmos. En condiciones de densidad relativamente elevadas, gran parte de las técnicas presentan resultados similares. Si consideramos un conjunto de entrenamiento del 80 %, no se aprecian diferencias estadísticamente significativas entre los seis mejores algoritmos ni en Movielens (UB, RSVD2, SVD++, RSVD, SO y TB) ni en Netflix (RSVD, RSVD2, SVD++, SO and TB).

Algunos autores han relacionado este hecho con un posible límite en la exactitud que podría llegar a alcanzar un algoritmo [Herlocker et al., 2004], y que estaría relacionada con la variabilidad natural que tenemos las personas, pues nuestra opinión sobre un producto será diferente en distintos momentos dependiendo de factores como el estado de ánimo, el contexto en que puntuamos un producto, etc. [Ellenberg, 2008].

Sin embargo, si bien es cierto que cualquier algoritmo va a estar sujeto a cierto error debido a esa variabilidad natural, los resultados que hemos obtenido demuestran que en gran medida el error que los algoritmos cometen está íntimamente relacionado con la falta de información. Es decir, la exactitud de las predicciones de un algoritmo está limitada por la cantidad de información que es capaz de extraer a partir de la matriz de puntuaciones. Lo cual a su vez está limitado por la cantidad de información que realmente está disponible, es decir, por la densidad de la matriz.

En otras palabras, existen tres factores que explican el error que puede cometer un algoritmo de recomendación: en primer lugar, la variabilidad natural de los individuos. En segundo lugar, la cantidad de información disponible, límite que se rebaja a medida que los usuarios interactúan con el sistema y puntúan diferentes productos. Y por último, las limitaciones de cada técnica para extraer información a partir de las

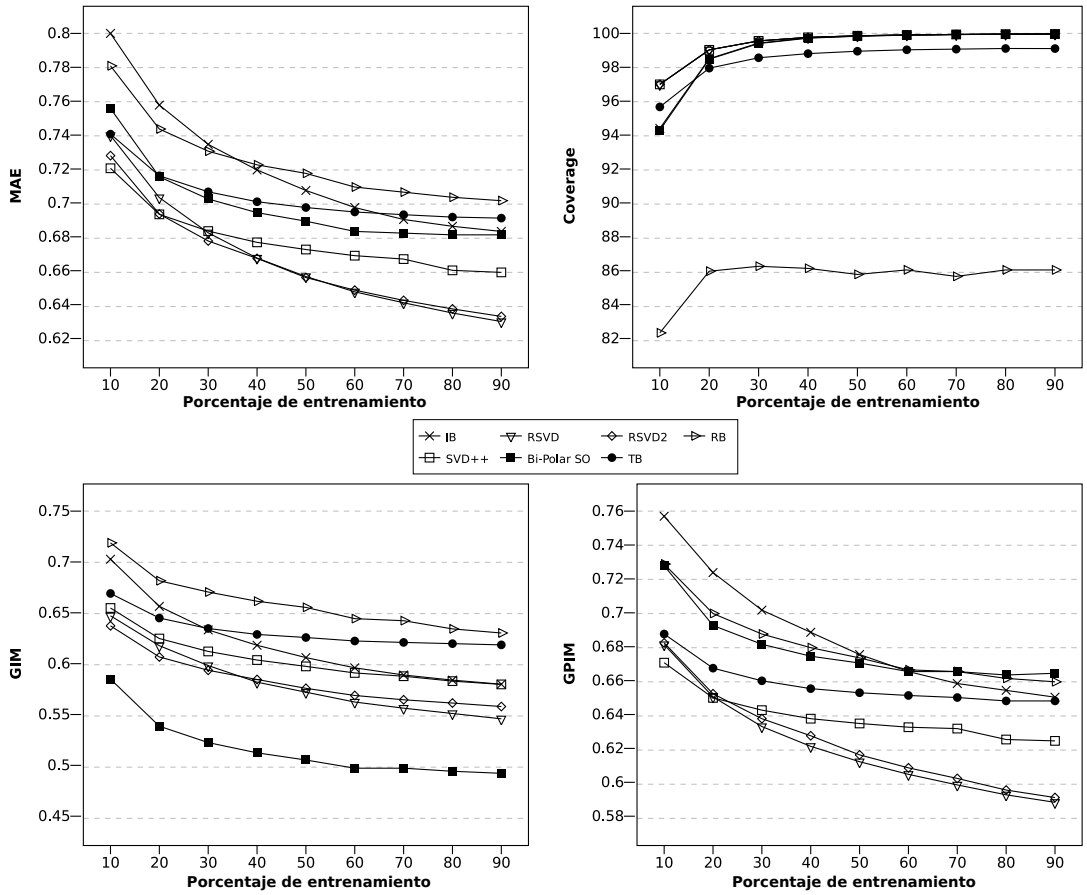


Figura 4.5: Error en la predicción y *coverage* en Netflix, según la densidad de la matriz.

puntuaciones. De las tres, esta última es la que más posibilidades tiene de ser mejorada gracias a la investigación y el desarrollo de nuevas técnicas, pues la primera es difícil de modelar y en gran medida, por tanto, un límite teórico, y la segunda depende más del éxito comercial de la aplicación y no tanto de las características del sistema de recomendación.

De hecho, la principal diferencia entre distintos algoritmos no está tanto en su exactitud o error, sino en la relación entre la cantidad de información disponible y el error cometido. En nuestra opinión, las mejores técnicas no son aquellas que presentan el error más pequeño pero en unas condiciones muy determinadas, sino aquellas que consiguen errores aceptables en condiciones adversas. Es decir, el impacto, desde el punto de vista del usuario, de mejorar la precisión en unas pocas décimas en condiciones favorables es mucho menor que una mejora de varios puntos en aquellas condiciones donde peores resultados se consiguen.

De hecho, si la cantidad de información disponible es elevada muchas de las técnicas estudiadas se comportan de manera similar, y por tanto la precisión en la predicción del algoritmo pierde importancia en relación a otros aspectos como la diversidad, eficiencia, etc. Sin embargo, a medida que disminuye la densidad, las diferencias entre algoritmos se acentúan, y algunas técnicas empeoran significativamente.

Por tanto, los experimentos en un contexto de baja densidad son más adecuados para evaluar la capacidad de un algoritmo concreto para interpretar la información disponible, extraer conocimiento útil a partir de esta, y utilizarlo para conseguir buenas predicciones. Dichos experimentos se han realizado utilizando la estrategia *Given-N*. Los resultados obtenidos se muestran en la Figuras 4.6 y 4.7. Para favorecer la claridad de las gráficas, sólo se muestran los resultados de los mejores algoritmos.

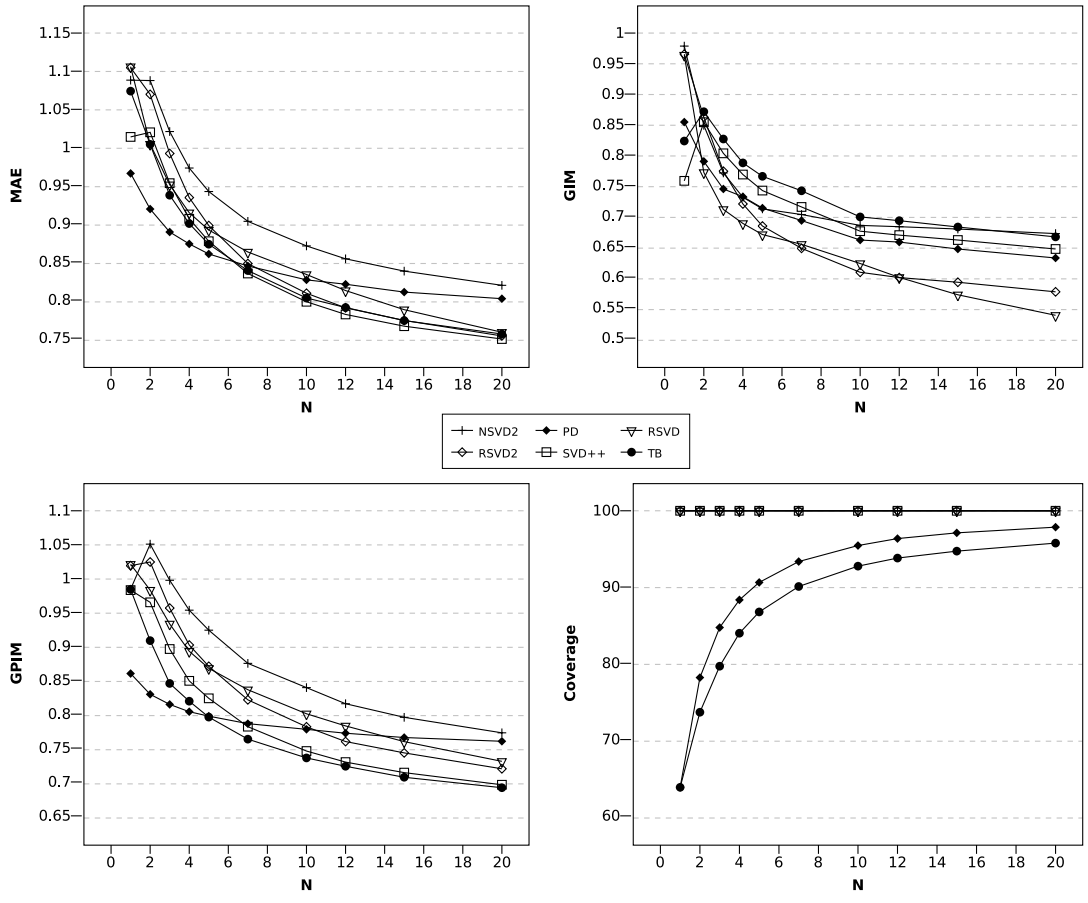


Figura 4.6: Resultados en la predicción de diferentes algoritmos en condiciones de baja densidad con la estrategia *Given-N*, en Movielens.

Observando estos resultados y los mostrados anteriormente, podemos comprobar

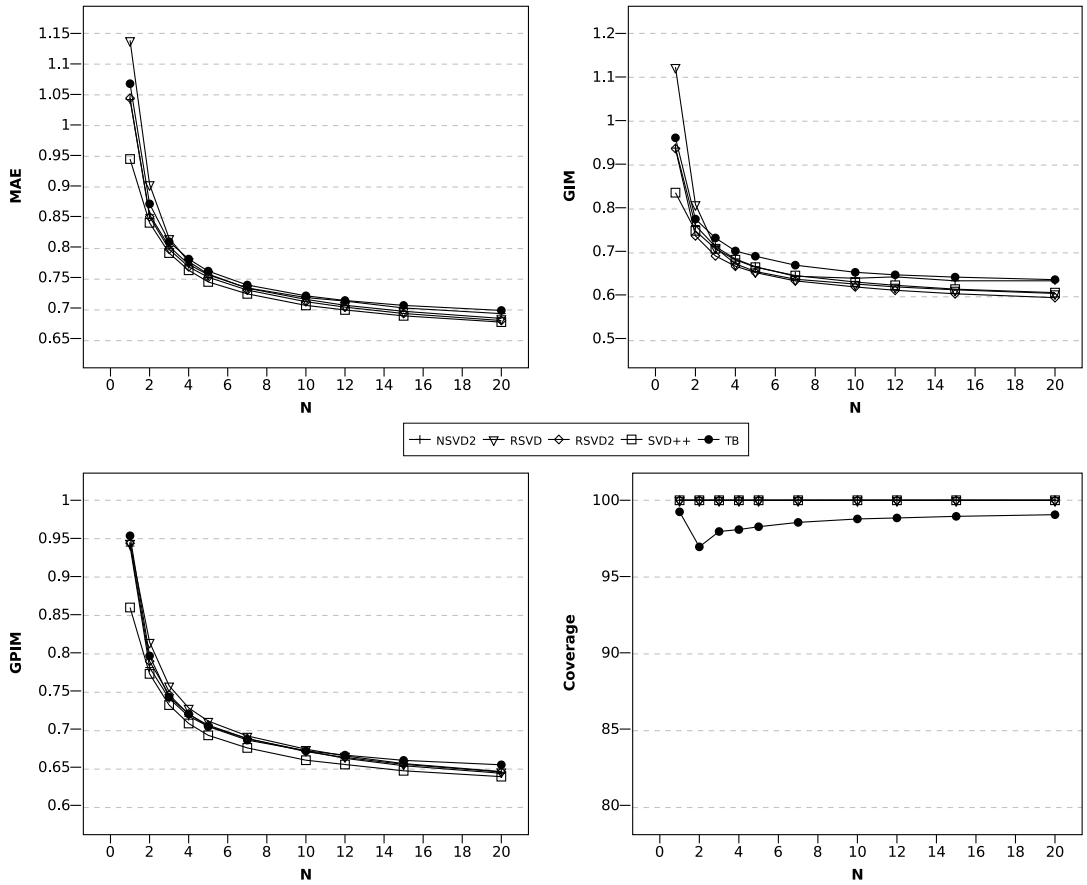


Figura 4.7: Resultados en la predicción de diferentes algoritmos en condiciones de baja densidad con la estrategia *Given-N*, en Netflix.

como el error cometido evoluciona de forma diferente según el algoritmo. Precisamente, esta evolución es una de las características que mejor define a un algoritmo y permite comparar entre sí diferentes técnicas. En la siguiente sección veremos como las técnicas basadas en memoria tienen una evolución diferente a las técnicas basadas en modelo.

4.4.2. Algoritmos basados en modelo frente a basados en memoria

En los algoritmos basados en memoria, el error disminuye rápidamente a medida que aumenta la densidad. En condiciones de densidad relativamente elevadas, estos algoritmos presentan muy buenos resultados. Sin embargo, a medida que esta disminuye el error se incrementa considerablemente. Esto es debido a que sólo utilizan una pequeña parte de la información disponible, es decir, la correlación entre usuarios (o productos, según el caso). Para poder calcular esa correlación, es necesario que ambos usuarios

hayan puntuado productos en común. Al disminuir la densidad de la matriz, también lo hacen las puntuaciones, por lo que el número de usuarios con puntuaciones en común será menor, y además la correlación estará basada en menos puntuaciones. Por tanto, el vecindario estará basado en menos usuarios, y a su vez la similitud entre ellos será más propensa a errores, de ahí que al final el error cometido sea mayor. De hecho, el *coverage* de estas técnicas disminuye notablemente al bajar la densidad, ya que en esos casos o bien no es posible calcular el vecindario de muchos usuarios, o bien no se puede predecir la puntuación de un producto ya que ningún vecino lo ha puntuado antes. No es por tanto difícil relacionar estos malos resultados con la falta de información de la que hablábamos en la sección anterior.

El algoritmo *Similarity Fusion*, al utilizar tanto las relaciones entre usuarios como las relaciones entre productos, utiliza mejor la información disponible, por lo que sus resultados en contextos de baja densidad son mejores, y al tiempo también presenta buenos resultados con mayor densidad.

Por otra parte, los algoritmos basados en modelo son mucho menos sensibles a los cambios de densidad de la matriz, presentando una curva mucho más suave. La razón es que la mayoría de estas técnicas capturan relaciones más complejas entre productos, usuarios y puntuaciones. Durante la fase de entrenamiento se utilizan algoritmos más potentes que la simple captura de correlaciones entre usuarios o productos, lo que permite extraer más información y por tanto obtener buenos resultados aún en contextos de baja densidad.

Sin embargo, si bien la información representada por los distintos modelos es más fácil de utilizar y en general más útil de cara a predecir una puntuación que la información presente directamente en la matriz de puntuaciones, su cantidad también es menor. Es decir, durante la construcción del modelo se pierde información.

Esto puede ser el resultado de una acción consciente, con el objetivo de tener un algoritmo más eficiente computacionalmente, como por ejemplo en las técnicas de reducción de la dimensionalidad (ver Sección 3.4.6), en la que la información menos relevante se descarta. En tales casos, a medida que aumenta la densidad los algoritmos basados en memoria, que no descartan información, pueden llegar a obtener mejores resultados.

Pero también puede ser resultado de que ciertos aspectos de la información no puedan ser representados por el modelo. Por ejemplo, el algoritmo de regresión lineal sólo considera las relaciones lineales entre productos, por lo que otro tipo de relaciones se perderán al construir el modelo. Es decir, los resultados de estos algoritmos van a depender del grado o bondad de ajuste del modelo a los datos realmente presentes en la matriz, por lo que algoritmos que funcionan bien en un dominio pueden no hacerlo

en otro.

En concreto, en los dos conjuntos de datos estudiados, los algoritmos basados en SVD obtienen muy buenos resultados. De hecho, las técnicas SVD con regularización destacan como las más precisas, en diversas situaciones, y si bien alguna de las técnicas basadas en memoria obtiene mejores resultados en condiciones de alta densidad, sólo nuestro algoritmo basado en tendencias es capaz de competir con estas en todos los casos.

Por el contrario, las técnicas de agrupamiento presentan muy malos resultados. De hecho, los resultados de nuestros experimentos muestran que incluso la media global es una mejor predicción que la media del grupo con los usuarios más próximos, en la que se basa la técnica de *cluster-based smoothing*. Tal como podemos observar en la Figura 4.8, a medida que aumentamos el número de grupos, también lo hace el error, al tiempo que disminuye el *coverage*.

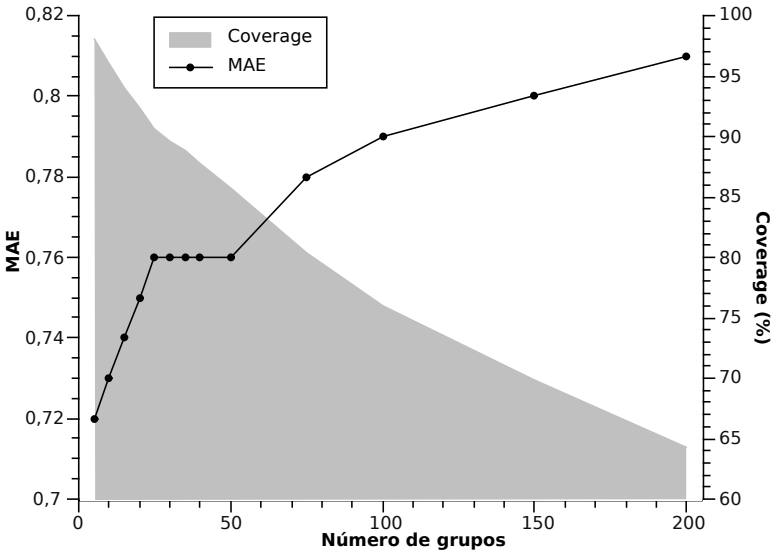


Figura 4.8: Evolución del error en la predicción y *coverage* según el número de grupos.

Esto podría parecer sorprendente, pues al final este algoritmo lo que hace es agrupar usuarios con gustos parecidos. En el dominio estudiado, podemos pensar que cada grupo corresponde a un conjunto de usuarios a los que les gusta el mismo género, director, actor, etc. A pesar de que esto pueda parecer adecuado, en la práctica es difícil que estos grupos existan. Que a un usuario le gusten las películas de acción no implica que no le gusten las comedias, o que le guste un director no quiere decir que le vayan a gustar todas sus películas. En la práctica, los gustos de la mayoría de usuarios son

demasiado dispersos y diferentes entre sí, lo que hace difícil clasificar a un usuario en una categoría concreta.

4.4.3. Algoritmo basado en tendencias: una técnica sencilla pero precisa

Por otra parte, el algoritmo basado en tendencias sí que presenta muy buenos resultados. De hecho, es el que ofrece los mejores resultados, junto con ciertas variantes de los algoritmos de factorización de la matriz, tal como se puede observar en las Figuras 4.2, 4.5, 4.6 y 4.7.

En condiciones de alta densidad, es sólo ligeramente inferior a algunas aproximaciones basadas en SVD y a técnicas basadas en vecinos. De hecho, en muchos casos los experimentos realizados no han mostrado diferencias estadísticamente significativas entre el algoritmo basado en tendencias y estos otros. Además, los resultados en las métricas GPIM y GIM muestran que comete menos errores en los productos relevantes, por lo que en general sus errores serán menos visibles para el usuario que los cometidos por otras técnicas. Destaca especialmente en GPIM, lo que indica que es un algoritmo con gran precisión, es decir, que comete pocos errores en aquellos productos para los que predice una puntuación elevada, lo que en general es un aspecto de trascendental importancia en aplicaciones reales. Además, sus resultados en GIM, si bien inferiores a otras técnicas, también son buenos, lo que indica que presenta unos resultados no sesgados, ni demasiado optimistas ni demasiado pesimistas.

En condiciones de baja densidad los resultados también son muy buenos, siendo el algoritmo que presenta globalmente los mejores resultados junto a algunas variantes de SVD. Con un conjunto de entrenamiento inferior al 50 %, supera en precisión a la mayor parte de técnicas, entre ellas las técnicas basadas en vecinos. De hecho, en condiciones de densidad extremadamente baja, de los algoritmos estudiados sólo RSVD, RSVD2 y SVD++ presentan resultados equivalentes. Además, al igual que PD y las técnicas basadas en SVD, su *coverage* apenas disminuye al reducirse la densidad de la matriz. Por ejemplo, con un conjunto de entrenamiento de sólo 5 puntuaciones por usuario, todavía es capaz de obtener una predicción para el 90 % de los productos, mientras la técnica basada en usuarios apenas cubre el 12 %, *Slope One* menos del 9 % y la técnica basada en productos ni siquiera alcanza en 6 %.

4.4.4. Eficiencia y escalabilidad de los algoritmos

Precisamente, es en la eficiencia computacional donde nuestro algoritmo basado en tendencias obtiene los mejores resultados, lo que es especialmente interesante pues

escalabilidad y eficiencia son aspectos claves para el éxito del sistema en las aplicaciones actuales.

En primer lugar, analizamos la complejidad computacional de los diferentes algoritmos, la cual es un buen indicador de la escalabilidad de cada uno de ellos.

Algoritmo	**Entrenamiento**	**Predicción**
User-Based	-	$O(mn)$
Item-Based	$O(mn^2)$	$O(n)$
Similarity Fusion	$O(n^2m + m^2n)$	$O(mn)$
Personality Diagnosis	$O(m^2n)$	$O(n)$
Regression-Based	$O(mn^2)$	$O(n)$
Slope One	$O(mn^2)$	$O(n)$
LSI/SVD	$O((m+n)^3)$	$O(1)$
RSVD	$O(mnk)$	$O(1)$
RSVD2	$O(mnk)$	$O(1)$
NSVD2	$O(mnk)$	$O(1)$
SVD++	$O(mn^2k)$	$O(1)$
Integrated model	$O(mn^2k)$	$O(1)$
Cluster-Based Smoothing	$O(mn\alpha + m^2n)$	$O(mn)$
Tendencies-Based	$O(mn)$	$O(1)$

Tabla 4.1: Complejidad computacional de los diferentes algoritmos estudiados.

La Tabla 4.1 muestra el análisis realizado[1]. Hemos separado la complejidad en dos partes, entrenamiento y predicción. La primera corresponde al coste de entrenamiento o construcción del modelo; la segunda, al coste de realizar una única predicción. En general, es preferible minimizar esta última, pues generalmente el modelo se construye *offline* y se utiliza para realizar un gran número de predicciones. Salvo en dominios que requieran la actualización frecuente del modelo, el tener una complejidad baja a la hora de realizar una predicción puede compensar una mayor complejidad en el entrenamiento. Como se puede ver, nuestro algoritmo basado en tendencias destaca como el más eficiente, con una complejidad de $O(mn)$ en entrenamiento y de $O(1)$ en predicción.

Estos resultados teóricos se confirman al estudiar los tiempos de ejecución. La Figura 4.9 muestra precisamente el tiempo de entrenamiento y predicción de los distintos algoritmos, utilizando el conjunto de datos Movielens y un conjunto de entrenamiento del 80 %, medidos en un PC con procesador Intel Quad Core Xeon a 2.66 GHz, 2×6

[1]Hemos asumido que en el algoritmo basado en usuarios los vecinos se calculan en tiempo de predicción.

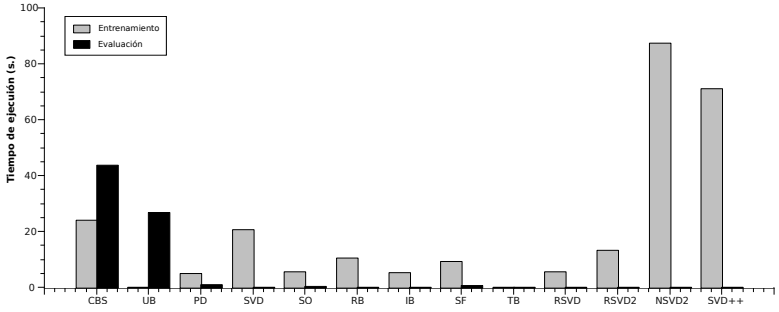


Figura 4.9: Tiempos de entrenamiento y predicción de diferentes algoritmos, utilizando el conjunto de datos Movielens.

MiB de cache y 1333 FSB, con 8 GiB de RAM[1]. El tiempo de predicción mostrado consiste en el tiempo necesario para predecir todas las puntuaciones presentes en el conjunto de evaluación (en torno a las 10.000 puntuaciones).

Como es de esperar, la mayoría de algoritmos basados en modelo presentan un tiempo de predicción muy bajo, pero en cambio la construcción del modelo puede ser bastante compleja. Por ejemplo, técnicas como SVD++, que presentan muy buenos resultados en cuanto a la precisión en la predicción, requieren un tiempo de entrenamiento muy elevado. Por otra parte, las técnicas basadas en usuario no necesitan una fase de entrenamiento, pero esto repercute en un mayor tiempo de predicción. En cambio, nuestra propuesta, el algoritmo basado en tendencias, obtiene muy buenos resultados en ambas fases. Por ejemplo, su tiempo de entrenamiento de tan sólo 13,2 ms. es varios órdenes de magnitud mejor que los algoritmos con resultados equivalentes en términos de precisión, como RSVD (5.433 ms.), RSVD2 (13.198 ms.) o SVD++ (71.025 ms.). En cuanto al tiempo de predicción, sus 12 ms. son similares a los obtenidos por otras aproximaciones basadas en modelo, y mucho mejores que técnicas basadas en memoria. En Netflix (con un conjunto de entrenamiento del 80 %), se obtienen resultados similares: el tiempo de entrenamiento de nuestro algoritmo es de tan sólo 4 s., mucho mejor que el de RSVD (640 s.), RSVD2 (860 s.) o SVD++ (9.700 s.).

4.5. Algoritmos basados en vecinos para predicción

A pesar de ser una de las técnicas más antiguas, los algoritmos basados en vecinos siguen siendo uno de los métodos de filtrado colaborativo más populares. En esta sección analizamos cómo determinadas decisiones afectan a sus resultados. En particular, nos

[1]Los algoritmos han sido implementados en Java. Obsérvese que diferentes implementaciones podrían obtener tiempos de ejecución diferentes.

centramos en las diferentes variantes de algoritmos basados en usuario. Su uso en la tarea de predicción contempla tres fases:

1. Cálculo de la similitud entre usuarios.
2. Selección del vecindario.
3. Predicción de la puntuación a partir de los vecinos.

En la Sección 4.5.1 estudiamos las dos posibles alternativas a la hora de elegir el vecindario. En la Sección 4.5.2 analizamos las limitaciones de las medidas de similitud más populares, y finalmente en la Sección 4.5.3 evaluamos las distintas técnicas de predicción. Los experimentos se han realizado usando la misma metodología que hemos usado en las pruebas presentadas en la sección anterior.

4.5.1. Selección del vecindario

Una correcta selección del vecindario tiene una gran importancia en los algoritmos basados en usuario, ya que los vecinos van a ser usados en las fases posteriores. Dado que la predicción se va a basar en las puntuaciones de los vecinos, una incorrecta selección del vecindario tendrá un gran impacto en la precisión final del algoritmo.

El vecindario escogido dependerá de dos factores: la medida de similitud empleada y la técnica usada para seleccionar los vecinos. En las Figuras 4.10 y 4.11 se muestran los resultados obtenidos con distintas medidas de similitud, usando como técnica de selección los vecinos más similares y el umbral de correlación, respectivamente.

En lo relativo a la técnica de selección de vecinos, los resultados que hemos obtenido son consistentes con los presentados por Herlocker et al. [2002]: la mejor estrategia es la selección de los k vecinos más similares. Podría parecer que cuanto menor sea el valor de k mejores resultados se van a obtener, ya que la recomendación estaría basada en los usuarios más parecidos al actual. Sin embargo, esto no es así, y tal y como se observa en la Figura 4.10, el error se incrementa considerablemente cuando se utilizan menos de 20 vecinos. De hecho, los resultados con 50 o 100 vecinos son mucho mejores que los obtenidos a partir de los 5 usuarios más parecidos al actual. Además, los resultados son consistentes independientemente de la medida de similitud empleada. Es decir, considerar un buen número de vecinos, aún cuando su similitud con el actual no sea demasiada, es mucho mejor que limitarse a unos pocos vecinos muy similares.

De hecho, si analizamos los resultados obtenidos con la técnica de umbral de correlación (ver Figura 4.11), se puede observar claramente este comportamiento. Al aumentar el umbral, también lo hace el error. Es importante destacar que la aparente mejora en

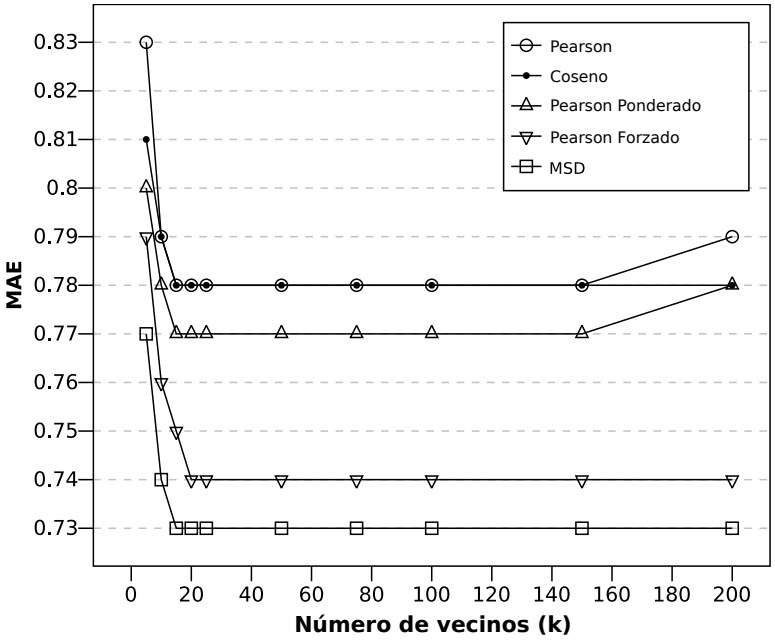


Figura 4.10: Error en la predicción de algoritmos kNN según el número de vecinos, con un conjunto de entrenamiento del 80 % en el conjunto de datos Movielens. Obsérvese que con pocos vecinos el error se incrementa considerablemente.

ciertos casos, como por ejemplo con el coeficiente de Pearson ponderado, se produce realmente con un *coverage* próximo a cero y corresponde a aquellos productos cuya puntuación es más sencilla de predecir. De hecho, Herlocker et al. [2002] mostraron como realmente ni siquiera existe mejora, pues si bien el error medio parece disminuir, esto se debe a que directamente el algoritmo no es capaz de predecir puntaciones para muchos productos. De haberse considerado sólo el error en los productos cuya puntuación sí es capaz de predecir, los resultados también empeorarían al aumentar el umbral. Precisamente, el *coverage* disminuye también al aumentar el umbral, lo que muestra que realmente es difícil encontrar usuarios muy similares.

En nuestra opinión, hay dos razones que explican este comportamiento:

- En primer lugar, la propia variabilidad de los usuarios, de la que ya se ha hablado, y que implica que si ya es difícil encontrar usuarios con un patrón de puntuaciones similar, más lo es encontrar usuarios cuyos gustos o intereses sean prácticamente iguales. La selección de un umbral de similitud demasiado alto restringe el algoritmo a ese limitado grupo de usuarios realmente similares, y de ahí los malos resultados, y en particular el bajo *coverage* obtenido. Por contra, al relajar el umbral se tienen en cuenta otros usuarios que, si bien pueden diferenciarse algo

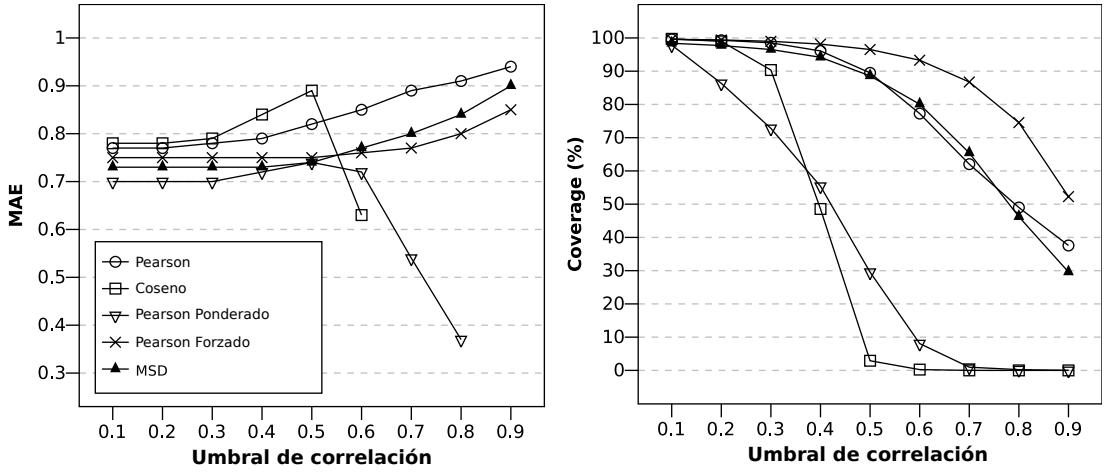


Figura 4.11: Error en la predicción y *coverage* de algoritmos basados en vecinos, según el umbral de similitud y un conjunto de entrenamiento del 80 %.

del usuario actual, en general tienen gustos similares y por tanto son igualmente útiles.

- Por otra parte, las limitaciones de las métricas de similitud. En general, las técnicas más populares se comportan razonablemente bien cuando dos usuarios comparten puntuaciones en un buen número de productos. Sin embargo, y especialmente en casos de baja densidad de la matriz, es bastante habitual que dos usuarios sólo hayan puntuado en común un puñado de productos, que muchas veces corresponden a productos populares y que no siempre son representativos del perfil real del usuario. Es decir, este tipo de técnicas, al basarse no en el perfil completo de los usuarios, sino sólo en aquellos productos que ambos han puntuado, muchas veces acaban comparando puntuaciones que no reflejan realmente los gustos o intereses de los usuarios, y de ahí los malos resultados. De hecho, tal y como hemos visto anteriormente, este tipo de algoritmos empeora significativamente al reducirse la densidad de la matriz, lo que está en gran parte relacionado con este problema.

4.5.2. Limitaciones de las medidas de similitud

Precisamente, las limitaciones de las medidas de similitud han sido objeto del estudio presentado en esta sección. En particular, hemos estudiado en torno a 25.000 relaciones entre usuarios, elegidas aleatoriamente a partir del conjunto de datos Movielens 10M, con el fin de analizar el problema presentado en la sección anterior, y explicar sus

causas.

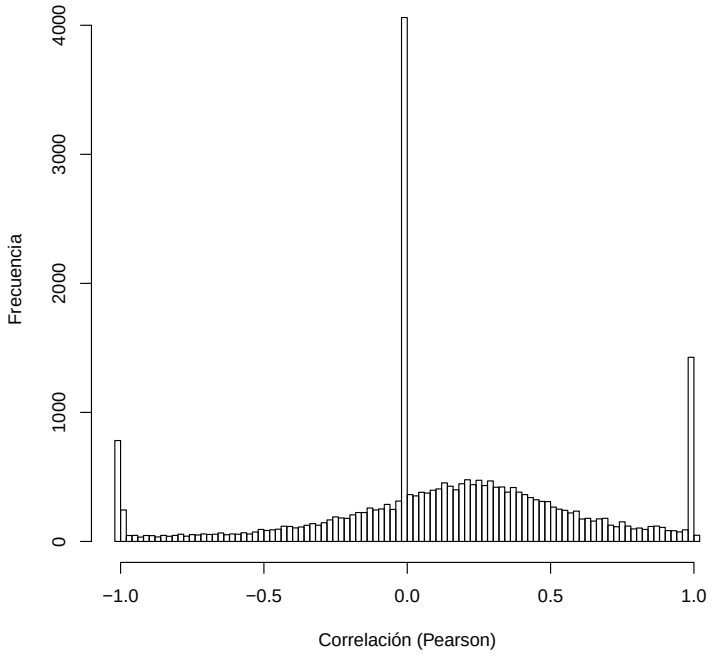


Figura 4.12: Histograma de la distribución del coeficiente de correlación de Pearson entre los usuarios analizados.

En primer lugar, tal y como se muestra en la Figura 4.12, podemos ver como la mayoría de los casos analizados presentan una correlación muy baja, entorno al 0, lo que prueba lo afirmado en la sección anterior acerca de la naturaleza de los usuarios y la dificultad de encontrar relaciones de similitud entre ellos debido a la falta de información. Puede sorprender, sin embargo, el gran número de usuarios con una correlación perfecta, sea positiva (cercana a 1,0) o negativa (cercana a -1,0), que parece contradecir lo anterior. Esto es, sin embargo, el resultado de una de las limitaciones más importantes de las medidas de similitud, que es su comportamiento poco fiable cuando el número de productos que ambos usuarios han puntuado es muy bajo. De hecho, tal y como se observa en la Figura 4.13, sólo en el caso de usuarios con pocas puntuaciones en común se llega a obtener una correlación elevada.

Obviamente, en dichos casos la medida de similitud no refleja realmente el parecido entre usuarios. Por ejemplo, si dos usuarios tienen gustos muy diferentes es posible que sólo tengan en común puntuaciones de alguna película especialmente reconocida

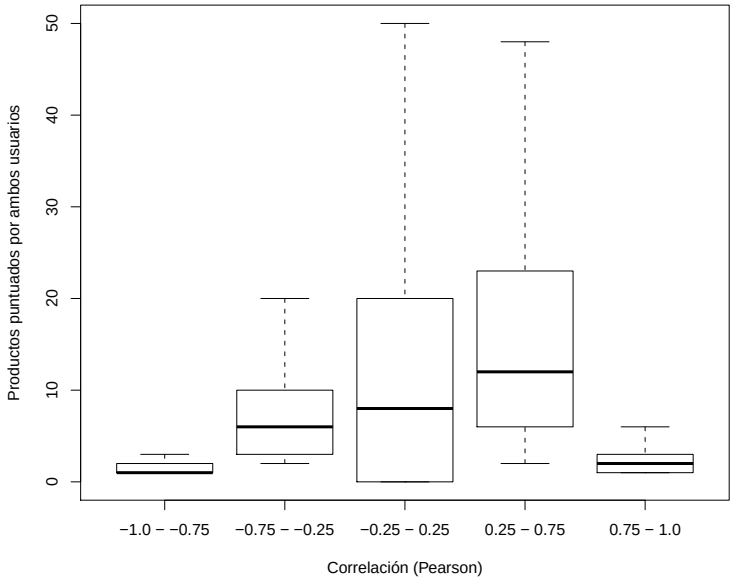


Figura 4.13: Número de productos que ambos usuarios han puntuado según el coeficiente de correlación de Pearson. Las observaciones atípicas se han ocultado para simplificar la gráfica y facilitar su comprensión.

o popular. Si ambos han otorgado una buena puntuación a estas películas (lo cual es bastante probable ya que estamos suponiendo que se trata de películas especialmente reconocidas), serán considerados similares aún cuando realmente no lo son. Un caso extremo serían usuarios que hayan puntuado únicamente dos productos en común, pues medidas de similitud como el coeficiente de Pearson los considerarían perfectamente correlacionados independientemente del valor de las puntuaciones. Estas limitaciones deben de tenerse en cuenta durante el diseño del sistema de recomendación.

En particular, el uso del coeficiente de Pearson ponderado mitiga este problema [Herlocker et al., 1999] al tener en cuenta el número de productos puntuados en común. El resultado, sin embargo, es una reducción en el número de usuarios con similitud elevada, tal y como se observa en la Figura 4.14. Es decir, estas se correspondían a usuarios que habían puntuado muy pocos productos en común, y desaparecen al ponderar la similitud. De todas formas, esto no es un problema, pues la mayor parte de ellas posiblemente fuesen erróneas, sino más bien un reflejo de la dificultad que tiene el encontrar similitudes entre usuarios.

En parte, esto se debe a la falta de información. Debido a que la matriz de puntuaciones es generalmente dispersa, es posible que usuarios que realmente tienen gustos

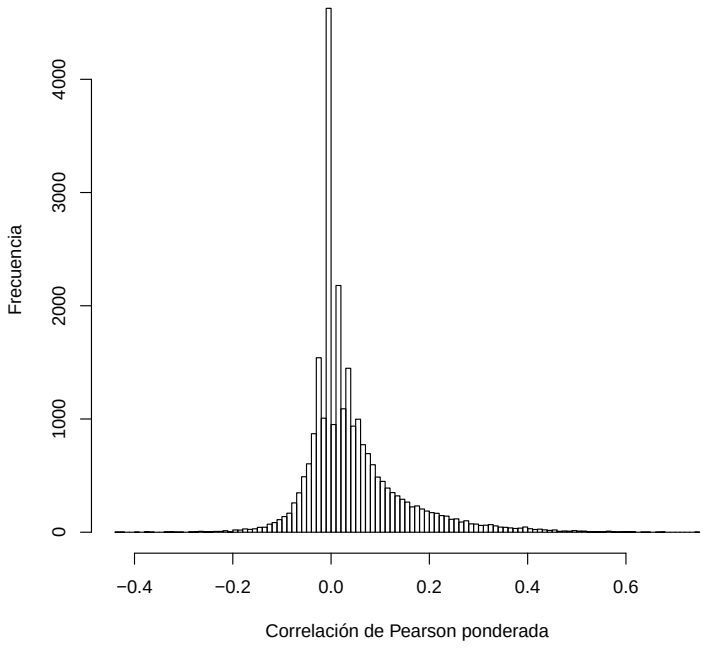


Figura 4.14: Histograma de la distribución del coeficiente de correlación de Pearson ponderado.

similares no hayan puntuado demasiados productos en común, lo que afecta al cálculo de la similitud, como hemos visto. De hecho, tal y como se observa en la Figura 4.15, el porcentaje de productos que dos usuarios tienen en común es generalmente muy bajo.

Sin embargo, la dispersión de la matriz no es el único motivo. Si bien usuarios con pocas puntuaciones tendrán más dificultades para encontrar usuarios semejantes, en la práctica usuarios muy activos también pueden sufrir este problema, tal y como se observa en la Figura 4.16. En este caso, el problema no se debe solamente a la falta de información, sino también a las diferencias de gustos entre usuarios. Como ya hemos mencionado anteriormente, aunque dos usuarios puedan compartir ciertos intereses, lo normal es que también difieran en otros. En estos casos, los productos que le interesan sólo a uno de los usuarios posiblemente no hayan sido puntuados por el otro, lo que disminuye el porcentaje de productos en común.

Se podría pensar, por tanto, que las aproximaciones basadas en productos son una mejor alternativa. Sin embargo, esto no es así, y tal y como se observa en la Figura 4.17 las variaciones de los usuarios también acaban reflejadas en la similitud entre productos.

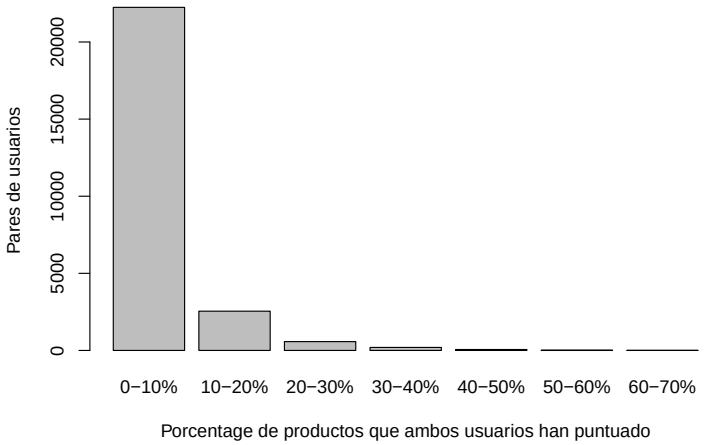


Figura 4.15: Distribución de las relaciones estudiadas según el porcentaje de productos puntuados por ambos usuarios, relativo a la suma de productos puntuados entre los dos.

Como conclusión, podemos afirmar que las medidas de similitud empleadas en la literatura son en ocasiones demasiado estrictas, y tienden a buscar usuarios demasiado similares. En ciertos casos, medidas de similitud más sencillas, como las que hemos presentado en [Cacheda et al., 2011b], son una mejor alternativa. Se basan en reconocer que si la información de que se dispone es escasa, las conclusiones que se pueden sacar de las medidas tradicionales va a ser posiblemente equivocada, y por tanto se deben utilizar medidas menos ambiciosas.

Por ejemplo, una de las medidas propuestas consiste en calcular la similitud en base a los productos que ambos usuarios han puntuado, sin prestar atención a la puntuación en sí. La idea es que si dos usuarios han puntuado los mismos productos, esto muestra en cierto modo un interés común, y en ausencia de más información conformarse con este concepto simplificado de similitud es mejor que intentar calcular la similitud en base a funciones de correlación o similares que darán malos resultados al no disponer de información suficiente.

4.5.3. Predicción de la puntuación

Por último, hemos analizado las dos técnicas usadas para la predicción de puntuación: la media ponderada por similitud de los vecinos, y la media normalizada (ver Sección 3.4.1.1). En particular, resulta especialmente interesante observar el impacto de la normalización.

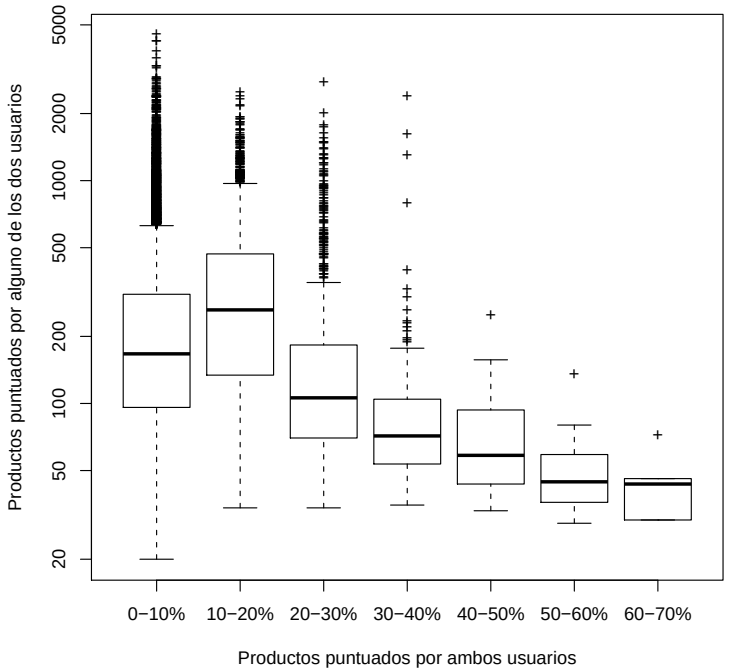


Figura 4.16: Distribución del número total de productos puntuados por cada par de usuarios, según el porcentaje de productos puntuados en común.

Tal y como se observa en la Figura 4.18, ésta mejora sustancialmente los resultados en todos los casos estudiados[1]. De hecho, bajo condiciones de alta densidad, los algoritmos que usan normalización son superiores a los que no la usan, independientemente de la medida de similitud empleada. A partir de cierto nivel de densidad, el peor de los algoritmos con normalización presenta mejores resultados que el mejor de los algoritmos sin normalización. Es decir, el uso de la normalización tiene un mayor impacto en los resultados que la medida de similitud empleada.

Esta es una observación particularmente interesante, pues históricamente los algoritmos de recomendación se han centrado en buscar las similitudes entre usuarios. Sin embargo, el modelar las diferencias entre ellos (en concreto, el uso de la normalización para corregir estas diferencias), tiene un mayor impacto en la precisión que la propia medida de similitud empleada.

Este hecho está directamente relacionado con las limitaciones de las medidas de similitud presentadas anteriormente, y en concreto con la dificultad de calcular de forma

[1]Este comportamiento ya había sido observado previamente por Herlocker et al. [2002]

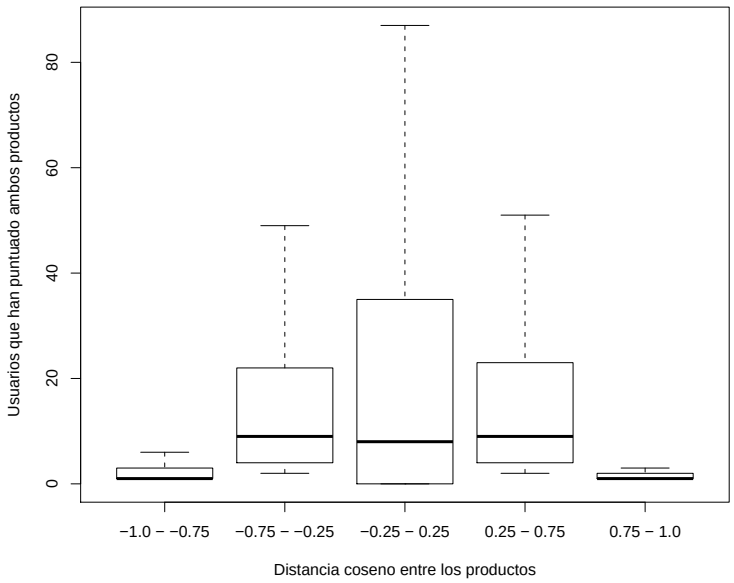


Figura 4.17: Número de usuarios que han puntuado ambos productos, según la similitud entre ellos.

fiable la similitud entre usuarios cuando la información disponible no es suficiente. En estos casos, las diferencias son más sencillas de calcular que las similitudes, y por tanto tener esto en cuenta mejora sustancialmente los resultados.

Precisamente, este es el motivo de los buenos resultados presentados por el algoritmo basado en tendencias. En lugar de buscar la similitud entre usuarios, intentamos modelar las diferencias en forma de tendencias, las cuales pueden ser calculadas de forma precisa incluso cuando la información disponible es muy limitada. El resultado es que un algoritmo sencillo basado en diferencias es capaz de realizar predicciones con precisión similar a algoritmos mucho más complejos basados en similitudes.

4.6. Conclusiones

En este capítulo se ha estudiado el comportamiento de la técnica de filtrado colaborativo en la tarea de predicción. Hemos realizado un amplio estudio de varias de las técnicas más populares, incluyendo la evaluación con diferentes condiciones de densidad de la matriz, lo que nos ha permitido extraer importantes conclusiones.

En primer lugar, hemos visto la importancia de caracterizar un algoritmo no sólo por el error cometido en un entorno concreto, sino por su evolución a medida que varían las

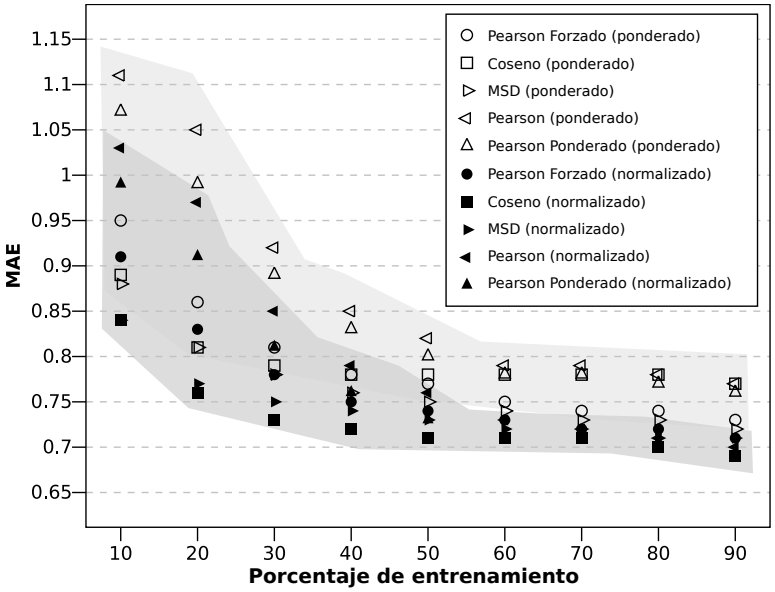


Figura 4.18: Efecto de la normalización en el error en la predicción.

condiciones. Esta evolución permite identificar la capacidad del algoritmo para extraer información a partir de las puntuaciones de los usuarios, y es un buen reflejo de cuál puede ser la precisión real del mismo en un entorno distinto a aquel para el que fue diseñado. En particular, hemos visto la diferencia entre algoritmos basados en memoria y algoritmos basados en modelo, así como la importancia de que el modelo se ajuste a los datos con los que va a ser usado. En concreto, mientras algoritmos basados en la factorización de la matriz ofrecen buenos resultados, las técnicas de agrupamiento presentan una precisión muy baja.

También hemos presentado dos nuevas métricas de evaluación, GIM y GPIM, que permiten una evaluación de la predicción más cercana a la percepción que tendrá el usuario del comportamiento del algoritmo, al tiempo que permiten detectar si una técnica es demasiado optimista o pesimista.

Finalmente, hemos propuesto la técnica de filtrado colaborativo basado en tendencias, que a pesar de su sencillez consigue resultados similares a otras técnicas mucho más complejas. Por una parte, este hecho nos hace dudar de la conveniencia del desarrollo de técnicas complejas, difíciles de entender, muy dependientes de las características concretas del dominio en que son usadas, y generalmente poco eficientes desde el punto de vista computacional, lo que dificulta su uso en entornos comerciales. Como hemos visto, técnicas sencillas obtienen similares resultados en términos de precisión y son mucho más fáciles de implementar y desplegar en aplicaciones reales. Por otra parte,

hemos comprobado que las medidas de similitud entre usuarios o productos no siempre obtienen los resultados esperados, debido en gran medida a la dificultad de su cálculo, especialmente en contextos de baja densidad. En estos casos, se debe optar por medidas más sencillas o por algoritmos centrados en buscar las diferencias entre usuarios o productos, que, como hemos visto en este capítulo, se pueden calcular de forma fiable sin necesidad de contar con grandes volúmenes de información.

4. TÉCNICAS DE FILTRADO COLABORATIVO EN LA TAREA DE PREDICCIÓN

Capítulo 5

Algoritmos kNN para recomendación

Tradicionalmente, la investigación en sistemas de recomendación se ha centrado en la tarea de predicción, viéndose la de recomendación relegada a un segundo plano. De hecho, sólo en el caso de sistemas que funcionan con preferencias implícitas o unarias ha tenido la recomendación un papel relevante. En sistemas basados en puntuaciones, la predicción ha centrado la mayor parte de los esfuerzos y apenas se pueden reseñar un puñado de trabajos dedicados a la recomendación, a pesar de la popularidad de esta tarea en diversos sistemas y aplicaciones comerciales.

En este capítulo nos centramos en ella, introduciendo técnicas y algoritmos que serán protagonistas en los restantes capítulos de esta tesis. En particular, estudiaremos las ventajas que los algoritmos kNN presentan en la tarea de recomendación. Estos algoritmos son muy populares gracias a aspectos como su sencillez, comportamiento intuitivo, o sus buenos resultados en multitud de escenarios diferentes, y, como veremos, son un candidato ideal para la tarea de recomendación.

En la Sección 5.1 comenzaremos con un repaso a los principales trabajos que han abordado esta tarea en los últimos años, y en la Sección 5.2 presentaremos cómo los algoritmos kNN pueden ser usados efectivamente en la recomendación. Posteriormente, en la Sección 5.3 describiremos un nuevo algoritmo kNN especialmente diseñado para esta tarea. Se presentará la metodología de evaluación (Sección 5.4), y finalmente estudiaremos los resultados obtenidos por nuestro algoritmo.

5.1. Introducción

Como hemos visto, la mayor parte de trabajos en el ámbito del filtrado colaborativo se han centrado en la tarea de predicción, es decir, en predecir la utilidad que un cierto producto puede tener para el usuario. El resultado de los esfuerzos dedicados a dicha tarea ha sido un gran avance en la calidad de los algoritmos en términos de precisión en la predicción, hasta tal punto que, tal y como hemos estudiado en el capítulo anterior, son aspectos como la variabilidad de los usuarios o la falta de información los que dificultan conseguir mejoras significativas respecto a las ya conseguidas.

Por el contrario, la tarea de recomendación, a pesar de estar presente en muchos sistemas comerciales, no ha recibido el mismo grado de atención por parte de la comunidad científica, y, en consecuencia, los avances en ella han sido de menor calado. Hace ya unos años, McLaughlin y Herlocker [2004] comprobaron como muchos de los algoritmos existentes ofrecían pobres resultados en recomendación, e incluso habían propuesto un nuevo algoritmo, *Belief Distribution*, diseñado con el objetivo de mejorar la experiencia del usuario. Más recientemente, Cremonesi et al. [2010] estudiaron la precisión en la recomendación de las técnicas actuales, diseñadas para la tarea de predicción. Sus experimentos demostraron que la mayor parte de algoritmos no se comportaban tan bien como cabría esperar, dejando claro que ambas tareas tienen grandes diferencias entre sí y que algoritmos con buenos resultados en predicción pueden no ser adecuados en la tarea de recomendación.

A pesar de ello, lo cierto es que los trabajos dedicados a recomendación han sido bastante escasos, y se han centrado en sistemas basados en puntuaciones unarias, principalmente de naturaleza implícita. Entre los dominios que han despertado mayor interés cabe destacar el comercio electrónico [Huang et al., 2007], donde las compras de productos pueden interpretarse como interés por parte del usuario, aunque también han sido objeto de estudio otros dominios como la recomendación de noticias [Das et al., 2007], la sugerencias de consultas a partir de las búsquedas anteriores [Baraglia et al., 2009], o la personalización de búsquedas en la web [Smyth, 2007].

Originalmente, los dominios con preferencias implícitas se habían abordado como un caso particular de los sistemas basados en puntuaciones, favoreciendo por tanto la aplicación de técnicas orientadas a la predicción, como por ejemplo algoritmos basados en productos [Karypis, 2001]. La asociación entre predicción con puntuaciones, y recomendación con preferencias implícitas o unarias, llevó incluso al extremo de descartar la información adicional aportada por las puntuaciones cuando el sistema se usaba en la recomendación [Deshpande y Karypis, 2004].

Hoy en día, sin embargo, los sistemas con preferencias unarias son objeto de estudio de una disciplina específica dentro de los sistemas de recomendación: la llamada *One Class Collaborative Filtering* [Pan et al., 2008]. Normalmente este problema se aborda usando dos estrategias: considerar como preferencia negativa la ausencia de información, es decir, el hecho de que un usuario no haya realizado una cierta acción sobre un producto; o simplemente tratar como preferencia desconocida esa ausencia de interacción por parte del usuario. Los algoritmos usados son variados e incluyen *weighted low-rank approximation* [Pan et al., 2008], *probabilistic latent semantic indexing* o factorización de la matriz [Pan y Scholz, 2009].

Precisamente, esa separación entre sistemas centrados en preferencias implícitas y sistemas basados en puntuaciones se ha tenido en cuenta en esta tesis, habiéndonos centrado en el segundo caso: el uso de sistemas basados en puntuaciones en la tarea de recomendación. En particular, hemos estudiado el uso de algoritmos basados en vecinos o kNN. En este capítulo nos centraremos en los resultados desde el punto de vista de la precisión de las recomendaciones. Presentaremos las características de este tipo de algoritmos en la tarea de recomendación, y propondremos nuevas técnicas para mejorar los resultados. En los próximos capítulos abordaremos otros aspectos, como el problema de *cold-start* o la eficiencia y escalabilidad de estos métodos.

A pesar de que los algoritmos kNN son una de las técnicas de recomendación más antiguas, siguen siendo hoy en día una de las más populares. De hecho, este tipo de algoritmos todavía presentan importantes ventajas sobre las técnicas desarrolladas en los últimos años, mucho más complejas y elaboradas [Desrosiers y Karypis, 2011]:

- **Simplicidad**, lo que convierte a estos algoritmos en una técnica fácil de entender y por tanto de implementar. De hecho, en posteriores capítulos veremos cómo implementarla de forma eficiente a partir de técnicas ampliamente utilizadas en el mundo de la RI.

- **Justificación de resultados**. El funcionamiento de estos algoritmos es muy intuitivo y permite derivar fácilmente el motivo de por qué un producto ha sido recomendado, lo que es muy útil de cara a justificar al usuario las razones de una recomendación concreta. Afirmaciones como "le hemos recomendado esta película porque le ha gustado esta otra" son muy útiles pues permiten al usuario hacerse una idea del tipo de producto recomendado, y decidir si es realmente una buena opción en ese momento. Las técnicas basadas en modelos más complejos no siempre permiten dar esta información.

- **Estabilidad y eficiencia**. A pesar de que en sistemas comerciales se están

añadiendo continuamente nuevos productos y usuarios, estos algoritmos no necesitan volver a entrenar complejos modelos, sino simplemente actualizar ciertas estructuras como el vecindario, lo que repercute también en una mayor eficiencia.

Además, son una técnica que ha demostrado su buen comportamiento en numerosos dominios y entornos diferentes, lo que la convierte en un candidato ideal en muchas aplicaciones comerciales. De hecho, tal y como veremos, supera a otro tipo de técnicas en la tarea de recomendación.

5.2. Algoritmos kNN para recomendación

Tradicionalmente, los algoritmos kNN, al igual que la mayor parte de algoritmos de filtrado colaborativo, se habían desarrollado con la tarea de predicción en mente (ver Sección 3.4.1). Su uso en la tarea de recomendación, sin embargo, requiere de pequeños cambios que permitan aprovechar sus numerosas ventajas.

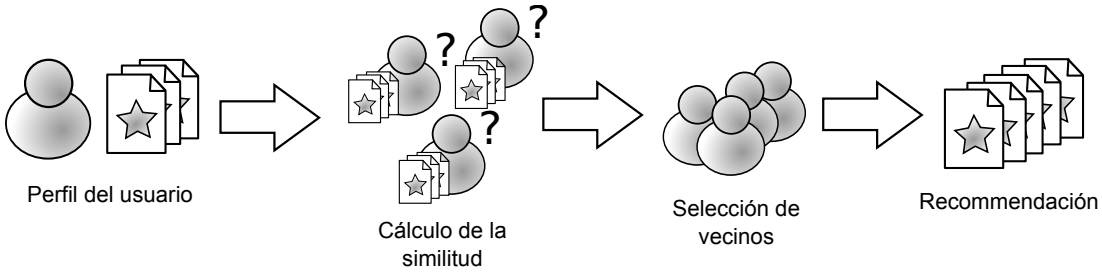


Figura 5.1: Funcionamiento de los algoritmos kNN.

El funcionamiento global de un algoritmo kNN es bastante sencillo. Tal como se refleja en la Figura 5.1, comprende tres fases principales: cálculo de similitudes entre usuarios, selección del vecindario y finalmente recomendación.

La principal diferencia entre predicción y recomendación está, precisamente, en esta última fase. En la predicción, en lugar de devolver una lista de productos, el algoritmo lo que hace es predecir la puntuación para un producto dado, previamente seleccionado por el usuario, a partir de las puntuaciones de los vecinos. Este funcionamiento puede extenderse de forma sencilla a la tarea de recomendación: simplemente se calcularía la puntuación para todos los productos, y se recomendarían aquellos con mayor puntuación. Esta aproximación, sin embargo, tiene varias limitaciones:

1. **Elevado coste computacional**, pues para hacer una recomendación a un único usuario se debe predecir su puntuación sobre todos los productos. Esta solución, por tanto, no es viable en aplicaciones con un gran número de productos.

2. **Muy sensible a pequeños errores en la predicción**. Como la lista de productos recomendados se calcula a partir de la predicción de la puntuación, un pequeño error en la predicción puede suponer una gran diferencia en el orden que un producto ocupa en la lista. Los sistema actuales trabajan con miles (o incluso millones) de productos, pero los usuarios sólo pueden distinguir entre unas pocas puntuaciones. En una escala de puntuaciones del 1 al 5, un error de una décima no va a ser perceptible por el usuario en términos de predicción, pero en la recomendación puede significar que la posición de un producto se vea alterada significativamente. Dado que el usuario apenas va a consultar los productos recomendados en las primeras posiciones (generalmente, 5 o 10 productos como mucho), estos pequeños errores pueden tener un gran impacto desde el punto de vista del usuario.

La solución a estos problemas pasa por evitar el paso adicional de predicción, y en su lugar proceder directamente a elegir los productos a recomendar. Para ello, en lugar de predecir la puntuación para cada producto, nos limitamos a calcular un peso ω_i para cada producto puntuado por los vecinos. La recomendación se haría escogiendo los N productos con mayor peso. Esta técnica tiene dos diferencias fundamentales con la técnica tradicional basada en la predicción.

En primer lugar, si bien intuitivamente peso y predicción están positivamente correlacionados, las similitudes entre ambos se acaban ahí. No es necesario, por ejemplo, que el peso esté acotado en una cierta escala, ni que dicha escala sea similar a la usada en las puntuaciones. De hecho, el valor de un peso no tiene ningún significado por sí mismo, y sólo lo adquiere en el momento en que es comparado con otro. Esto permite desarrollar sistemas de recomendación menos sensibles a los errores de predicción mencionados anteriormente.

Por otra parte, sólo es necesario calcular los pesos para los productos que han sido valorados al menos por un vecino, en lugar de las predicciones, que tradicionalmente se calculan para todos los productos. Esto tiene un gran impacto en la eficiencia del algoritmo, en parte porque el número de cálculos a realizar es mucho menor, pero también porque simplifica la selección de vecinos. Tal como habíamos mencionado en la Sección 3.4.1, en la predicción suele ser necesario seleccionar un vecindario diferente para cada producto, precisamente para asegurarnos de que los vecinos lo han puntuado. Sin embargo, en la recomendación esto ya no es necesario, lo que simplifica el proceso.

En cuanto a los métodos empleados, tanto para el cálculo de similitudes entre usuarios como para el cálculo de pesos, se podrían utilizar métodos tradicionales. Por ejemplo, la correlación de Pearson o la distancia coseno como medida de similitud (ecuacio-

nes 3.2 y 3.4 respectivamente), y la media ponderada de puntuaciones (Ecuación 5.1) como método para el cálculo de pesos.

$$w_i = \frac{\sum_{u \in \mathcal{N}(a)} s_{au} r_{ui}}{\sum_{u \in \mathcal{N}(a)} s_{au}} \tag{5.1}$$

Otra posibilidad es el desarrollo de nuevos métodos más adecuados a la tarea de recomendación. En particular, en la próxima sección se presenta un nuevo algoritmo, *Good Items kNN*, en el que las medidas de similitud y cálculo de pesos se redefinen para considerar únicamente productos bien puntuados. Como veremos, esto permite importantes mejoras en la calidad de las recomendaciones.

5.3. *Good Items* k*NN*

Al igual que los algoritmos kNN tradicionales presentados en la sección anterior, el algoritmo *Good Items kNN* que describiremos en esta sección comprende las tres fases principales reflejadas en la Figura 5.1. La principal novedad de nuestra aproximación está en las técnicas usadas para el cálculo de similitud entre usuarios y la selección de productos a recomendar a partir de los vecinos, que han sido especialmente diseñadas para la tarea de recomendación.

5.3.1. Selección del vecindario

En un algoritmo kNN, la selección del vecindario viene determinada por el valor del parámetro k y por la medida de similitud entre usuarios empleada, siendo esta última un factor de gran impacto en la selección de un vecindario adecuado y, en consecuencia, de una buena recomendación.

Las medidas de similitud tradicionales, al ser diseñadas para la tarea de predicción, se basan en encontrar usuarios con un patrón de puntuaciones similar. De hecho, todas las puntuaciones de los usuarios son tratadas por igual, con independencia de la puntuación realmente otorgada. Esto puede llevar a que dos usuarios sean considerados similares a pesar de coincidir únicamente en productos mal valorados. Si bien tal comportamiento puede ser razonable en la tarea de predicción, pues al fin y al cabo su objetivo es predecir la puntuación de un producto sea esta buena o mala, en la tarea de recomendación supone una limitación importante que se une a los problemas ya comentados en la Sección 4.5.2. Y es que en la tarea de recomendación el objetivo del sistema es el recomendar buenos productos, y que dos usuarios coincidan en señalar qué productos no son adecuados, no quiere decir que vayan a tener los mismos gustos.

Nuestra aproximación, por tanto, es que a la hora de calcular la similitud sólo se tengan en cuenta aquellos productos que ambos usuarios han considerado relevantes. En particular, hemos descartado aquellos productos puntuados por debajo de un cierto umbral θ. Parece razonable asumir que si el algoritmo debe recomendar buenos productos, pues sólo se deberían tener en cuenta las coincidencias en estos. Esta idea, que será por tanto la primera novedad introducida por nuestro algoritmo, queda reflejada en una nueva medida de similitud, *best items similarity*, definida en la siguiente ecuación:

$$s(a,u) = \frac{\sum_{i \in \mathcal{I}^{(a,u;\theta)}} r_{ai} r_{ui}}{\left|\mathcal{I}^{(a,u;\theta)}\right|} \tag{5.2}$$

donde el conjunto $\mathcal{I}^{(a,u;\theta)}$ está compuesto por aquellos productos que ambos usuarios han valorado por encima del umbral θ, es decir, $\mathcal{I}^{(a,u;\theta)} = \left\{i \in \mathcal{I}^{(a,u)} | r_{ai} \geq \theta, r_{ui} \geq \theta\right\}$. Tras seleccionar los productos que pertenecen a dicho conjunto, la similitud se calcula como la suma del producto de las puntuaciones que ambos usuarios han otorgado a cada producto. Cuanto mayor sea la puntuación otorgada, por tanto, mayor será la similitud. Por otra parte, el resultado se pondera dividiéndolo por el número de productos considerados, para evitar que usuarios con un gran número de puntuaciones tengan más posibilidades de formar parte del vecindario, lo cual no es deseable.

Esta nueva aproximación favorecerá a usuarios a los que les han gustado los mismos productos que al usuario actual, en lugar de a usuarios con un patrón de puntuaciones similar. Es de esperar que este tipo de usuarios sean mejores vecinos en el caso de la recomendación, donde el objetivo es recomendar buenos productos.

5.3.2. Selección de los productos a recomendar

Una vez calculado el vecindario, la lista de productos a recomendar se obtiene a partir de los productos puntuados por los vecinos, tal y como hemos comentado en la sección anterior. Sin embargo, hemos introducido dos variaciones diseñadas específicamente para la tarea de recomendación:

1. **Considerar sólo productos bien valorados**. Con las técnicas tradicionales, todos los productos puntuados por los vecinos van a ser posibles candidatos para formar parte de la lista de recomendación. De hecho, incluso aquellos productos con malas puntuaciones pueden ser recomendados. Esto es razonable, ya con medidas de similitud tradicionales como Pearson puede haber vecinos negativamente correlacionados con el usuario actual, y por tanto un producto mal puntuado por dicho vecino puede llegar a ser una buena recomendación. Sin embargo, nuestra medida de similitud se centra en escoger vecinos a los que les gustan los mismos

productos que al usuario actual, y por tanto dichas recomendaciones no son una buena elección. Por tanto, hemos adaptado la técnica de selección de productos a nuestra medida de similitud, descartando aquellos productos que han recibido una puntuación inferior a un cierto umbral θ', no necesariamente igual al umbral θ utilizado en la medida de similitud.

2. **Dar más peso a productos puntuados por muchos vecinos**. Si usamos una técnica tradicional para el cálculo de pesos, como la presentada en la sección anterior, el peso asociado a cada producto se suele normalizar teniendo en cuenta la similitud de cada vecino que había puntuado dicho producto. El objetivo de dicha técnica, que también se utiliza habitualmente en la tarea de predicción, es evitar dar demasiado peso a un producto sólo por el hecho de que haya sido puntuado por muchos vecinos, lo que, en principio, parece una medida más que razonable.

   Sin embargo, en la práctica esto implica que al final todos los productos valorados por los vecinos reciben una importancia similar, independientemente de su relación real con los gustos del usuario. Este problema se puede explicar de forma sencilla utilizando un ejemplo. Supongamos un usuario al que le gustan las películas de acción. Es probable, por tanto, que otros usuarios a los que también les gusta este tipo de películas acaben formando parte del vecindario. Hasta este punto, todo parece correcto: hemos seleccionado un vecindario formado por usuarios que comparten la afición del usuario actual por las películas de acción. Sin embargo, a la hora de seleccionar los productos a recomendar, se tendrán en cuenta todos los productos puntuados por los vecinos (o al menos los bien valorados), con independencia de su relación con los productos que realmente le gustan al usuario actual. Por ejemplo, si a uno de los vecinos, aparte de películas de acción, le gustan películas de terror, dichos productos se acabarían recomendando a pesar de no guardar relación con aquellos productos que realmente motivaron que este usuario formase parte del vecindario.

   Para evitar este problema, o al menos minimizarlo, hemos optado por eliminar la normalización de pesos, para así favorecer aquellos productos puntuados por muchos vecinos. Es de esperar que esto incremente las posibilidades de que los productos recomendados sean precisamente aquellos relacionados con los productos que han hecho que el usuario formase parte del vecindario. Consideremos nuevamente el ejemplo anterior: como al usuario le gustaban las películas de acción, a cada vecino le gustarán varias películas de acción, y quizás otro tipo de películas.

Dado que las películas de acción son precisamente las que todos los vecinos tienen en común, con nuestra técnica lo más probable es que acaben recibiendo un mayor peso, y por tanto la recomendación estará formada principalmente por este tipo de películas, que son precisamente las que le gustan al usuario.

Ambas variaciones quedan reflejadas en nuestra fórmula para el cálculo de pesos, que se muestra en la siguiente ecuación:

$$\omega_i = \sum_{u \in N(a), r_{ui} > \theta'} r_{ui} s(a, u) \tag{5.3}$$

Es decir, cada vecino contribuye al peso de un producto una cantidad proporcional a la puntuación otorgada a dicho producto y a su similitud con el usuario actual. Pero sólo contribuye en caso de que su puntuación no sea inferior al umbral θ', pues en caso contrario no es tenido en cuenta.

5.4. Evaluación de la recomendación: métricas y metodología

5.4.1. Métricas

Tradicionalmente, tal y como se describe en la Sección 2.6, la tarea de recomendación se ha evaluado principalmente mediante el uso de métricas de precisión en la clasificación y ordenación. Las primeras se centran en evaluar si el sistema es capaz de distinguir entre aquellos productos adecuados para el usuario y aquellos que no lo son, siendo los primeros, obviamente, los que serían finalmente recomendados. Las métricas de precisión en la ordenación, por el contrario, son más estrictas y evalúan no sólo que el sistema sea capaz de distinguir entre productos adecuados y no adecuados, sino que también sea capaz de distinguir cuál de dos productos es la mejor opción. Es decir, que sea capaz de ordenar correctamente los distintos productos según las preferencias del usuario.

En la mayoría de aplicaciones de los sistemas de recomendación, sin embargo, el conseguir una correcta ordenación es un objetivo demasiado ambicioso y, en general, no particularmente útil. En primer lugar, porque el usuario normalmente no necesita recibir una lista ordenada con los mejores productos, sino simplemente un conjunto de productos adecuados. En segundo lugar, por la propia dificultad a la hora de seleccionar qué producto o productos son los mejores, tanto por las limitaciones de las propias técnicas de recomendación como sobre todo por la variabilidad del usuario. Especialmente, en multitud de dominios el usuario no necesita (ni quiere) el mejor producto.

	Recomendados	No recomendados
Relevante	Verdadero Positivo (VP)	Falso Negativo (FN)
No relevante	Falso Positivo (FP)	Verdadero Negativo (VN)

Tabla 5.1: Clasificación del resultado de la recomendación según la relevancia del producto.

Pensemos por ejemplo en la recomendación de películas, donde el producto preferido para el usuario en un momento dado no tiene por que ser aquel al que le daría la mayor puntuación, pues podría preferir una película no tan buena pero más adecuada a su estado de ánimo, personas con las que la va a ver, etc. Por tanto, en general preferiremos la evaluación utilizando métricas de precisión en la clasificación, si bien esta no siempre ha sido la opción escogida en la literatura.

Generalmente, en la evaluación se seleccionan un conjunto de usuarios, y se obtiene una recomendación para cada uno de ellos. Luego se comparan los productos recomendados con aquellos realmente relevantes. Tal y como se muestra en la Tabla 5.1, se pueden dar cuatro casos, según un producto sea o no relevante para el usuario y haya sido o no recomendado:

A partir de estos datos se pueden definir varias métricas, entre las que destacan la *precisión* (P) y la exhaustividad o *recall* (R), definidas según las ecuaciones 5.4 y 5.5, y que miden el ratio entre los productos relevantes que han sido recomendados, y, respectivamente, el total de productos recomendados o el total de productos relevantes.

$$P = \frac{VP}{VP + FP} \tag{5.4}$$

$$R = \frac{VP}{VP + FN} \tag{5.5}$$

La precisión proporciona una medida de la calidad de la lista de productos recomendados, es decir, la probabilidad de que realmente un producto recomendado sea una buena opción. En cambio, el *recall* evalúa la probabilidad de que un producto relevante sea recomendado al usuario. Ambas métricas están inversamente relacionadas, y en general al aumentar el tamaño de la lista de productos recomendados también lo hace el *recall*, al tiempo que empeora la precisión. Por tanto, se suele tener en cuenta la relación entre ambas a la hora de evaluar la calidad de un algoritmo en cuanto a la clasificación.

Una alternativa es el uso de la métrica $F1$ que, tal y como se muestra en la ecuación 5.6, tiene en cuenta ambas medidas.

$$F1 = \frac{2PR}{P+R} \tag{5.6}$$

En el caso de los sistemas de recomendación, el número de productos mostrados al usuario suele ser un número fijo no demasiado elevado (por ejemplo, 5), por lo que el valor de precisión y *recall* en los primeros N productos ($P@N$ y $R@N$) es una buena medida del comportamiento del algoritmo.

En otros casos puede ser interesante estudiar la evolución de ambas, para lo que se puede utilizar la curva de la precisión frente al *recall* (o curva P-R) [van Rijsbergen, 1979], que mide la precisión del algoritmo en diferentes niveles de *recall*. Otra alternativa es el uso de la precisión media o MAP (*Mean Average Precision*) [Manning et al., 2008], que aproxima el área bajo la curva P-R.

Las curvas ROC (*Receiver Operating Characteristic*), que representan la relación entre productos relevantes y no relevantes que han sido recomendados también se utiliza en la evaluación de algoritmos de recomendación [Herlocker et al., 2004]. Mientras la curvas P-R enfatizan la proporción de productos relevantes recomendados, las curvas ROC lo hacen con los productos no relevantes que han sido recomendados. Dependerá de la aplicación cuál de ellas es preferible mejorar.

5.4.2. Metodologías de evaluación tradicionales y limitaciones

En la práctica, la evaluación de la recomendación supone un desafío importante, consistente en determinar qué productos son realmente relevantes [Herlocker et al., 2004]. En una evaluación *online* se puede preguntar al usuario, pero si la evaluación se realiza *offline*, el problema es más complejo. Si disponemos de la puntuación del usuario, se podrían considerar relevantes aquellos productos cuya puntuación supere un cierto umbral, y como no relevantes aquellos que no lo hagan. El problema es que no conoceremos la puntuación real para la mayoría de productos. Por tanto, es más que probable que la recomendación hecha por el sistema contenga muchos productos que no sabremos si son o no relevantes. En dichos casos, se pueden considerar diferentes soluciones.

La primera sería ignorar su efecto, bien eliminándolos directamente de la lista de recomendación [Cacheda et al., 2011a], o bien otorgándoles una puntuación neutral [Breese et al., 1998]. Otra alternativa es considerarlos productos no relevantes [McLaughlin y Herlocker, 2004], solución similar a la adoptada generalmente en otros contextos como el de RI [Manning et al., 2008].

Normalmente esto es lo que se hace en la evaluación de sistemas con puntuaciones unarias. En dichos casos, los productos se consideran relevantes en caso de haber si-

do puntuados, y no relevantes en el caso contrario [Huang et al., 2007; Karypis, 2001; Sarwar et al., 2000a]. Cuando las puntuaciones pueden reflejar diferentes grados de relevancia, una propuesta alternativa es forzar al algoritmo a seleccionar la recomendación entre varios productos, de los cuales sólo se conoce la puntuación de uno de ellos. Si la recomendación del sistema incluye dicho producto, se considera un acierto [Cremonesi et al., 2010].

Como forma de evitar el problema de qué hacer con los productos cuya puntuación se desconoce, Redpath et al. [2010] han propuesto una métrica alternativa, *approval rate*, definida como el ratio entre los productos recomendados que realmente son relevantes y los productos recomendados cuya puntuación se conoce (sea o no positiva). En la práctica, es similar a ignorar aquellos productos cuya puntuación real no es conocida.

Realmente, en caso de optar por una evaluación *offline* este problema siempre va a estar presente de una u otra manera. Como alternativa, se puede proceder a una evaluación con usuarios [Shani y Gunawardana, 2011], en la que personas reales interactúan con el sistema y, por tanto, pueden valorar si los productos recomendados son buenos realmente, evitando así el problema anterior. Sin embargo, este tipo de estudios son mucho más complicados de llevar a cabo, en gran parte debido a que requieren más tiempo y presentan un coste elevado. Además, los experimentos generalmente no son repetibles, lo que dificulta la comparación entre distintas técnicas.

5.4.3. Evaluación de la clasificación: una metodología alternativa

Al contrario de lo que sucede en el ámbito de los sistemas de recomendación, en la recuperación de información sí existe un método de evaluación generalmente aceptado, conocido como el paradigma de Cranfield [Voorhees, 2002]. Consiste en utilizar una colección de documentos, un determinado número de necesidades de información o consultas, y un conjunto de documentos que son relevantes para cada una de ellas. Durante la evaluación, estos se comparan con la respuesta real del motor de búsqueda, consulta a consulta, utilizando la métrica o métricas deseadas (por ejemplo, la precisión del sistema).

La metodología de evaluación de la recomendación que proponemos en esta tesis es precisamente la adaptación de esa metodología a la evaluación de sistemas de recomendación. En primer lugar, el conjunto de consultas se sustituye por un conjunto de usuarios de evaluación. Para cada usuario, el sistema generará una lista de productos recomendados, que posteriormente se compara con los juicios de relevancia de los que se dispone, es decir, las puntuaciones. La idea es dividir el perfil de los usuarios de evaluación en dos partes: una destinada a la generación de la recomendación, y por tanto

conocida por el algoritmo, y otra destinada a la evaluación y que por tanto se oculta. Si la puntuación supera un cierto umbral, se considera que el producto es relevante para el usuario.

El parámetro ε determina el porcentaje de puntuaciones que formarán parte del conjunto de evaluación. Al aumentar su valor, dispondremos de más puntuaciones con las que comparar el resultado de la recomendación, y por tanto menor será el impacto de la limitación en la evaluación *offline* mencionado anteriormente. Sin embargo, también se reducirá la información disponible acerca de cada usuario, lo que repercute en la calidad de la recomendación, al igual que sucedía en la tarea de predicción, tal como hemos discutido en el capítulo anterior.

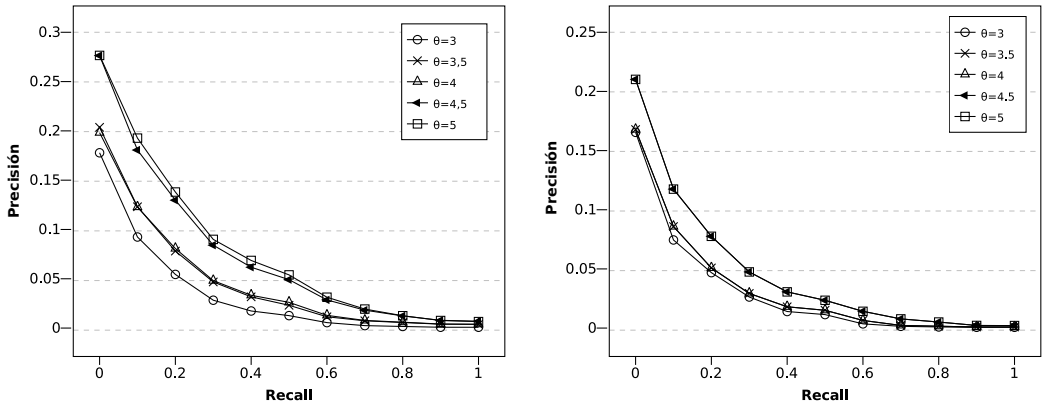
De hecho, dada la importancia de estudiar el comportamiento de los algoritmos en diferentes condiciones, hemos introducido un segundo parámetro en la evaluación, τ, que controla el porcentaje de puntuaciones de cada usuario que realmente van a ser usadas.

En resumen, el primer paso es la selección aleatoria de los usuarios de evaluación. Para cada uno de ello, reservamos un porcentaje ε de sus puntuaciones de cara a la evaluación posterior. Posteriormente, de cada usuario se escoge un porcentaje τ de puntuaciones, y el resto son descartadas. Con esa información, el algoritmo recomendará una lista de productos a cada usuario de evaluación, lista que se comparará con las puntuaciones reservadas, calculando el resultado para las métricas de evaluación deseadas.

5.5. Experimentos y resultados

Para evaluar el comportamiento de nuestro algoritmo, hemos seguido la metodología de evaluación que se acaba de describir. En concreto, hemos utilizado 1.000 usuarios de evaluación, y distintos valores de los parámetros ε y τ como se verá indicado a lo largo de la sección. En cuanto a los conjunto de datos, hemos usado Netflix y Movielens 10M.

Para el análisis estadístico de los resultados hemos empleado el análisis de la varianza (ANOVA). A pesar de que las precondiciones necesarias para este test no se cumplen estrictamente (entre otras cosas, las métricas dan resultados acotados y por tanto los errores no siguen realmente una distribución normal), en la práctica estas pequeñas violaciones de las hipótesis del modelo tienen un impacto limitado en los resultados de la prueba, y de hecho el ANOVA se utiliza habitualmente para validar estadísticamente experimentos en RI [Hull, 1993]. Todos los resultados mencionados a continuación son estadísticamente significativos con un nivel de significación $\alpha = 0{,}05$, excepto en los



(a) Movielens 10M **(b)** Netflix

Figura 5.2: Curvas P-R del algoritmo GIkNN para diferentes valores de umbral θ, calculadas con $k = 25$, $\tau = 80\,\%$ y $\varepsilon = 20\,\%$.

casos específicamente mencionados.

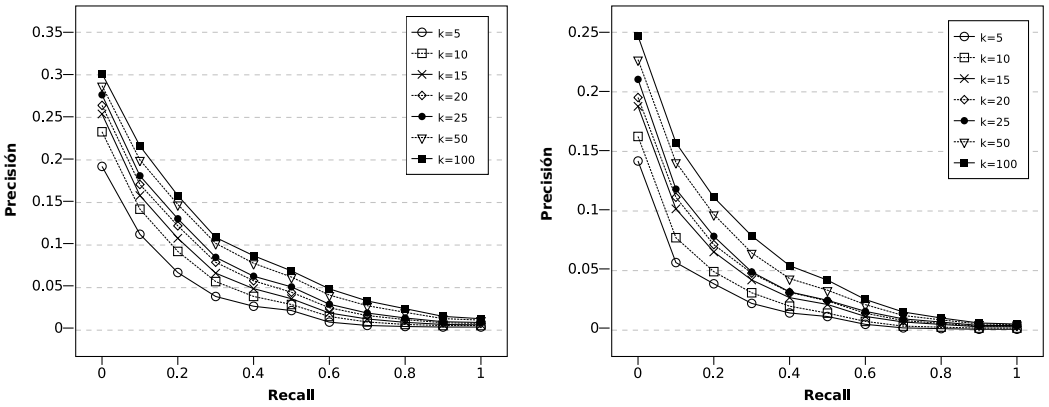
5.5.1. Impacto de los parámetros del algoritmo

En primer lugar, hemos estudiado el comportamiento de nuestro algoritmo en función de los parámetros θ y k, que determinan, respectivamente, el umbral de relevancia para que un producto sea considerado en el cálculo de similitud entre vecinos, y el número máximo de vecinos que se van a usar en la fase de recomendación. El tercer parámetro del algoritmo, θ' ha sido fijado a 4,0 (en una escala de puntuaciones del 1 al 5), para hacerlo coincidir con el umbral de relevancia empleado en la fase de evaluación, y el estudio de comportamiento en función de dicho parámetro no presenta mayor interés. Nos hemos centrado, por tanto, en estudiar el impacto de los parámetros θ y k en la calidad de las recomendaciones.

Tal como se puede deducir a partir de las curvas P-R mostradas en las Figuras 5.2a y 5.2b, en ambos conjuntos de datos los mejores resultados se obtienen cuando θ se fija a un valor elevado, en concreto, con $\theta = 4{,}5$ y $\theta = 5{,}0$[1]. Estos resultados permiten confirmar la hipótesis que habíamos considerado inicialmente: en la tarea de recomendación, los mejores vecinos son aquellos a los que les gustan los mismos productos que al usuario actual, con independencia de que coincidan o no en malos productos.

Por otra parte, si estudiamos la precisión y el *recall* por separado, vemos que este

[1]Obsérvese que los resultados en Netflix son iguales en ambos casos, debido a que dicho conjunto de datos sólo considera puntuaciones consistentes en números enteros.

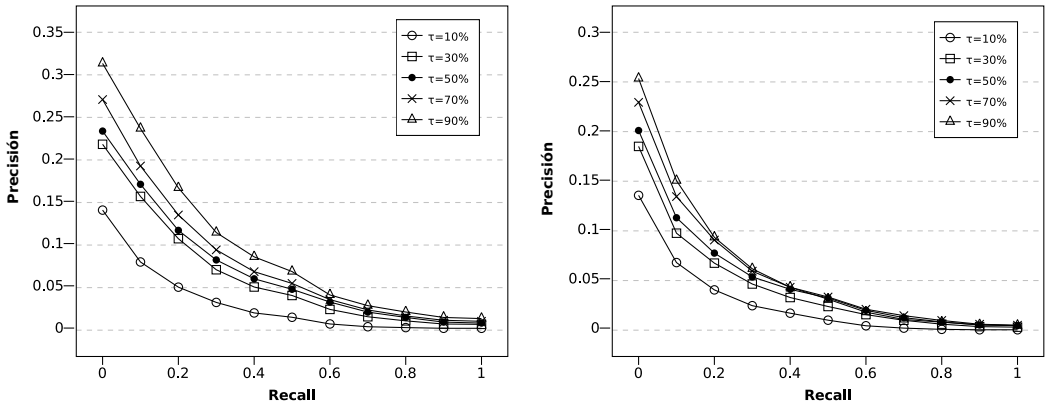


(a) Movielens 10M **(b)** Netflix

Figura 5.3: Curvas P-R del algoritmo GIkNN con diferente número de vecinos, k, calculadas con $\theta = 4.5$, $\tau = 80\,\%$ y $\varepsilon = 20\,\%$.

parámetro influye sobre todo en la precisión, y con porcentajes de entrenamiento elevados. En Movielens 10M, y con $\tau = 90\,\%$, la mejora en $P@5$ con $\theta = 5$ respecto a $\theta = 4$ supera el 70 %, y en $P@10$ llega al 80 %. En Netflix, la mejora supera el 80 % en $P@5$, y es ligeramente inferior al 70 % en $P@10$. En ambos casos, a medida que se reduce el porcentaje de entrenamiento también lo hace el porcentaje de mejora, pero aún así la precisión al usar valores altos de θ es mejor en prácticamente todos los experimentos realizados. La mejora en *recall*, por otra parte, es menor, y sólo es estadísticamente significativa con porcentajes de entrenamiento elevados. De hecho, con $\tau < 50\,\%$ el *recall* se ve empeorado ligeramente, lo cual se explica por el hecho de que al descartar productos con peor valoración es más difícil encontrar usuarios parecidos. En cualquier caso, las mejoras en precisión siguen siendo un buen motivo para tender al uso de valores de θ elevados, como norma general.

Por otra parte, el número de vecinos, k, sí que tiene un importante impacto tanto en precisión como en *recall*, independientemente del porcentaje destinado al conjunto de entrenamiento. Tal y como se observa en las Figuras 5.3a y 5.3b, los resultados son mejores al aumentar el número de vecinos. Sin embargo, las diferencias entre niveles próximos no siempre son estadísticamente significativas. La mejora en *recall* es esperada, ya que el algoritmo utiliza las puntuaciones de los vecinos para la recomendación de productos, por lo que cuanto mayor sea k mayor será el número de productos entre los que escoger la recomendación. Establecer k a un valor relativamente elevado es deseable porque además repercute en una mayor diversidad y *coverage*, al mismo tiempo que en una mayor precisión.



(a) Movielens 10M **(b)** Netflix

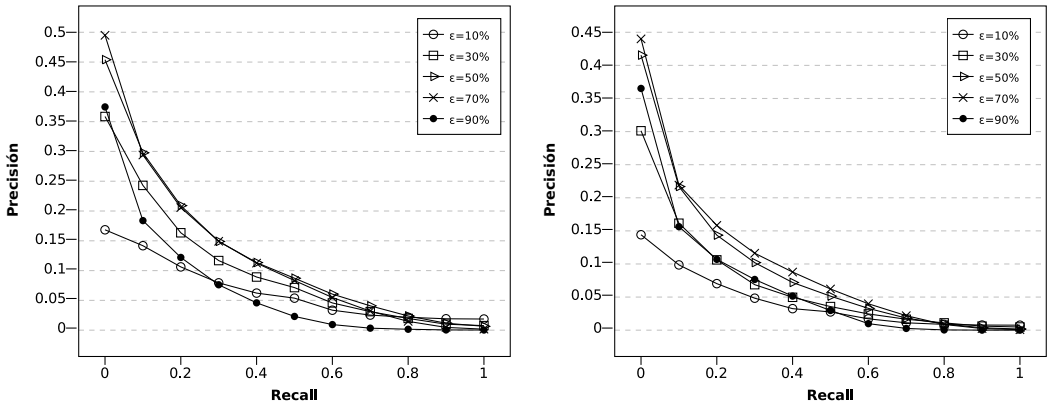
Figura 5.4: Curvas P-R del algoritmo GIkNN según el porcentaje de puntuaciones destinado al conjunto de entrenamiento, τ, calculadas con $\theta = 4.5$, $k = 25$ y $\varepsilon = 20\,\%$.

La mejora en precisión puede parecer más sorprendente, pero realmente es lo que habíamos esperado y es consistente con los resultados obtenidos en la tarea de predicción (ver Sección 4.5.1). A parte del menor impacto de un posible error en el cálculo de la similitud, que afecta a todas las técnicas, nuestro algoritmo funciona mejor con un mayor número de vecinos, precisamente gracias a que su mecanismo de selección de productos a recomendar da prioridad a productos bien puntuados por muchos vecinos. Tener por tanto muchos vecinos ayuda a evitar que sean recomendados productos que sólo han gustado a unos pocos vecinos y que posiblemente no tengan relación con los gustos del usuario actual, tal y como hemos discutido en la Sección 5.3.2.

5.5.2. Resultados según el conjunto de entrenamiento

Además del estudio de los parámetros del algoritmo, hemos analizado el impacto de los parámetros que controlan el porcentaje de puntuaciones que formarán parte de los conjuntos de entrenamiento y evaluación, τ y ε. Tal y como se muestra en las Figuras 5.4a y 5.4b, en lo relativo a τ los resultados son los esperados: cuanto mayor sea el porcentaje de puntuaciones en el conjunto de entrenamiento, mejores son los resultados. Esto se debe a que el algoritmo tiene más información disponible para seleccionar los productos a recomendar, y por tanto estos son más adecuados a los gustos del usuario.

En cuanto a ε, los resultados obtenidos requieren una explicación más detallada. En primer lugar, tal y como se observa en las Figuras 5.5a y 5.5b, con la excepción de $\varepsilon = 90\,\%$, las recomendaciones parecen mejorar al aumentar el porcentaje de puntuaciones en el conjunto de evaluación. Sin embargo, más que una mejora real esto se debe



(a) Movielens 10M **(b)** Netflix

Figura 5.5: Curvas P-R del algoritmo GIkNN según el porcentaje de puntuaciones destinado al conjunto de evaluación, ε, calculadas con $\theta = 4.5$, $k = 25$ y $\tau = 80\,\%$.

a las limitaciones de la metodología de evaluación empleada, en la que se considera no relevante cualquier producto que no forma parte del conjunto de evaluación. Por tanto, cuando ε es pequeño, hay pocas puntuaciones en el conjunto de evaluación, y las posibilidades de que un producto sea considerado no relevante aumentan. Esto no significa, sin embargo, que los resultados sean peores; es simplemente una limitación de la evaluación *offline*, tal y como hemos discutido en la Sección 5.4.

Por otra parte, cuando ε es muy grande ($\varepsilon = 90\,\%$), se produce otro fenómeno que causa que los resultados empeoren: la mayoría de puntuaciones formarán parte del conjunto de evaluación, reduciéndose por tanto la información disponible en el conjunto de entrenamiento. El algoritmo tendrá entonces menos información para generar buenas recomendaciones, y la precisión disminuye, esta vez no ya debido a las limitaciones de la metodología sino a un peor comportamiento del algoritmo. En particular, este es un caso específico del problema de *cold-start* que abordaremos en el siguiente capítulo.

5.5.3. *GIkNN* frente a otros algoritmos de recomendación

Finalmente, hemos comparado la precisión de nuestro algoritmo *GIkNN* con la de otros algoritmos populares en la actualidad. En concreto, con el algoritmo RSVD (ver Sección 3.4.6.2), el algoritmo basado en tendencias presentado en el capítulo anterior, y un algoritmo kNN basado en la similitud coseno (Sección 3.4.1) [1]. En todos los casos

[1]También hemos utilizado como alternativa el coeficiente de correlación de Pearson, más popular en el mundo de la recomendación, pero sus resultados fueron muy inferiores y no han sido considerados en la siguiente discusión.

los parámetros de los algoritmos han sido optimizados previamente usando validación cruzada.

Al igual que habíamos hecho en la evaluación de la predicción mostrada en el capítulo anterior, hemos analizado la evolución según el porcentaje de puntuaciones que forman parte del conjunto de entrenamiento, con el objetivo de comparar los distintos algoritmos en condiciones tanto de baja como de alta densidad de información.

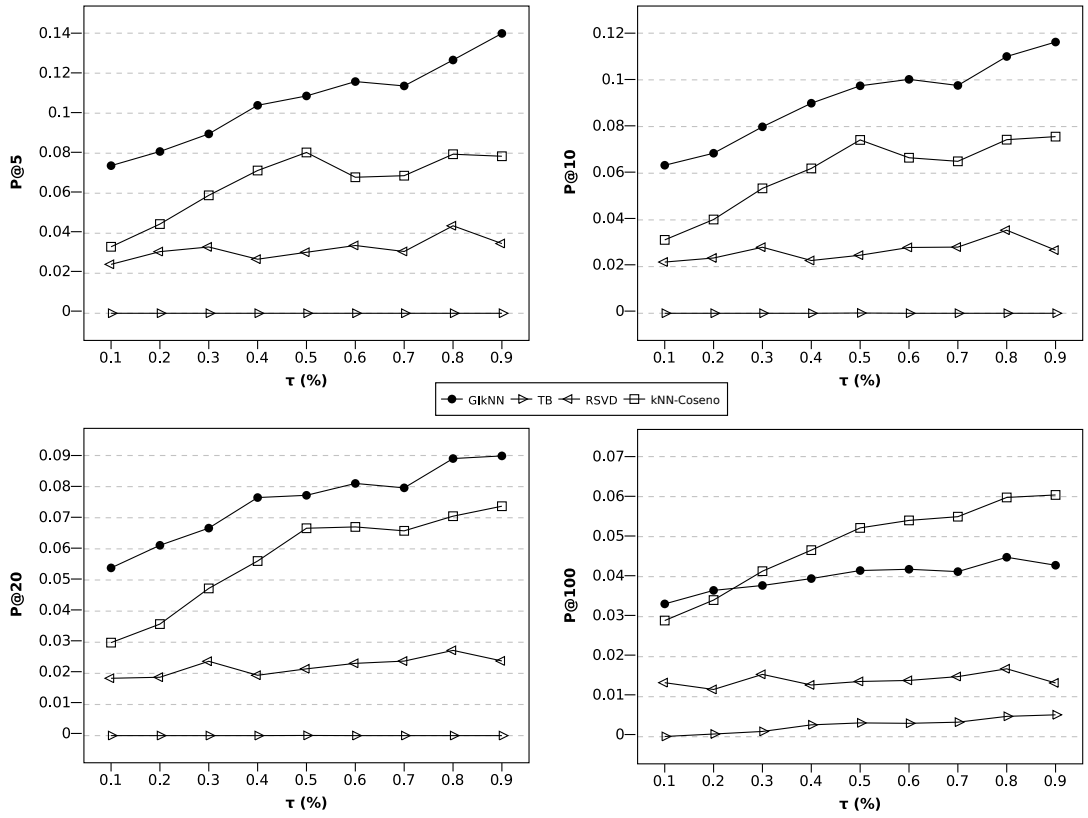


Figura 5.6: Evolución de $P@N$ según el tamaño del conjunto de entrenamiento, τ, en Movielens 10M.

En primer lugar, hemos estudiado la evolución de la precisión. Los resultados para Movielens 10M y Netflix se muestran, respectivamente, en las Figuras 5.6 y 5.7. Como se puede observar, hemos estudiado la precisión teniendo en cuenta la recomendación de distinto número de productos: 5, 10, 20, y 100. En general, como hemos discutido anteriormente, en un sistema de recomendación es más interesante el estudio de la precisión con listas de tamaño reducido. Sin embargo, también mostramos los resultados de $P@100$ con el objetivo de ver la tendencia del algoritmo a medida que aumenta el número de productos a recomendar.

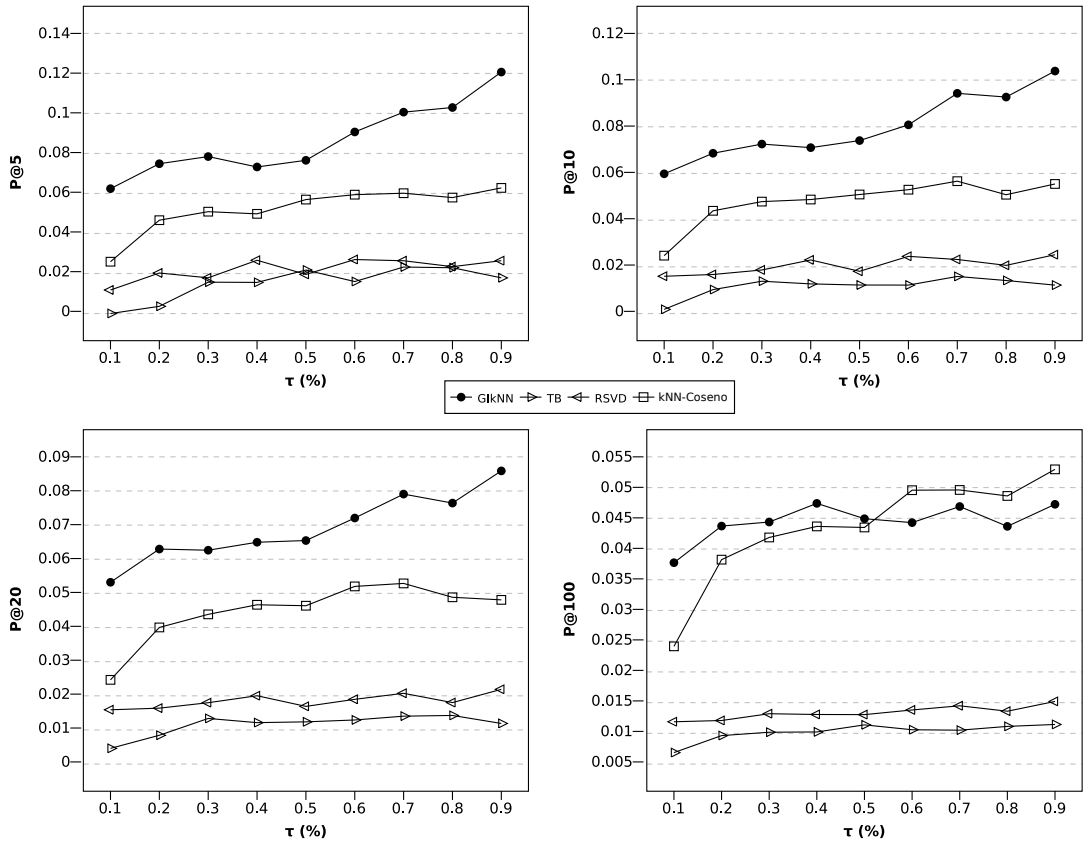


Figura 5.7: Evolución de $P@N$ según el tamaño del conjunto de entrenamiento, τ, en Netflix.

En primer lugar, se puede comprobar como, en la tarea de recomendación, los algoritmos kNN son muy superiores a algoritmos como RSVD y TB, que sí habían obtenido resultados mejores en la tarea de predicción. En particular, el algoritmo GIkNN es el que mejor resultados obtiene, con excepción de $P@100$ donde se ve superado por un algoritmo kNN basado en la similitud coseno. En todos los demás casos la precisión del algoritmo GIkNN es muy superior, con mejoras superiores al 50 % en la mayoría de casos, y próximas o incluso superiores al 100 % en muchos de ellos. La mejora respecto a los otros algoritmos es incluso superior. Además, los resultados son consistentes y estadísticamente significativos a lo largo de diferentes valores de τ, es decir, al variar la densidad del conjunto de entrenamiento, y en ambos conjuntos de datos. El algoritmo $GIkNN$ es el que mejor precisión tiene, seguido de kNN-Coseno y a mayor distancia RSVD y TB. De hecho, el algoritmo basado en tendencias presentado en la sección anterior no tiene un buen comportamiento en la tarea de recomendación, lo que contrasta

con sus buenos resultados en la tarea de predicción.

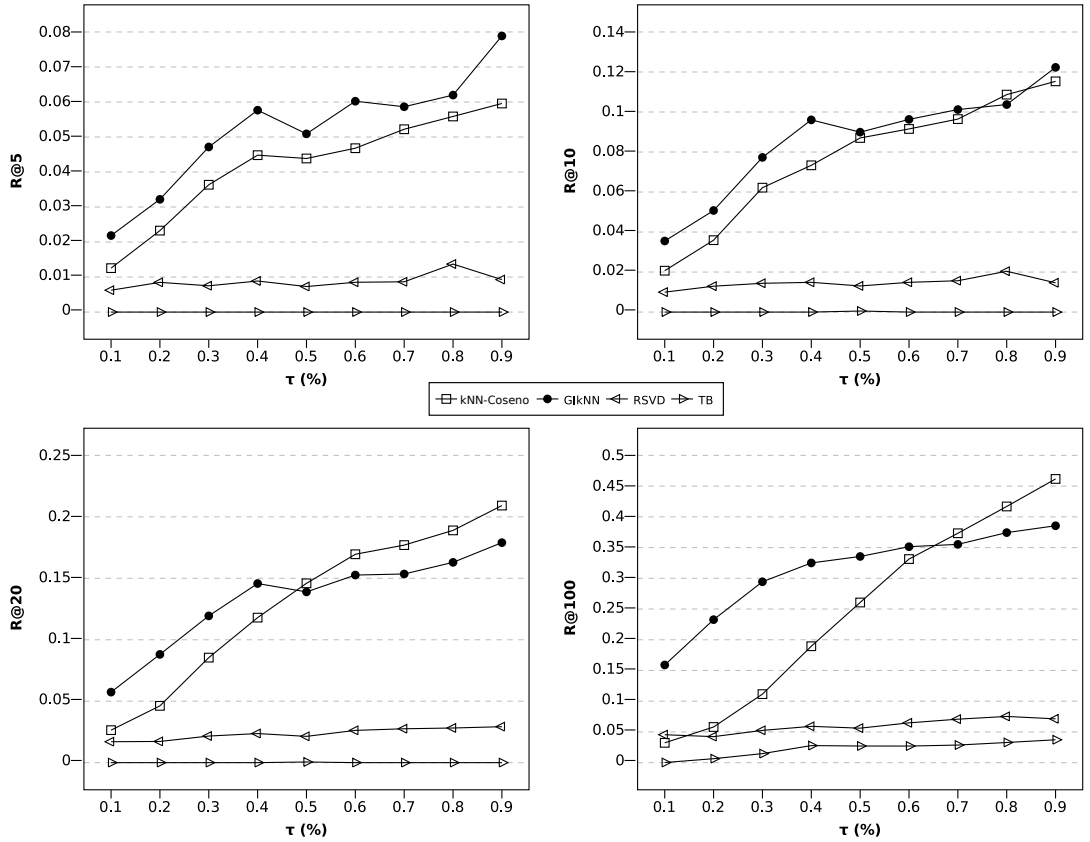


Figura 5.8: Evolución de $R@N$ según el tamaño del conjunto de entrenamiento, τ, en Movielens 10M.

En cuanto al *recall*, los resultados de los algoritmos kNN, y especialmente nuestra propuesta GIkNN, son también muy superiores a los obtenidos con otro tipo de algoritmos, tal como se observa en las Figuras 5.8 y 5.9. Mientras RSVD y TB apenas mejoran el *recall* al aumentar la densidad de la matriz, en los algoritmos kNN lo hace de forma substancial. Además, la técnica GIkNN sigue siendo muy superior a estas otras técnicas en casos de baja densidad.

Al contrario que en el caso de la precisión, donde generalmente es más interesante considerar listas de pequeño tamaño, la medida de $R@100$ es especialmente interesante pues permite evaluar en cierto modo el *coverage* del algoritmo. Mientras es de esperar que la medida de $R@5$, por ejemplo, ofrezca resultados discretos (ya que el algoritmo recomendará sólo 5 productos relevantes en el mejor de los casos), el uso de una lista de recomendación de gran tamaño ayuda a valorar la capacidad del algoritmo de cubrir

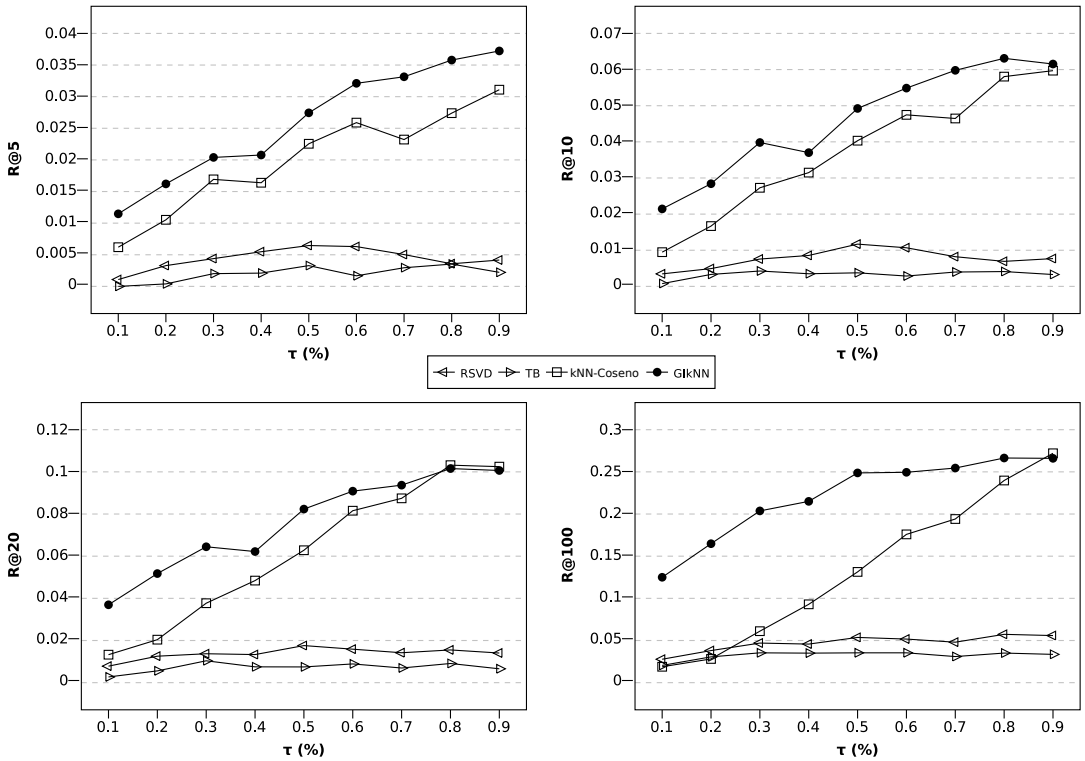
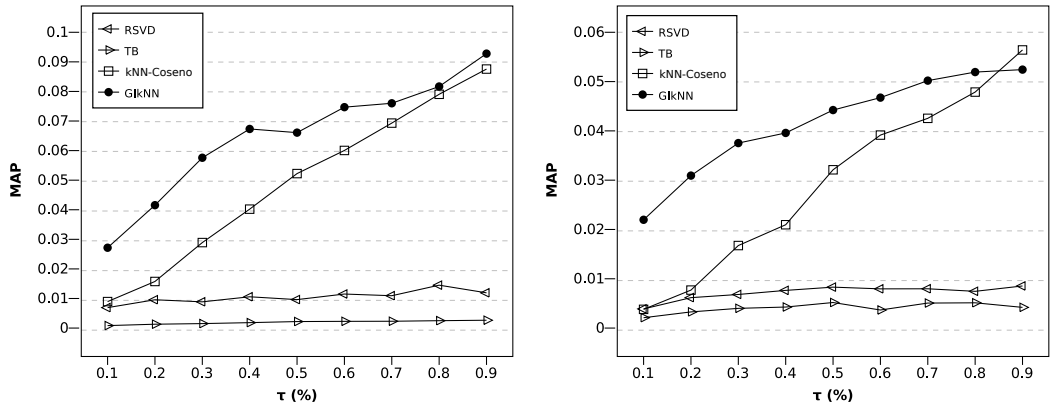


Figura 5.9: Evolución de $R@N$ según el tamaño del conjunto de entrenamiento, τ, en Netflix.

un amplio abanico de productos. En particular, vemos que los algoritmos basados en vecinos sobrepasan el 25 % de *recall* en Netflix, y llegan a superar el 40 % en Movielens 10M, lo cual supone un gran resultado teniendo en cuenta que otras técnicas apenas superan ligeramente el 5 %. Esto demuestra que los buenos resultados en precisión no se deben simplemente a que los algoritmos se limitan a hacer recomendaciones obvias, sino que realmente se ajustan a lo que el usuario espera, es decir, ofrecer una gran variedad de productos.

Finalmente, en las Figuras 5.10a y 5.10b se muestran los resultados en MAP. Como era de esperar teniendo en cuenta los resultados analizados hasta ahora, los algoritmos kNN se comportan mejor que otro tipo de técnicas, y de ellos es nuestra propuesta GIkNN la que mejores resultados obtiene. En particular, las mayores diferencias respecto a kNN-Coseno se obtienen cuando el conjunto de entrenamiento contiene un porcentaje de puntuaciones bajo. Esto es debido a que medidas de similitud tradicionales como coseno son difíciles de calcular de forma precisa con poca información, mientras la medida que proponemos es más sencilla y menos sensible a errores, con lo



(a) Movielens 10M **(b)** Netflix

Figura 5.10: Evolución de MAP según el tamaño del conjunto de entrenamiento, τ.

que se mejoran los resultados con baja densidad, sin que se vean afectados al aumentar esta.

5.6. Conclusiones y futuros trabajos

Tras haber estudiado la tarea de predicción en el capítulo anterior, en este nos hemos centrado en la tarea de recomendación, especialmente interesante por su destacado papel en sistemas comerciales y por su gran utilidad para el usuario.

Hemos comprobado como ambas tareas presentan importantes diferencias, y como algoritmos que presentan muy buenos resultados en la predicción no son siempre adecuados para la tarea de recomendación. En cambio, hemos visto como los algoritmos kNN sí ofrecen buenos resultados. Además, hemos propuesto una variante de este tipo de algoritmos especialmente diseñada para la tarea de recomendación, la cual permite obtener resultados sensiblemente mejores que las técnicas tradicionales. En particular, hemos propuesto una nueva medida de similitud entre usuarios y una nueva técnica para el cálculo de pesos de cara a la selección de productos a recomendar a partir de los vecinos. En futuros trabajos pretendemos estudiar posibles optimizaciones de ambas técnicas.

En lo relativo a la medida de similitud, hemos identificado las mejoras siguientes:

- Ponderar la similitud en función de los productos que ambos usuarios han puntuado, disminuyendo por tanto la posibilidad de incluir en el vecindario usuarios que apenas tengan productos en común con el usuario actual. Sería una técnica

similar a la usada en el coeficiente de Pearson ponderado [Herlocker et al., 1999][1].

- Ponderar la contribución de cada producto a la medida de similitud en función del grado de información que aporte dicho producto. Por ejemplo, una coincidencia en un producto muy popular es menos importante que una en un producto más específico o desconocido. De igual forma, coincidencias en productos que generan gran controversia aportan más información que productos generalmente aceptados y bien valorados. Se deberán estudiar los factores que determinan la importancia de un producto a la hora de determinar los gustos de un usuario, y ponderar su peso en función de esta, analizando el impacto real en la recomendación posterior.

- Tener en cuenta los desacuerdos entre usuarios. Actualmente sólo tenemos en cuenta productos que ambos usuarios han puntuado positivamente, pero en futuros trabajos pretendemos tener en cuenta los desacuerdos, es decir, productos que gustan sólo a uno de los usuarios. Un gran número de desacuerdos podría indicar diferencias entre usuarios que no hemos considerado en esta primera aproximación.

En cuanto al cálculo de pesos y selección de los productos a recomendar, en futuros trabajos se considerará la normalización de puntuaciones con el objetivo de tener en cuenta las diferencias entre usuarios, técnica que presenta buenos resultados en la tarea de predicción. También se estudiará la posible aplicación de un filtrado adicional previo a la recomendación, como por ejemplo obviar productos populares, los cuales no son especialmente útiles, o la diversificación de la lista de recomendaciones.

Finalmente, en este capítulo también hemos estudiado las limitaciones de las métricas y metodologías de evaluación que han sido utilizadas en la literatura, y hemos propuesto una metodología alternativa basándonos en la utilizada habitualmente en la evaluación de sistemas de recuperación de información, en nuestra opinión mucho más adecuada.

[1] Ver Sección 3.4.1.1.

Capítulo 6

Expansión del perfil

Las técnicas kNN presentan muy buenos resultados tanto en la tarea de predicción como en la tarea de recomendación, especialmente cuando la matriz de puntuaciones es relativamente densa, es decir, se dispone de suficiente información acerca de los usuarios. Sin embargo, cuando el número de puntuaciones disminuye, también lo hace la calidad de los resultados. Este problema es especialmente preocupante en el caso de nuevos usuarios, los cuales, precisamente por haber empezado a usar el sistema recientemente, apenas habrán puntuado unos pocos productos. En estos casos, el sistema tendrá poca información sobre sus gustos o intereses, lo que repercute negativamente en la calidad de las recomendaciones.

En este capítulo se introduce la técnica de expansión del perfil, consistente en aprovechar la información presente acerca de otros usuarios y productos para así ampliar el perfil de esos "nuevos usuarios" con puntuaciones generadas automáticamente, lo que repercute en una mejora de la calidad de los productos recomendados y, por tanto, aumenta la satisfacción del usuario y las posibilidades de fidelización del mismo con el sistema.

El capítulo se estructura de la siguiente forma. En la Sección 6.1 describimos en detalle el problema mencionado, y repasamos alguna de las soluciones que podemos encontrar en la literatura. También introducimos las técnicas de expansión de consultas, usadas en recuperación de información e íntimamente relacionadas con nuestra propuesta. Posteriormente, la Sección 6.2 introduce la técnica de expansión del perfil, sus fundamentos y las principales variantes. En la Sección 6.3 se describen los experimentos realizados para evaluar el comportamiento de las mismas y se presentan las principales conclusiones.

6.1. Introducción

6.1.1. El problema de *cold-start*

Como hemos visto en los capítulos anteriores, las técnicas de filtrado colaborativo, y en particular los algoritmos kNN, obtienen muy buenos resultados cuando se tiene mucha información sobre los usuarios, es decir, cuando estos han puntuado un gran número de productos, pero empeoran a medida que dicha información disminuye.

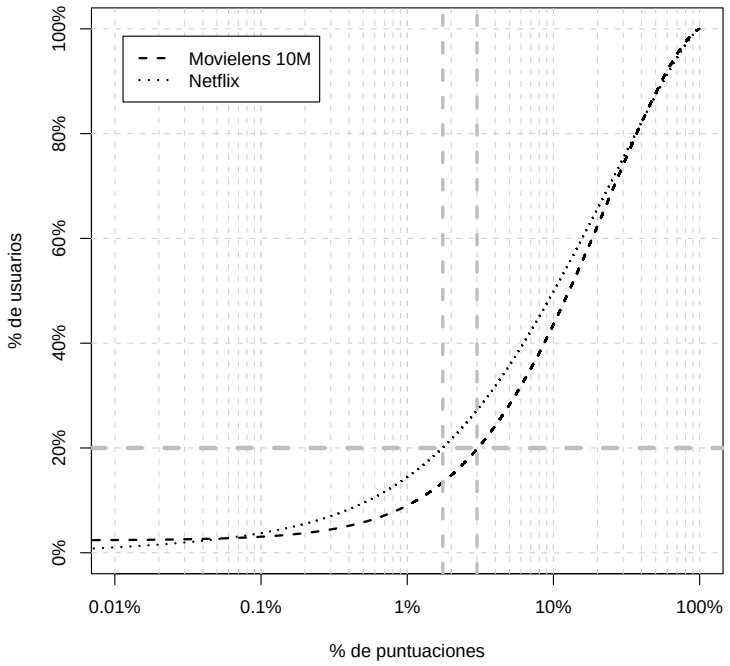


Figura 6.1: Distribución de las puntuaciones según el porcentaje de usuarios.

En muchas aplicaciones reales sucede que la información es por lo general escasa. Por ejemplo, en un sistema de recomendación de películas, lo normal es que un usuario sólo vea un pequeño porcentaje de las películas que se estrenan cada año, por lo que difícilmente va a puntuar más que unos pocos productos. Este comportamiento puede observarse en la Figura 6.1: en los conjuntos de datos Movielens 10M y Netflix, hay un 20 % de usuarios que sólo aportan un pequeño porcentaje de las puntuaciones, es decir, apenas se dispone de información acerca de los mismos. En otros dominios (por ejemplo, la recomendación de páginas web), las puntuaciones serán todavía más escasas.

Además, hay ciertas situaciones en las que este problema se acentúa [Park y Chu,

2009]:

- Cuando se añade un *nuevo producto* al sistema, que por tanto todavía nadie ha puntuado.
- Cuando un *nuevo usuario* se registra en el sistema.
- Cuando se lanza un *nuevo sistema* o aplicación de recomendación.

Estas situaciones se engloban en el problema conocido como *cold-start*, que afecta en general a todos los algoritmos de filtrado colaborativo. En la literatura podemos encontrar distintas aproximaciones para abordar cada situación en particular, muchas de las cuales están basadas en técnicas híbridas que además de las puntuaciones utilizan fuentes de información adicionales [Balabanović y Shoham, 1997].

Respecto al problema del nuevo producto, las puntuaciones se suelen combinar con información basada en contenido, es decir, información acerca del propio producto, como por ejemplo el género, director, o reparto en caso de la recomendación de películas. La información basada en contenido puede emplearse simplemente para estimar las puntuaciones de los usuarios y por tanto mejorar su perfil. Por ejemplo, Melville et al. [2002] propusieron el uso de un clasificador de texto bayesiano para estimar las puntuaciones desconocidas a partir de distintas características de los productos. Posteriormente, el perfil completo se utilizaba en un algoritmo de filtrado colaborativo tradicional. Otros autores han estudiado el uso de *bots*, usuarios virtuales que puntúan automáticamente ciertos productos según sus características o contenido [Good et al., 1999; Park et al., 2006]. Finalmente, en otros casos la información basada en contenido se incorpora como parte del modelo [Gunawardana y Meek, 2008; Schein et al., 2002].

Algunos autores también han empleado sistemas híbridos para abordar el problema del nuevo usuario. En este caso, se han estudiado fuentes de información adicionales, como datos demográficos [Lam et al., 2008; Nguyen et al., 2007], o relaciones de confianza o amistad en una red social [Papagelis et al., 2005a].

En general, es sencillo obtener información adicional sobre los productos, lo que convierte a las aproximaciones híbridas en una solución aceptable para este problema. Sin embargo, en el caso de nuevos usuarios la situación es diferente: sea cual sea la información empleada (edad, sexo, dirección, etc.), es el usuario el que tiene que suministrarla. Por tanto, sistemas como Movielens han optado por solicitar al usuario que, en lugar de proporcionar información de otro tipo, simplemente puntúe inicialmente un cierto número de productos, lo que permite el uso de un sistema de filtrado colaborativo puro.

En realidad, no es una mala idea solicitar información al usuario, ya que a cambio este espera obtener mejores recomendaciones, y no le importará dedicar cierto tiempo a introducir esta información. De hecho, las técnicas de filtrado colaborativo se basan precisamente en que el usuario estará dispuesto a ofrecer continuamente información sobre sus gustos en forma de puntuaciones, con el objetivo de mejorar su perfil y obtener un mayor beneficio del sistema de recomendación. Sin embargo, pedir demasiada información al principio sí puede ser un problema. Por ejemplo, si los productos que se muestran al usuario no son elegidos correctamente, el perfil inicial puede llegar a ser inútil, con la consiguiente decepción del usuario. Para evitar esto, algunos autores han propuesto determinadas técnicas para intentar optimizar esta fase inicial de recogida de información [Golbandi et al., 2011; Rashid et al., 2008].

Aún así, en muchos casos es mejor intentar aprovechar la información disponible, por muy escasa que esta pudiese ser, antes de molestar al usuario solicitándole demasiada información. Si se ofrecen recomendaciones razonablemente buenas desde el principio, el usuario continuará usando el sistema, ofreciendo progresivamente más información. Algunos autores han seguido este camino, desarrollando algoritmos que se comporten relativamente bien con nuevos usuarios. Por ejemplo, Ahn [2008] propuso una nueva medida de similitud para algoritmos kNN, basada en la proximidad, impacto y popularidad de las puntuaciones de los usuarios, con el objetivo de reemplazar a las medidas tradicionales como el coeficiente de correlación de Pearson, y conseguir un mejor resultado con nuevos usuarios. Así mismo, Piccart et al. [2010] propusieron un modelo probabilístico basado en grafos que también mejoraba los resultados en situaciones de *cold-start*.

6.1.2. Técnicas de expansión de consultas

En el mundo de la recuperación de información existe un problema muy similar al de nuevo usuario. Al igual que un perfil de un usuario que ha empezado a usar el sistema recientemente y, por tanto, apenas aporta información sobre sus preferencias, en un buscador una consulta con pocos términos o términos ambiguos puede no dar suficiente información acerca de la necesidad concreta del usuario. Es decir, a partir de la consulta el sistema no puede inferir con exactitud qué está buscando el usuario, de igual forma que un perfil compuesto de pocas puntuaciones puede no ser suficiente para estimar sus preferencias o gustos.

En el caso de sistemas de recuperación de información, las técnicas de *relevance feedback* y expansión de consultas o *query expansión* se desarrollaron precisamente para mejorar los resultados y aproximarse lo más posible a las necesidades reales de los

usuarios.

Las técnicas de *relevance feeback* se basan en analizar la respuesta de los usuarios al conjunto de documentos recuperado inicialmente. Se utiliza por tanto un proceso iterativo, donde la consulta (y por tanto los resultados obtenidos) se revisan a partir de la interacción del usuario. Por ejemplo, tras presentarse el resultado obtenido a partir de la consulta inicial, es posible que el usuario seleccione o acceda a los aquellos documentos que considera más relevantes. Esta información puede ser usada por el sistema para refinar la consulta y devolver una nueva lista de resultados más adecuada. Por ejemplo, el algoritmo de Rocchio [Rocchio, 1971] actualiza la consulta con el conjunto de términos que maximiza la distancia (vectorial) entre los documentos relevantes (aquellos a los que ha accedido el usuario) y los no relevantes.

En los sistemas de recomendación, *relevance feedback* es un mecanismo natural, pues se espera que los usuarios aporten información acerca de la relevancia de los productos recomendados, por medio de las puntuaciones. Sin embargo, puede llegar a ser un proceso lento, pues al contrario que un sistema de recuperación de información tradicional, donde el título o resumen de un documento puede dar una idea de su relevancia, en un sistema de recomendación el usuario tendría que conocer el producto antes de poder valorarlo. Además, esto suele tener un importante coste asociado (sea en dinero o en tiempo). En la práctica, en los sistemas de recomendación, *relevance feedback* es la forma natural de aprendizaje del sistema, pero por propia definición no es una solución al problema del nuevo usuario.

Alternativamente, los sistemas de recuperación de información pueden hacer uso de las técnicas de *expansión de consultas*, también llamadas *implicit relevance feedback*, que al contrario de las anteriores trabajan directamente a partir de la información disponible en la consulta inicial, y por tanto no requieren la posterior interacción del usuario. Generalmente estas técnicas se clasifican en *análisis local* y *análisis global* [Baeza-Yates y Ribeiro-Neto, 2008].

En el análisis local, el conjunto de documentos recuperado a partir de la consulta se examina con el objetivo de obtener términos relacionados, a partir de los cuales se expandiría la consulta original, dando lugar a un nuevo resultado (que se enviaría al usuario). Por ejemplo, la técnica de agrupamiento local [Attar y Fraenkel, 1977] extrae determinados términos presentes en la lista de resultados original.

El análisis global, por el contrario, está basado en información obtenida a partir de todos los documentos del sistema. Normalmente, estas técnicas construyen una especie de diccionario que relaciona distintos términos entre sí. La consulta se expandiría posteriormente con términos relacionados con aquellos que forman parte de la misma. El

diccionario se construye generalmente a partir de información extraída de la colección de documentos, como relaciones entre términos [Qiu y Frei, 1993] o análisis estadísticos [Crouch y Yang, 1992], pero también podría ser obtenida de otras fuentes, como por ejemplo WordNet [Stark y Riesenfeld, 1998], diccionarios de sinónimos o mediante minería de *query logs* [Fitzpatrick y Dent, 1997].

La técnica de expansión del perfil presentada en la próxima sección adapta esta idea al mundo del filtrado colaborativo.

6.2. Técnicas de expansión del perfil

En los algoritmos de filtrado colaborativo, el perfil de los usuarios, compuesto por las puntuaciones que han otorgado a distintos productos, tiene una importancia fundamental pues, al contrario de lo que sucede con otras técnicas, es la única fuente de información a la hora de calcular una recomendación.

El problema del nuevo usuario adquiere por tanto una dimensión crítica, pues el sistema no será capaz de ofrecer recomendaciones adecuadas a usuarios con pocas puntuaciones. En particular, en los algoritmos kNN este problema es especialmente preocupante, pues el perfil del usuario se usa directamente para calcular su similitud con otros usuarios y a partir de las mismas seleccionar el vecindario. Si este perfil es muy pequeño, las medidas de similitud no funcionarán adecuadamente y ofrecerán resultados incorrectos[1], lo que tendrá un gran impacto en fases posteriores. En concreto, el vecindario estará formado por usuarios que no serán necesariamente similares, y la posterior recomendación será poco menos que aleatoria.

La técnica de expansión del perfil o *Profile Expansion* (PE) consiste en incrementar el perfil de aquellos usuario con pocas puntuaciones, añadiendo productos adicionales al mismo. Este perfil expandido será el que se use posteriormente en la fase de cálculo de similitudes, minimizando el problema anteriormente mencionado y, por tanto, mejorando el cálculo de las mismas.

Formalmente, dado un usuario $u \in \mathcal{U}$ y su correspondiente perfil, P_u, definido según la Ecuación 3.1, el propósito de las técnicas de expansión del perfil es el de calcular un nuevo *perfil extendido*, $P'_u = P_u \cup P^e_u$, donde P^e_u es el conjunto de pares producto-puntuación que se han añadido al perfil. Es decir, $P^e_u = \left\{(i, r^e_i) \,|r^e_i \in \mathcal{R}, i \in \mathcal{I} - \mathcal{I}^{(u)}\right\}$. Además, el número de productos a añadir está limitado por un parámetro l, es decir, $|P^e_u| \leq l$.

En la práctica, es posible emplear diferentes técnicas de cara a elegir los productos

[1] En la Sección 4.5 se ha presentado una detallada discusión respecto a este asunto.

con que expandir el perfil, así como la puntuación a ellos otorgada. Dichas técnicas puede ser clasificadas de acuerdo a dos criterios:

1. *Técnicas locales frente a técnicas globales.* De forma similar a las técnicas de expansión de consultas que hemos discutido previamente, las técnicas de expansión del perfil pueden ser locales o globales. Las técnicas locales están basadas en los resultados obtenidos tras calcular una recomendación a partir del perfil original. Por otro lado, las técnicas globales calculan el perfil extendido a partir de información que no depende del perfil del usuario.

2. *Basadas en usuarios o basadas en productos.* Las técnicas basadas en productos eligen los elementos con los que extender el perfil a partir de un conjunto de productos. Por otro lado, las técnicas basadas en usuarios parten de un conjunto de usuarios, y eligen los elementos para expandir entre los productos por ellos puntuados.

	Global	**Local**
Basadas en productos	Productos similares a aquellos en el perfil del usuario, de acuerdo a información global, tal como opiniones de otros usuarios, características de los productos, etc.	Productos recomendados por el algoritmo a partir del perfil original del usuario.
Basadas en usuarios	Productos puntuados por usuarios similares, teniendo en cuenta información global como por ejemplo datos demográficos.	Productos valorados por los vecinos calculados a partir del perfil del usuario.

Tabla 6.1: Resumen de los productos con los que expandir el perfil, según las distintas técnicas.

En la Tabla 6.1 se muestra un resumen de las distintas técnicas y con qué productos expanden el perfil cada una de ellas. En el resto de esta sección se describirán más en detalle alguna de las alternativas propuestas. En particular, en este trabajo hemos empleado técnicas basadas únicamente en información colaborativa, es decir, en las puntuaciones otorgadas por los distintos usuarios. Estas técnicas tienen la ventaja de que no necesitan más información que aquella ya disponible en la matriz de puntuaciones. Sin embargo, es importante destacar la posibilidad de usar otro tipo de información al expandir el perfil. Por ejemplo, información basada en contenido, como la descripción de los productos u otro tipo de características. Muchos de los sistemas de

recomendación híbridos (ver Sección 2.5) pueden ser adaptados a la expansión del perfil, dando lugar a algoritmos basados en filtrado colaborativo pero que también hacen uso de información adicional de cara a mejorar la calidad de los perfiles. Es decir, más que un conjunto de técnicas, la expansión del perfil puede describirse como un marco para abordar el problema del nuevo usuario.

6.2.1. Técnicas globales basadas en productos

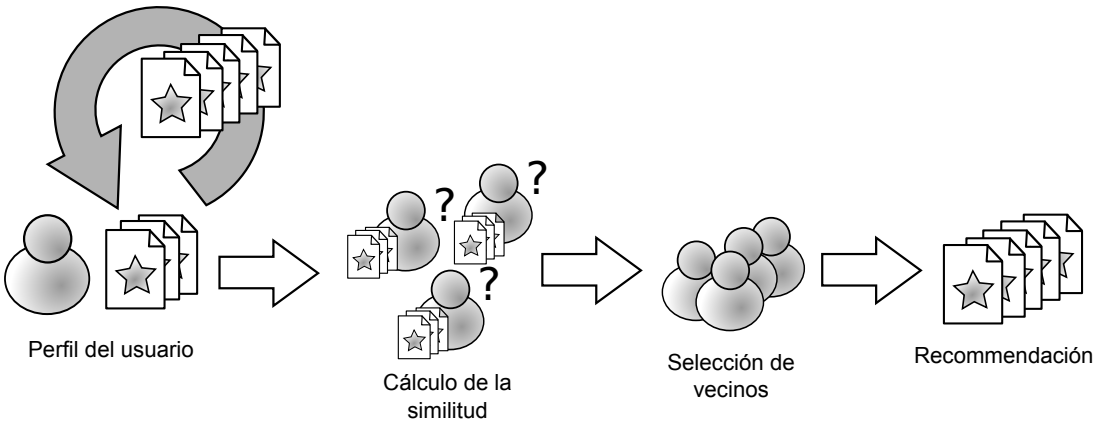


Figura 6.2: Técnicas de expansión del perfil globales basadas en productos.

Las técnicas globales basadas en productos expanden el perfil de un usuario con otros productos similares a aquellos que ya formaban parte del mismo en un principio. Tal y como se observa en la Figura 6.2, estas técnicas actúan al principio del proceso, antes de que se lleve a cabo cualquier otra operación como podría ser la selección de vecinos, por lo que no disponen de más información que el perfil original del usuario.

En este trabajo hemos estudiado una aproximación puramente colaborativa, que se basa en las puntuaciones disponibles en la matriz. Alternativamente, se podrían haber usado otras fuentes de información. Por ejemplo, información acerca del contenido de los productos, tales como el director o el género en caso de recomendación de películas.

En una aproximación colaborativa, las puntuaciones de distintos productos se comparan con el objetivo de encontrar los productos más parecidos a aquellos presentes en el perfil del usuario. Para tal fin, es necesario definir una función de similitud $s : \mathcal{I} \times \mathcal{I} \rightarrow \Re$ capaz de comparar dos productos entre sí. En los experimentos presentados en la Sección 6.3, se ha usado la función de similitud coseno (Ecuación 3.4), si bien se podría haber usado cualquier otra función de similitud, entre ellas las presentadas en la Sección 3.4.1.1.

Una vez calculadas todas las similitudes, el perfil se expande con los l productos

más similares. La puntuación que se asigna a dichos productos en el perfil expandido se calculará a partir de la puntuación que tenían los productos que existían originalmente en el perfil del usuario, ponderada según la similitud con los mismos. De esta forma, la puntuación final será más parecida a la que el usuario había dado a los productos similares. El proceso, como el lector habrá reconocido, es muy similar al de los algoritmos de filtrado colaborativo basados en productos, descritos en la Sección 3.4.1.2.

Una limitación de esta técnica es que la elección de los productos más similares es una operación costosa. En la práctica, es razonable precalcular todas las similitudes y mantener listas con los productos más parecidos a cada uno de ellos.

6.2.2. Técnicas locales basadas en productos

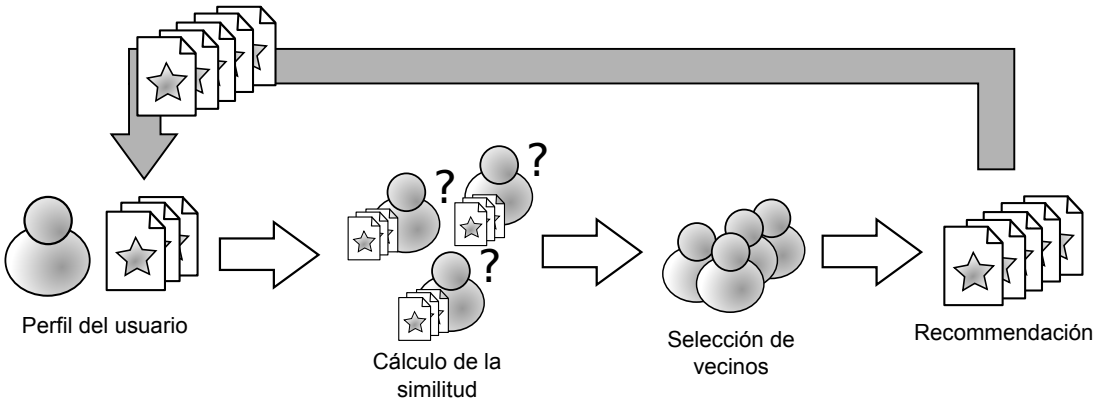


Figura 6.3: Técnicas de expansión del perfil locales basadas en productos.

Tal y como se muestra en la Figura 6.3, las técnicas locales basadas en productos expanden el perfil del usuario a partir de los productos recomendados originalmente. Es decir, se llevan a cabo dos recomendaciones. En la recomendación inicial, se utiliza el perfil original del usuario, sin expandir. Si ese perfil original es muy pequeño, los productos recomendados habrán sido calculados en base a muy poca información sobre las preferencias del usuario, y por tanto pueden no ser los más adecuados. Sin embargo, sí son un reflejo de la información que el sistema tiene sobre el usuario y sus gustos. Por tanto, en una segunda fase, el sistema expande el perfil original con estos nuevos productos, completando el perfil del usuario y permitiendo que, en la segunda recomendación, tanto el vecindario encontrado como la lista de productos a recomendar sean mucho más adecuados. Esta segunda recomendación es la que se presenta al usuario.

En la práctica, no es buena idea considerar todos los productos presentes en la recomendación original. Si el tamaño de la lista es muy grande o el perfil original

muy pequeño, es posible que algunos de los productos no sea realmente adecuado. En general, limitar el número de productos con los que expandir el perfil es una buena práctica. Por tanto, nos quedaremos solamente con los l productos que encabezan la recomendación inicial. Se podrían llegar a usar técnicas más complejas, como buscar la diversidad de productos teniendo en cuenta otro tipo de factores, pero no han sido considerados en este trabajo.

6.2.3. Técnicas globales basadas en usuarios

Estas técnicas consisten en localizar usuarios similares al usuario actual, y expandir el perfil con productos que esos usuarios han puntuado. Al igual que las técnicas globales basadas en productos, la expansión se lleva a cabo previamente a cualquier otra fase del algoritmo de recomendación, utilizando información global de cara a encontrar usuarios parecidos. Por ejemplo, se podría utilizar información demográfica, como la edad, sexo, lugar de residencia, ocupación laboral, nivel de estudios, etc.

Es importante destacar que, en caso de usar únicamente información colaborativa (es decir, las puntuaciones presentes en la matriz) para encontrar los usuarios similares, estas técnicas son muy parecidas a las técnicas locales basadas en usuarios, que presentaremos a continuación. La razón es que en ese caso, el conjunto de usuarios seleccionados por similitud con el usuario actual va a ser parecido al vecindario que se obtendría aplicando el algoritmo de recomendación con el perfil original. Por tanto, dado que en este trabajo nos limitamos al uso de información colaborativa, esta técnica no será objeto de evaluación.

6.2.4. Técnicas locales basadas en usuarios

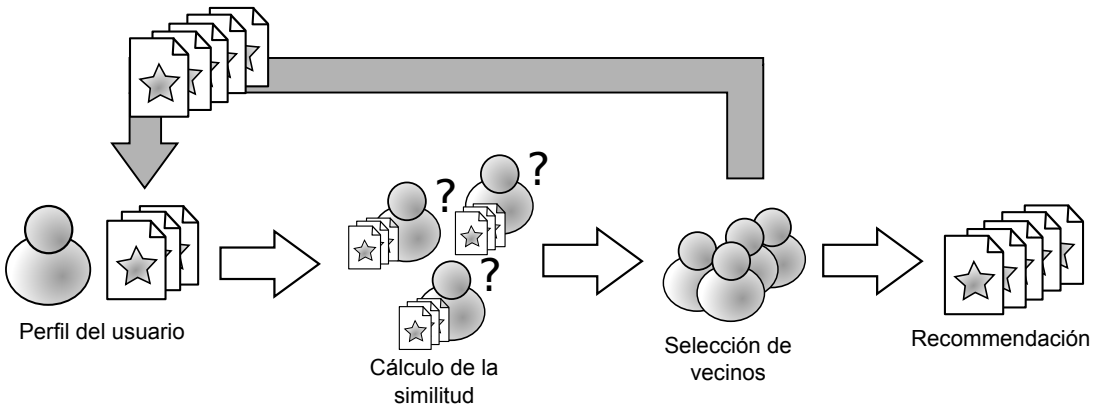


Figura 6.4: Técnicas de expansión del perfil locales basadas en usuarios.

Tal y como se muestra en la Figura 6.4, estas técnicas parten del vecindario calculado a partir del perfil original. Consisten en seleccionar productos que han sido puntuados por los usuarios que forman parte de este vecindario. La idea es sencilla: a partir del perfil original, dejamos que el algoritmo de recomendación encuentre los usuarios más parecidos al usuario actual. Es de suponer, por tanto, que sus puntuaciones serían parecidas a las que efectuaría el usuario actual, y por tanto pueden ser utilizadas para expandir su perfil.

Obviamente, si el perfil original es pequeño, el vecindario, aunque sea el mejor que el algoritmo puede seleccionar teniendo en cuenta la limitada información de que dispone, no será ni mucho menos perfecto. Por tanto, es deseable seleccionar cuidadosamente las puntuaciones con las que se va a expandir realmente el perfil. Para tal fin se pueden usar distintas técnicas, entre las cuales se han considerado las siguientes:

- **Productos mejor puntuados**. Consiste en expandir el perfil con los l productos mejor puntuados por los vecinos. En caso de que un producto fuese puntuado por varios vecinos, se hallaría la media ponderada según la similitud de cada vecino. Esta técnica es similar a la aproximación local basada en productos, es decir, a expandir el perfil con los productos recomendados originalmente, ya que estos productos corresponderán a los mejor valorados por los vecinos. Sin embargo, mientras la recomendación sólo contendrá productos bien valorados (ya que no tiene sentido recomendar malos productos), no hay en principio ningún impedimento para expandir el perfil con productos que han recibido una baja puntuación. Por tanto, siempre y cuando l sea lo suficientemente grande, esta técnica se diferenciará de las técnicas locales basadas en productos.

- **Productos más puntuados**. En este caso, el perfil se expande con los l productos que han sido puntuados por más vecinos, independientemente de la puntación otorgada. Esta aproximación es especialmente interesante, ya que incrementa las posibilidades de que los productos elegidos para expandir el perfil sean parecidos a aquellos en que se ha basado el vecindario, es decir, a aquellos que realmente interesan al usuario. De hecho, responde a la misma idea que habíamos planteado en el algoritmo GIkNN presentado en la Sección 5.3: productos puntuados por muchos vecinos posiblemente estén más relacionados con los gustos del usuario. Al fin y al cabo, son los que los vecinos tienen en común.

- **Vecinos locales**. Este método es similar a las técnicas globales basadas en productos, ya que expanden el perfil con los productos más parecidos a los que forman parte del perfil del usuario. Sin embargo, mientras las técnicas globales

tienen en cuenta todas las puntuaciones disponibles en la matriz, en este caso sólo se consideran las puntuaciones otorgadas por los vecinos.

- **Agrupamiento local de usuarios**. Al igual que el método anterior, esta técnica calcula las similitudes entre productos teniendo en cuenta únicamente las puntuaciones de los vecinos. Sin embargo, esta similitud se calcula empleando un método completamente diferente. En concreto, se utiliza un método inspirado en la técnica de *local association matrix* [Attar y Fraenkel, 1977] usada en recuperación de información para la expansión de consultas. Consiste en calcular la similitud entre dos productos multiplicando la puntuación que han recibido de cada vecino. Es decir, dado un producto del sistema, $i \in \mathcal{I}$, y un producto perteneciente al perfil del usuario actual, $j \in \mathcal{I}^{(a)}$, la similitud entre ambos se calcularía según la siguiente ecuación:

$$s(i,j) = \frac{\sum_{n \in \mathcal{N}(a)} r_{ni} r_{nj}}{|\mathcal{N}(a)|} \tag{6.1}$$

6.3. Experimentos y resultados

A continuación, presentamos los resultados obtenidos por las diferentes técnicas. La evaluación se ha realizado utilizando la metodología que hemos presentado en el capítulo anterior. Con el objetivo de simular una condición de nuevo usuario, para cada usuario de evaluación nos hemos quedado únicamente con N puntuaciones, siguiendo por tanto una estrategia *Given-N*. Estas puntuaciones formarán parte del conjunto de entrenamiento, junto a las pertenecientes al resto de usuarios. Las puntuaciones no seleccionadas pasarán a formar parte del conjunto de evaluación.

6.3.1. Resultados de las técnicas globales basadas en productos

En esta sección se muestran los resultados obtenidos para las las técnicas globales basadas en productos, en particular el método basado en la similitud coseno mencionado en la Sección 6.2.1. Hemos estudiado tanto el comportamiento general de esta técnica como la influencia del parámetro l, que determina el número máximo de productos a añadir al perfil del usuario. En las Figuras 6.5 y 6.6 mostramos las curvas P-R obtenidas tanto con 2 como con 10 puntuaciones por usuario, es decir, siguiendo una aproximación *Given-2* y *Given-10*.

En general, podemos observar que las técnicas globales basadas en productos ofrecen muy buenos resultados, especialmente en la precisión de los primeros productos de

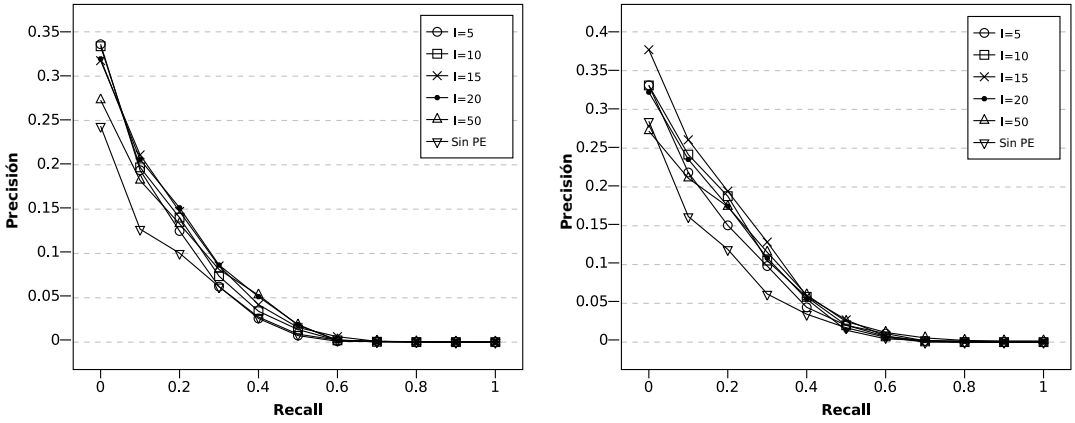


Figura 6.5: Curvas P-R de las técnicas globales basadas en productos en el conjunto de datos Movielens 10M, según el número de productos añadidos al perfil, l. Resultados con 2 (izquierda) y 10 (derecha) puntuaciones para cada usuario de evaluación.

	Movielens	Netflix
$l = 5$	**0.138**	0.125
$l = 10$	**0.140**	**0.144**
$l = 15$	0.130	0.109
$l = 20$	0.127	0.102
$l = 50$	0.106	0.082
Sin PE	0.077	0.094

Tabla 6.2: P@5 de la técnica de expansión del perfil global basada en productos, según el número de productos añadidos al perfil, l, con $N = 2$. Se han resaltado los mejores valores.

la lista de recomendaciones. Precisamente, estos productos son los que tienen mayor posibilidad de ser seleccionados, por lo que cabe esperar que estas técnicas de expansión del perfil mejoren significativamente la satisfacción de los nuevos usuarios. En particular, tal como se muestra en la Tabla 6.2, las técnicas globales basadas en productos ofrecen una gran mejora en P@5 respecto al algoritmo sin expansión del perfil. Los mejores resultados se producen con $N = 2$, es decir, cuando el usuario apenas ha interactuado con el sistema. Este resultado es esperado, pues esa es una situación muy extrema en la que el algoritmo apenas tiene información del usuario, y demuestra el buen comportamiento de la expansión del perfil. De hecho, la mejora llega a ser del 80 % en algunos casos, y además es estadísticamente significativa, independientemente del valor de l[1].

[1]Exceptuando valores grandes de l, en concreto, con $l = 50$, como se explicará a continuación.

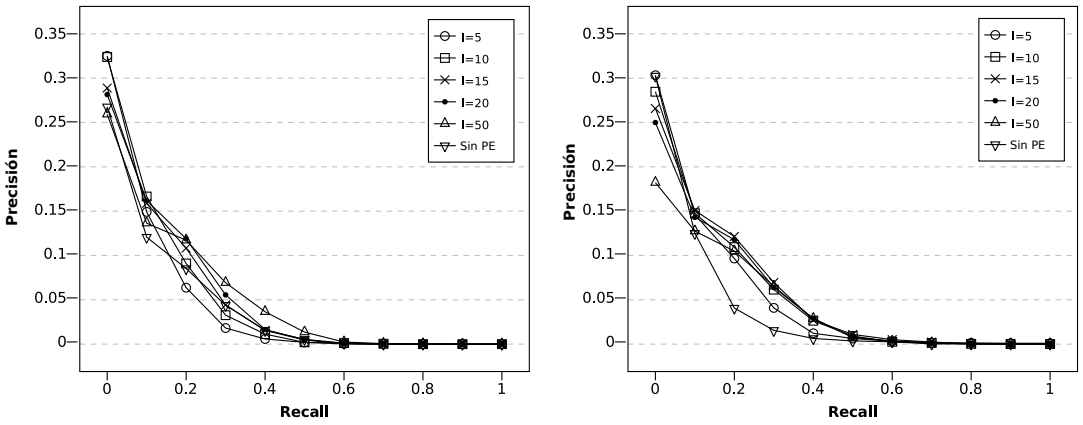


Figura 6.6: Curvas P-R de las técnicas globales basadas en productos en el conjunto de datos Netflix, según el número de productos añadidos al perfil, l. Resultados con 2 (izquierda) y 10 (derecha) puntuaciones para cada usuario de evaluación.

Por el contrario, a medida que N aumenta, el beneficio de expandir el perfil es menor, lo cual también es un resultado esperado, pues al tener más información el algoritmo se comporta bien sin necesidad de la expansión. Por ejemplo, con $N = 10$ no se obtienen mejoras significativas con el conjunto de datos Netflix, aunque sí se consiguen mejoras en torno al 50 % en Movielens.

En cuanto al número de productos con el que se debe expandir el perfil, l, los mejores resultados se han obtenido usando un valor pequeño, especialmente con el conjunto de entrenamiento *Given-2*. Cuando se usa un valor de l demasiado grande, el perfil se acaba expandiendo con productos no demasiado similares a los existentes en el perfil original, lo que empeora los resultados. Los mejores resultados han sido obtenidos con $l \leq 10$. Sin embargo, cuando el perfil original es mayor (por ejemplo, con $N = 10$), se tiene más información y es posible incrementar ligeramente l sin que eso afecte negativamente al resultado final. De hecho, en el conjunto de datos Movielens 10M, con $N = 10$ se consiguen mejores resultados con un l mayor. En general, los mejores resultados se obtienen cuando l es ligeramente mayor que el tamaño del perfil original.

Finalmente, hemos estudiados el impacto en MAP, métrica que tiene en cuenta todos los productos recomendados, independientemente de su posición. Como puede observarse en la Figura 6.7, las técnicas globales basadas en productos también mejoran esta métrica. Los resultados con $l = 5$, sin embargo, son modestos, en particular si los comparamos con los obtenidos en P@5, donde sí se comportaba muy bien. La razón es que el perfil expandido sigue sin ser lo suficientemente grande para conseguir una lista grande de productos adecuados. De todas formas, creemos que mejorar la precisión

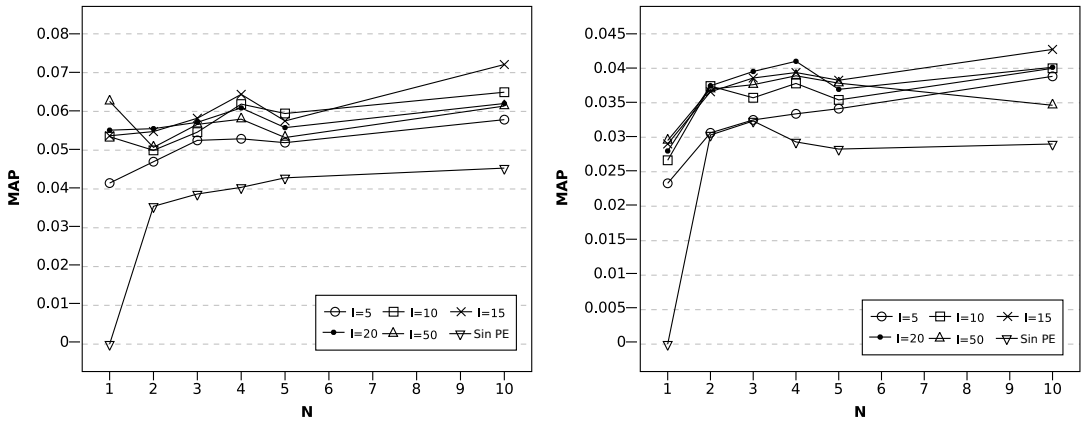


Figura 6.7: Evolución de la MAP para las técnicas de expansión globales basadas en productos con distinto tamaño de expansión, l, según el tamaño original del perfil, N. Resultados calculados en Movielens 10M (izquierda) y Netflix (derecha).

de los primeros productos es más adecuado en la mayoría de aplicaciones, si bien, lógicamente, esto dependerá de cada caso concreto. Como hemos visto, el parámetro l se podría ajustar para conseguir el resultado deseado.

6.3.2. Resultados de las técnicas locales basadas en productos

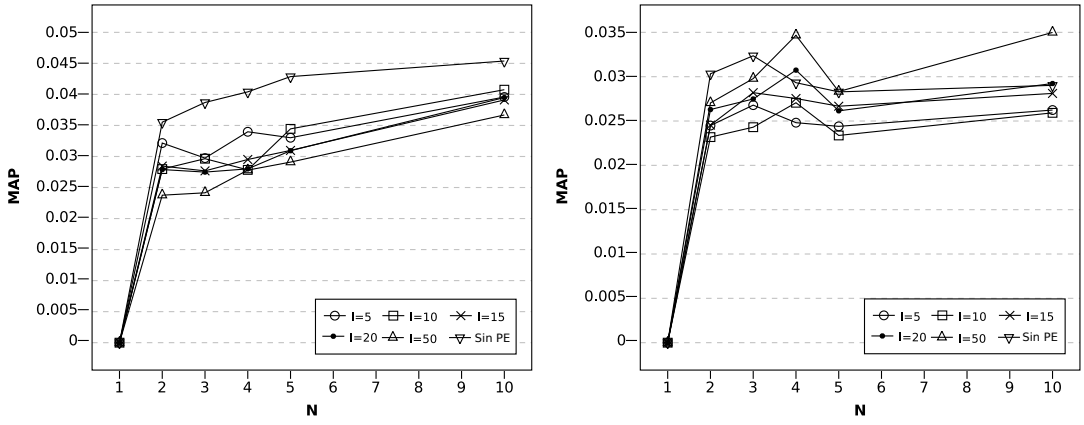


Figura 6.8: Evolución de la MAP para las técnicas de expansión locales basadas en productos con distinto tamaño de expansión, l, según el tamaño original del perfil, N. Resultados calculados en Movielens 10M (izquierda) y Netflix (derecha).

Al contrario que las técnicas globales basadas en productos, las técnicas locales no presentan buenos resultados, y en muchos casos incluso llegan a empeorar los resultados obtenidos sin expansión del perfil, como se muestra en la Figura 6.8.

El problema de estas técnicas es que la expansión está basada en los productos recomendados inicialmente. Si el sistema no tiene suficiente información acerca del usuario, estos productos no serán una buena recomendación, y el perfil se expandirá con productos que no son adecuados para el usuario, dando lugar a una posterior recomendación todavía peor. Al contrario que en RI, en filtrado colaborativo no es una buena técnica el asumir que la recomendación original es relativamente correcta. De hecho, la expansión de consultas en RI es más parecida a las técnicas locales basadas en usuarios, que se presentan a continuación. La razón es que en RI la consulta se utiliza para obtener una lista de documentos, de la misma forma que en filtrado colaborativo el perfil se usa para calcular el vecindario. La recomendación de productos a partir de los vecinos es un paso adicional necesario en estos sistemas, y cuando el perfil original es pequeño introduce errores adicionales.

6.3.3. Resultados de las técnicas locales basadas en usuarios

Finalmente, en esta sección se presentan los resultados obtenidos para las técnicas locales basadas en usuario presentadas en la Sección 6.2.4: productos mejor puntuados, productos más puntuados, vecinos locales y agrupamiento local de usuarios.

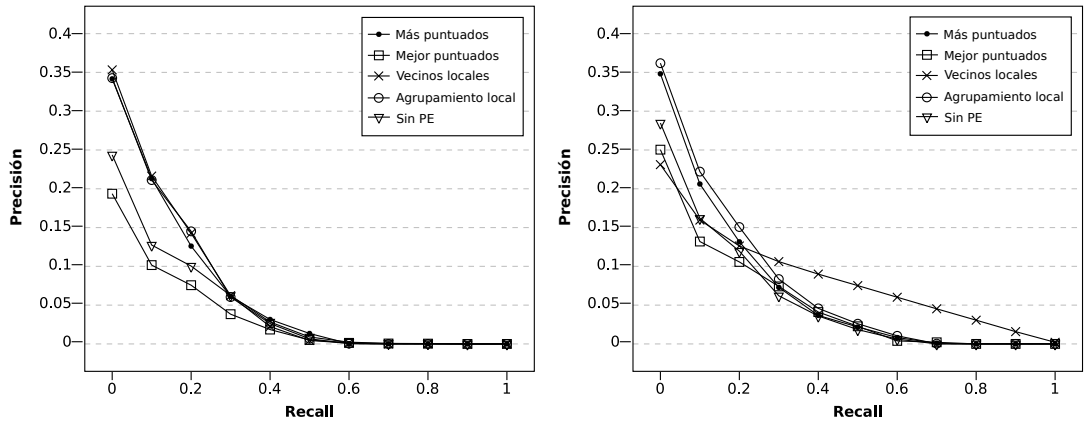


Figura 6.9: Curvas P-R de las técnicas locales basadas en usuarios en el conjunto de datos Movielens 10M, con $l = 10$. Resultados con 2 (izquierda) y 10 (derecha) puntuaciones para cada usuario de evaluación.

En las Figuras 6.9 y 6.10 se muestran las curvas P-R de cada método, obtenidas con $N = 2$ y $N = 10$. En primer lugar, se puede observar que la técnica "productos mejor puntuados" no funciona demasiado bien: en Netflix es equivalente a no expandir el perfil en absoluto, y en Movielens 10M es incluso peor. A la vista de los resultados obtenidos por las técnicas locales basadas en productos, este resultado es el esperado,

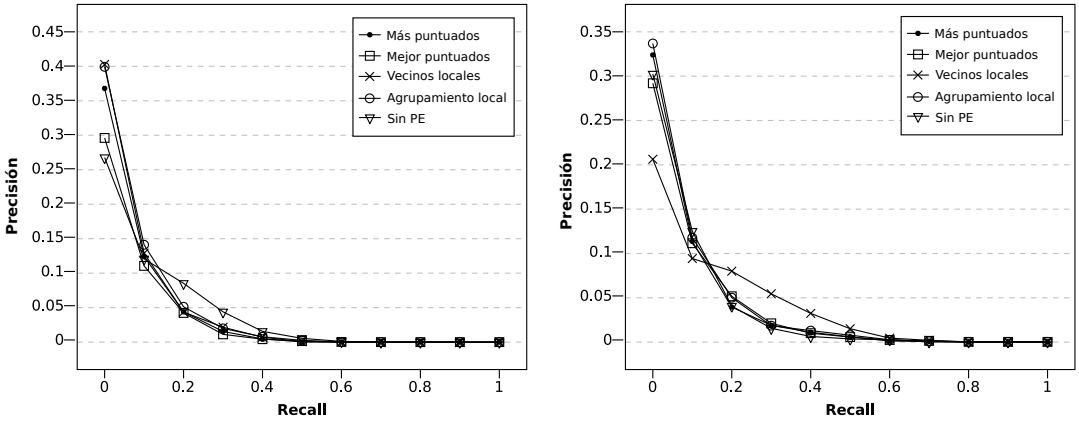


Figura 6.10: Curvas P-R de las técnicas locales basadas en usuarios en el conjunto de datos Netflix, con $l = 10$. Resultados con 2 (izquierda) y 10 (derecha) puntuaciones para cada usuario de evaluación.

	Más puntuados		Agrupamiento local		Vecinos locales	
	Movielens	Netflix	Movielens	Netflix	Movielens	Netflix
$l = 5$	0.143	**0.163**	0.116	**0.166**	0.129	**0.169**
$l = 10$	0.142	0.154	0.141	**0.160**	0.142	**0.173**
$l = 15$	**0.155**	0.154	**0.165**	0.145	0.139	0.124
$l = 20$	**0.157**	0.145	0.154	0.136	**0.175**	0.125
$l = 50$	0.122	0.122	0.121	0.120	0.134	0.123
Sin PE	0.077	0.094	0.077	0.094	0.077	0.094

Tabla 6.3: P@5 de diferentes técnicas locales de expansión del perfil basadas en productos, según el número de productos añadidos al perfil, l, con $N = 2$. Se han resaltado los mejores valores.

pues ambas aproximaciones son bastante parecidas.

Sin embargo, las restantes técnicas sí se comportan bien, mejorando significativamente la precisión obtenida. Con $N = 2$, todos los métodos (a excepción del antes mencionado) obtienen mejoras notables en ambos conjuntos de datos, especialmente en las primeras posiciones de la lista de recomendaciones. En la Tabla 6.3 se muestran los resultados detallados de P@5. Puede verse como todas las técnicas dan lugar a mejoras superiores al 80 % en Netflix, y al 100 % en Movielens 10M, superando incluso a las técnicas globales basadas en productos.

Además, con $N = 10$ también obtienen mejoras superiores a las técnicas globales basadas en productos. Sin embargo, mientras los métodos de "productos más puntuados" y "agrupamiento local de usuarios" obtienen muy buenos resultados en las primeras

posiciones de la lista, la técnica de "vecinos locales" no se comporta especialmente bien en esas posiciones (con $N = 10$), y por contra tiende a mejorar la precisión global. Estos resultados se confirman al estudiar la evolución de la MAP según el número de puntuaciones en el perfil original, N, tal como se muestra en la Figura 6.11.

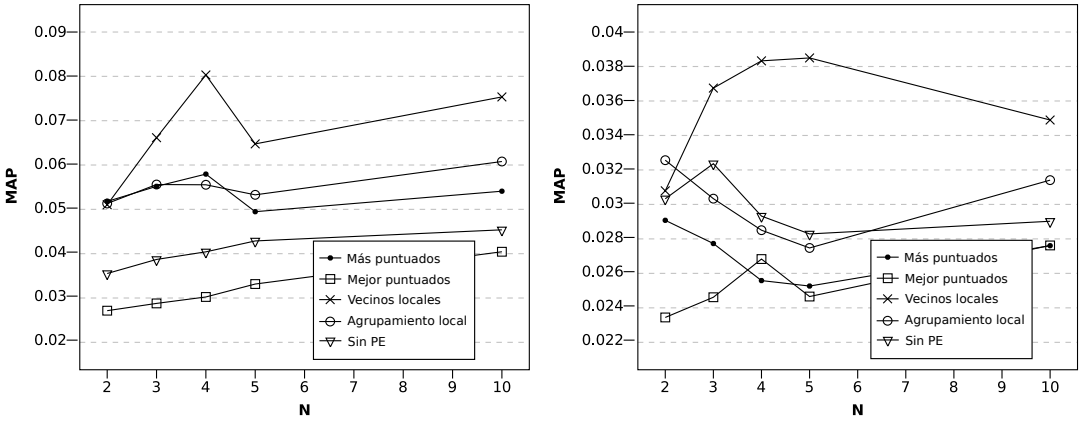


Figura 6.11: Evolución de la MAP para las técnicas de expansión locales basadas en usuarios con $l = 10$, según el tamaño original del perfil, N. Resultados calculados en Movielens 10M (izquierda) y Netflix (derecha).

Por tanto, si se busca mejorar la precisión media, la técnica de "vecinos locales" es la más adecuada, mientras los métodos de "productos más puntuados" y "agrupamiento local de usuarios" son una mejor opción si lo que se desea es recomendar un número pequeño de productos. En particular, la técnica de "productos más puntuados" destaca por su sencillez.

Finalmente, en cuanto al número de productos con los cuales expandir el perfil, l, en Netflix los mejores resultados se obtienen con un número pequeño, mientras en Movielens 10M un número ligeramente mayor es más adecuado. En general, es una buena ideal aumentar l a medida que el usuario interactúa con el sistema y por tanto tiene más puntuaciones en su perfil.

6.4. Conclusiones y futuros trabajos

En este capítulo abordamos el problema del nuevo usuario, que en mayor o menor medida afecta a todos los sistemas de recomendación, y en especial a los basados en filtrado colaborativo. En concreto, hemos propuesto un conjunto de técnicas diseñadas con el objetivo de minimizar este problema. Todas ellas se basan en la expansión del perfil del usuario, por lo que pueden agruparse bajo un marco común en el que también

tendrían cabida otro tipo de métodos. La idea es aprovechar la información disponible para incrementar el número de puntuaciones en el perfil del usuario actual, con el objetivo de mejorar el conocimiento que el algoritmo de recomendación va a tener acerca del usuario.

En particular, hemos propuesto varias técnicas basadas en esta idea, las cuales pueden clasificarse en: globales basadas en productos, locales basadas en productos, y locales basadas en usuarios. Con la excepción de esta última, especialmente pensada para algoritmos basados en vecinos, las técnicas de expansión del perfil se podrían utilizar con cualquier algoritmo de filtrado colaborativo, si bien nos hemos centrado precisamente en su aplicación a algoritmos kNN.

La principal ventaja de los métodos propuestos es que están basados enteramente en las puntuaciones de los usuarios, es decir, en información colaborativa, y por lo tanto no necesitan información adicional a aquella que va a utilizar el algoritmo de recomendación, lo cual supone una importante ventaja frente a la forma en que tradicionalmente se ha abordado el problema del nuevo usuario.

La evaluación realizada muestra que tanto las técnicas globales basadas en productos como las técnicas locales basadas en usuarios se comportan especialmente bien. En particular, permiten mejorar significativamente la precisión de los productos recomendados en las primeras posiciones, lo que tiene un impacto importante en la percepción del usuario acerca de la calidad del sistema. La mejora alcanza el 80 % en las técnicas globales basadas en productos, y el 100 % en las técnicas locales basadas en usuario.

En cambio, las técnicas locales basadas en productos obtienen malos resultados. Esto se debe a que la recomendación original, basada en un perfil muy pequeño, no es adecuada para el usuario y por tanto tampoco es una buena fuente de productos a partir de los cuales expandir el perfil. Un motivo de este comportamiento es que los algoritmos kNN no se comportan demasiado bien cuando la información disponible es limitada, especialmente cuando se usan ciertas medidas de similitud. Una alternativa a estudiar en futuros trabajos es la utilización de otro tipo de algoritmos de recomendación en esta primera fase, el resultado de los cuales se usaría para expandir el perfil. También se podrían utilizar dos medidas de similitud diferentes, una en cada fase.

En general, las técnicas locales basadas en usuario son la mejor aproximación, especialmente los métodos de "productos más puntuados" y "agrupamiento local de usuarios". Este último está inspirado en una técnica tradicional de expansión de consultas usada en RI, lo que demuestra que ambos problemas tienen similares características, y es posible adaptar técnicas de expansión de consultas a la expansión del perfil.

Por otro lado, si bien las técnicas globales basadas en productos ofrecen resultados

inferiores, siguen siendo una buena alternativa, pues son sencillas y capaces de ofrecer buenas recomendaciones a usuarios con una única puntuación. En el futuro se estudiará cómo información acerca del contenido de los productos puede ser usada en este tipo de técnicas. En general, esta información es sencilla de conseguir y puede aportar importantes beneficios. Igualmente, se estudiarán las posibilidades de las técnicas globales basadas en usuarios, en las que datos demográficos o relaciones de amistad en redes sociales pueden ser especialmente interesantes.

Parte III

Algoritmos kNN eficientes y escalables

Capítulo 7

Técnicas para mejorar la eficiencia de los algoritmos kNN

En la actualidad, los sistemas de recomendación son usados en infinidad de aplicaciones y dominios, en muchos de los cuales deben manejar una gran cantidad de usuarios y productos. La eficiencia y escalabilidad de los algoritmos empleados es, por tanto, uno de los factores claves para su éxito en sistemas comerciales. Sin embargo, tradicionalmente la investigación se ha centrado en la mejora de la precisión, dando lugar a técnicas complejas difícilmente utilizables en entornos reales con gran volumen de datos.

Esto contrasta con ámbitos como el de recuperación de información, donde siempre se ha dado una gran importancia a aspectos relacionados con el rendimiento de los sistemas. En este capítulo se introduce nuestra propuesta para unificar el mundo de la RI con el filtrado colaborativo, con el objetivo de aprovechar las contribuciones que se han hecho en ese ámbito a la mejora de la eficiencia, sin renunciar a algoritmos de recomendación exitosos y sobradamente contrastados. En particular, nos hemos centrado en mejorar la eficiencia de los algoritmos basados en vecinos, técnica muy popular gracias a sus buenos resultados en términos de precisión, simplicidad, etc.

En la Sección 7.1, se presentan los principales trabajos relacionados con la optimización de algoritmos de filtrado colaborativo publicados hasta la fecha. A continuación, en la Sección 7.2 se describe la técnica propuesta, presentando una implementación de los algoritmos kNN basada en el modelo vectorial usado en RI, y describiendo cómo optimizar el cálculo gracias al uso de índices para almacenar la matriz de puntuaciones. Finalmente, en la Sección 7.3 se describe la técnica de preselección de vecinos, cuyo objetivo es mejorar los tiempos de recomendación sin que la precisión del sistema se vea afectada significativamente. En los dos capítulos siguientes se propondrán técnicas

adicionales que mejoran aún más el rendimiento de estos sistemas.

7.1. Eficiencia y escalabilidad de los sistemas de recomendación

Los sistemas de recomendación han experimentado un auge espectacular en los últimos años. De su uso en dominios específicos, en aplicaciones con pocos usuarios o de naturaleza relativamente estática se ha pasado a un uso generalizado, que incluye desde portales de comercio electrónico con millones de usuarios a dominios tan dinámicos como la recomendación de noticias.

En este nuevo escenario, la eficiencia y escalabilidad de las técnicas a emplear pasa a ser un factor clave, que puede determinar el éxito o fracaso de un determinado sistema. El desarrollo de nuevas técnicas que mejoren los aspectos relacionados con el rendimiento no sólo es necesario en las aplicaciones de hoy en día, sino que es imprescindible para expandir el uso de sistemas de recomendación a dominios con grandes posibilidades como la recomendación de búsqueda en la web, contenidos de redes sociales, etc., caracterizados por un gran volumen de datos y una componente dinámica y de caducidad de la información muy importante.

Sin embargo, en la literatura apenas se pueden encontrar propuestas en este sentido. En general, el interés de la comunidad investigadora ha estado más centrado en la mejora de otros aspectos, fundamentalmente los relacionados con la precisión de los resultados, especialmente en la tarea de predicción.

Los escasos trabajos centrados en la mejora de la eficiencia han abordado el problema desde dos puntos de vista diferentes: el diseño de algoritmos eficientes, o la implementación eficiente de técnicas bien conocidas y suficientemente contrastadas. Algunos autores se han centrado en el diseño de nuevos algoritmos, basados en variantes sencillas de métodos más complejos, o bien en modelos sencillos y por tanto fáciles de calcular [Lemire, 2003; Lemire y Maclachlan, 2005]. Un ejemplo de este enfoque sería el algoritmo basado en tendencias presentado en la Sección 4.2. En otros casos, los autores se han centrado en diseñar algoritmos pensados para su uso en entornos distribuidos, donde la eficiencia y escalabilidad se consigue gracias a repartir el trabajo entre distintas máquinas. Por ejemplo, Das et al. [2007] propusieron el uso de *MapReduce* [Dean y Ghemawat, 2008] para distribuir el cálculo de su algoritmo de recomendación de noticias. A veces, el distribuir el algoritmo es necesario debido a la propia naturaleza del sistema, como en el caso de algoritmos pensados para redes P2P, donde la carga se distribuye entre los distintos nodos. Un ejemplo de este método sería el propuesto por

Wang et al. [2006c], basado en un modelo probabilístico. Otro ejemplo de algoritmo adaptado a la naturaleza del sistema es el propuesto por Ali y van Stam [2004] para la recomendación de programas de televisión en el sistema TiVo, que aprovecha que cada usuario siempre accede desde su receptor para dividir el procesamiento entre cliente y servidor.

El principal inconveniente de abordar el problema de la eficiencia mediante el diseño de nuevos algoritmos es que, a pesar de presentar un mejor rendimiento, pueden ser inferiores a las técnicas tradicionales en otros aspectos. Por ejemplo, pueden presentar peores resultados en términos de precisión, diversidad, etc., características igualmente importantes y en los que técnicas tradicionales se comportan perfectamente. Otras veces el algoritmo es simplemente menos genérico, y si bien presenta buenos resultados en una tarea o dominio concreto, puede no presentarlos al usarlo en un entorno diferente a aquel para el que había sido diseñado. Por ejemplo, nuestro algoritmo basado en tendencias obtiene muy buenos resultados en predicción, pero bastante discretos en recomendación.

Precisamente, técnicas tradicionales como los algoritmos basados en vecinos destacan por presentar muy buenos resultados en multitud de dominios y condiciones, y es por eso que en lugar de diseñar nuevos algoritmos, algunos autores han centrado sus esfuerzos en mejorar la implementación de este tipo de técnicas. La gran ventaja de esta aproximación es que permite aprovechar, en entornos exigentes computacionalmente, un algoritmo contrastado y con multitud de ventajas [Desrosiers y Karypis, 2011].

Las soluciones propuestas en la literatura involucran diferentes aspectos del algoritmo, centrándose sobre todo en la fase de cálculo de vecinos, que es también la más costosa y generalmente el principal cuello de botella de estas aproximaciones. Algunos autores se centraron en la actualización del sistema a medida que los usuarios introducían nuevas puntuaciones. En lugar de tener que volver a procesar la matriz completa para calcular la similitud entre usuarios, Papagelis et al. [2005b] proponían mantener almacenado el resultado de varios cálculos intermedios, de forma que la similitud se pudiese actualizar fácilmente y así mantener el vecindario actualizado permanentemente. Otros autores han optado por mejorar los tiempos del cálculo del vecindario mediante mejoras en la implementación, por ejemplo mediante el uso de *MapReduce* para distribuir la carga entre diferentes máquinas [Jiang et al., 2011; Schelter et al., 2012].

Un enfoque particularmente interesante es intentar emplear técnicas ampliamente extendidas en el ámbito de la RI para mejorar la implementación de algoritmos kNN. Por ejemplo, Cöster y Svensson [2002] estudiaron el uso de un índice invertido para el cálculo de la similitud entre usuarios, consiguiendo mejoras en el tiempo de selección

del vecindario. Más recientemente, Bellogín et al. [2011] estudiaron la relación entre los algoritmos kNN y los métodos clásicos de RI, proponiendo un *framework* que unificaba ambas aproximaciones. Si bien no abordaron aspectos relacionados con la eficiencia, aproximar el mundo de la recomendación y la recuperación de información tiene implicaciones importantes a este respecto.

Al contrario que lo sucedido en recomendación, la eficiencia siempre ha sido considerada un elemento fundamental a tener en cuenta en el diseño de sistemas de recuperación de información. Las lecciones aprendidas durante el desarrollo de sistemas de RI capaces de manejar *terabytes* de información suponen un conocimiento de gran valor acerca de técnicas, algoritmos y arquitecturas que permiten optimizar diferentes aspectos relacionados con la escalabilidad y eficiencia de estos sistemas. En lugar de reinventar la rueda, el aprovechar esta experiencia en el mundo de los sistemas de recomendación puede suponer un gran beneficio. Precisamente, en esta tesis hemos abordado la eficiencia y escalabilidad de los sistemas de recomendación partiendo de las soluciones desarrolladas en RI.

Sin embargo, para poder hacer esto es necesario abordar la tarea de recomendación de una forma similar a la resolución de consultas en un sistema de RI. En la siguiente sección mostramos como los algoritmos kNN mostrados en el Capítulo 5 pueden implementarse fácilmente utilizando un enfoque basado en RI, y proponemos el uso de índices para el cálculo de vecindario y recomendación, extendiendo la propuesta de Cöster y Svensson [2002]. Posteriormente, veremos como gracias a este enfoque se pueden conseguir importantes mejoras de rendimiento basadas en técnicas ampliamente extendidas en el mundo de la recuperación de información.

7.2. Algoritmos kNN basados en un modelo vectorial

El modelo vectorial es una de las aproximaciones más populares en la recuperación de información, siendo la base de los potentes buscadores disponibles actualmente. Es por ello que existen numerosas técnicas para mejorar la eficiencia de distintos aspectos de un sistema basado en este modelo, desde la construcción y almacenamiento de índices, el procesamiento de las consultas, la ejecución en entornos distribuidos, etc.

La utilidad de implementar algoritmos kNN siguiendo un modelo similar está por tanto fuera de toda duda, y esta aproximación será precisamente el objetivo de esta sección. Se estudiará cómo el modelo vectorial puede aplicarse a algoritmos kNN, y se discutirá su implementación utilizando índices.

7.2.1. El modelo vectorial

Las técnicas de recuperación de información abordan el problema de cómo satisfacer la necesidad de información de los usuarios. Generalmente, estos expresan sus necesidades de información en forma de una consulta, y el sistema devuelve una lista ordenada de documentos relevantes para la consulta proporcionada. Formalmente, un sistema de RI puede verse como una función $S : q \rightarrow \{d_1, d_2, \ldots, d_k\}$, con $d_i \in D$, siendo D el conjunto de documentos disponible (o colección), y q una consulta. En muchos sistemas, los documentos son vistos simplemente como textos, y las consultas son realmente un conjunto de términos. Intuitivamente, la relevancia de un documento va a estar relacionada con la presencia en el mismo de los términos de la consulta efectuada.

En los modelos de RI clásicos, los documentos se modelan generalmente como simples conjuntos de palabras, es decir, se representan únicamente con los términos que contienen, descartándose aspectos como el formato e incluso la estructura del mismo. Es decir, $d = \{t_1, t_2, \ldots\}$, con $t_i \in T$, siendo T el conjunto de términos[1]. Dado que no todos los términos en un documento tienen la misma importancia, a cada uno de ellos se le asignará un valor numérico que indica su *peso* en el documento, es decir, la importancia que tiene el término para describir el contenido semántico del documento.

El modelo vectorial [Salton et al., 1975] tiene en cuenta el peso de los diferentes términos, representando cada documento $d_i \in D$ como un vector con tantos elementos como términos, y donde el valor de cada elemento refleja el peso del correspondiente término en el documento. Es decir, $\vec{d_i} = \{w_{1,i}, w_{2,i}, \ldots w_{n,i}\}$, donde n es el número de términos en la colección, y cada $w_{j,i} \geq 0$ representa el peso del término $t_j \in T$ en el documento d_i. Si un término no aparece en el documento, se le asigna un peso igual a 0. De igual forma, las consultas pueden representarse como un vector $\vec{q} = \{w_{1,q}, w_{2,q}, \ldots w_{n,q}\}$. Es decir, una consulta puede verse como un pequeño documento donde cada término presente en la consulta tiene también un peso asociado.

El cálculo del valor de cada peso viene determinado por el modelo de pesos usado, y generalmente se hace a partir de algún dato acerca de los términos o documentos en la colección. Por ejemplo, un modelo de pesos muy sencillo es el *tf-idf*, mostrado en la Ecuación 7.1, y donde el peso de un término se calcula a partir del número de veces que aparece en el documento (es decir, la frecuencia del término o *tf*), y el número de documentos que contienen dicho término (*df*). La idea de este modelo es dar más peso a aquellos términos que aparecen muchas veces en un documento (y por tanto tienen más

[1] Téngase en cuenta la diferencia entre término y palabra, ya que en un sistema de RI es habitual ignorar palabras comunes (por ejemplo, preposiciones) o reducir diferentes variaciones de género o número en un único término.

importancia en el mismo), pero evitando dar demasiado peso a términos que son muy comunes y por tanto aparecen en gran número de documentos, ya que estos términos no aportan realmente información acerca de la naturaleza del documento.

$$w_{t,d} = tf_{t,d} \log \frac{|D|}{df_t} \tag{7.1}$$

Una de las grandes ventajas del modelo vectorial es que es muy sencillo obtener los documentos relevantes para una consulta, pues únicamente se debe calcular la similitud entre el vector de la consulta, $\vec{q}$ y el vector de cada documento, $\vec{d_i}$. Los documentos más similares a la consulta serán, por tanto, los más relevantes. Como medida de similitud se pueden usar distintos métodos, siendo el coseno del ángulo entre ambos vectores (distancia coseno) uno de los más sencillos y populares:

$$sim(d_i, q) = \frac{\vec{d_i} \bullet \vec{q}}{|\vec{d_i}| \times |\vec{q}|} \tag{7.2}$$

7.2.2. El modelo vectorial en recomendación

Para poder usar el modelo vectorial en la tarea de recomendación, debemos modelar de forma adecuada las distintas fases del algoritmo. En particular, un algoritmo de recomendación basado en vecinos puede verse como la suma de dos funciones:

1. Una función de cálculo del vecindario, $N : a \rightarrow \{u_1, u_2, \ldots, u_k\}$, con $a, u_i \in \mathcal{U}$, la cual a partir de un usuario obtiene un conjunto de usuarios similares.

2. Una función de recomendación, $R : N(a) \rightarrow \{i_1, i_2, \ldots, i_n\}$, con $i_x \in \mathcal{I}$, que a partir del vecindario obtiene la lista de productos a recomendar.

El cálculo del vecindario con un modelo vectorial se puede derivar directamente del proceso empleado para resolver consultas en RI, simplemente substituyendo documentos por usuarios y términos por productos. Al igual que hacíamos en RI, los usuarios se representan como vectores de tantos elementos como productos en el sistema. La diferencia fundamental es que mientras en RI cada elemento del vector representa el peso o importancia del término en el documento, en un sistema de recomendación representaría la preferencia del usuario sobre el correspondiente producto, es decir, su puntuación. Podemos por tanto representar un usuario como un vector de puntuaciones, tal como se muestra en la siguiente ecuación:

$$\vec{u} = \{r_{u1}, r_{u2}, \ldots, r_{un}\} \tag{7.3}$$

Esto supone una pequeña ventaja frente a los sistemas de recuperación de información, especialmente si estamos hablando de sistemas con preferencias explícitas, y es que es el usuario el que directamente indica su grado de interés por los distintos productos. Es decir, la relevancia de un producto es proporcionada por una persona, en lugar de ser estimada a partir de ciertas estadísticas de la colección.

Asimismo, al igual que en RI la consulta se representaba como un vector de términos, y por tanto era tratada como cualquier otro documento, en la fase de cálculo de vecinos la "consulta" es, de hecho, otro usuario, en concreto el usuario actual. Es decir, mientras en RI el proceso de consulta parte de un "documento" (la consulta) para dar como resultado una lista de documentos, en el cálculo del vecindario se parte de un usuario para obtener una lista de usuarios. Por tanto, nuestra consulta es realmente el perfil del usuario actual, que podemos representar con el vector $\vec{a}$ al igual que hemos hecho con el resto de usuarios:

$$\vec{a} = \{r_{a1}, r_{a2}, \ldots, r_{an}\} \tag{7.4}$$

A partir de esta representación, es evidente que calcular el vecindario en un algoritmo kNN es equivalente a resolver una consulta en un sistema de RI. Es decir, consiste en calcular la similitud entre los vectores $\vec{a}$ y cada vector $\vec{u}$, $\forall u : u \in U, u \neq a$. Para el cálculo de la misma se puede utilizar una medida tradicional como la distancia coseno, o bien otro tipo de técnicas como la presentada en la Sección 5.2.

De igual forma, la fase de recomendación también puede representarse siguiendo un modelo vectorial. En comparación con RI, sin embargo, en esta fase habría que substituir los documentos por productos, y los términos por usuarios, al revés de como se había hecho en el cálculo del vecindario.

Básicamente, cada producto se representaría con un vector con tantos elementos como usuarios en el sistema, representando cada elemento la preferencia de dicho usuario por el producto. Al igual que en el caso anterior, las puntuaciones son usadas como indicativo de la preferencia, con lo que no es necesario ningún tipo de estimación, y el vector que representa a un producto $i \in \mathcal{I}$ sería el mostrado en la siguiente ecuación:

$$\vec{i} = \{r_{1i}, r_{2i}, \ldots, r_{mi}\} \tag{7.5}$$

La representación de la consulta es, en este caso, un poco diferente, pues en la fase de recomendación el equivalente a la consulta serían los vecinos del usuario actual, calculados en la fase anterior. Sin embargo, el vecindario puede también representarse como un vector $\vec{n_a}$ de longitud m (es decir, tantos elementos como usuarios en el sistema), donde cada elemento del mismo representa la similitud entre el usuario correspondiente

y el usuario actual, siendo esta 0 en caso de que el usuario no pertenezca al vecindario. Es decir:

$$\vec{n_a} = \{w_1, w_2, \ldots, w_m\} \tag{7.6}$$

con:

$$w_u = \begin{cases} s(a,u) & if\ u \in \mathcal{N}(a) \\ 0 & otherwise \end{cases} \tag{7.7}$$

Es decir, el vecindario se representa como un "producto" virtual, y el proceso de recomendación consistirá en buscar productos similares al mismo. De esta forma, la selección de los productos a recomendar pasa a ser equivalente a la selección de vecinos, sólo que en lugar de partir de un usuario para obtener usuarios similares, partimos de un "producto" para obtener productos similares. Nuevamente, el proceso se reduce al cálculo de similitud entre vectores. Eso sí, en esta fase se podrían utilizar otro tipo de medidas de similitud, como por ejemplo la presentada en la Sección 5.3.2. En ese caso, la similitud vendría dada por el peso ω_i.

7.2.3. Índices de puntuaciones

Por sí mismo, el uso del modelo vectorial tal como ha sido presentado en la sección anterior no supone una mejora del rendimiento respecto a una implementación más tradicional de un algoritmo kNN. De hecho, no es particularmente eficiente. La selección del vecindario implica calcular la similitud entre m vectores, cada uno de ellos de n elementos, lo que supone una complejidad computacional de $O(mn)$. Posteriormente, se han de seleccionar los k vecinos más próximos, lo que implica ordenar las similitudes calculadas, es decir, una complejidad $O(m\log(m))$. Finalmente, en la recomendación se ha de calcular la similitud entre n vectores de longitude m —$O(mn)$— y ordenarlos —$O(n\log(n))$— para seleccionar finalmente los N productos a recomendar.

Sin embargo, en la práctica la mayor parte de elementos de los distintos vectores van a tener el valor 0, y pueden ser ignorados durante los cálculos. Esto se debe a que la matriz de puntuaciones es extremadamente dispersa, y es un comportamiento similar a lo que sucede en RI (la mayor parte de términos sólo están presente en unos pocos documentos). Al igual que en dicho caso, se puede hacer uso de una estructura en forma de índice para calcular de forma eficiente tanto el vecindario como la recomendación.

En particular, para implementar un algoritmo kNN usando el modelo vectorial van a ser necesarios dos índices:

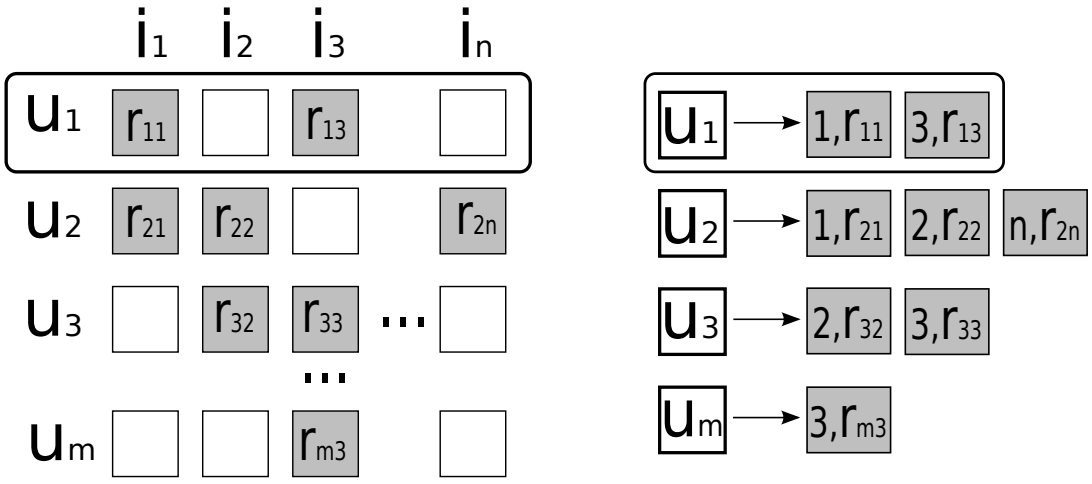


Figura 7.1: El índice del perfil del usuario se calcula directamente a partir de las filas de la matriz de puntuaciones.

- En primer lugar, el *índice de perfiles de usuario* (Figura 7.1), que permite obtener directamente el perfil de un usuario, es decir, los elementos del vector $\vec{u}$. Formalmente, puede verse como una función $f_u : \mathcal{U} \rightarrow 2^{\mathcal{I} \times \mathcal{R}}$ tal que $f_u(u) = \{(i, r_{ui}) | i \in \mathcal{I}^{(u)}\}$. Este índice se construiría directamente a partir de las filas de la matriz de puntuaciones, y estaría compuesto por dos ficheros: el fichero con la *lista de puntuaciones de usuarios* y el *índice de usuarios*, donde para cada usuario se almacena la posición de su lista de puntuaciones en el fichero anterior. El fichero con la lista de puntuaciones almacena el identificador y puntuación para cada producto puntuado por el usuario, es decir:

$$\langle (i, r_{ui}) \, | i \in \mathcal{I}_u \rangle \tag{7.8}$$

  Lógicamente, el índice sólo almacena aquellos productos puntuados por el usuario, por lo que las necesidades de almacenamiento son mucho menores que el tamaño teórico de la matriz. Típicamente, por debajo del 2 % [Cacheda et al., 2011a]. Obsérvese, sin embargo, que cualquier vector $\vec{u}$ puede recrearse fácilmente a partir de dicho índice.

- En segundo lugar, el *índice invertido de perfiles de usuario* (Figura 7.2), que almacena, para cada producto, las puntuaciones que ha recibido de los distintos usuarios. También puede verse como una función $f_i : \mathcal{I} \rightarrow 2^{\mathcal{U} \times \mathcal{R}}$ tal que $f_i(i) = \{(u, r_{ui}) | u \in \mathcal{U}^{(i)}\}$. Dicho índice se utiliza durante la fase de selección del vecindario, facilitando el cálculo de similitudes entre usuarios. Al igual que en el

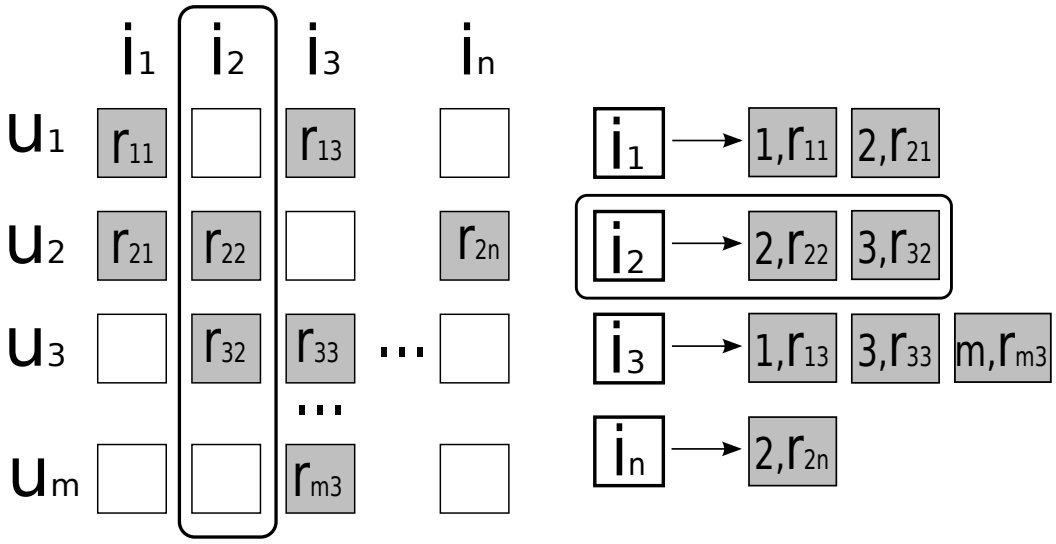


Figura 7.2: El índice invertido de perfiles de usuario se calcula a partir de las columnas de la matriz de puntuaciones.

caso anterior, este índice consta de dos ficheros, el *índice de productos* y la *lista de puntuaciones de productos*.

7.2.4. Cálculo de vecinos y recomendación usando índices

Los algoritmos kNN presentados en la Sección 5.2 pueden implementarse de forma eficiente a partir de los dos índices presentados en la sección anterior, gracias al enfoque basado en el modelo vectorial que también ha sido presentado anteriormente.

En primer lugar, es necesario calcular el vecindario, lo que requiere el cálculo de la similitud entre el vector que representa el usuario actual y los vectores correspondientes al resto de usuarios. Asumiendo el uso de la medida de similitud coseno (Ecuación 3.4), el proceso a seguir sería el mostrado en el Algoritmo 1. En caso de usarse otra medida de similitud el proceso sería similar.

El proceso requiere una consulta inicial al índice de perfiles de usuario, para así obtener el perfil del usuario actual. Luego, para cada producto en dicho perfil, se consulta el índice invertido para obtener la lista de usuarios que lo han puntuado. La similitud de cada usuario se actualiza con la contribución de dicho producto, y el proceso se repite hasta haber procesado todos los productos del perfil. El factor de normalización mostrado en el algoritmo corresponde al denominador de la función coseno, que al ser constante para cada usuario se puede calcular durante la construcción del índice y almacenarlo junto a este, para evitar tener que volver a calcularlo cada vez.

Algorithm 1 Cálculo del vecindario para un usuario $a \in \mathcal{U}$

```
1: for all u ∈ U do
2:    Similitud[u] ← 0
3: end for
4: p_a ← f_u(a) {acceso al índice de perfiles de usuario}
5: for all (i, r_ai) ∈ p_a do
6:    p_i ← f_i(i) {acceso al índice invertido de perfiles de usuario}
7:    for all (u, r_ui) ∈ p_i do
8:       Similitud[u] ← Similitud[u] + r_ai * r_ui
9:    end for
10: end for
11: for all u ∈ U do
12:    w ← factor de normalización para el usuario u
13:    Similitud[u] ← Similitud[u]/w
14: end for
15: ordenar Similitud
16: return k usuarios con similitud más elevada
```

En la práctica, esta es sólo una de las posibles formas de calcular el vecindario a partir de los dos índices, y se corresponde con la técnica *item-at-a-time*, que no es más que el equivalente de la aproximación *term-at-a-time* (TAAT) usada en RI [Turtle y Flood, 1995]. Alternativamente, se puede usar una técnica *user-at-a-time*, que sería el equivalente a la técnica *document-at-a-time* (DAAT) [Turtle y Flood, 1995]. En ese caso, la similitud se calcularía usuario a usuario, accediendo al índice invertido para cada producto en paralelo. Además, en RI se han propuesto varias optimizaciones a estas técnicas básicas [Ding y Suel, 2011], muchas de las cuales podrían ser adaptadas para su uso en recomendación.

Una vez calculado el vecindario, los productos a recomendar se seleccionan entre aquellos que han recibido mayor puntuación por parte de los vecinos. A cada producto se le asigna un peso ω, resultado del cálculo de la similitud entre el vector que representa al vecindario y los vectores que representan a cada uno de los productos valorados por los vecinos. Los N productos con mayor peso serán recomendados. Para el cálculo del peso se pueden utilizar distintas técnicas, representando el Algoritmo 2 el cálculo en caso de emplearse la medida basada en la media ponderada por similitud, presentada en la Ecuación 5.1.

El proceso es similar al usado en el cálculo del vecindario. En este caso, se sigue un proceso *user-at-a-time*, donde el peso de cada producto se calcula vecino a vecino. En primer lugar, se obtiene la lista de vecinos. Para cada vecino, se utiliza el índice de perfiles de usuario para obtener la puntuación otorgada a cada producto y así actualizar

Algorithm 2 Recomendación para un usuario $a \in \mathcal{U}$

```
1: for all i ∈ I do
2:    Numerador[i] ← 0
3:    Denominador[i] ← 0
4: end for
5: N(a) ← N(a) {acceso al vecindario del usuario actual}
6: for all (u, s_ua) ∈ N(a) do
7:    p_u ← f_u(u) {acceso al índice de perfiles de usuario}
8:    for all (i, r_ui) ∈ p_u do
9:       Numerador[i] ← Numerador[i] + r_ui * s_au
10:      Denominador[i] ← Denominador[i] + s_au
11:   end for
12: end for
13: for all i ∈ I do
14:   Numerador[i] ← Numerador[i]/Denominador[i]
15: end for
16: ordenar Numerador
17: return N productos con mayor Numerador
```

su peso con la contribución de dicho vecino. Una vez que todos los vecinos han sido procesados, se escogen los N productos con mayor peso.

7.3. Preselección de vecinos

La selección del vecindario es una de las fases más costosas en los algoritmos kNN. Requiere calcular la similitud entre el usuario actual y el resto de usuarios, lo que a su vez requiere comparar entre sí las distintas puntuaciones otorgadas por cada uno de ellos. En las aplicaciones actuales, que cuentan con millones de usuarios y productos, esta fase puede llevar varios segundos y convertirse en un cuello de botella que impida el cálculo de recomendaciones en tiempo real.

Una solución sencilla pero muy eficaz es llevar a cabo la selección de vecinos en un proceso *offline*, de forma parecida a la fase de entrenamiento en algoritmos basados en modelo. De hecho, es la aproximación seguida habitualmente en sistemas comerciales [Linden et al., 2003].

Sin embargo, esta optimización podría tener un impacto negativo en la calidad de las recomendaciones, ya que los vecinos no estarían basados en el perfil actual del usuario. A medida que el usuario puntúa nuevos productos, su perfil contiene más información sobre el mismo, lo que permitiría al sistema escoger mejores vecinos. Además, si los gustos o intereses del usuario cambiasen con una frecuencia lo suficientemente rápida,

los vecinos seleccionados podrían no reflejar los gustos o preferencias actuales de los usuarios.

En esta sección analizamos la técnica de preselección de vecinos, estudiando tanto las mejores en rendimiento como el impacto en precisión.

7.3.1. Algoritmos kNN con preselección de vecinos

El funcionamiento de un algoritmo kNN con preselección de vecinos sería muy similar al del correspondiente algoritmo sin preselección, y podría implementarse fácilmente mediante el uso de índices, tal y como se describía en la Sección 7.2.4.

En una implementación sin preselección, cada recomendación (Algoritmo 2) requiere la correspondiente ejecución del algoritmo de cálculo de vecinos (Algoritmo 1). Sin embargo, con la preselección de vecinos cada uno de ellos se ejecuta en momentos diferentes.

El algoritmo de cálculo de vecinos se ejecutaría en un proceso *offline*, en el que se seleccionarían k vecinos para cada usuario del sistema. Estos se almacenarían en una estructura en forma de índice, el *índice de vecinos*, con una estructura similar a la de los índices de puntuaciones descritos en la Sección 7.2.3 y donde cada usuario estaría asociado a la lista de vecinos, junto a la similitud con cada uno de ellos.

Por otra parte, el algoritmo de recomendación se ejecutaría cada vez que un usuario solicitase una recomendación, con la única diferencia de que en lugar de calcular el vecindario en ese momento, este se obtendría consultando el índice de vecinos. Exceptuando esta diferencia, el funcionamiento del algoritmo sería igual que el presentado en la sección anterior, e implicaría el acceso al índice de perfiles de usuario. Al contrario que el índice de vecinos, este último sí está continuamente actualizado[1]. Esto permite que la fase de recomendación sí tenga en cuenta las ultimas puntuaciones de los usuarios, lo que supone una diferencia importante respecto a los algoritmos basados en modelo, en los que suele ser necesario volver a entrenar el modelo para considerar la nueva información.

Esta es precisamente una de las principales ventajas de esta técnica respecto a algoritmos basados en modelo. Por ejemplo, esto implica que productos introducidos recientemente en el sistema pueden ser recomendados aunque no hayan sido tenidos en cuenta durante la fase de selección del vecindario. En la práctica, esto significa que la frecuencia de actualización del vecindario puede ser menor que la necesaria en un algoritmo basado en modelo. Además, puede realizarse de forma incremental o incluso

[1] Posiblemente no en tiempo real, pero sí con una frecuencia mucho mayor que la actualización del vecindario.

utilizar frecuencias de actualización diferentes en función de la actividad del usuario. Por ejemplo, usuarios que interactúan habitualmente con el sistema pueden necesitar una actualización frecuente del vecindario, mientras que usuarios que sólo lo hacen de forma esporádica no necesitan una actualización tan frecuente. Este tipo de actualizaciones no es generalmente posible en un algoritmo basado en modelo.

7.3.2. Experimentos y resultados

Para evaluar el posible impacto negativo, en cuanto a la precisión de la recomendación, de usar la preselección de vecinos, hemos estudiado la evolución de los resultados según el tiempo que transcurre entre la preselección y la recomendación. Para ello, hemos aprovechado que en el conjunto de datos Netflix está disponible la fecha en que se realizó una puntuación. En concreto, las aproximadamente 100 millones de puntuaciones presentes en dicho conjunto de datos han sido realizadas entre octubre de 1998 y diciembre de 2005.

Nuestro experimento ha consistido en evaluar el comportamiento del algoritmo a lo largo de los diez primeros meses de 2005, considerando dos situaciones diferentes. En la primera de ellas, hemos hecho uso de la preselección de vecinos. En particular, nos hemos quedado con las puntuaciones anteriores al 1 de enero de 2005, y a partir de las mismas hemos calculado el vecindario para cada uno de los 1.000 usuarios elegidos aleatoriamente como parte del conjunto de evaluación. Es decir, hemos simulado una preselección del vecindario efectuada el primer día del año. Luego, hemos realizado una recomendación al final de cada uno de los meses, teniendo en cuenta las puntuaciones otorgadas hasta dicho mes, pero sin volver a calcular el vecindario.

En la segunda de las situaciones, sin embargo, hemos vuelto a calcular el vecindario justo antes de la recomendación. Es decir, en la primera de las situaciones las puntuaciones adicionales sólo se utilizan a la hora de realizar la recomendación, mientras que en la segunda también se utilizan para el cálculo del vecindario. Estamos comparando, por tanto, una situación en la que el vecindario se preselecciona con una en la que el vecindario se calcula para cada recomendación. Nuestra hipótesis es que la preselección del vecindario supone una gran ventaja en términos de eficiencia sin empeorar demasiado la calidad de las recomendaciones.

En primer lugar, hemos estudiado las mejoras de rendimiento debidas al uso de la preselección de vecinos. Para ello, hemos medido el tiempo necesario para cada recomendación, empleando un PC con procesador Intel Pentium 4 a $3,20$ GHz y 256 MiB de RAM. Los resultados se muestran en la Figura 7.3. Como se puede observar, el uso de preselección reduce significativamente los tiempos de recomendación. Sin

preselección, una recomendación tarda varios segundos o incluso minutos, debido al coste necesario para el cálculo del vecindario, mientras con la preselección el tiempo se reduce a unas pocas décimas. De media, la mejora es de dos órdenes de magnitud, pero lo más importante es que marca la diferencia entre un algoritmo que funciona en tiempo real y otro que sólo sería útil para dominios con muy pocos usuarios o que sólo necesitasen recomendaciones *offline*.

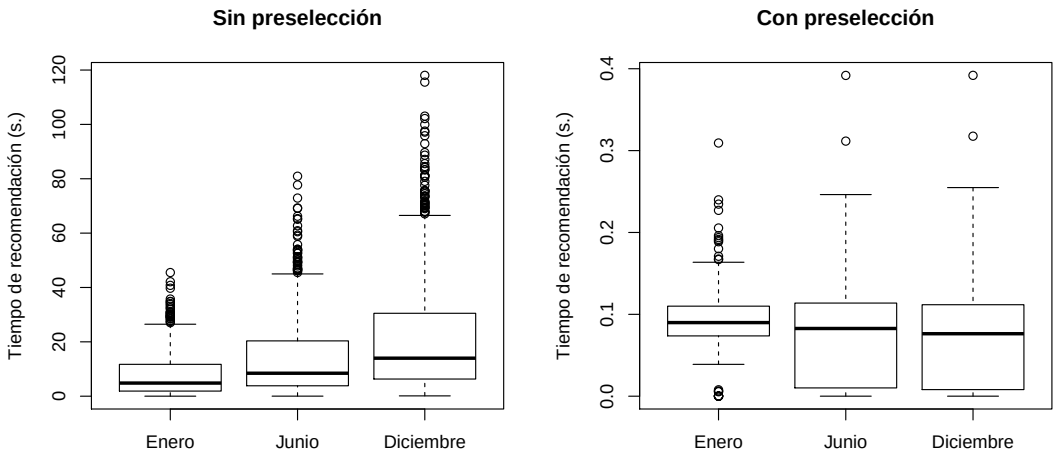


Figura 7.3: Tiempo de recomendación (en segundos), con y sin preselección de vecinos.

Además, al aumentar el número de puntuaciones con el uso del sistema, el tiempo necesario para la fase de recomendación se mantiene más o menos constante (pues sólo tiene en cuenta las puntuaciones de los vecinos), mientras el necesario para la fase de selección del vecindario sí aumenta significativamente. Por tanto, la preselección permite mantener unos tiempos de recomendación estables a pesar de que el número de puntuaciones vaya aumentando.

Para evaluar el posible descenso de la calidad de las recomendaciones al usar la preselección de vecinos, hemos medido la precisión y *recall* con listas de recomendación de 5 productos. Tal y como se ve en la Tabla 7.1, en los primeros meses no se observa un descenso de la precisión especialmente significativo debido al uso de preselección. Al contrario que muchas de las técnicas basadas en modelo, las nuevas puntuaciones sí se usan en la fase de recomendación, por lo que se tienen en cuenta nuevos productos y opiniones, con lo que se consiguen buenos resultados pese a usar un vecindario calculado muchos días antes. En los últimos meses, sin embargo, el usar un vecindario desactualizado ya comienza a ser un problema y la precisión es inferior a la del algoritmo sin

preselección[1].

	P@5		R@5	
	Con	Sin	Con	Sin
Ene	0,042	0,042	0,0053	0,0053
Feb	0,033	0,038	0,0035	0,0051
Mar	0,028	0,032	0,0040	0,0045
Abr	0,033	0,046	0,0052	0,0097
May	0,026	0,033	0,0053	0,0058
Jun	0,021	0,031	0,0043	0,0058
Jul	0,014	0,028	0,0038	0,0051
Ago	0,009	0,014	0,0089	0,0048
Sep	0,005	0,018	0,0075	0,0094
Oct	0,007	0.013	0,0017	0,0054

Tabla 7.1: P@5 y R@5 con y sin preselección de vecinos.

Por supuesto, la frecuencia con la que es necesario proceder a la actualización del vecindario será dependiente del dominio. En la recomendación de películas en que hemos evaluado esta técnica, el vecindario sigue siendo útil después de varios meses, pero en otros dominios este tiempo podría ser menor. De igual forma, para ciertos usuarios puede ser necesario recalcular el vecindario con mayor frecuencia. Por ejemplo, usuarios que empiezan a usar el sistema y de los cuales todavía no se tiene demasiada información. En cualquier caso, la técnica de preselección del vecindario tiene importantes ventajas y muy pocos inconvenientes.

[1]Obsérvese que el descenso de precisión en los últimos meses, con y sin preselección, es debido a que existen pocos productos en el conjunto de evaluación, es decir, es una limitación de la metodología de evaluación y no del algoritmo, tal y como se discute en la Sección 5.4.2.

Capítulo 8

Técnicas de compresión de la matriz de puntuaciones

En las aplicaciones de hoy en día, es bastante común que los sistemas de recomendación deban manejar volúmenes de información muy elevados, del orden de los millones de usuarios o productos, y varios miles de millones de puntuaciones. En dichos casos, el tamaño de la matriz de puntuaciones puede llegar a convertirse en un problema importante, tanto por el coste asociado a almacenar la información como por el tiempo necesario para procesarla.

En ámbitos como el de recuperación de información, es habitual el uso de técnicas de compresión para disminuir el tamaño del índice [Witten et al., 1999, cap. 5], lo que permite reducir los costes relacionados con el almacenamiento, y, al mismo tiempo, ayudan a mejorar el rendimiento del sistema, ya que un índice más pequeño reduce el volumen de datos a ser transferidos, y repercute en un aprovechamiento más eficiente de la memoria *cache*.

En este capítulo se estudia el uso de técnicas de compresión de la matriz de puntuaciones, y cómo técnicas usadas habitualmente en recuperación de información pueden usarse en filtrado colaborativo. En la Sección 8.1 se presentan los beneficios de la compresión, se introducen los fundamentos de la misma y se hace un pequeño repaso de su uso en RI. Posteriormente, en la Sección 8.2, se demuestra como este tipo de técnicas pueden ser aplicadas a filtrado colaborativo, y como esto redunda en una reducción significativa tanto del tamaño de la matriz de puntuaciones como del tiempo necesario para calcular la recomendación. Finalmente, en la Sección 8.3, se introducen las técnicas de reasignación de identificadores, y se presenta una nueva técnica que funciona especialmente bien en sistemas de filtrado colaborativo.

8.1. Introducción a las técnicas de compresión

El propósito de las técnicas de compresión es conseguir representar la misma información utilizando menos espacio, lo que tiene importantes aplicaciones en multitud de ámbitos. En particular, los sistemas de recuperación de información han utilizado este tipo de técnicas durante años, siendo hoy en día un elemento esencial para reducir los costes de almacenamiento y posibilitar el procesamiento eficiente de consultas, especialmente en dominios con gran cantidad de datos como por ejemplo la búsqueda en la Web.

En esta sección se introducen los fundamentos de este tipo de técnicas, sus beneficios y su uso en la compresión de índices invertidos en RI. En la próxima sección se estudiará cómo aplicarlas en la compresión de la matriz de puntuaciones en filtrado colaborativo.

8.1.1. Beneficios y fundamentos de las técnicas de compresión

En RI, la compresión del índice invertido es una técnica ampliamente utilizada, y fundamental en la construcción de sistemas capaces de manejar *terabytes* de información. En particular, las técnicas de compresión aportan los siguientes beneficios:

- **Reducción de los costes de almacenamiento**. Dado que la compresión reduce el tamaño del índice, es obvio que los costes necesarios para el almacenamiento se verán también reducidos. En RI es habitual conseguir ratios de compresión de 1:4 [Witten et al., 1999, cap. 5].

- **Mejor aprovechamiento de la memoria**. Los ordenadores actuales cuentan con una jerarquía de varios niveles de memoria, cada uno de ellos caracterizado por un coste, velocidad de acceso y capacidad diferentes. Los niveles más rápidos, como los registros o la *cache* de la CPU, son también los más caros, y cuentan con una capacidad muy limitada. Su uso se reduce al almacenamiento temporal de la información que está siendo procesada por la CPU. En el otro extremo se encuentran los discos magnéticos, que ofrecen una gran capacidad de almacenamiento a bajo precio, pero cuyo tiempo de acceso es varios órdenes de magnitud más lento. Por ejemplo, mientras un acceso a la *cache* L1 de un procesador moderno no tarda más que unos pocos nanosegundos, el tiempo de acceso medio a un disco magnético es de varios milisegundos. La capacidad de almacenamiento de estos últimos, sin embargo, es muy superior, y además con un coste sensiblemente más económico, lo que permite su uso para el almacenamiento masivo de información. En un nivel intermedio estaría la memoria RAM, mucho más rápida

que un disco magnético y más económica que la circuitería de la CPU, y además con mayor capacidad que esta última. Los sistemas operativos modernos intentan mantener almacenada en la memoria RAM aquella información a la que el sistema accede con mayor frecuencia, para así evitar tener que acceder a los lentos discos magnéticos. Este sistema, conocido como *cache* de disco, ayuda a incrementar la eficiencia del sistema. Sin embargo, dado que la capacidad de los discos es mucho mayor que el tamaño de la memoria RAM, no es posible almacenar en la *cache* toda la información necesaria. El sistema operativo monitoriza el uso de la información almacenada en la *cache*, reemplazando datos antiguos o a los que se accede ocasionalmente con aquellos que son consultados frecuentemente, intentando maximizar el número de accesos que se resuelven directamente sin necesidad de acceder al disco. Las técnicas de compresión, al reducir el espacio necesario para almacenar la información, permiten almacenar más datos en la *cache* y, por tanto, ayudan a que se aproveche mejor este mecanismo, lo que mejora significativamente los tiempos de acceso.

- **Transferencias de datos más rápidas**. Además, al ocupar menos espacio, la información comprimida puede ser transferida más rápidamente por ejemplo entre el disco y la memoria RAM, lo que también ayuda a reducir los tiempos de acceso. Por supuesto, el algoritmo de descompresión debe ser lo suficientemente rápido para poder aprovechar esta ventaja, de ahí que en la práctica se utilicen técnicas sencillas con tiempos de descompresión muy reducidos.

Las técnicas de compresión aprovechan las características de la distribución de los elementos a comprimir (o símbolos) para conseguir almacenarlos de forma eficiente. En cada dominio, el conjunto de símbolos válidos, o alfabeto, será diferente.

Por ejemplo, consideremos la estructura conocida como índice invertido que se usa habitualmente en RI para almacenar la relación entre términos y documentos [Manning et al., 2008, cap. 1]. Dicha estructura almacena, para cada término, una lista con los identificadores de los documentos en los cuales aparece. En este caso, el alfabeto estaría formado por todos los identificadores posibles, y cada uno de ellos sería un símbolo. Si usásemos un código de longitud fija (es decir, todos los símbolos se representan con el mismo número de bits), necesitaríamos $\lceil \log_2 N \rceil$ bits para representar un símbolo, con N el número de documentos.

Por supuesto, no todos los documentos aparecen en el índice con igual probabilidad. Si los identificadores que aparecen con mayor frecuencia se codificasen con menos bits, el número total de bits necesario para almacenar el índice sería menor. Las técnicas

de compresión hacen uso de códigos de longitud variable, en los que el número de bits dedicado a cada símbolo depende de la frecuencia con la que aparece, y por tanto se consiguen mayores ratios de compresión.

Si asumimos que los símbolos son independientes, el número medio de bits necesario para codificar un símbolo estará acotado inferiormente por la entropía [Shannon, 1948] de la distribución de probabilidad. La entropía, que representa la cantidad media de información por símbolo que contiene un determinado alfabeto, se denota con la letra H y se calcula según la siguiente ecuación:

$$H = \sum_{s} -p(s) \cdot \log_2 p(s) \tag{8.1}$$

Donde s representaría un símbolo del alfabeto (en nuestro ejemplo, un identificador de un documento), y $p(s)$ la probabilidad de ocurrencia de dicho símbolo, es decir, la probabilidad de que un término esté presente en dicho documento. Por tanto, la entropía nos da una medida del grado de compresión que puede llegar a obtenerse. Por otra parte, en dominios como la recuperación de información los símbolos no son independientes, por lo que se pueden desarrollar códigos que codifican varios símbolos consecutivos y alcanzan ratios de compresión mayores.

En la práctica, la distribución de probabilidad de la información a comprimir no se conoce a priori. Las técnicas de compresión utilizan un determinado modelo para estimar esta distribución y, por tanto, determinar el número de bits con el que se codificará cada símbolo. Cuanto más próximo esté este modelo a la distribución de probabilidad real, mejores ratios de compresión se podrán alcanzar.

8.1.2. Técnicas de compresión de índices

En el caso de los índices invertidos, se obtienen muy buenos resultados si, en lugar de codificar los identificadores de los documentos, se codifican las diferencias entre ellos [Witten et al., 1999, cap. 5]. Es decir, si un término aparece en los documentos con identificadores $(3, 4, 7, 10, 12)$, dicha lista se almacenaría como $(3, 1, 3, 3, 2)$. La distribución de estas diferencias ha sido ampliamente estudiada, y existen diferentes modelos específicamente diseñados que obtienen muy buenos resultados pese a estar basados en códigos sencillos. Podemos clasificar estos métodos en dos tipos [Witten et al., 1999, cap. 5]: métodos globales, que asumen una misma distribución de probabilidad para todos los términos; y métodos locales, donde el modelo depende de un conjunto de parámetros calculados de forma independiente para cada uno de ellos. Por ejemplo, se podría considerar como parámetro la frecuencia concreta de cada término. Estos

parámetros se almacenarían como parte del índice. Por supuesto, los métodos globales también pueden depender de ciertos parámetros, pero en ese caso su valor sería el mismo para todos los términos.

Los métodos globales no parametrizados están basados en la observación de que la distribución de las diferencias en una lista de documentos depende de la frecuencia del término en cuestión. En caso de términos frecuentes, las diferencias son pequeñas, lo cual es obvio pues, si el término es frecuente, lo será porque aparece en muchos documentos. Sin embargo, para términos poco frecuentes las diferencias son mucho mayores. Por tanto, se utilizan códigos de longitud variable, en donde las diferencias pequeñas se representan con menos bits. Por ejemplo, los códigos γ y δ propuestos por Elias [1975] se comportan mucho mejor que un código de longitud fija en estos casos. El problema de este tipo de códigos es que asumen una distribución de probabilidad fija, independientemente de las características concretas de la colección de documentos a indexar.

Una alternativa mucho mejor es considerar modelos que dependen de ciertas características de la información a comprimir. Por ejemplo, los códigos Golomb [Golomb, 1966] presentan muy buenos resultados. Asumen que las diferencias siguen una distribución geométrica correspondiente a ensayos de Bernoully con una cierta probabilidad p de que un término aparezca en un documento. Esta probabilidad es un parámetro del modelo, que se calculará a partir de los datos a indexar. A pesar de que esta técnica parte de premisas claramente equivocadas (como por ejemplo, la independencia entre pares término-documento), en la práctica obtiene muy buenos resultados.

Finalmente, los métodos locales mejoran esta idea, al calcular los parámetros del modelo de forma independiente para cada término. Por ejemplo, el modelo de Bernoully local también utiliza un código Golomb [Witten et al., 1999, cap. 5], pero la probabilidad p será diferente para cada término. De esta forma, los códigos se ajustan según la frecuencia de cada término, con lo que se consiguen mayores ratios de compresión, a costa de tener que almacenar un parámetro diferente para cada lista. Otro método que también funciona muy bien es el modelo *Skewed Golomb* [Witten et al., 1999, cap. 5], el cual es similar al anterior pero utiliza códigos más pequeños para las diferencias pequeñas, sin penalizar demasiado las diferencias grandes.

8.2. Técnicas de compresión para filtrado colaborativo

A pesar de ser una técnica muy extendida en ámbitos como la recuperación de información, en la literatura apenas pueden encontrarse trabajos dedicados a estudiar los

beneficios de la compresión en el ámbito del filtrado colaborativo. Cöster y Svensson [2002] mencionan el uso de códigos γ, δ y Golomb para la compresión de los identificadores de productos en la matriz de puntuaciones, pero sin analizar qué beneficios aportan o cuál de ellos presenta mejores tasas de compresión. Además, ni siquiera utilizan estas técnicas en los experimentos que llevan a cabo.

Se puede decir, por tanto, que hasta la fecha no existe ningún trabajo dedicado a estudiar hasta qué punto los sistemas de recomendación se pueden beneficiar de este tipo de técnicas de compresión. En este capítulo se lleva a cabo dicho estudio, centrándonos en algoritmos de filtrado colaborativo y, por tanto, en la compresión de la matriz de puntuaciones. En particular, en una representación de la matriz basada en índices, tal y como ha sido presentada en la Sección 7.2.3.

Por supuesto, el uso de técnicas de compresión en filtrado colaborativo exige considerar las particularidades de este nuevo dominio. En particular, en comparación con el uso de compresión en RI, se deben abordar los siguientes problemas:

- ¿Cómo comprimir los identificadores de usuarios y productos? Se debe estudiar si la aproximación utilizada en RI, es decir, la codificación de las diferencias entre identificadores, es adecuada para la compresión de la matriz de puntuaciones, o por contra, debemos buscar técnicas alternativas. Además, se debe determinar qué códigos son los más adecuados en este dominio, lo que está íntimamente relacionado con la distribución de las puntuaciones entre usuarios y productos.

- ¿Cómo comprimir las puntuaciones? En los sistemas de recomendación, los índices almacenan también las puntuaciones de los usuarios, las cuales generalmente pertenecen a un conjunto finito distribuido de forma no uniforme (normalmente las puntuaciones altas son más frecuentes), y que por tanto puede ser comprimido de forma eficiente.

- Finalmente, un sistema de recomendación maneja dos índices, el índice de perfiles de usuario y el índice invertido de perfiles de usuario, por lo que habría que considerar la técnica de compresión utilizada en cada uno de ellos. Al ser la distribución de las puntuaciones diferente en cada caso, usar un método diferente en cada índice puede ser un buena opción. Además, si se utiliza la técnica de preselección de vecinos, deberemos considerar también la compresión de este índice adicional.

Estas cuestiones serán abordadas a lo largo de esta sección. Para la realización de este estudio, hemos hecho uso de dos conjuntos de datos pertenecientes al ámbito de la

	Número de puntuaciones	Usuarios	Productos
Netflix	100.480.507	480.189	17.770
Movielens 10M	10.000.054	69.878	10.677
LibimSeTi	17.359.346	135.359	168.791

Tabla 8.1: Características de los conjuntos de datos empleados en el estudio de las técnicas de compresión.

recomendación de películas (Netflix [Bennett y Lanning, 2007] y Movielens 10M), y otro que contiene datos reales pertenecientes a un servicio de citas por Internet (LibimSeTi [Brozovsky y Petricek, 2007]). Como puede verse en la Tabla 8.1, los dos primeros contienen muchos más usuarios que productos, mientras LibimSeTi tiene un número similar de ambos.

8.2.1. Compresión de los identificadores de usuarios y productos

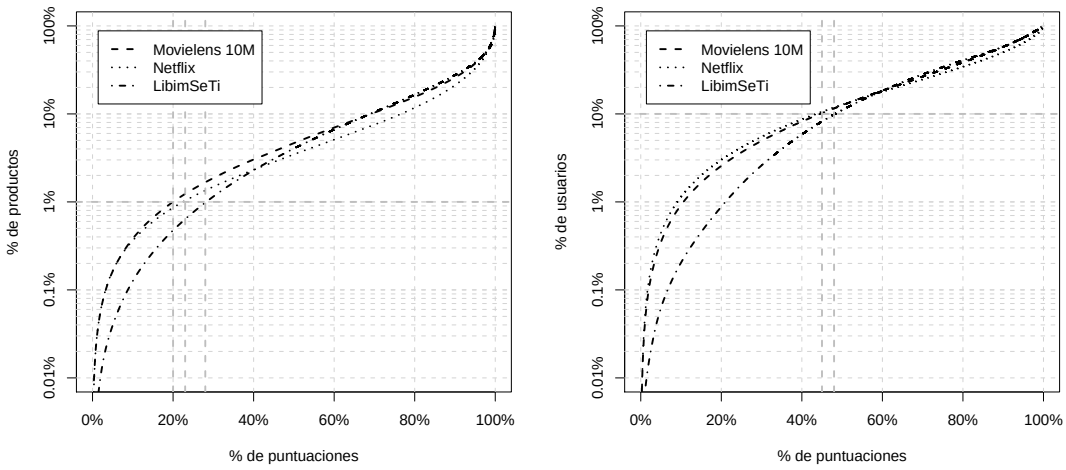


Figura 8.1: Distribución de las puntuaciones según el porcentaje de productos (izquierda) y de usuarios (derecha).

La compresión eficiente de los identificadores requiere conocer la distribución de los mismos y, en particular, comprobar si la técnica de compresión de diferencias entre identificadores, popular en RI, ofrece buenos resultados en este caso.

Por tanto, en primer lugar analizamos la distribución de puntuaciones para los tres conjuntos de datos utilizados. Tal y como se observa en la Figura 8.1, la distribución de

puntuaciones a lo largo de los diferentes productos presenta una gran variabilidad. Por un lado, existe un 1 % de los productos que concentra más del 20 % de las puntuaciones, mientras por otra parte tenemos en torno a un 60 % de productos que apenas recibe un 5 %. Podemos por tanto clasificar los productos en tres subconjuntos:

- Productos populares. Correspondería a ese 1 % de productos que son puntuados por muchos usuarios. Las razones de que dichos productos reciban tantas puntuaciones responde a una serie de razones generalmente externas al propio sistema de recomendación. Por ejemplo, un producto podría alcanzar esta popularidad gracias a su calidad, utilidad, o buen precio en relación a un producto de la competencia; pero también la podría alcanzar por otros motivos, como podría ser una campaña publicitaria agresiva, una buena opinión por parte de un crítico reconocido, etc. En general estos productos van a ser bien valorados, aunque esto no siempre se cumple y ciertos productos reciben una mala puntuación a pesar de su popularidad.

- Productos corrientes. Correspondería a aquellos en la zona intermedia de la Figura 8.1, es decir, productos que reciben un número razonable de puntuaciones, pero sin llegar a ser extremadamente populares. Digamos que son productos que muchos usuarios conocen, pero por diversos motivos no han alcanzado la popularidad de los anteriores.

- Productos desconocidos. Estos se corresponden a ese 60 % de productos que apenas recibe un 5 % de las puntuaciones. Aquí se englobarían productos específicos, que sólo gustan o son de utilidad para un conjunto o comunidad concreta de usuarios (por ejemplo, libros de carácter técnico), productos que en su momento no han tenido una buena acogida, productos que no son asequibles al público en general, etc. En ámbitos como el comercio electrónico, este tipo de productos corresponde a lo que se conoce como *long tail* [Anderson, 2006]. En general, son productos de especial interés desde el punto de vista del sistema de recomendación, ya que este grupo incluye productos que pueden ser de gran utilidad para el usuario, pero que este desconoce.

En cuanto a los usuarios, el comportamiento es similar, sólo que las diferencias son menos pronunciadas. Por un lado, se observa que un 10 % de los usuarios concentra cerca de la mitad de las puntuaciones. Son los usuarios más activos, aquellos que usan el sistema con frecuencia. Por otra parte, tenemos usuarios que lo usan mucho menos, y por tanto poseen menos puntuaciones. Nuevamente, las causas para este comportamiento

	Netflix	Movielens 10M	LibimSeTi
Longitud fija	15,00	14,00	18,00
γ	9,57	5,60	11,24
δ	9,10	5,68	10,07
Global Golomb	7,84	6,77	11,02
Local Golomb	6,89	6,28	10,12
Skewed Golomb	7,19	5,17	8,95

Tabla 8.2: Número medio de *bits* por identificador en el índice de perfiles de usuario.

son ajenas al sistema de recomendación. Por ejemplo, un usuario podría ser activo porque tiene un especial interés o afición por los productos que ofrece el sistema, porque adquiere productos frecuentemente al tener un gran poder adquisitivo, etc. Por otra parte, que un usuario tenga pocas puntuaciones podría deberse a su descontento con el sistema o comercio, a que sólo se ha registrado para hacer una compra puntual (por ejemplo, para hacer un regalo) pero no le interesan particularmente este tipo de productos, o bien a que, debido a sus condiciones económicas, no puede adquirir productos con tanta frecuencia.

En cualquier caso, e independientemente de los motivos, tanto productos como usuarios muestran una diferencia entre aquellos que acumulan un gran número de puntuaciones, y aquellos cuyas puntuaciones son puramente testimoniales. Por tanto, parece razonable utilizar una técnica similar a la empleada en RI, es decir, en lugar de almacenar los identificadores en el índice, se almacenarían las diferencias entre ellos. Obviamente, los productos populares (y los usuarios activos), los cuales son responsables de la mayor parte de datos almacenados en el índice, presentarían diferencias pequeñas. Por tanto, sería idóneo utilizar una técnica de compresión que representase con menos bits los números más pequeños y frecuentes, reduciéndose así el tamaño global del índice.

En las Tablas 8.2 y 8.3 se muestra la cantidad media de bits necesaria para codificar un identificador (es decir, una diferencia), para el índice de perfiles de usuario y el índice invertido de perfiles de usuario, respectivamente. Como cabría esperar teniendo en cuenta la discusión anterior, el beneficio de la compresión es muy grande, suponiendo una reducción del 50 % del tamaño del índice en la mayoría de los casos, e incluso cercana al 75 % en algunas situaciones.

En general, los métodos locales obtienen tasas de compresión más elevadas, lo que también es de esperar dado que la distribución de probabilidad asumida por estos está parametrizada para cada lista, y por tanto es más cercana a la realidad. En cuanto al

	Netflix	Movielens 10M	LibimSeTi
Binary	19,00	17,00	18,00
γ	5,88	7,33	14,60
δ	5,93	7,24	12,73
Global Golomb	7,63	7,52	11,51
Local Golomb	5,38	5,55	9,52
Skewed Golomb	5,05	5,81	9,91

Tabla 8.3: Número medio de *bits* por identificador en el índice invertido de perfiles de usuario.

rendimiento de los métodos globales, debemos destacar el buen comportamiento de los códigos γ y δ en ciertas ocasiones, lo cual podría sorprender en un principio. Por ejemplo, en el índice de perfiles de usuario para Movielens 10M, presentan tasas de compresión muy próximas, e incluso mejores, a los métodos locales. Sin embargo, esto se debe a la distribución de puntuaciones que presenta este conjunto de datos en particular. Tal y como se observa en la Figura 8.2 (gráfica de la izquierda), en Movielens 10M los elementos más populares son aquellos que tienen identificadores más pequeños. Por tanto, los códigos γ y δ son una buena alternativa. De hecho, en la Figura 8.3 (gráfica de la izquierda) se puede ver como la distribución de probabilidad asumida por estos códigos se asemeja bastante a la distribución real de las puntuaciones en dicho conjunto de datos.

Sin embargo, esta es una situación que se produce en un conjunto de datos destinado a investigación como puede ser Movielens 10M, pero es poco frecuente en el mundo real. En la práctica, la situación más habitual es que no haya relación entre el identificador del producto y el número de puntuaciones que ha recibido. Es decir, se parecerán más a la distribución de puntuaciones en Netflix (gráfica de la derecha en la Figura 8.2). En dichos casos, la distribución que asumen los códigos globales como γ y δ es muy diferente a la distribución real (Figura 8.3, derecha).

En la Sección 8.3 presentaremos una técnica que se aprovecha precisamente de este comportamiento para así incrementar significativamente las tasas de compresión alcanzables en aplicaciones reales.

8.2.2. Compresión de las puntuaciones

En general, las puntuaciones son números reales, por lo que deberían ser almacenadas como números en coma flotante. Sin embargo, en la práctica la mayoría de aplicaciones consideran únicamente un conjunto finito de posibles puntuaciones. Por

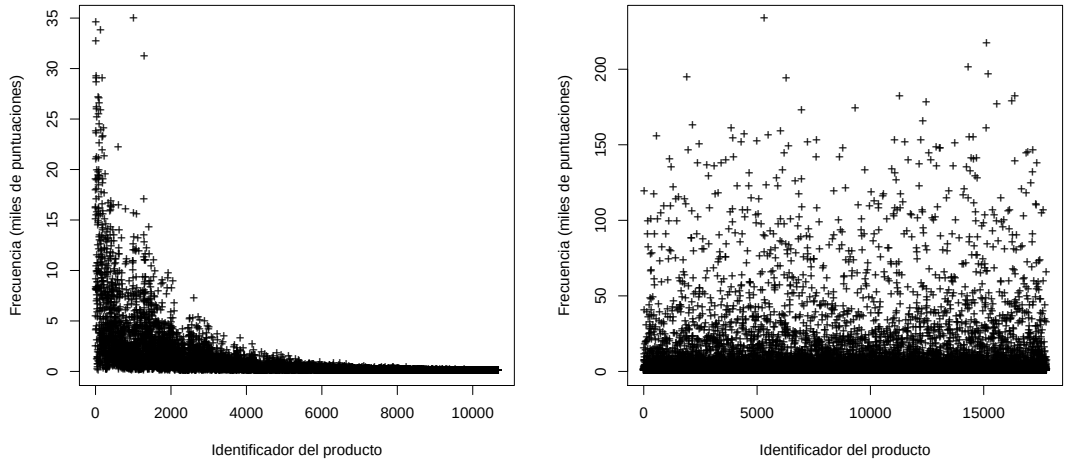


Figura 8.2: Número de puntuaciones según el identificador del producto, en Movielens 10M (izquierda) y Netflix (derecha).

tanto, en lugar de almacenar el valor real de la puntuación, se podría almacenar el índice dentro del conjunto. Por ejemplo, si las posibles puntuaciones en un sistema fuesen $\{1, 1.5, 2, 2.5, 3\}$, la puntuación 1 se representaría con el índice 1, la puntuación 1,5 con el 2, y así sucesivamente. Dado que el número de posibles puntuaciones es generalmente bajo, un sencillo código de longitud fija apenas necesitaría unos pocos bits para codificar una puntuación. Por ejemplo, en el caso de Netflix, que considera 5 puntuaciones diferentes, sólo serían necesarios 3 bits.

A pesar de todo, esta aproximación implica considerar que las puntuaciones están uniformemente distribuidas, lo cual no suele ser el caso. Tal y como se observa en la Figura 8.4, la distribución de puntuaciones es diferente en cada dominio. Por ejemplo, en la recomendación de películas las puntuaciones positivas son mucho más frecuentes que las negativas. El uso de un código de longitud variable podría aprovechar esto y obtener mejores ratios de compresión, lo cual podría significar un ahorro de espacio considerable en matrices de gran tamaño. Por ejemplo, podríamos usar un código Huffman [Huffman, 1952] siempre y cuando conociésemos la distribución de puntuaciones antes de indexar, como en el caso de indexación en dos pasos. Esto reduciría el número de bits necesario para almacenar una puntuación a 2,15 en Netflix, 2,79 en Movielens 10M y 3,24 en LibimSeTi, lo que supone una mejora del 28,44 %, 30,23 % y 18,99 %, respectivamente, en relación al tamaño necesario en caso de usar una codificación de longitud fija.

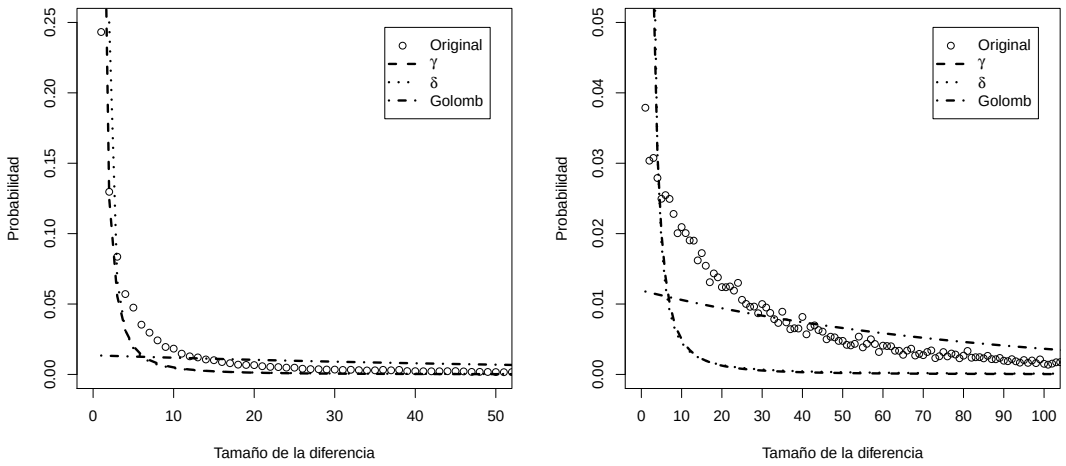


Figura 8.3: Distribución de probabilidad de las diferencias entre identificadores, en Movielens 10M (izquierda) y Netflix (derecha), comparada con la distribución de probabilidad asumida por distintos métodos de codificación globales.

8.2.3. Mejoras en el rendimiento gracias a la compresión

Junto a la reducción del espacio necesario para almacenar la matriz, el otro gran beneficio del uso de la compresión es la mejora del rendimiento que, tal y como se ha comentado, se consigue gracias a un mejor aprovechamiento de la memoria *cache* y a la disminución de la cantidad de datos que es necesario transferir desde el disco.

Para estudiar las mejoras de rendimiento, hemos hecho uso del conjunto de datos Netflix, por ser el de mayor tamaño de los que tenemos acceso. Sin embargo, aún así es muy pequeño en comparación con el volumen de datos existente en muchas aplicaciones reales. En una máquina actual con varios gigabytes de RAM, la matriz de puntuaciones puede almacenarse completamente en memoria, restando importancia a los beneficios de la compresión. Por tanto, para evaluar la mejora en rendimiento obtenida gracias a la compresión, en nuestros experimentos hemos utilizado un PC relativamente antiguo, con una CPU Intel Pentium 4 a 3,20 GHz y solamente 256 MiB de memoria RAM. De esta forma, la relación entre el tamaño de la matriz y las capacidades hardware de nuestro entorno de pruebas son más semejantes a las de un entorno real. Esta es una aproximación utilizada habitualmente en la evaluación del rendimiento en RI [Badue et al., 2007].

Para realizar los experimentos, se han seleccionado 500 usuarios de forma aleatoria, calculando una recomendación para cada uno de ellos. Se ha utilizado una implementa-

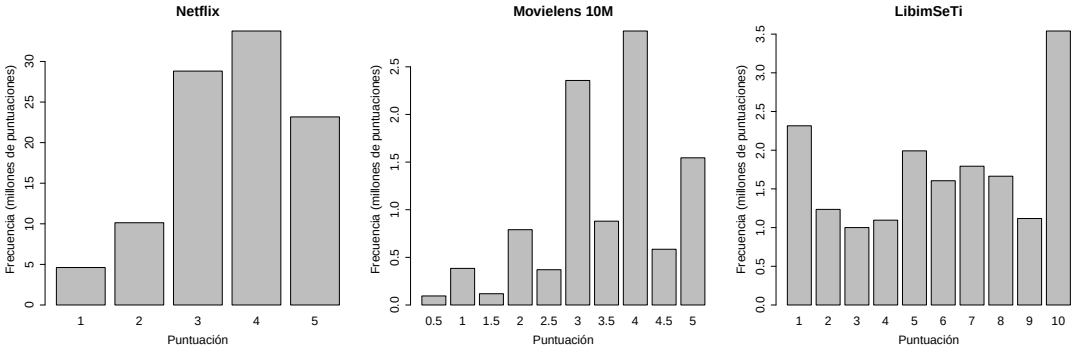


Figura 8.4: Histograma de las puntuaciones en diferentes conjuntos de datos.

ción en Java del algoritmo kNN introducido en la Sección 7.2.4. Las recomendaciones se han calculado una tras otra, para que el algoritmo se pueda beneficiar de la posible presencia de datos usados anteriormente en la *cache*, simulando por tanto un entorno real, donde continuamente se están calculando nuevas recomendaciones. Hemos medido el tiempo necesario para acceder al índice del perfil del usuario, y también al índice invertido, usado durante el cómputo del vecindario. Se ha realizado un Análisis de la Varianza (ANOVA) para comprobar si los resultados son estadísticamente significativos, así como el método de Scheffé para los contrastes múltiples, con un nivel de significación del 5 %.

En la Figura 8.5 se muestra el tiempo de acceso al índice de perfiles de usuario para los distintos métodos de codificación estudiados, así como el tiempo de acceso obtenido con un código de longitud fija. Se puede observar claramente que el uso de técnicas de compresión da lugar a una mejora significativa del rendimiento. Sin compresión, el tiempo de acceso medio por recomendación es de 127 ms., mientras que con la técnica que mejor resultados ofrece (*Local Skewed Golomb*) se reduce a 68 ms., una mejora de casi un 50 %. Los resultados son estadísticamente significativos. Con otras técnicas las mejoras obtenidas son similares: con γ, 70 ms. y con Golomb local, 71 ms. El código δ es ligeramente peor. De las técnicas estudiadas, el código Golomb global es la peor alternativa, pero aún así es significativamente mejor que no usar compresión en absoluto, obteniendo una mejora entorno al 25 %, de media.

Por otra parte, tal y como se muestra en la Figura 8.6, los tiempos de acceso al índice invertido no muestran una mejora significativa con el uso de técnicas de compresión. Teniendo en cuenta la mejora que se había obtenido en cuanto a la reducción del tamaño del índice (ver Tabla 8.3), esto podría resultar extraño, especialmente considerando que en el acceso al índice de perfiles de usuario sí se produce un aumento importante del

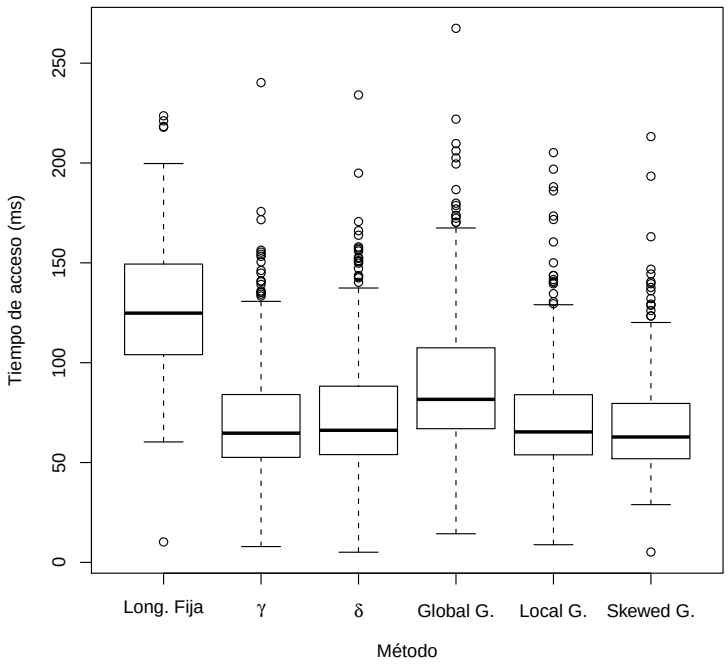


Figura 8.5: Tiempos de acceso al índice de perfiles de usuario, según el método de codificación empleado.

rendimiento.

Sin embargo, se puede explicar a partir del patrón de acceso al índice en cada caso. Tal y como se observa en la gráfica de la izquierda de la Figura 8.7, en el caso del acceso a los perfiles de usuarios, podemos ver como hay unos pocos (en torno a 100) a los que se accede frecuentemente, mientras que apenas se accede a la gran mayoría de ellos (unos 480.000). En este caso, el uso de la *cache* pasa a tener una gran importancia: si los perfiles de estos usuarios a los que se accede frecuentemente se mantuviesen en memoria, la mejora del rendimiento sería notable. Las técnicas de compresión, al reducir el tamaño de cada perfil, ayudan a que se aproxime esta situación y por tanto mejoran el rendimiento.

Sin embargo, el acceso a los productos (gráfica de la derecha), muestra un patrón ligeramente diferente. En este caso, los productos a los que se accede frecuentemente representan un porcentaje importante (2.000 de 17.000), por lo que, incluso usando técnicas de compresión, no es posible almacenar todos en la *cache*, y la mejora del rendimiento no será tan importante.

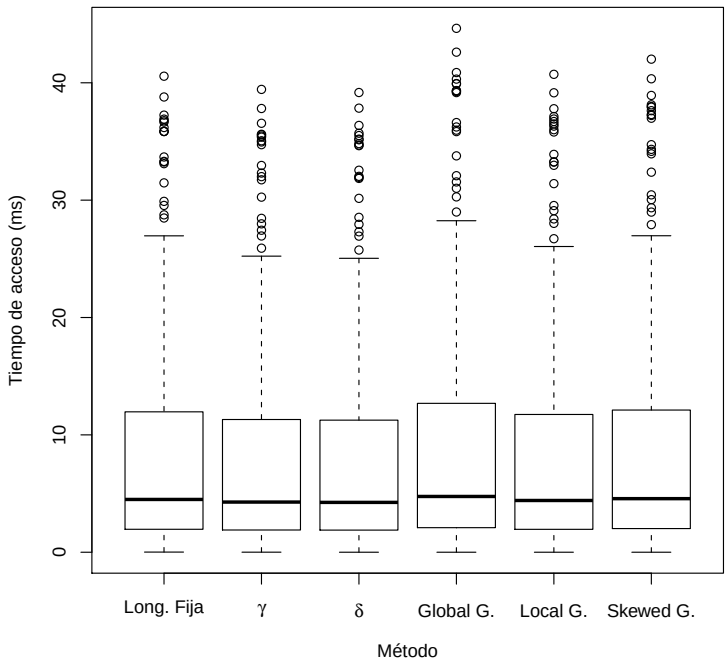


Figura 8.6: Tiempos de acceso al índice invertido de perfiles de usuario por vecindario, según el método de codificación empleado.

Es decir, la configuración hardware que hemos usado en nuestros experimentos es adecuada sólo para el primero de los casos. En la práctica, el tamaño de la memoria *cache* debe ser escogido en función del patrón de acceso a los índices, con el objetivo de que los perfiles a los que se accede frecuentemente permanezcan en la *cache* el mayor tiempo posible. Al reducir de manera importante el espacio necesario para almacenarlos, las técnicas de compresión son un factor clave para lograr ese objetivo.

8.3. Optimización de la asignación de identificadores en filtrado colaborativo

Como hemos visto, las técnicas de compresión de identificadores utilizan códigos que requieren un menor número de bits en caso de diferencias pequeñas. Por tanto, una forma de conseguir mejores tasas de compresión es asignar los identificadores de productos de tal forma que se redujese el tamaño de las diferencias en el índice. En la mayoría de las aplicaciones estos identificadores son internos y arbitrarios, por lo

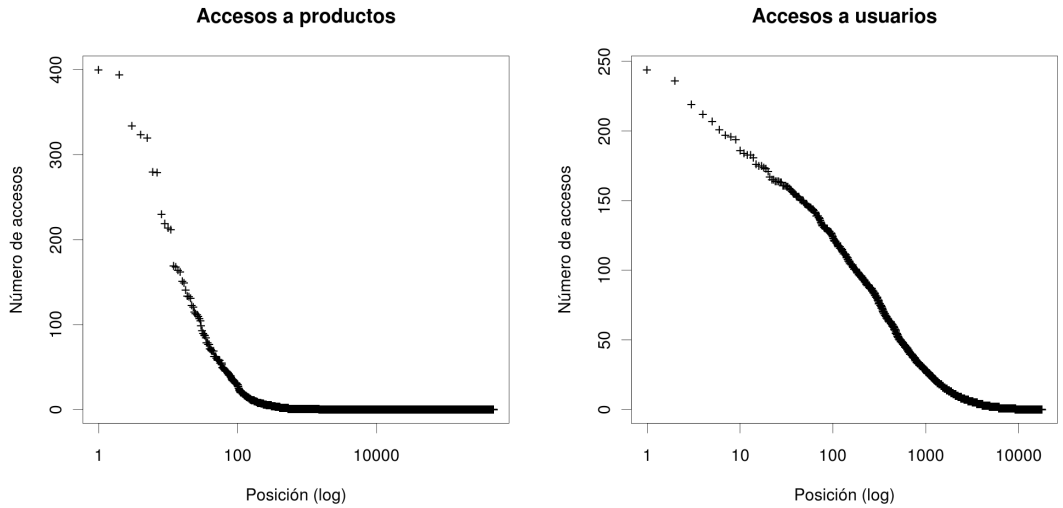


Figura 8.7: Número de accesos al perfil de cada usuario y producto, ordenado según popularidad.

que pueden ser cambiados libremente. Esta técnica, que se conoce con el nombre de reasignación de identificadores, ha sido recientemente propuesta en el ámbito de la recuperación de información, y, como veremos, puede ser usada para la compresión de la matriz de puntuaciones.

En general, este problema puede caracterizarse como un problema de minimización, orientado a buscar la asignación óptima, es decir, aquella que minimiza el tamaño del índice. Así planteado, este es un problema NP-completo [Blanco y Barreiro, 2005], pero en la práctica se han propuesto diferentes técnicas que permiten aproximar la solución. En el contexto de la RI, la mayoría de las soluciones asumen que, si se asignan identificadores próximos a documentos parecidos (es decir, con muchos términos en común), las diferencias serán menores. Las técnicas propuestas hasta la fecha pueden clasificarse en [Ding et al., 2010]:

- **Aproximaciones *Top-Down***. Parten de la colección de documentos y la dividen según la similitud entre documentos, utilizando técnicas basadas en grafos [Blandford y Blelloch, 2002] o *clustering* [Silvestri et al., 2004].

- **Aproximaciones *Bottom-Up***. Parten de cada documento de forma independiente, y los van agrupando según su similitud. La mayor parte de técnicas propuestas están basadas en *clustering* [Silvestri et al., 2004], o grafos, basándose en el problema del viajante [Blanco y Barreiro, 2006; Ding et al., 2010].

- **Aproximaciones basadas en la ordenación**. Son técnicas mucho más sencillas que las anteriores, que se limitan a ordenar los documentos según un determinado criterio, como por ejemplo según la URL [Silvestri, 2007]. Aún así obtienen buenos resultados.

La técnica que proponemos en esta Sección se encuadraría en esta última categoría. En lugar de complejas técnicas como la mayoría de las propuestas en RI, hemos optado por una técnica sencilla y eficiente, ideal para ser usada en aplicaciones con millones de usuarios y productos, y que se comporta especialmente bien en el ámbito del filtrado colaborativo. Tal y como hemos discutido anteriormente, en un sistema de recomendación existirán unos pocos productos muy populares, y un gran número de productos que sólo conocen y puntúan unos pocos usuarios. Como hemos visto en la Sección 8.2, si por ejemplo un producto es muy popular, recibirá puntuaciones de muchos usuarios, y, por tanto, las diferencias entre identificadores serán más pequeñas. Las técnicas de compresión se basan en este hecho para así reducir el tamaño de la matriz.

Sin embargo, en esta sección abordamos este mismo hecho con un punto de vista diferente: si un producto es muy popular, no sólo tendrá un perfil muy grande, sino que también aparecerá en los perfiles de muchos usuarios diferentes. Por tanto, si asignamos los identificadores más pequeños a los productos más populares, es muy posible que como resultado consigamos que los perfiles de un gran número de usuarios contengan en sus primeras posiciones productos con identificadores pequeños. Por tanto, las diferencias entre ellos también serán pequeñas, y se podrán alcanzar tasas de compresión mayores.

Resumiendo, nuestra técnica lo que hace es reasignar los identificadores según la frecuencia de los productos (o usuarios), en orden descendente. Los productos más populares recibirán los identificadores más pequeños.

La Figura 8.8 muestra la frecuencia según el identificador del producto, antes y después de la reasignación. Se puede observar como en la distribución original no hay relación entre identificadores y frecuencia, lo que hace probable que en el perfil de un usuario haya productos con identificadores tanto grandes como pequeños, dando lugar a diferencias grandes. Por otra parte, tras la reasignación la mayoría de puntuaciones corresponden a identificadores pequeños, y por tanto es de esperar que la mayoría de diferencias también lo sean.

Para confirmar este hecho, en la Figura 8.9 mostramos la distribución de probabilidad de las diferencias entre identificadores de productos, antes y después de la reasignación. Puede verse que, tras la reasignación, los identificadores pequeños son más probables, lo que reducirá el número de bits necesario para almacenar un identifi-

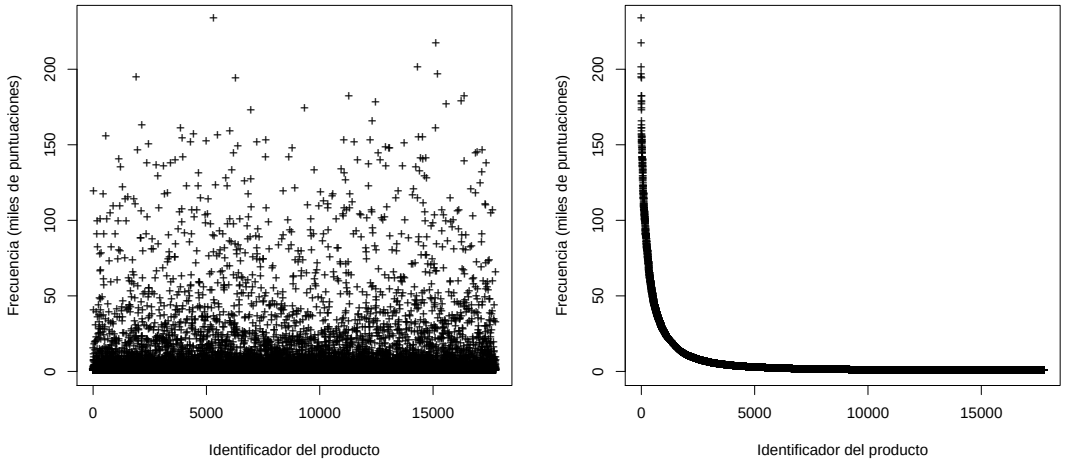


Figura 8.8: Número de puntuaciones por producto, antes (izquierda) y después (derecha) de la reasignación de identificadores, en el conjunto de datos Netflix.

cador, por término medio. En concreto, si usamos un código δ, el número medio de bits por diferencia se reduce de 9,10 a 5,35 (una mejora del 41 %), y con un código γ, de 9,57 a 5,25 (mejora del 45 %). Tras la reasignación, se consiguen ratios de compresión de 1:3 incluso con un método de codificación global, mejorando significativamente los resultados obtenidos con métodos locales sin reasignación.

Además, en la Figura 8.9 se puede observar que, tras la reasignación, la distribución de las diferencias sigue la ley de Zipf. En particular una con un exponente α de $1,44$. Este hecho es especialmente interesante ya que un conjunto de valores enteros distribuidos segundo una regla de Zipf con exponente menor que 2 puede ser almacenado de forma eficiente utilizando la codificación ζ [Boldi y Vigna, 2005]. Por tanto, si usamos esta técnica, podemos obtener mejoras adicionales. En particular, podemos reducir el número de bits necesarios a 4,97 por término medio, menos que los necesarios con otros métodos de codificación y bastante mejor que los 15 bits necesarios en este conjunto de datos en caso de no utilizar ningún tipo de compresión.

Además, el uso de la codificación ζ tras la reasignación de identificadores también ayuda a mejorar el rendimiento, tal y como se muestra en la Figura 8.10. En concreto, el tiempo medio de acceso al perfil de un usuario se reduce de 6,78 ms. (*Skewed Golomb* sin reasignación) a 6,05, una mejora pequeña pero significativa. Es importante destacar que con el método de reasignación que hemos presentado, una técnica de codificación global que haga uso de la codificación ζ mejora a técnicas de compresión locales, tanto

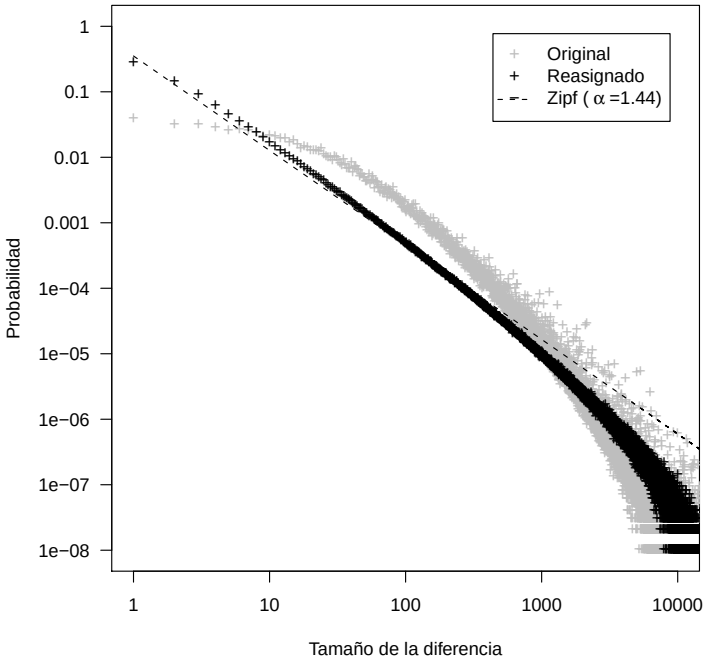


Figura 8.9: Distribución de probabilidad de las diferencias entre identificadores de productos, antes y después de la reasignación. Obsérvese la escala logarítmica en ambos ejes.

en tasa de compresión como en tiempos de acceso.

Finalmente, debemos destacar que esta técnica es también muy eficiente a la hora de reasignar los identificadores, con una complejidad de $O(nlog(n))$ en tiempo y $O(n)$ en espacio. En aplicaciones a gran escala, el tiempo necesario para la reasignación es despreciable comparado con los beneficios que aporta a la hora de calcular las recomendaciones. Además, las frecuencias relativas de usuarios y sobre todo de productos no suelen cambiar drásticamente, especialmente en períodos cortos de tiempo, por lo que en aplicaciones que necesitan actualizar el índice con frecuencia, no es necesario realizar una nueva reasignación cada vez.

8.4. Conclusiones

En este capítulo hemos estudiado el uso de técnicas de compresión de la matriz de puntuaciones, y las ventajas que con ellas se consiguen.

En primer lugar, estas técnicas permiten reducir de manera importante el espacio

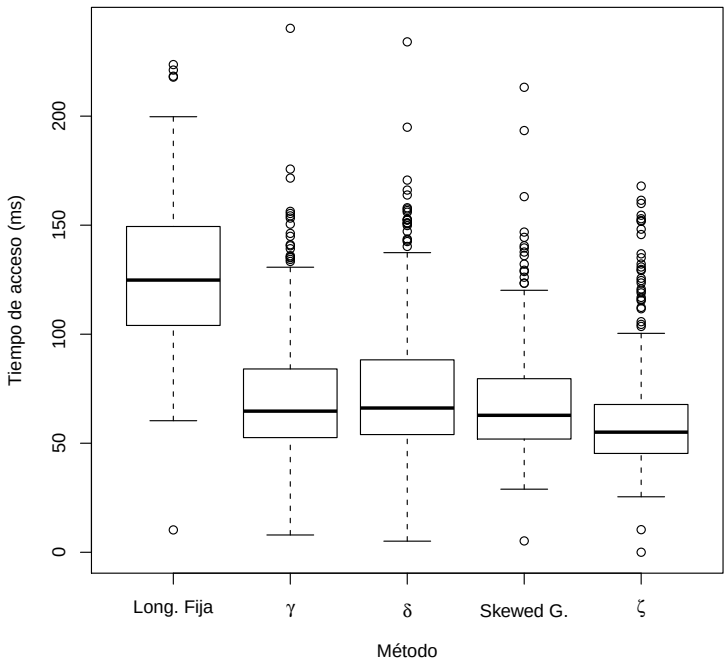


Figura 8.10: Tiempos de acceso al índice de perfiles de usuario tras la reasignación de identificadores, según el método de codificación empleado.

necesario para almacenar la matriz. Los mejores resultados han sido obtenidos con los métodos locales, si bien códigos globales como γ y δ son suficientes para conseguir buenos ratios de compresión. La combinación de la codificación tanto de identificadores como de puntuaciones permite que, por ejemplo, en el conjunto de datos Netflix el tamaño de la matriz comprimida sea sólo el 43 % del tamaño original. Además, hemos mostrado como la compresión puede reducir los tiempos de recomendación hasta un 50 %.

Finalmente, en la última parte del capítulo hemos propuesto una técnica para la reasignación de identificadores, la cual da lugar a mejoras adicionales tanto en rendimiento como en espacio. Esta técnica, basada en asignar los identificadores más bajos a los productos y usuarios con más puntuaciones, es especialmente efectiva y puede implementarse fácilmente con un proceso de indexación en dos pasos. Además, si se usa junto a un código ζ para la compresión de identificadores, los ratios de compresión son todavía mayores.

Capítulo 9

Sistemas de recomendación distribuidos

En sistemas con millones de usuarios y productos, la distribución de datos y el procesamiento paralelo de los mismos son técnicas imprescindibles para poder satisfacer todas las recomendaciones solicitadas en un tiempo razonable. Se bien en ámbitos como la recuperación de información estas técnicas se utilizan desde hace años, su aplicación en sistemas de recomendación apenas ha sido estudiada.

En este capítulo analizamos los factores que afectan al rendimiento de un algoritmo kNN, y proponemos una arquitectura distribuida que mejora significativamente tanto el tiempo de respuesta como el volumen de recomendaciones calculadas por unidad de tiempo. En la Sección 9.1 describimos de forma global la arquitectura propuesta. Posteriormente, en la Sección 9.2 presentamos dos técnicas para la distribución de la matriz de puntuaciones: partición por productos y partición por usuarios.

Para evaluar el rendimiento de las soluciones propuestas, hemos simulado el comportamiento de un sistema real. El uso de un simulador nos permite estimar el comportamiento del sistema con diferentes configuraciones y un elevado número de máquinas, con un coste muy inferior al necesario en caso de usar dispositivos reales. En la Sección 9.3 se presenta el modelo de simulación empleado y se valida su comportamiento. Finalmente, en la Sección 9.4 se describen los experimentos realizados y se analizan los resultados obtenidos.

9.1. Introducción

El procesamiento paralelo y la distribución de datos son técnicas esenciales para que los sistemas de recuperación de información modernos sean capaces de responder

miles de consultas por segundo. En RI, son especialmente populares técnicas como la replicación del sistema o la partición y distribución del índice [Büttcher et al., 2010]. La replicación es la más sencilla de las dos, y consiste simplemente en crear varias copias o réplicas del sistema, cada una de las cuales contiene toda la información y elementos necesarios para poder procesar una consulta de forma independiente. Por tanto, cada réplica puede procesar una consulta diferente al mismo tiempo, lo que aumenta el *throughput* del sistema, es decir, el número de consultas que pueden ser procesadas por unidad de tiempo.

Por otra parte, la partición del índice consiste en dividirlo en varios trozos que serán asignados a nodos de procesamiento diferentes. Por tanto, el índice se distribuye entre varios nodos, cada uno de los cuales obtendrá un resultado parcial de las consultas a partir de la información de la que dispone en su trozo de índice. Finalmente, un nodo adicional se encarga de combinar los resultados parciales y calcular la respuesta final. Cada consulta es procesada por varios nodos en paralelo, cada uno de los cuales sólo necesita hacer una pequeña parte del trabajo. Por tanto, el tiempo de procesamiento necesario va a ser menor que en un sistema no distribuido, lo que incrementa el *throughput* a la par que reduce el *tiempo de respuesta*.

Un sistema de RI distribuido está formado generalmente por dos componentes: el coordinador o *broker* y los servidores de consulta o *query servers* (QS), conectados a través de una red de área local (LAN) de gran velocidad y ancho de banda [Cacheda et al., 2007]. El coordinador está encargado de recibir las consultas de los usuarios, distribuirlas entre los diferentes servidores de consulta y, finalmente, combinar los resultados obtenidos por cada QS y enviar la respuesta final al usuario. Cada servidor de consulta almacena una parte del índice, y está encargado de resolver las distintas consultas en esa parte, obteniendo un resultado parcial que enviará al coordinador.

El modelo que proponemos para filtrado colaborativo está precisamente basado en esta misma idea: la matriz de puntuaciones se divide entre un puñado de servidores, los cuales realizan la mayor parte del trabajo, y un servidor centralizado que se encarga de la coordinación entre todos ellos y de la interacción con el usuario. En la Figura 9.1 se puede observar una representación de la arquitectura propuesta. En concordancia con la nomenclatura utilizada en RI, daremos a los nodos de trabajo el nombre de servidores de recomendación (RS), y al servidor centralizado el de coordinador. Las tareas que ambos han de realizar, sin embargo, difieren bastante de las que sus homónimos realizan en RI. En la recomendación, nuestro modelo sigue los siguientes pasos:

1. El coordinador recibe una solicitud de recomendación.

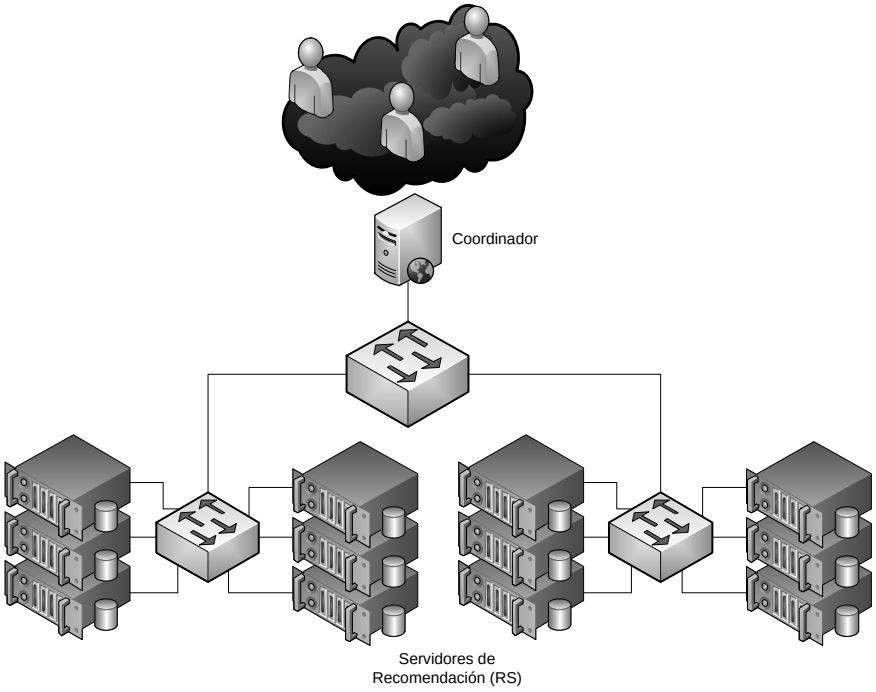


Figura 9.1: Esquema de la arquitectura de un sistema de recomendación distribuido.

2. El coordinador obtiene el vecindario del usuario actual.
3. El coordinador envía la lista de vecinos a cada RS, solicitando la recomendación.
4. Cada RS calcula una recomendación parcial y se la envía al coordinador.
5. Finalmente, el coordinador combina los resultados y muestra la recomendación final al usuario.

Las operaciones que es necesario realizar en cada paso son dependientes de la técnica de partición empleada, y se detallarán en las próximas secciones. Es importante destacar que, de forma similar a como se hace en RI, en nuestro modelo el índice de perfiles de usuario se divide y distribuye entre los diferentes servidores de recomendación. Sin embargo, el índice de vecinos se almacena completo en el coordinador, y es este el responsable de recuperar la lista de vecinos de cada usuario. Se podría pensar que esta decisión no es la más adecuada, ya que podría dar lugar a un cuello de botella en el coordinador, y ser fuente de problemas de escalabilidad. Sin embargo, el índice de vecinos es generalmente pequeño, debido a que sólo se almacena un pequeño número

de vecinos por cada usuario. Por ejemplo, con diez millones de usuarios y $k = 25$, y suponiendo que los identificadores de usuario se almacenan sin comprimir, dedicando 4 bytes a cada uno de ellos, el índice apenas ocuparía 1 GB. Por tanto, en una máquina actual el índice se podrá almacenar enteramente en memoria RAM, la cual tiene unos tiempos de acceso muy bajos, tal y como se ha discutido en la Sección 8.1. En la práctica, el tiempo necesario para recuperar el vecindario es despreciable en comparación a las otras fases de la recomendación, como también lo es el coste necesario para enviar la lista de vecinos a cada servidor de recomendación.

Finalmente, en caso de ser necesario el coordinador también podría ser replicado o incluso distribuido, lo cual es habitual en RI [Cacheda et al., 2007], si bien no ha sido una opción considerada en este trabajo.

Por contra, el acceso al perfil del usuario sí es costoso. Primero, porque su tamaño puede llegar a ser grande, haciendo imposible que todos los perfiles sean almacenados en memoria. Y segundo, porque para cada recomendación debemos acceder al perfil completo de cada vecino. Por tanto, optamos por dividir el índice de perfiles de usuario, distribuyendo el mismo entre los distintos servidores de recomendación, lo que tiene dos grandes ventajas:

- En primer lugar, cada RS sólo tiene que realizar una pequeña parte del trabajo, con lo que el tiempo necesario será menor.

- Además, como cada RS almacena sólo un trozo del índice, las posibilidades de que los datos accedidos más frecuentemente estén disponibles en la *cache* serán mayores, lo que repercute en una mejora del rendimiento.

9.2. Técnicas de partición de la matriz de puntuaciones

En esta tesis se proponen dos técnicas para dividir el índice de perfiles de usuario: la partición por productos y la partición por usuarios. En la partición por productos, cada RS es responsable de un subconjunto de los productos, y mantiene un perfil parcial de cada usuario, compuesto por las puntuaciones otorgadas a esos productos. Por otro lado, en la partición por usuarios, cada RS almacena el perfil completo de una parte de los usuarios. En ambos casos, usuarios o productos se distribuyen uniformemente entre los distintos servidores de recomendación. La Figura 9.2 muestra gráficamente ambas técnicas, que procedemos a describir con más detalle a continuación.

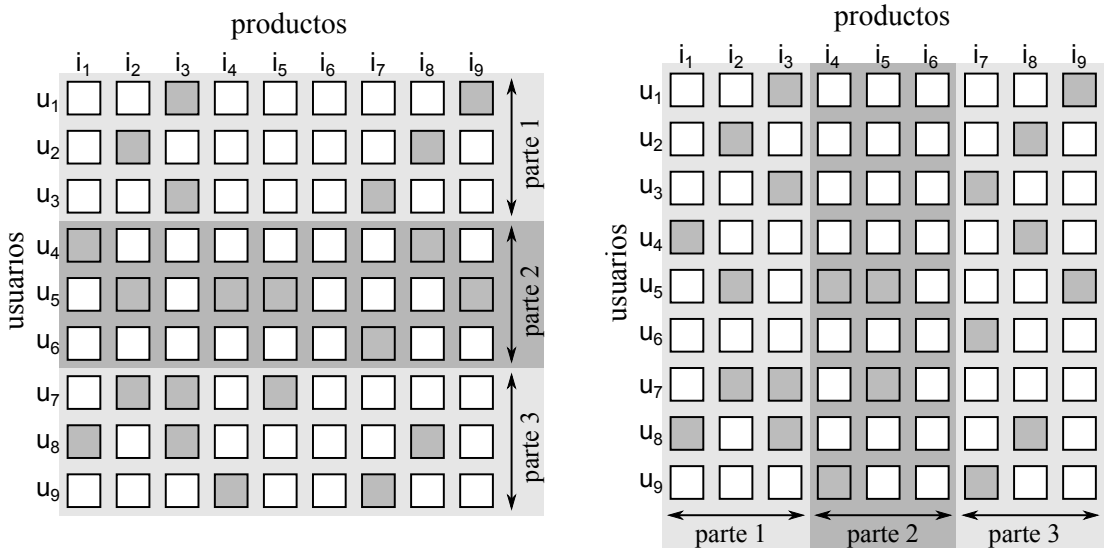


Figura 9.2: Técnicas de partición por usuarios (izquierda) y por productos (derecha).

9.2.1. Partición por productos

En la partición por productos, cada RS tiene asignado un subconjunto de los productos disponibles en el sistema. Formalmente, el conjunto de productos se divide aleatoriamente entre los servidores de recomendación disponibles, es decir, $\mathcal{I} = \{\mathcal{I}_1 \cup \mathcal{I}_2 \cup \ldots \cup \mathcal{I}_n\}$, con n el número de RS y cumpliéndose que $\mathcal{I}_x \cap \mathcal{I}_y = \varnothing, \forall x, y | x \neq y$. Por tanto, cada RS almacenará un perfil parcial para cada usuario, que contendrá únicamente las puntuaciones otorgadas a los productos de los cuales es responsable.

El funcionamiento global del sistema es el siguiente:

1. El coordinador recupera el vecindario del usuario actual, y se lo envía a los servidores de recomendación. Esto se puede hacer de forma eficiente mediante un paquete *multicast*.

2. Cada RS recupera el perfil parcial del usuario actual y de los distintos vecinos.

3. Cada RS procesa sus perfiles parciales, descartando productos ya puntuados por el usuario actual, y combinando los perfiles de cada vecino. Al final, se seleccionan los N productos mejor valorados.

4. Estos N productos se envían al coordinador.

5. Finalmente, el coordinador ordena los distintos perfiles parciales recibidos, y selecciona los N productos con mejor valoración global como recomendación para el usuario.

La principal ventaja de esta aproximación está en el paso 3, y es que cada RS puede directamente descartar un gran número de productos, gracias a que puede calcular el peso final que recibirá cada producto a él asignado. Esto es así debido a que todas las puntuaciones recibidas por un producto están almacenadas en el mismo RS. Por tanto, la cantidad de productos que se enviarán al coordinador es pequeña (como mucho, N productos por RS), y el paso final en el coordinador es muy sencillo, pues se limita a seleccionar los N productos mejor valorados a partir de n listas ya ordenadas.

Por otra parte, sus principales limitaciones consisten en que todos los servidores necesitan trabajar en todas las recomendaciones solicitadas al sistema, y además cada uno de ellos debe acceder a $k+1$ perfiles (los de los vecinos más el usuario actual). Esto podría requerir varios accesos al disco, y acabar convirtiéndose en un cuello de botella, tal y como se discute en la Sección 9.4.

9.2.2. Partición por usuarios

La partición por usuarios es similar a la anterior, pero en este caso cada RS tendrá asignado un subconjunto de los usuarios, almacenando el perfil completo de cada uno de ellos. El funcionamiento del algoritmo es por tanto diferente, y consiste en los siguientes pasos:

1. El coordinador recupera el vecindario del usuario actual, y se lo envía a cada servidor de recomendación. Técnicamente esto no es necesario, ya que cada RS sólo necesita conocer los vecinos que forman parte de su subconjunto de usuarios, pero en la práctica es más eficiente enviar un único paquete *multicast* que paquetes diferentes a cada RS.

2. Cada RS recupera el perfil de los vecinos que tuviese asignado. Aquel que tenga asignado el perfil del usuario actual también accederá a este.

3. Cada RS combina los diferentes perfiles y envía el resultado al coordinador.

4. Finalmente, el coordinador combina los distintos perfiles recibidos, los ordena, y finalmente recomienda al usuario los N productos mejor valorados.

En este caso, el número de accesos a disco necesarios en cada RS es menor, ya que no tiene que recuperar todos los perfiles, sino sólo aquellos que tenga asignados. De hecho, en caso de que se use un número elevado de servidores de recomendación, muchos de ellos no tendrán que trabajar en absoluto, posibilitando el procesamiento simultáneo de diferentes recomendaciones. Por otra parte, al contrario que en la partición por

productos, cada RS sólo conoce el peso parcial de cada producto, ya que otros vecinos podrían también contribuir al mismo. Por tanto, los servidores de recomendación no pueden descartar ningún producto, y todos ellos deben ser enviados al coordinador, lo cual incrementa la carga en la red.

De todas formas, esto no implica que esta técnica ocasione una mayor carga que la partición por productos, ya que en esta otra todos los servidores de recomendación deben responder al coordinador, incluso aquellos que no tengan productos que enviar. Esto se debe a que en la partición por productos, el coordinador no puede adivinar cuantos servidores tienen realmente datos que transmitir, por lo que tiene que esperar a que todos respondan antes de calcular la recomendación final. Sin embargo, en la partición por usuarios, el coordinador únicamente tiene que esperar a recibir los $k + 1$ perfiles, lo que hace innecesario que un RS necesite confirmar que no tiene datos que enviar. Por tanto, cuando el sistema cuenta con un número elevado de servidores de recomendación, la carga en la red será menor con la técnica de partición por usuarios. En la Sección 9.4.2 retomaremos esta discusión.

9.3. Diseño del modelo de simulación

Para evaluar el rendimiento de las arquitecturas de filtrado colaborativo distribuidas, hemos implementado un simulador orientado a eventos discretos, basado en el entorno de simulación JavaSim [Little, 1999]. El objetivo es poder usar los resultados obtenidos a través de la simulación para predecir el comportamiento de un sistema real. Para tal fin, diseñamos un modelo que simula los distintos pasos necesarios para realizar una recomendación, teniendo en cuenta los componentes involucrados en los mismos (disco, memoria, red, procesador...)

La simulación es una técnica usada habitualmente para la evaluación del rendimiento en varios campos como el de la recuperación de información [Cacheda et al., 2007; Ribeiro-Neto y Barbosa, 1998]. Su principal ventaja es el poder evaluar un gran número de configuraciones y arquitecturas diferentes, en muy poco tiempo y sin necesidad de afrontar grandes gastos en adquisición de infraestructura hardware, configuración y/o mantenimiento.

En esta sección se presentará un novedoso modelo de simulación orientado a la evaluación del rendimiento de sistemas de filtrado colaborativo, y se describirán los pasos llevados a cabo para su diseño y evaluación. Se comenzará describiendo el modelo de simulación para un sistema de filtrado colaborativo no distribuido, basado en el algoritmo kNN presentado en el Capítulo 7. Posteriormente, presentaremos la evaluación

del mismo. Finalmente, se estudiará cómo se ha ampliado el modelo para simular el rendimiento de los sistemas distribuidos presentados en la sección anterior.

Para el diseño del modelo de simulación, hemos realizado un gran número de experimentos en hardware real, identificando los parámetros más importantes a la hora de caracterizar el rendimiento de un sistema de filtrado colaborativo. Como mostraremos en la próximas líneas, estos parámetros pueden clasificarse en parámetros dependientes del hardware (por ejemplo, el tiempo de acceso a disco), y parámetros dependientes del dominio o aplicación (por ejemplo, el número de puntuaciones de un usuario).

Para estimar los parámetros dependientes del hardware, hemos realizado un conjunto de experimentos en hardware real, midiendo los tiempos obtenidos y el impacto de cada parámetro. Se han utilizado 1.000 usuarios elegidos aleatoriamente a partir del conjunto de datos Netflix. Al igual que en el Capítulo 8, hemos hecho uso de un equipo algo antiguo, para ser consistentes con el tamaño del conjunto de datos empleado. En concreto, hemos usado un PC con un procesador Intel Pentium 4 a 3, 20 GHz y 512 MiB de RAM. En cuanto a los parámetros dependientes del conjunto de datos, los hemos obtenido a partir de los datos reales para los usuarios seleccionados en la evaluación.

9.3.1. Modelo analítico del rendimiento de un algoritmo kNN

En primer lugar, hemos diseñado un modelo analítico que permita reproducir el comportamiento, en términos de rendimiento, de un algoritmo de filtrado colaborativo no distribuido. En particular, el algoritmo kNN con preselección de vecinos presentado en el Capítulo 7. A pesar de ser un modelo sencillo, hemos logrado capturar los factores más importantes a la hora de predecir el rendimiento de un algoritmo de filtrado colaborativo. De hecho, tal y como mostraremos en la Sección 9.3.2, nuestro modelo permite aproximar con gran exactitud los tiempos necesarios para calcular una recomendación en un sistema real.

Partiendo del algoritmo kNN considerado, el tiempo total necesario para calcular una recomendación, t, viene dado por:

- El tiempo necesario para recuperar el vecindario del índice de vecinos, tn.
- El tiempo del acceso al perfil, tp, es decir, el tiempo necesario para recuperar el perfil de un usuario desde el índice de perfiles de usuario. Debemos tener en cuenta que el algoritmo necesitará acceder al perfil del usuario actual, así como a los perfiles de los k vecinos.
- El tiempo de combinación, tm, necesario para calcular el peso de un producto,

w, combinando las puntuaciones y similitud de los productos puntuados por los vecinos.

- El tiempo de ordenación, ts, necesario para seleccionar los N productos mejor valorados, es decir, con mayor peso.

Por tanto, dado un usuario $a \in \mathcal{U}$, el tiempo de recomendación se calcularía de la siguiente forma:

$$t_a = tp_a + tn_a + \sum_{k \in \mathcal{N}(a)} tp_k + tm_a + ts_a \tag{9.1}$$

Tanto tp como tn incluyen el tiempo necesario para leer datos del disco. Sin embargo, estos datos podrían estar disponibles en la *cache*, lo que reduce significativamente los tiempos de acceso.

El estimar cuándo un perfil (o vecindario) en particular está disponible en la *cache* no es un problema trivial. Depende del tamaño de la *cache* e índice, así como del algoritmo de reemplazo utilizado y de la secuencia de accesos realizados con anterioridad.

De cara a proponer un modelo de simulación eficaz, hemos estudiado el comportamiento de la *cache* según la relación entre su tamaño y el del índice. A partir del conjunto de datos Netflix, hemos seleccionado aleatoriamente 10.000 usuarios. Calculamos una recomendación para los primeros 6.000, de tal forma que se inicialice la *cache* con una cierta cantidad de datos válidos, y a continuación medimos el porcentaje de aciertos obtenido para los restantes 4.000 usuarios, es decir, el porcentaje de perfiles que estaban en la *cache*. Los resultados se muestran en la Figura 9.3. El eje x representa el tamaño de la *cache* en función del porcentaje de perfiles que cabrían en ella, respecto al número total de perfiles en el índice. El número entre paréntesis sería el porcentaje respecto al número total de perfiles a los que se ha accedido durante el experimento.

A la vista de estos resultados, hemos considerado que la *cache* se puede simular eficazmente usando un modelo sencillo, donde se asume que un determinado porcentaje de accesos van a ser servidos directamente por la *cache*. Por tanto, y en función de ese porcentaje, tanto tp como tn vendrán determinados bien por el tiempo de acceso a disco, $td(s)$, o bien por el tiempo de acceso a la cache, $tc(s)$. En ambos casos, s representa el número de identificadores a leer, es decir, el tamaño del perfil o del vecindario.

9.3.1.1. Simulación del acceso a disco

Acceder al disco requiere de 4 pasos: (1) se indica al disco qué datos se desean transferir; (2) la cabeza lectora del disco se desplaza hasta la pista que contiene los

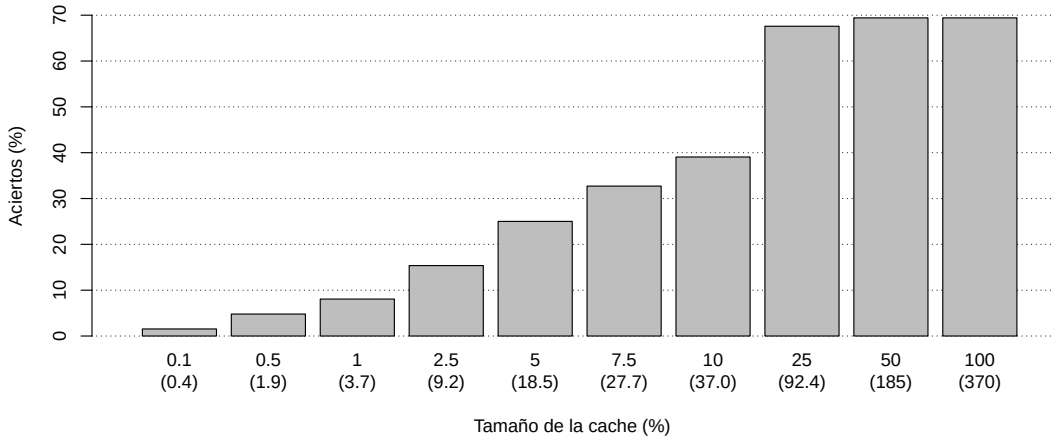


Figura 9.3: Porcentaje de aciertos según el tamaño de la *cache*.

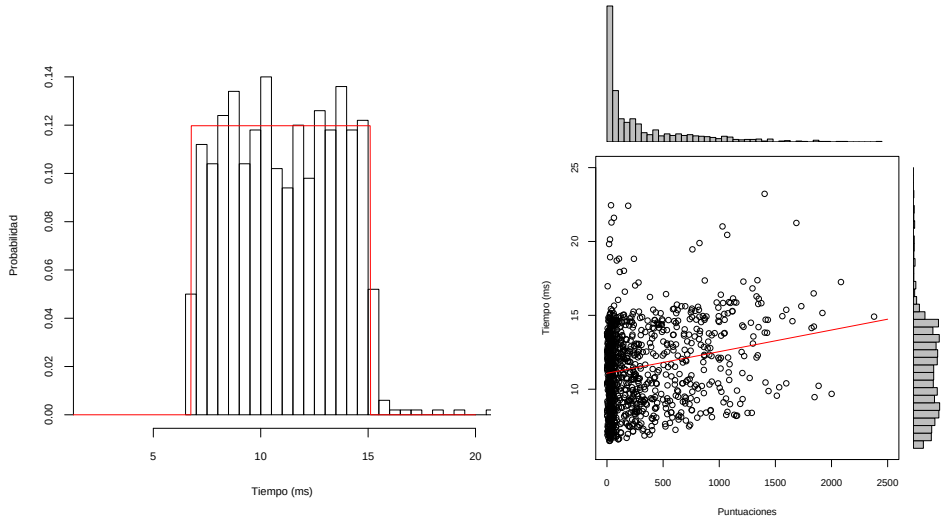
datos a leer; (3) el disco rota hasta que los datos a leer estén bajo la cabeza lectora; y finalmente (4) los datos se leen y transfieren a la memoria principal. Los pasos (1) a (3) son independientes de la cantidad de datos a transferir, y los denotaremos como *tiempo de búsqueda*. El tiempo necesario para el último paso será denominado *tiempo de transferencia*, y sí depende de la cantidad de datos a transferir.

En el caso de tn, esta cantidad es constante y directamente relacionada con el número de vecinos, k. En los experimentos realizados, hemos utilizado $k = 25$. Por tanto, los pasos (1), (2) y (4) son prácticamente constantes, y la variabilidad dependerá sobre todo del paso (3) [1]. En un disco rotando a 7.200 revoluciones por minuto (RPM), el tiempo necesario para ese paso, conocido como *latencia de rotación*, sigue una distribución uniforme entre 0 (la cabeza ya está situada sobre los datos a leer) y $8,33$ ms. (se necesita una rotación completa). Este comportamiento se puede ver en la Figura 9.4a. Incluso cuando la latencia de rotación es cercana a 0, existe un cierto retardo, que corresponde al resto de pasos.

Por otra parte, en el acceso al perfil del usuario, la cantidad de datos a transferir es variable y depende en gran medida del tamaño del perfil. De hecho, tal y como se muestra en la Figura 9.4b, existe una relación lineal entre ambos.

Por tanto, podemos estimar el tiempo de acceso a disco, $td(s)$, como:

[1] Es cierto que el paso (2) varía ligeramente según la posición de la cabeza lectora respecto a la pista a leer, pero ese tiempo es muy pequeño y puede ser despreciado.



(a) Histograma del tiempo de acceso al vecindario.

(b) Tiempo de acceso al perfil, según el número de puntuaciones.

Figura 9.4: Tiempos de acceso a disco, sin *cache*.

$$td(s) = \tau_i + U(0, \tau_l) + s * \tau_r \tag{9.2}$$

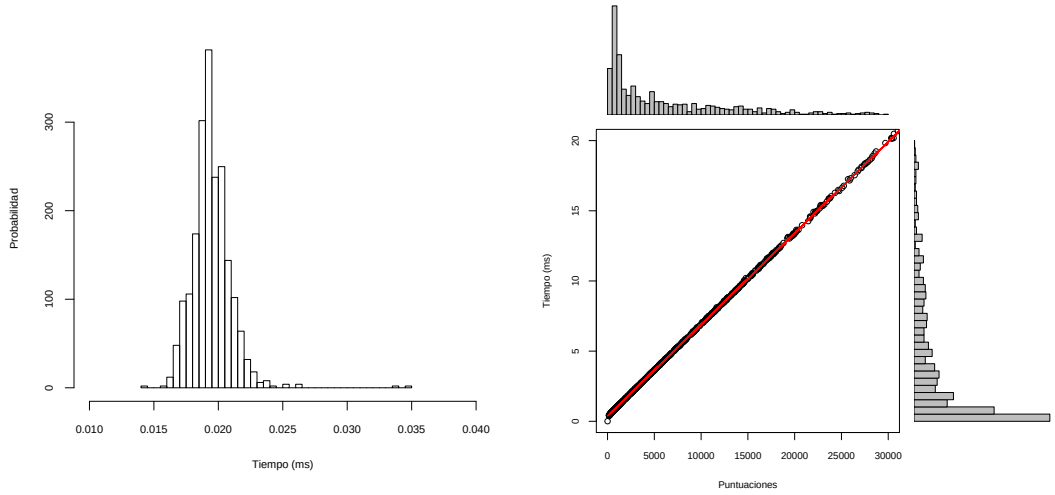
Donde U representa la distribución uniforme, τ_i es un parámetro que representa el conjunto de retardos de naturaleza fundamentalmente constante, τ_l es la latencia máxima de rotación del disco, y τ_r el tiempo medio necesario para leer y transferir un identificador desde el disco. τ_i, τ_l y τ_r son parámetros dependientes del hardware, que toman, respectivamente, los valores $6,75$, $8,33$ y $0,0020$ ms. en nuestro entorno de pruebas. Por otra parte, s es un parámetro dependiente del conjunto de datos, cuyo valor será diferente para cada usuario, y será por tanto un parámetro del modelo de simulación. En el mismo, tanto tp como tn se estimarán a partir del valor de $td(s)$.

9.3.1.2. Simulación de la memoria *cache*

Como se puede observar en las Figuras 9.5a y 9.5b, el tiempo de acceso a la *cache* es mucho menor que el acceso a disco, y se puede simular con el siguiente modelo:

$$tc(s) = \tau_{c1} + s * \tau_{c2} \tag{9.3}$$

donde τ_{c1} representa una latencia constante propia de la *cache*, y τ_{c2} el tiempo medio necesario para transferir un único identificador. Ambos son parámetros dependientes del hardware, que toman los valores $0,015$ y $0,00065$ ms., respectivamente.



(a) Histograma del tiempo de acceso al vecindario.

(b) Tiempo de acceso al perfil, según el número de puntuaciones.

Figura 9.5: Tiempos de acceso a disco, cuando los datos están en la *cache*.

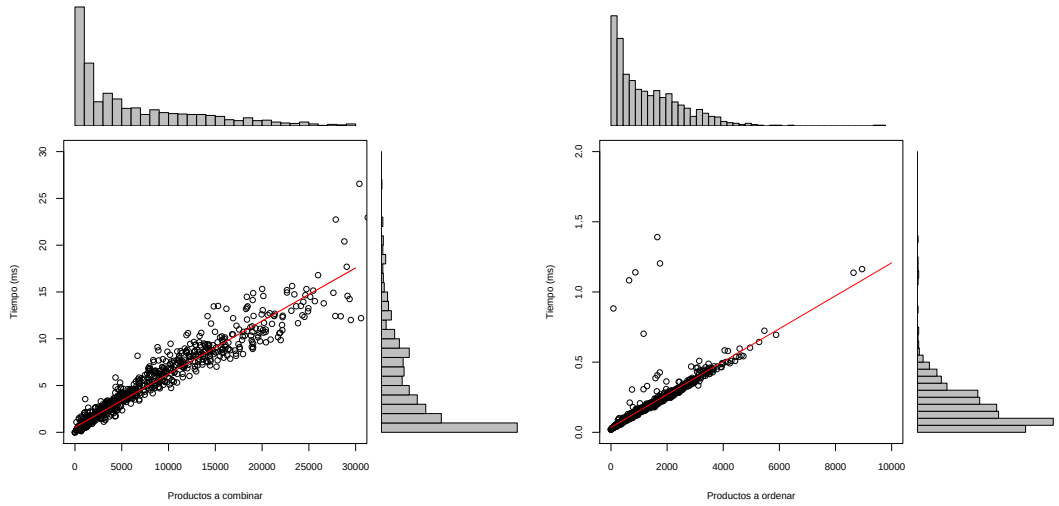
9.3.1.3. Simulación de los tiempos de combinación y ordenación

Una vez que se ha obtenido el vecindario, y accedido a los perfiles de los vecinos, el algoritmo necesita combinar y calcular el peso de los productos puntuados por los vecinos. Tal y como se muestra en la Figura 9.6a, el tiempo necesario, tm, es linealmente proporcional al número total de productos puntuados por los vecinos, por lo que puede ser simulado de la siguiente forma:

$$tm(s') = \tau_{m1} + s' * \tau_{m2} \tag{9.4}$$

donde τ_{m1} y τ_{m2} son parámetros dependientes del hardware, que toman los valores $0,55$ y $0,00055$ ms. en nuestros experimentos. s' es el número total de productos puntuados por los vecinos, es decir, $s' = \sum_{k \in \mathcal{N}(a)} \left|\mathcal{I}^{(k)}\right|$, para el usuario actual a. Naturalmente, s' es un parámetro dependiente del conjunto de datos, diferente para cada usuario.

Finalmente, el algoritmo debe ordenar los productos según el peso calculado en la fase anterior, y recomendar al usuario los N mejor valorados. Tal y como se observa en la Figura 9.6b, ts también está relacionado linealmente, pero en este caso con el número total de productos restantes una vez finalice la fase de combinación, que denotaremos como s''. Este número será por supuesto inferior a s', pues muchos vecinos habrán puntuado productos en común, y se puede estimar usando la siguiente ecuación:



(a) Tiempo de combinación de productos, según el número de puntuaciones.

(b) Tiempo de ordenación según el número de productos a ordenar.

Figura 9.6: Tiempos de combinación y ordenación de los productos.

$$s'' = \alpha_0 * s' + \alpha_1 * \sqrt{s'} \tag{9.5}$$

tomando α_0 y α_1 los valores $0,06$ y $11,05$, respectivamente, en el conjunto de datos Netflix. El tiempo de ordenación puede finalmente estimarse como:

$$ts(s'') = \tau_{s1} + s'' * \tau_{s2} \tag{9.6}$$

con τ_{s1} y τ_{s2} parámetros dependientes del hardware, y que en nuestros experimentos toman los valores $0,040$ y $0,00012$ ms., respectivamente.

9.3.2. Evaluación del modelo de simulación

Para evaluar el modelo de simulación presentado en la sección anterior, hemos comparado sus resultados con los obtenidos en un sistema real. En primer lugar, hemos seleccionado de forma aleatoria un nuevo conjunto de 1.000 usuarios a partir del conjunto de datos Netflix. Para cada usuario, usamos el simulador para estimar el tiempo necesario para realizar una recomendación, y los comparamos con los obtenidos en un sistema real. Finalmente, ajustamos un modelo lineal, para comprobar si los tiempos reales se pueden predecir a partir de los tiempos obtenidos con la simulación.

Como se observa en la Figura 9.7, existe una relación lineal entre ambos tiempos. De hecho, presenta un coeficiente de determinación R^2 del 99 %, lo que indica que nuestro

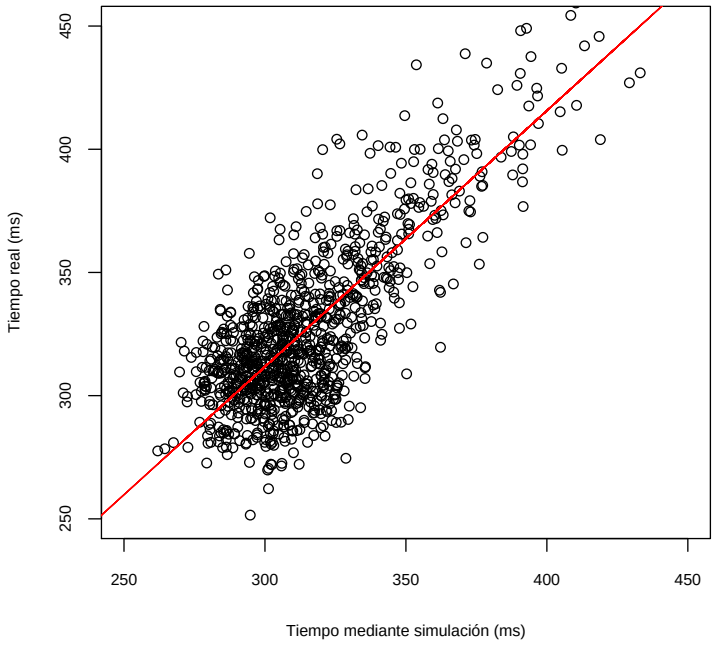


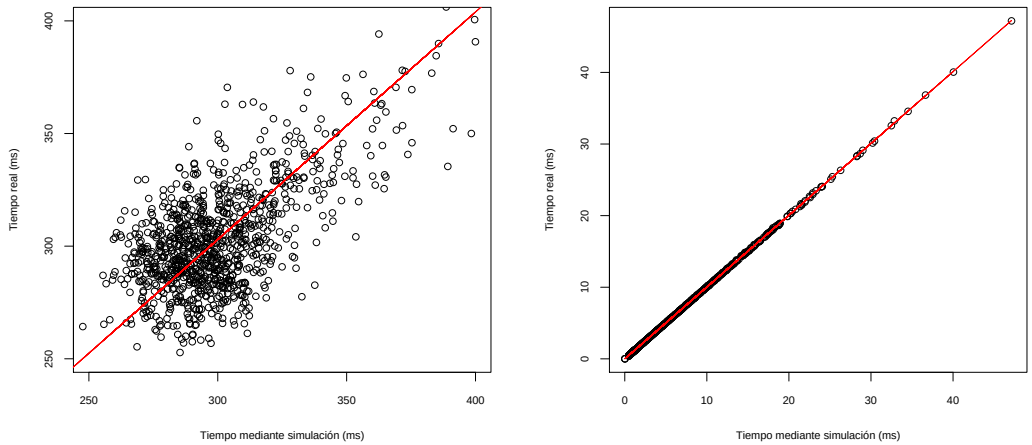
Figura 9.7: Relación entre el tiempo de recomendación real y el obtenido mediante la simulación. Los datos atípicos han sido eliminados de la figura.

modelo de simulación explica la mayor parte de la variabilidad presente en los tiempos. La variabilidad restante es debida principalmente al tiempo de búsqueda del disco, que nosotros modelamos como una variable aleatoria uniformemente distribuida.

Por otra parte, también hemos evaluado cada una de las fases de la recomendación de forma independiente, obteniendo resultados similares. En cuanto al tiempo de ordenación, un R^2 del 96 %; en cuanto al tiempo de combinación, del 97 %; y en cuanto al acceso al perfil, del 99 % en caso de considerar acceso al disco (Figura 9.8a), y del 100 % si asumimos que los datos están en la *cache* (Figura 9.8b).

9.3.3. Modelo de simulación para sistemas distribuidos

En esta sección estudiamos cómo se puede modificar el modelo presentado anteriormente para simular el rendimiento en los sistemas distribuidos presentados en la Sección 9.2. En un sistema distribuido, el tiempo de recomendación depende del tiempo invertido por el coordinador, el tiempo invertido por los servidores de recomendación, y el tiempo necesario para la comunicación entre ambos. Se debe tener en cuenta que el



(a) Tiempo de acceso a disco. **(b)** Tiempo de acceso a *cache*.

Figura 9.8: Tiempos de acceso al perfil reales frente a simulados. Los datos atípicos han sido eliminados de la figura.

coordinador debe esperar a recibir las recomendaciones parciales, pero que los distintos servidores pueden trabajar en paralelo.

Los tiempos de coordinador y servidores de recomendación se pueden calcular a partir del modelo presentado anteriormente, considerando por supuesto las tareas a realizar por ambos, que, como hemos visto, son diferentes según la técnica de partición empleada.

En cuanto al tiempo de comunicación, hemos simulado una LAN conmutada en la cual los diferentes nodos están conectados mediante *switches* (conmutadores) de 48 puertos. Por tanto, cuando simulamos infraestructuras con más de 48 nodos, necesitamos considerar varios *switches*. En dichos casos, hemos optado por una topología en forma de árbol, donde el coordinador se conecta al *switch* raíz, y los servidores de recomendación a las hojas.

Hemos simulado el funcionamiento básico de los *switches*: cuando reciben un paquete por uno de los puertos, se lo reenvían al puerto o puertos de destino correspondientes. El comportamiento simulado es el de *store-and-forward*, en el cual el *switch* espera a recibir el paquete completo antes de reenviarlo. Por supuesto, un *switch* puede procesar varios paquetes en paralelo, siempre y cuando los puertos de entrada y salida sean diferentes. En caso de que varios paquetes deban ser enviados simultáneamente por el mismo puerto, son encolados y enviados uno a uno. Hemos considerado colas independientes para cada puerto.

Además, hemos simulado el retardo de propagación en cada segmento, teniendo en cuenta el tamaño del paquete, el tiempo necesario para la propagación de las señales electromagnéticas a través del cable, y el ratio de transmisión de la red. En concreto, hemos seguido el modelo presentado por Cacheda et al. [2007]. Se ha considerado una red *Fast Ethernet*, con un ancho de banda de 100 Mbps.

El tamaño de cada paquete ha sido calculado independientemente, teniendo en cuenta la información enviada. Se ha considerado un tamaño de 4 bytes para los identificadores de productos y usuarios, y 4 bytes adicionales para codificar los pesos y similitudes. Además, se ha considerado una cabecera de 16 bytes para almacenar información como el identificador de cada recomendación, el tipo de paquete, etc., a los que debemos sumar el tamaño necesario por las cabeceras UDP (8 bytes), IP (20 bytes) y Ethernet (26 bytes). Finalmente, hemos considerado un MTU de 1500 bytes, el habitual en redes *Fast Ethernet*, fragmentando los paquetes cuando sea necesario. Para los paquetes enviados desde el coordinador a todos los servidores de recomendación se han considerado paquetes *multicast*.

9.4. Resultados

Usando el modelo de simulación presentado en la sección anterior, hemos evaluado el rendimiento de la partición por productos y por usuarios a la hora de calcular las recomendaciones para 1.000 usuarios elegidos aleatoriamente del conjunto de datos Netflix.

Los resultados obtenidos se presentan en esta sección. Comenzaremos estudiando el tiempo de respuesta, para analizar a continuación la carga del sistema. Posteriormente veremos la importancia de la memoria *cache* y el desequilibrio entre RS presente en cada una de las dos arquitecturas propuestas.

9.4.1. Estudio del tiempo de respuesta

En primer lugar, hemos evaluado el tiempo de respuesta de cada técnica, es decir, el tiempo necesario para calcular una recomendación. En este primer conjunto de experimentos, se ha ignorado el uso de memoria *cache*, por lo que se asume que los perfiles siempre se deben recuperar del disco.

La Figura 9.9 muestra los resultados obtenidos según el número de servidores de recomendación considerados. Puede observarse que la partición por usuarios obtiene mejores resultados, que es lo que cabría esperar teniendo en cuenta que, al no considerar el uso de *cache*, el tiempo de búsqueda en disco se convierte en el principal cuello de

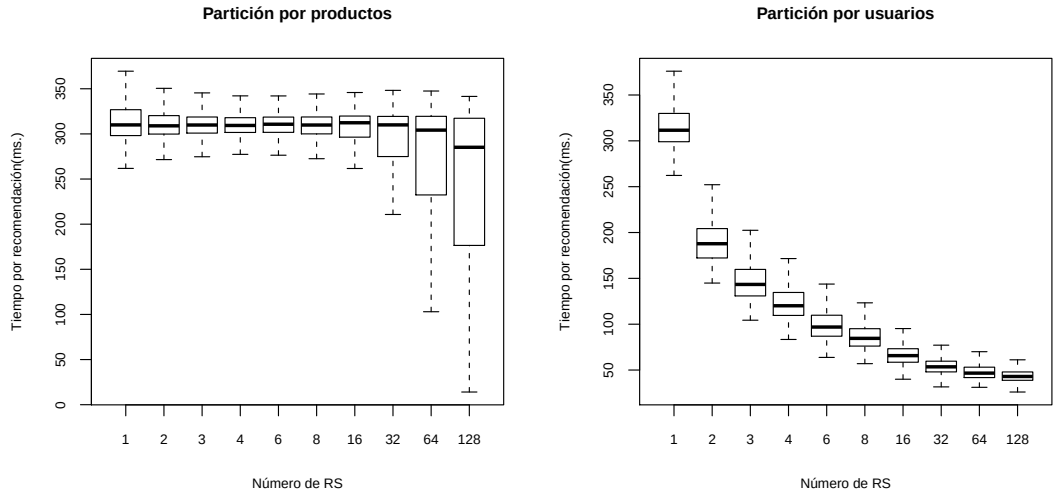


Figura 9.9: Tiempo de respuesta por recomendación, sin considerar el uso de *cache*.

botella. Usando la técnica de partición por usuarios, cada RS debe acceder a un pequeño número de perfiles, por lo que los tiempos de búsqueda se reparten entre varios RS y se pueden realizar en paralelo. De hecho, cuantos más servidores de recomendación tengamos, menos accesos a disco tiene que hacer cada uno de ellos, y por tanto el tiempo de respuesta disminuye. Por el contrario, en la técnica de partición por productos, todos los RS necesitan acceder a todos los perfiles (parciales, en este caso), y el tiempo de respuesta no mejora demasiado. De hecho, sólo llega a mejorar algo cuando el número de RS es muy elevado, ya que en dichos casos el perfil parcial para algún usuario puede ser vacío (debido a que no ha puntuado ninguno de los productos asignados a dicho RS) y, por tanto, se puede evitar el acceso al disco.

Debemos tener en cuenta, sin embargo, que este hecho está íntimamente relacionado con el tamaño del perfil de los usuarios. Si el perfil es relativamente pequeño, el tiempo de búsqueda será mucho mayor que el tiempo necesario para leer y transferir el perfil del disco, y se convertirá por tanto en el cuello de botella. Este es el caso de Netflix y es el motivo de que la partición por usuarios, como hemos visto, obtenga mejores resultados. Esta es una particularidad de los sistemas de recomendación basados en preferencias explícitas, que los diferencia de otros ámbitos como la recuperación de información, o de otro tipo de sistemas de recomendación: en general, los usuarios puntúan un número pequeño de productos y por tanto los perfiles no serán demasiado grandes. Por otra parte, si se considerasen sistemas con preferencias implícitas, o incluso aproximaciones basadas en productos (ver Sección 3.4.1.2), los perfiles serían mayores, y el tiempo de

búsqueda perdería peso respecto al tiempo de transferencia.

En dichos casos, la partición por productos puede ser mejor alternativa que la partición por usuarios. En esta última, al aumentar el número de servidores de recomendación, llegará un punto donde cada RS sólo debe acceder a un perfil, pero tendrá que transferirlo entero. Añadir más servidores no arregla el problema. Sin embargo, en la partición por productos, el perfil parcial será más pequeño a medida que aumentamos el número de servidores, por lo que si el cuello de botella está precisamente en el tiempo de transferencia, el tiempo de respuesta se verá reducido.

Esta tendencia se observa parcialmente en nuestros resultados: el tiempo de respuesta mejora a medida que se añaden más RS, pero mientras en la partición por usuarios la mejora es mayor cuando el número de servidores es pequeño, en la partición por productos lo es con muchos servidores.

Para resumir, podemos decir que, en relación al tiempo de respuesta, la partición por productos funciona mejor con perfiles grandes y muchos servidores de recomendación, y la partición por usuarios con perfiles pequeños y no tantos servidores.

9.4.2. Estudio de la carga del sistema

En cualquier caso, el tiempo de respuesta es sólo una de las dos caras de la moneda. En una aplicación real no sólo importa lo rápido que el sistema es capaz de generar una recomendación, sino también la cantidad de recomendaciones que es capaz de calcular por unidad de tiempo. Es decir, el *throughput*. En ausencia de estudios que nos permitan simular la solicitud de recomendaciones en sistemas reales, en su lugar hemos calculado la carga del sistema, es decir, el porcentaje de los recursos que realmente están trabajando en un momento dado. Cuanto menor sea la carga, mayor será el número de recursos que tendremos libres en un momento dado, y que por tanto podríamos dedicar a procesar otra recomendación. Existe, por tanto, una relación entre el porcentaje de carga y el *throughput*.

En la Figura 9.10 se muestra el porcentaje medio de carga según el número de servidores de recomendación, para ambas técnicas. Puede verse como la partición por usuarios se comporta mejor, que es una vez más lo que cabría esperar, pues mientras en la partición por productos todos los RS tienen que trabajar durante el cálculo de una recomendación, en la partición por usuarios sólo lo han de hacer aquellos que tengan asignados el perfil del usuario actual o de alguno de los vecinos.

De todas formas, es importante destacar que en la afirmación anterior hemos ignorado dos aspectos fundamentales: la carga en el coordinador y la carga en la red. Aunque disminuyésemos la carga de los servidores de recomendación, en caso de que

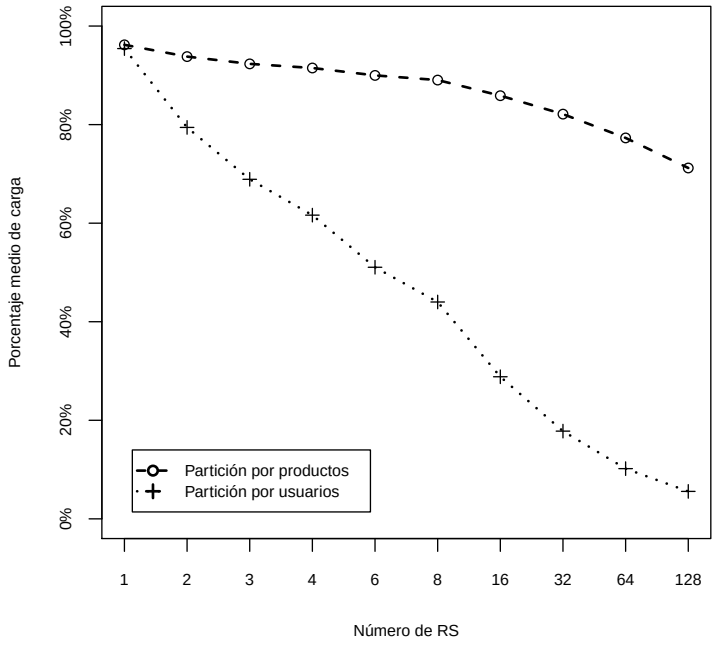


Figura 9.10: Porcentaje medio de carga, según el número de servidores de recomendación.

la red o el coordinador estuviesen colapsados no conseguiríamos mejorar el *throughput*. Por tanto, debemos analizar también las diferencias entre ambas técnicas en lo relativo a estos dos aspectos.

En el coordinador, las principales diferencias se deben a los tiempos de combinación y ordenación, pues el acceso al vecindario se realiza en ambos casos. En la partición por productos, cada RS se encarga de combinar y ordenar sus resultados parciales, por lo que el trabajo final del coordinador se limita a obtener los productos mejor valorados a nivel global. Por contra, en la partición por usuarios es el coordinador el que debe realizar el grueso de las tareas de combinación y ordenación, y por tanto, la carga es mayor en este último caso, tal y como se puede observar en la Figura 9.11. Sin embargo, la diferencia entre ambas técnicas es pequeña, y despreciable frente al tiempo total necesario por la recomendación. Por tanto, en las configuraciones estudiadas el coordinador no es en ningún caso un cuello de botella ni un aspecto diferenciador entre los dos métodos de partición.

Debemos mencionar que aunque en la figura pueda dar la impresión de que en el caso de la partición por usuarios los tiempos aumentan con el número de servidores

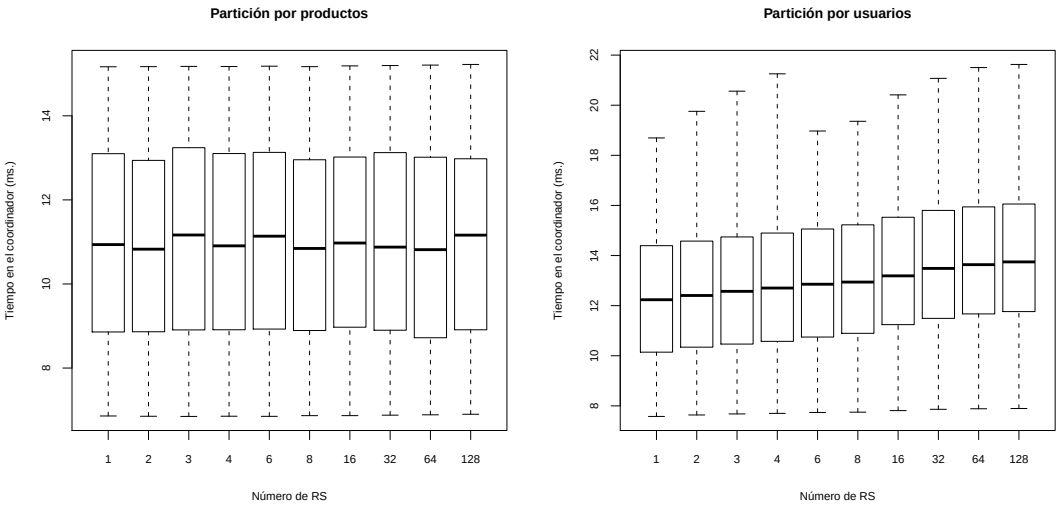


Figura 9.11: Tiempo de procesamiento en el coordinador, según el número de servidores de recomendación.

de recomendación, esto sólo sucede en configuraciones con pocos servidores. Se debe a que cuando tenemos pocos RS, cada uno de ellos posiblemente sea responsable de varios perfiles, y por tanto puede combinarlos y ordenarlos, reduciendo la carga en el coordinador. A medida que el número de servidores aumenta, también aumenta el número de resultados parciales, y por tanto la carga en el coordinador. Sin embargo, ese número nunca va a ser mayor de $k + 1$, por lo que llega un momento en que aumentar el número de servidores no aumenta el número de resultados parciales, y los tiempos se estabilizan. En el caso de la partición por productos eso no es así: cuantos más servidores tengamos, más resultados parciales han de ser procesados en el coordinador. Sin embargo, su tarea se reduce a elegir los productos con mayor puntuación global, lo que se hace en centésimas de milisegundo y es despreciable frente al tiempo de acceso al vecindario. Por tanto, podemos concluir que para que el coordinador se convierta en un cuello de botella, el número de servidores tendría que aumentar significativamente, y en tal caso posiblemente la red se convirtiese en un problema mucho antes, como analizaremos a continuación.

Precisamente, la Figura 9.12 muestra los resultados relativos a la red. El eje y representa el tiempo medio necesario para transmitir un paquete, mientras el eje x representa el tiempo de red agregado, es decir, la suma del tiempo necesario para transmitir todos los paquetes pertenecientes a la misma recomendación. Por ejemplo, si una recomendación necesitase de 2 paquetes, enviados desde 2 RS diferentes, el primero

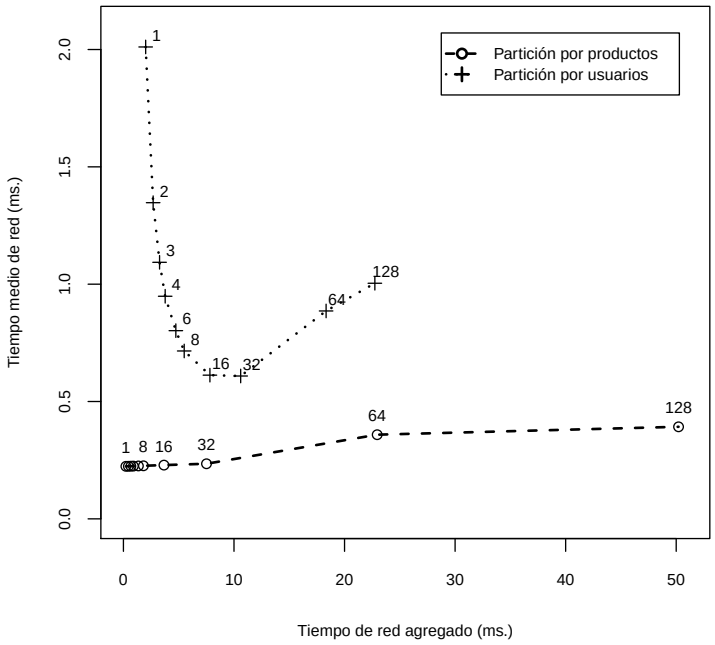
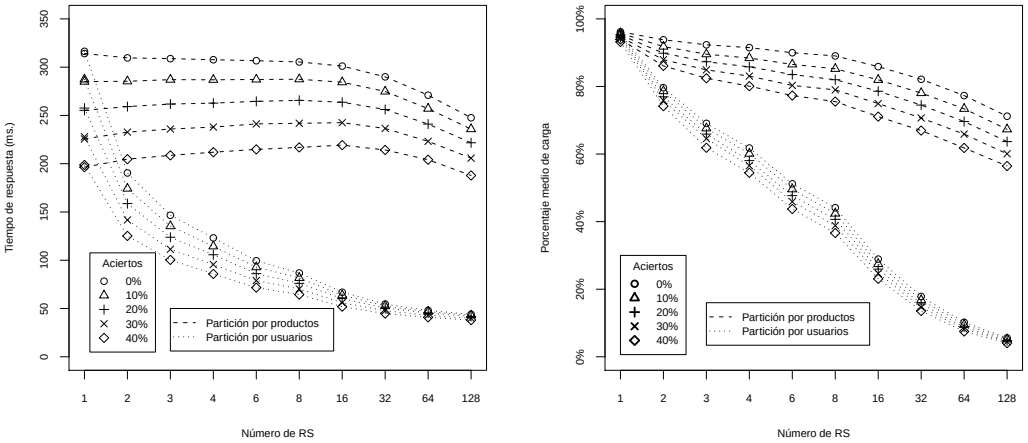


Figura 9.12: Uso de la red según el número de servidores de recomendación.

de los cuales tarda 1 ms. en llegar al coordinador, y el segundo 2 ms., el tiempo medio sería de $1,5$ ms., y el tiempo agregado sería de 3 ms. Obsérvese que el tiempo agregado es una medida artificial, y no quiere decir que la red estuvo ocupada durante 3 ms. Por ejemplo, pudo suceder que realmente el segundo paquete sólo tardó 1 ms. en atravesar la red, pero estuvo 1 ms. adicional esperando en una cola a que el coordinador acabase de recibir el primer paquete. Es decir, que en la práctica puede que ambos paquetes fuesen transmitidos en un intervalo de sólo 2 ms. Sin embargo, es una buena medida de la carga real de la red.

De hecho, en cierto modo el tiempo medio por paquete representa el tiempo de respuesta, mientras el tiempo agregado representa la carga. Considerar ambas métricas nos ayuda a entender mejor el uso que hacen de la red ambas técnicas.

En la partición por productos, cada RS puede descartar algunos productos, y al coordinador sólo se envían un máximo de N resultados. Por tanto, el tiempo medio por paquete es menor, ya que estos son pequeños y no necesitan ser fragmentados. Sin embargo, debido a que todos los servidores deben responder al coordinador, la carga de red aumenta significativamente según aumenta el número de servidores, hasta



(a) Tiempo de respuesta **(b)** Porcentaje medio de carga

Figura 9.13: Tiempo de respuesta y carga según el porcentaje de aciertos de la *cache*.

el punto de que la red puede llegar a convertirse en un cuello de botella en caso de que tengamos muchos servidores. Además, aumentar el número de servidores tampoco reduce el tiempo por paquete, ya que esto apenas afecta a su tamaño.

Por el contrario, en la partición por usuarios el tiempo medio por paquete es mayor, ya que los paquetes son más grandes y a menudo necesitan ser fragmentados, pero por otra parte la carga de red es menor, porque sólo algunos RS necesitan realmente responder al coordinador. Además, según aumenta el número de RS también disminuye el tamaño de cada paquete, y por tanto el tiempo de transmisión necesario.

Finalmente, es importante destacar que el aumento de tiempo de red entre 32 y 64 RS no está relacionado con el tamaño del paquete, sino con el hecho de que debemos atravesar un mayor número de *switches*. Esto es debido a que en la simulación estamos considerando *switches* de 48 puertos, por lo que con 64 y 128 RS debemos usar dos niveles de *switches*.

9.4.3. Influencia del uso de la *cache*

Los resultados según el porcentaje de aciertos de la *cache* se muestran en las Figuras 9.13a y 9.13b. Por una parte, podemos observar que el uso de la *cache* no influye demasiado en las particularidades de cada técnica, y por tanto el comportamiento es, en general, similar al reportado anteriormente. Por otra, y como cabría esperar, cuanto mayor sea el porcentaje de aciertos, mejores serán los resultados obtenidos, lo cual es

lógico ya que el acceso a *cache* es varios órdenes de magnitud más rápido que el acceso a disco.

Por otra parte, sí se observa que el aumento de aciertos de *cache* supone una mejora mayor en la partición por productos. Esto también es esperado, ya que el mayor cuello de botella de esa técnica era el tiempo de búsqueda en disco, el cual se evita en caso de que los datos estén en la *cache*. Por contra, en la partición por usuarios el beneficio es menor, especialmente si el número de servidores es grande, ya que con tal de que un único RS necesite acceder a disco, el coordinador tendrá que esperar por él y el tiempo de respuesta no se verá afectado significativamente.

9.4.4. Desequilibrio entre servidores de recomendación

Por último, hemos estudiado el desequilibrio existente en las configuraciones analizadas. Basándonos en la definición presentada por Badue et al. [2007], definimos el desequilibrio como el cociente entre el tiempo máximo y medio necesario para computar una recomendación en los distintos servidores. Un desequilibrio cercano a 1 significa que la carga está bien balanceada entre los distintos servidores, lo cual es generalmente deseable. Cuanto más dependa el tiempo de recomendación de un servidor particularmente lento, mayor será el desequilibrio.

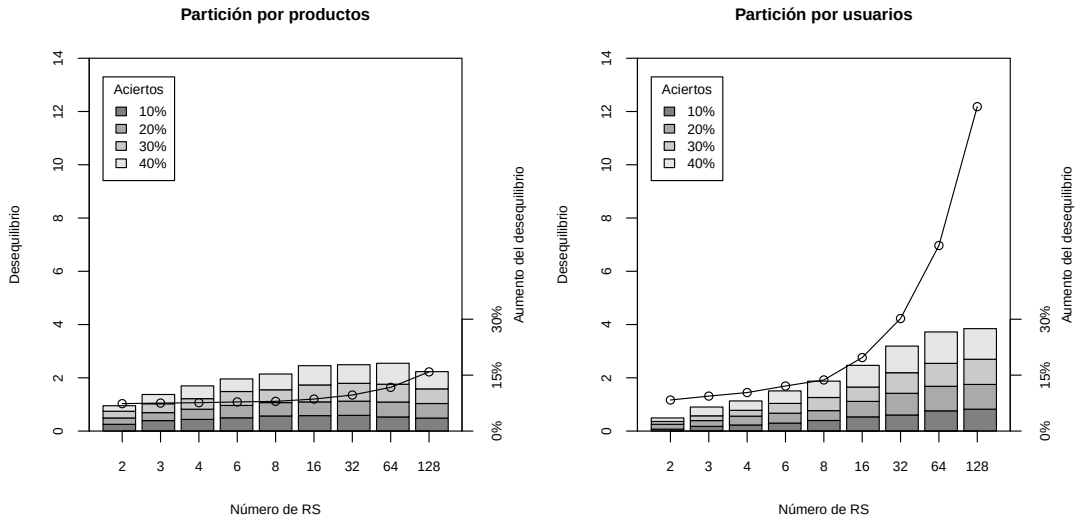


Figura 9.14: Desequilibrio medio para la partición por productos y usuarios. Las líneas y puntos representan el desequilibrio con la *cache* desactivada. Las barras, el incremento del desequilibrio con distintos porcentajes de aciertos de *cache*, expresado como el porcentaje relativo al desequilibrio sin *cache*.

La Figura 9.14 muestra el desequilibrio medio para ambas técnicas. Hemos estudiado

cómo varia el desequilibrio en relación al número de servidores de recomendación y al porcentaje de aciertos de *cache*, comprobando que ambos factores tienen una influencia importante en el mismo.

La partición por usuarios es la que presenta un mayor desequilibrio, lo que se debe a que, mientras en la partición por productos todos los RS necesitan recuperar un perfil parcial, en la partición por usuarios sólo lo tendrán que hacer algunos RS. Cuando el número de RS es grande, muchos de ellos estarán desocupados durante el procesamiento de una recomendación en concreto. En principio, esto puede parecer un problema poco importante, porque esos servidores podrían dedicarse a procesar otras recomendaciones en paralelo. Sin embargo, tal y como se había discutido en el Capítulo 8 (ver Figura 8.7), el acceso a los perfiles de los usuarios suele ser muy sesgado: hay unos pocos usuarios que forman parte de un gran número de vecindarios, debido a que son usuarios muy activos, con muchas puntuaciones a productos populares. Esto se debe tener en cuenta a la hora de hacer la partición, para evitar este tipo de desequilibrios. En la partición por productos este problema tiene mucha menos importancia.

Por otra parte, en cuanto a la influencia de la *cache*, el resultado es el esperado: cuando mayor el porcentaje de aciertos, mayor el desequilibrio, ya que los servidores que operan sobre la *cache* serán más rápidos. Además, podemos observar diferencias entre ambas técnicas de partición. En la partición por usuarios, el aumento del desequilibrio debido a la *cache* aumenta con el número de RS. Esto es debido a que con muchos servidores, es poco probable que un servidor tenga que acceder a más de un perfil. Si está en la *cache* el servidor responderá muy rápido, pero si no está será mucho más lento, y por tanto las diferencias entre servidores serán grandes. Por contra, cuando el número de servidores es pequeño, cada uno de ellos tendrá que acceder a varios perfiles, alguno de los cuales estará en *cache*, y otros no lo estarán, por lo que al final las diferencias entre servidores son más pequeñas. En la partición por productos, aunque tengamos muchos servidores, todos deberán acceder a varios perfiles parciales, por lo que la influencia de la *cache* será similar en todos ellos, y por tanto el desequilibrio será menor.

9.5. Conclusiones y trabajos futuros

En este capítulo hemos propuesto y evaluado dos técnicas para distribuir algoritmos kNN, ambas basadas en la partición de la matriz de puntuaciones. En la partición por usuarios, cada trozo contiene un subconjunto de los perfiles de los usuarios. En cambio, en la partición por productos, cada trozo consiste en un perfil parcial de cada usuario,

donde sólo se almacenan las puntuaciones de un subconjunto de los productos.

Hemos mostrado como ambas técnicas permiten incrementar el *throughput* y reducir el tiempo de respuesta de las recomendaciones. En general, hemos visto que la partición por usuarios es una técnica más adecuada, al menos en aplicaciones donde los perfiles de los usuarios son relativamente pequeños, situación habitual en muchos sistemas basados en preferencias explícitas. Al aumentar el número de RS, el tiempo de respuesta se reduce más rápido con la técnica de partición por usuarios. Además, la carga media en cada RS también es más pequeña, y lo mismo sucede con la carga en la red, lo que hace que esta técnica sea más escalable. Por contra, en sistemas con perfiles de usuario relativamente grandes (lo que podría suceder en aplicaciones basadas en preferencias implícitas), la partición por productos podría ser una buena alternativa.

Para la evaluación de las distintas técnicas hemos simulado el comportamiento del sistema, lo que permite realizar multitud de experimentos de forma sencilla, rápida, y mucho más económica que con el uso de un sistema real. Hemos diseñado un modelo de simulación específico para filtrado colaborativo a partir de los usados en RI, estudiando los principales parámetros que caracterizan el rendimiento de estas técnicas, e incluyendo las diferentes fases del proceso de recomendación: acceso a perfil de usuario y vecindario, combinación de resultados y ordenación de los mismos. El modelo ha sido comparado con un sistema real para validar sus resultados.

En futuros trabajos pretendemos extender este estudio a arquitecturas más complejas o que combinen la distribución con la replicación del sistema completo o parte del mismo. También tenemos pensado mejorar el modelo de simulación, considerando fuentes de variación adicionales, incluyendo la simulación de la *cache* o el modelo de red.

Capítulo 10

Conclusiones y trabajos futuros

En esta tesis hemos desarrollado técnicas que mejoran diversos aspectos de los sistemas de recomendación, principalmente aquellos relacionados con la eficiencia y escalabilidad. Hemos visto como se pueden diseñar algoritmos sencillos capaces de ofrecer resultados similares o incluso mejores a técnicas más complicadas, y hemos comprobado la utilidad de los algoritmos kNN y su buen comportamiento en recomendación. Entre otros aspectos, hemos visto cómo se pueden beneficiar de técnicas desarrolladas en el ámbito de la recuperación de información, lo que no sólo permite mejorar la eficiencia de estos sistemas, sino que da lugar a líneas de investigación especialmente prometedoras que se desarrollarán en trabajos futuros.

En este capítulo presentamos en primer lugar las conclusiones obtenidas durante el desarrollo de esta tesis, para introducir a continuación las principales ideas y cuestiones que han quedado pendientes y que se abordarán en el futuro.

10.1. Principales conclusiones

En esta tesis se han abordado diversos problemas que afectan a los sistemas de recomendación basados en filtrado colaborativo, en especial aquellos relacionados con la eficiencia y escalabilidad de estos sistemas. Con un enfoque centrado en el desarrollo de técnicas sencillas y fáciles de usar en aplicaciones reales, durante la elaboración de este trabajo se han sacado diversas conclusiones que se expondrán a lo largo de esta sección.

En primer lugar, hemos comprobado que es posible contar con técnicas de recomendación precisas y eficaces sin necesidad de recurrir a complejos modelos con multitud de factores y parámetros a estimar. Las ventajas de utilizar algoritmos sencillos son numerosas e incluyen:

- Una mejor comprensión del funcionamiento del algoritmo y sus características, lo que simplifica el diseño de variantes o pequeñas modificaciones para adaptarse a condiciones o requisitos particulares de la aplicación.
- Una mayor eficiencia, tanto en el entrenamiento del sistema como en la recomendación, lo que permite su uso en entornos con gran volumen de usuarios o productos.
- Una implementación más sencilla, lo que facilita la optimización de la misma.
- Menor dependencia de contar con suficientes datos para entrenar y adaptar el sistema al dominio en que va a ser usado, lo que en general repercute en un mejor funcionamiento en condiciones de baja densidad o cuando los datos difieren de las condiciones de entrenamiento.

En nuestra opinión, las ventajas que ofrecen este tipo de técnicas tienen una mayor importancia que el conseguir mejorar la precisión unas pocas décimas a cambio de complicar significativamente el diseño del algoritmo, que sin embargo ha sido el objetivo perseguido por gran parte de los trabajos que podemos encontrar en la literatura.

Como hemos demostrado en esta tesis, es posible desarrollar algoritmos sencillos que no sólo son capaces de competir con técnicas mucho más complejas en aspectos tales como la precisión de las recomendaciones, sino que incluso ofrecen resultados superiores cuando se tiene en cuenta el funcionamiento global del algoritmo en multitud de condiciones, su evolución con la densidad de información, etc. En concreto, hemos propuesto dos técnicas que destacan por sus buenos resultados y sencillez.

En el Capítulo 4 hemos presentado la técnica de filtrado colaborativo basado en tendencias, algoritmo basado en modelar de forma sencilla las diferencias entre usuarios y productos. El enfoque es particularmente novedoso, por cuanto la mayor parte de técnicas existentes se han centrado en representar o identificar las similitudes entre ellos. Hemos comprobado como, en general, las diferencias son más fáciles de calcular que las similitudes, con lo que nuestro enfoque no sólo da lugar a un sistema más eficiente, sino que permite obtener buenos resultados en condiciones de baja densidad. Mejora así los resultados de técnicas tradicionales en esos casos, donde generalmente se habían obtenido resultados más pobres. En gran medida esto se debe a la relativa complejidad de los modelos y medidas de similitud usadas tradicionalmente, lo que dificulta su cálculo cuando la información disponible es escasa. En cambio, las tendencias pueden ser calculadas eficazmente a partir de menos información, lo que permite obtener resultados más precisos en condiciones adversas.

Posteriormente, **en el Capítulo 5 hemos presentado la técnica *Good Items* *k*NN**, la cual incorpora varias mejoras a los algoritmos kNN, especialmente diseñadas para la tarea de recomendación. Hemos comprobado la importancia de simplificar el cálculo de las medidas de similitud para mejorar el comportamiento cuando no se dispone de suficiente información. También hemos comprobado las diferencias entre las tareas de predicción y recomendación, y la importancia de diseñar técnicas dedicadas especialmente a la tarea en que se va a utilizar el algoritmo. En particular, hemos propuesto una nueva medida de similitud así como una nueva forma de seleccionar los productos a recomendar, las cuales mejoran significativamente la precisión del algoritmo.

Además, las variantes introducidas nos han permitido comprobar la versatilidad de los algoritmos kNN, y la facilidad con la que se pueden adaptar a un propósito concreto, muestra evidente de las ventajas de técnicas sencillas e intuitivas. De hecho, los algoritmos kNN son muy populares en la literatura por este y otros motivos. Sus detractores critican sobre todo su pobre comportamiento en situaciones de *cold-start*, es decir, cuando se tiene poca información acerca de las preferencias de usuarios o productos, o su bajo rendimiento debido sobre todo al coste de buscar similitudes entre usuarios y seleccionar los vecinos.

Sin embargo, en este trabajo hemos visto como los principales problemas que sufren estas técnicas pueden ser abordados de forma satisfactoria con relativamente poco esfuerzo. **En el Capítulo 6 hemos presentado la técnica de expansión del perfil, la cual resuelve satisfactoriamente el problema de *cold-start*** en caso de nuevos usuarios para los que apenas se dispone de puntuaciones, consiguiendo mejoras de la precisión de hasta el 100 % en estos casos. Además, no necesitan más información que aquella ya presente en la matriz de puntuaciones, lo que es una novedad frente a la mayoría de aproximaciones presentadas con anterioridad. Inspirada en las técnicas de expansión de consultas, su buen funcionamiento sirve de ejemplo de las ventajas que puede tener el acercar el mundo de la recomendación con el de recuperación de información, sin duda una de las principales conclusiones que se han sacado durante el desarrollo de este trabajo.

De hecho, esta aproximación ha tenido una gran importancia en la última parte de la tesis, en la cual hemos presentado numerosas técnicas para mejorar la eficiencia de los algoritmos kNN. **En el Capítulo 7 hemos visto como la preselección de vecinos mejora significativamente el tiempo necesario para realizar la recomendación**, sin que se produzca un deterioro importante de su calidad. La ventaja de esta aproximación frente a técnicas basadas en modelo es que permite tener en cuenta las nuevas puntuaciones sin necesidad de volver a calcular el vecindario, mostrando que

los algoritmos kNN pueden ser tan eficientes como la mayoría de las técnicas basadas en modelo.

Además, una de las principales ventajas de los algoritmos kNN es que pueden ser implementados utilizando un enfoque similar al empleado en RI, en concreto basándonos en el modelo vectorial. De esta forma, estos algoritmos pueden beneficiarse de muchas de las técnicas que permiten a los motores de búsqueda manejar cantidades ingentes de información y aún así ser capaces de responder en unos pocos milisegundos a las consultas de millones de usuarios. Estas técnicas representan el resultado de años de investigación, y el poder aprovecharlas en filtrado colaborativo supone una oportunidad inmejorable. Entre otras ventajas, posibilita la recomendación a gran escala y amplía las posibilidades de este tipo de técnicas, permitiendo su uso eficaz en nuevos dominios y aplicaciones hasta ahora prohibitivos debido al volumen de datos que potencialmente se tendría que manejar.

En particular, **en el Capítulo 7 hemos visto cómo se pueden implementar de forma eficiente mediante el uso de índices**. La naturaleza dispersa de la matriz de puntuaciones hace adecuada una implementación de este tipo, lo que agiliza tanto el cálculo de vecinos como la fase de recomendación. Posteriormente, **en el Capítulo 8 mostramos cómo técnicas similares a las utilizadas en RI para la compresión de índices pueden ser empleadas en filtrado colaborativo**, reduciendo el tamaño de la matriz de puntuaciones en una proporción 1:3, lo que repercute tanto en la reducción de costes de almacenamiento como en el tiempo necesario para calcular una recomendación. En concreto, se ha estudiado la compresión de identificadores y puntuaciones utilizando diferentes métodos de codificación, así como una novedosa técnica de reasignación de identificadores.

Finalmente, **en el Capítulo 9 hemos explorado la recomendación en entornos distribuidos**, imprescindibles para garantizar la escalabilidad de los sistemas en aplicaciones especialmente exigentes. Hemos propuesto dos arquitecturas que permiten la mejora de los tiempos de respuesta y *throughput* de las recomendaciones, consistentes en distribuir la matriz de puntuaciones y su procesamiento entre diversos nodos: la distribución basada en usuarios y la distribución basada en productos. En general la primera es más adecuada en caso de que el tamaño de los perfiles de los usuarios no sea demasiado elevado, lo que suele ser más habitual en sistemas basados en preferencias explícitas. Para la evaluación se ha desarrollado un modelo de simulación, y se ha comprobado su relación con el comportamiento de un sistema real, lo que demuestra la utilidad de este tipo de técnicas para el estudio y evaluación del rendimiento.

10.2. Futuros trabajos

Este trabajo ha abordado diversos aspectos que afectan a la calidad y eficiencia de los sistemas de filtrado colaborativo, en especial los relacionados con los algoritmos basados en vecinos. Precisamente debido a la variedad de problemas abordados y técnicas propuestas, así como a la lógica limitación temporal que nos debemos marcar a la hora de realizar una tesis doctoral, en el camino han quedado pendientes aspectos que no ha sido posible abordar y que serán objeto de futuros trabajos.

En primer lugar, el algoritmo *Good Items kNN* presentado en el Capítulo 5 obtiene muy buenos resultados pese a ser una primera aproximación al problema. Es interesante, por tanto, dedicar futuros trabajos a profundizar en esta técnica y en posibles mejoras a la misma. De hecho, ya en ese mismo capítulo se han presentado numerosas mejoras tanto a la medida de similitud como al método para el cálculo de pesos, las cuales, de confirmarse nuestras expectativas, mejorarían significativamente los ya de por sí prometedores resultados de este algoritmo. Otra posibilidad sería el estudio de este algoritmo en contextos de preferencias unarias, es decir, en filtrado colaborativo *one class*.

Una situación similar se produce con las técnicas de expansión del perfil presentadas en el Capítulo 6, donde también hemos obtenido resultados muy buenos a pesar de haber estudiado un conjunto limitado de métodos. En futuros trabajos se estudiarán diversas alternativas, muchas de ellas ya introducidas en dicho capítulo. Entre ellas, destaca la posibilidad de convertir la expansión del perfil en un mecanismo para combinar diferentes tipos de algoritmos, es decir, en el desarrollo de sistemas híbridos. A mayores del algoritmo kNN, se utilizaría una segunda técnica cuyo objetivo sería únicamente obtener los productos a partir de los cuales expandir el perfil. Por ejemplo, un algoritmo basado en contenido se podría aplicar como una técnica global basada en productos, o un algoritmo demográfico como una técnica global basada en usuarios. Se podría incluso utilizar otro algoritmo de filtrado colaborativo, dando lugar a una nueva generación de técnicas locales basadas en productos. En este caso, se elegiría un algoritmo que se comporte especialmente bien en contextos de baja densidad.

Otra posibilidad a explorar es una expansión del perfil de orden superior, es decir, el aplicar la expansión del perfil repetidas veces. El número de veces determinaría el orden de la técnica, y posiblemente un mayor orden mejoraría los resultados en las condiciones más extremas.

Finalmente, el éxito de la implementación de algoritmos kNN basándonos en un enfoque similar al empleado en recuperación de información propuesto en la última parte

de este trabajo da lugar a numerosas líneas de investigación que serán desarrolladas en el futuro.

Por una parte, es especialmente prometedor el uso de técnicas orientadas a mejorar la eficiencia de los algoritmos, aspecto que ha concentrado gran parte de los esfuerzos dedicados al desarrollo de este trabajo. El principal objetivo a abordar sigue siendo la optimización de la fase de selección de vecinos, la más costosa en algoritmos kNN. Si bien hemos visto como la preselección minimiza en gran medida este problema, en ciertos dominios va a seguir siendo necesario actualizar el vecindario con cierta frecuencia.

Afortunadamente, es muy posible que técnicas utilizadas en RI puedan aportar nuevas optimizaciones en este tipo de tareas. Los mecanismos que hemos estudiado para la compresión de la matriz de puntuaciones o la ejecución en entornos distribuidos han presentado resultados excelentes, pero realmente son una adaptación de técnicas básicas en el mundo de la RI. El estudio de aproximaciones más avanzadas es, sin lugar a dudas, una de las líneas de investigación que parecen más prometedoras para el futuro.

De igual forma, sería muy interesante el adaptar otro tipo de técnicas que pese a su popularidad en RI no ha sido posible abordar en este trabajo. En particular, las técnicas de *prunning* parecen especialmente interesantes en este ámbito. Consisten en descartar aquella información que se estima que no va a tener un impacto elevado en la selección de resultados, con el objetivo de reducir los datos que realmente se deben procesar y por tanto mejorar los tiempos de consulta, el tamaño del índice, etc. Con el algoritmo GIkNN presentado en este trabajo, por ejemplo, las puntuaciones negativas no van a ser usadas en ningún momento, por lo que podrían ser directamente eliminadas de la matriz. Esta y otras técnicas serán estudiadas en futuros trabajos.

Por otra parte, las ventajas de unificar RI y recomendación van más allá de soluciones a los problemas de eficiencia. Muchas de las técnicas desarrolladas para mejorar la precisión de los motores de búsqueda posiblemente se puedan adaptar con éxito a la recomendación, y serán estudiadas en futuros trabajos. Además, el representar la recomendación según un modelo vectorial y tratarlo como si fuese la consulta en un buscador da lugar a variaciones interesantes. Por ejemplo, un problema particularmente atractivo es la recomendación según el contexto [Adomavicius y Tuzhilin, 2008], donde la respuesta del sistema vendría determinada no sólo por las preferencias del usuario, sino también por el contexto en el que se solicita la recomendación: tiempo, lugar, compañía, etc. En un modelo vectorial, se podría abordar de forma sencilla mediante una técnica de *reducción del perfil*, en el que se eliminarían del perfil del usuario aquellas

puntuaciones llevadas a cabo en un contexto diferente al actual. Por ejemplo, en un sistema de recomendación de destinos de vacaciones, podrían eliminarse aquellas puntuaciones realizadas en un periodo del año distinto al actual. Otra posibilidad sería la recomendación para grupos de personas, donde como perfil se utilizaría la intersección de los perfiles de todos ellos.

En definitiva, esta tesis no sólo ha supuesto un buen número de contribuciones al mundo del filtrado colaborativo, sino que abre las puertas a numerosas y prometedoras lineas de investigación para el futuro.

Glosario

A continuación se presenta una lista de abreviaturas y términos usados a lo largo del presente documento, acompañados de una breve descripción. En muchos casos esta descripción se ha basado en la definición disponible en la Wikipedia[1].

CPU Del inglés *Central Processing Unit*, y también conocido simplemente como procesador o microprocesador, es el componente principal del ordenador y otros dispositivos programables, que interpreta las instrucciones contenidas en los programas y procesa los datos.

Ethernet Familia de tecnologías para la transmisión de datos en redes de área local (LAN).

GB Abreviatura de *gigabyte*, unidad de almacenamiento de la información, equivalente a 10^9 bytes.

GiB Abreviatura de *gibibyte*, unidad de almacenamiento de la información, equivalente a 2^{30} bytes.

IP Del inglés *Internet Protocol*, es un protocolo de comunicación para el envío de datagramas entre redes de computadores, y cuyas principales funciones son el direccionamiento y enrutamiento.

LAN Del inglés *Local Area Network* o red de área local, es una red que interconecta computadoras en una oficina, laboratorio, o edificio, caracterizada por una extensión geográfica limitada, altas tasas de transferencia y el uso de un medio de comunicación privado.

Mbps Unidad de medida de la tasa de transferencia en una red de comunicaciones equivalente a 1.000.000 de bits por segundo.

[1] http://es.wikipedia.org

MiB Abreviatura de *mebibyte*, unidad de almacenamiento de la información, equivalente a 2^{20} bytes.

MTU Del inglés *Maximum Transfer Unit*, es un término de redes de computadoras que expresa el tamaño en bytes de la unidad de datos más grande que puede enviarse usando un protocolo de comunicaciones.

Multicast Envío de la información en una red a múltiples destinos simultáneamente.

P2P Del inglés *peer-to-peer*, red de computadoras en la que todos o algunos aspectos funcionan sin clientes ni servidores fijos, sino una serie de nodos que se comportan como iguales entre si, es decir, actúan simultáneamente como clientes y servidores respecto a los demás nodos de la red.

RAM Del inglés *random-access memory*, espacio de almacenamiento temporal en un ordenador, que se utiliza como memoria de trabajo para el sistema operativo, los programas y la mayoría del software.

RI Recuperación de Información, ciencia de la búsqueda de información en documentos electrónicos y cualquier tipo de colección documental digital.

Switch Dispositivo de interconexión de redes de computadoras que opera en la capa de enlace, cuya función es interconectar dos o más segmentos de red, pasando datos de un segmento a otro de acuerdo con la dirección MAC de destino de las tramas en la red. En español se conoce con el nombre de conmutador.

UDP Del inglés *User Datagram Protocol*, es un protocolo del nivel de transporte basado en el intercambio de datagramas, que no necesita en establecimiento previo de una conexión, ni tiene confirmación ni control de flujo.

Bibliografía

Gediminas Adomavicius y Alexander Tuzhilin. Context-aware recommender systems. En *Proceedings of the 2008 ACM conference on Recommender systems*, RecSys '08, páginas 335–336, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-093-7. 212

Hyung Jun Ahn. A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem. *Inf. Sci.*, 178:37–51, Enero 2008. ISSN 0020-0255. 126

Jae-wook Ahn, Peter Brusilovsky, Jonathan Grady, Daqing He, y Sue Yeon Syn. Open user profiles for adaptive news systems: help or harm? En *Proceedings of the 16th international conference on World Wide Web*, WWW '07, páginas 11–20, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-654-7. 29

Kamal Ali y Wijnand van Stam. Tivo: making show recommendations using a distributed collaborative filtering architecture. En *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '04, páginas 394–401, New York, NY, USA, 2004. ACM. ISBN 1-58113-888-1. 147

Xavier Amatriain, Josep M. Pujol, y Nuria Oliver. I like it... i like it not: Evaluating user ratings noise in recommender systems. En *Proceedings of the 17th International Conference on User Modeling, Adaptation, and Personalization: formerly UM and AH*, UMAP '09, páginas 247–258, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-02246-3. 26

Chris Anderson. *The Long Tail: Why the Future of Business Is Selling Less of More*. Hyperion, 2006. ISBN 1401302378. 19, 168

R. Attar y A. S. Fraenkel. Local feedback in full-text retrieval systems. *J. ACM*, 24: 397–417, Julio 1977. ISSN 0004-5411. 127, 134

Esma Aïmeur, Gilles Brassard, José; M. Fernandez, y Flavien Serge Mani Onana. Alambic: a privacy-preserving recommender system for electronic commerce. *Int. J. Inf. Secur.*, 7(5):307–334, Septiembre 2008. ISSN 1615-5262. 42

C. S. Badue, R. Baeza-Yates, B. Ribeiro-Neto, A. Ziviani, y N. Ziviani. Analyzing imbalance among homogeneous index servers in a web search system. *Inf. Process. Manage.*, 43:592–608, Mayo 2007. ISSN 0306-4573. 172, 203

Ricardo Baeza-Yates y Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Publishing Company, USA, 2nd edition, 2008. ISBN 0321416910, 9780321416919. 127

BIBLIOGRAFÍA

Marko Balabanović y Yoav Shoham. Fab: content-based, collaborative recommendation. *Commun. ACM*, 40(3):66–72, 1997. ISSN 0001-0782. 32, 35, 125

Ranieri Baraglia, Fidel Cacheda, Victor Carneiro, Diego Fernandez, Vreixo Formoso, Raffaele Perego, y Fabrizio Silvestri. Search shortcuts: a new approach to the recommendation of queries. En *Proceedings of the third ACM conference on Recommender systems*, RecSys '09, páginas 77–84, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-435-5. 100

Chumki Basu, Haym Hirsh, y William Cohen. Recommendation as classification: using social and content-based information in recommendation. En *Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, AAAI '98/IAAI '98, páginas 714–720, Menlo Park, CA, USA, 1998. American Association for Artificial Intelligence. ISBN 0-262-51098-7. 32

Alejandro Bellogín, Jun Wang, y Pablo Castells. Text retrieval methods for item ranking in collaborative filtering. En Paul Clough, Colum Foley, Cathal Gurrin, Gareth Jones, Wessel Kraaij, Hyowon Lee, y Vanessa Mudoch, editores, *Advances in Information Retrieval*, volumen 6611 de *Lecture Notes in Computer Science*, páginas 301–306. Springer Berlin / Heidelberg, 2011. 148

James Bennett y Stam Lanning. The netflix prize. En *Proceedings of KDD Cup and Workshop*, KDDCup '07, páginas 3–6, San Jose, California, USA, 2007. ACM. 5, 33, 68, 75, 167

Roi Blanco y Alvaro Barreiro. Characterization of a simple case of the reassignment of document identifiers as a pattern sequencing problem. En *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, páginas 587–588, New York, NY, USA, 2005. ACM. ISBN 1-59593-034-5. 176

Roi Blanco y Álvaro Barreiro. TSP and cluster-based solutions to the reassignment of document identifiers. *Inf. Retr.*, 9:499–517, Septiembre 2006. ISSN 1386-4564. 176

Dan Blandford y Guy Blelloch. Index compression through document reordering. En *Proceedings of the Data Compression Conference*, DCC '02, páginas 342–351, Washington, DC, USA, 2002. IEEE Computer Society. 176

Paolo Boldi y Sebastiano Vigna. Codes for the World Wide Web. *Internet Math.*, 2(4): 407–429, 2005. 178

Kurt D. Bollacker, Steve Lawrence, y C. Lee Giles. Citeseer: an autonomous web agent for automatic retrieval and identification of interesting publications. En *Proceedings of the second international conference on Autonomous agents*, AGENTS '98, páginas 116–123, New York, NY, USA, 1998. ACM. ISBN 0-89791-983-1. 29

Dirk Bollen, Bart P. Knijnenburg, Martijn C. Willemsen, y Mark Graus. Understanding choice overload in recommender systems. En *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, páginas 63–70, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-906-0. 5

John S. Breese, David Heckerman, y Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. En *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, UAI'98, páginas 43–52, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. ISBN 1-55860-555-X. 35, 47, 54, 57, 73, 109

Lukas Brozovsky y Vaclav Petricek. Recommender system for online dating service. En *Proceedings of Znalosti 2007 Conference*, Ostrava, 2007. VSB. 167

Robin Burke. Knowledge-based recommender systems. *Encyclopedia of Library and Information Science*, 69(32):180–200, 2000. 31

Robin Burke. Hybrid web recommender systems. En Peter Brusilovsky, Alfred Kobsa, y Wolfgang Nejdl, editores, *The Adaptive Web*, volumen 4321 de *Lecture Notes in Computer Science*, páginas 377–408. Springer-Verlag, Berlin, Heidelberg, 2007. ISBN 978-3-540-72078-2. 32

Stefan Büttcher, Charles Clarke, y Gordon V. Cormack. *Information Retrieval: Implementing and Evaluating Search Engines*. The MIT Press, 2010. ISBN 0262026511, 9780262026512. 182

Fidel Cacheda, Victor Carneiro, Vassilis Plachouras, y Iadh Ounis. Performance analysis of distributed information retrieval architectures using an improved network simulation model. *Inf. Process. Manage.*, 43(1):204–224, 2007. ISSN 0306-4573. 11, 182, 184, 187, 196

Fidel Cacheda, Víctor Carneiro, Diego Fernández, y Vreixo Formoso. Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems. *ACM Trans. Web*, 5:2:1–2:33, Febrero 2011a. ISSN 1559-1131. 34, 109, 153

Fidel Cacheda, Victor Carneiro, Diego Fernández, y Vreixo Formoso. Improving k-nearest neighbors algorithms: practical application of dataset analysis. En *Proceedings of the 20th ACM international conference on Information and knowledge management*, CIKM '11, páginas 2253–2256, New York, NY, USA, 2011b. ACM. ISBN 978-1-4503-0717-8. 93

Liren Chen y Katia Sycara. Webmate: a personal agent for browsing and searching. En *Proceedings of the second international conference on Autonomous agents*, AGENTS '98, páginas 132–139, New York, NY, USA, 1998. ACM. ISBN 0-89791-983-1. 29

Paul-Alexandru Chirita, Wolfgang Nejdl, y Cristian Zamfir. Preventing shilling attacks in online recommender systems. En *Proceedings of the 7th annual ACM international workshop on Web information and data management*, WIDM '05, páginas 67–74, New York, NY, USA, 2005. ACM. ISBN 1-59593-194-5. 42

Dan Cosley, Shyong K. Lam, Istvan Albert, Joseph A. Konstan, y John Riedl. Is seeing believing?: how recommender system interfaces affect users' opinions. En *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '03, páginas 585–592, New York, NY, USA, 2003. ACM. ISBN 1-58113-630-7. 25

Rickard Cöster y Martin Svensson. Inverted file search algorithms for collaborative filtering. En *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '02, páginas 246–252, New York, NY, USA, 2002. ACM. ISBN 1-58113-561-0. 10, 147, 148, 166

BIBLIOGRAFÍA

Paolo Cremonesi, Yehuda Koren, y Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. En *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, páginas 39–46, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-906-0. 29, 100, 110

Carolyn J. Crouch y Bokyung Yang. Experiments in automatic statistical thesaurus construction. En *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '92, páginas 77–88, New York, NY, USA, 1992. ACM. ISBN 0-89791-523-2. 128

Abhinandan S. Das, Mayur Datar, Ashutosh Garg, y Shyam Rajaram. Google news personalization: scalable online collaborative filtering. En *Proceedings of the 16th international conference on World Wide Web*, WWW '07, páginas 271–280, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-654-7. 24, 100, 146

Jeffrey Dean y Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, Enero 2008. ISSN 0001-0782. 146

A. P. Dempster, N. M. Laird, y D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977. 57

Mukund Deshpande y George Karypis. Item-based top-N recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, 2004. ISSN 1046-8188. 51, 100

Christian Desrosiers y George Karypis. A comprehensive survey of neighborhood-based recommendation methods. En Ricci et al. [2011b], páginas 107–144. ISBN 978-0-387-85819-7. 7, 49, 101, 147

Shuai Ding y Torsten Suel. Faster top-k document retrieval using block-max indexes. En *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, SIGIR '11, páginas 993–1002, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0757-4. 155

Shuai Ding, Josh Attenberg, y Torsten Suel. Scalable techniques for document identifier assignment ininverted indexes. En *Proceedings of the 19th international conference on World wide web*, WWW '10, páginas 311–320, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-799-8. 176

Zhicheng Dou, Ruihua Song, y Ji-Rong Wen. A large-scale evaluation and analysis of personalized search strategies. En *Proceedings of the 16th international conference on World Wide Web*, WWW '07, páginas 581–590, New York, NY, USA, Mayo 2007. ACM. ISBN 978-1-59593-654-7. 24

P. Elias. Universal codeword sets and representations of the integers. *Information Theory, IEEE Transactions on*, 21(2):194 – 203, Marzo 1975. ISSN 0018-9448. 165

Jordan Ellenberg. This Psychologist Might Outsmart the Math Brains Competing for the Netflix Prize. *Wired Magazine*, 16(3), 2008. 26, 70, 78

A. Felfernig y R. Burke. Constraint-based recommender systems: technologies and research issues. En *Proceedings of the 10th international conference on Electronic commerce*, ICEC '08, páginas 3:1–3:10, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-075-3. 31

A. Felfernig, K. Isak, K. Szabo, y P. Zachar. The VITA financial services sales support environment. En *Proceedings of the 19th national conference on Innovative applications of artificial intelligence - Volume 2*, IAAI'07, páginas 1692–1699. AAAI Press, 2007. ISBN 978-1-57735-323-2. 30

Larry Fitzpatrick y Mei Dent. Automatic feedback using past queries: social searching? En *Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '97, páginas 306–313, New York, NY, USA, 1997. ACM. ISBN 0-89791-836-3. 128

Simon Funk. Netflix Update: Try This at Home. `http://sifter.org/~simon/journal/20061211.html`, 2006. 61

Thomas George y Srujana Merugu. A scalable collaborative filtering framework based on co-clustering. En *Proceedings of the Fifth IEEE International Conference on Data Mining*, ICDM '05, páginas 625–628, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2278-5. 56

Nadav Golbandi, Yehuda Koren, y Ronny Lempel. Adaptive bootstrapping of recommender systems using decision trees. En *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM '11, páginas 595–604, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0493-1. 126

Jennifer Golbeck. Generating predictive movie recommendations from trust in social networks. En *Proceedings of the 4th international conference on Trust Management*, iTrust'06, páginas 93–104, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 3-540-34295-8, 978-3-540-34295-3. 31

David Goldberg, David Nichols, Brian M. Oki, y Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992. 5, 30, 44

Ken Goldberg, Theresa Roeder, Dhruv Gupta, y Chris Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Inf. Retr.*, 4(2):133–151, 2001. ISSN 1386-4564. 24

S. Golomb. Run-length encodings (corresp.). *Information Theory, IEEE Transactions on*, 12 (3):399–401, Julio 1966. ISSN 0018-9448. 165

Nathaniel Good, J. Ben Schafer, Joseph A. Konstan, Al Borchers, Badrul Sarwar, Jon Herlocker, y John Riedl. Combining collaborative filtering with personal agents for better recommendations. En *Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*, AAAI '99/IAAI '99, páginas 439–446, Menlo Park, CA, USA, 1999. American Association for Artificial Intelligence. ISBN 0-262-51106-1. 32, 125

Georg Groh y Christian Ehmig. Recommendations in taste related domains: collaborative filtering vs. social filtering. En *Proceedings of the 2007 international ACM conference on Supporting group work*, GROUP '07, páginas 127–136, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-845-9. 31

BIBLIOGRAFÍA

Asela Gunawardana y Christopher Meek. Tied boltzmann machines for cold start recommendations. En *Proceedings of the 2008 ACM conference on Recommender systems*, RecSys '08, páginas 19–26, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-093-7. 125

Ido Guy, Naama Zwerdling, David Carmel, Inbal Ronen, Erel Uziel, Sivan Yogev, y Shila Ofek-Koifman. Personalized recommendation of social software items based on social relations. En *Proceedings of the third ACM conference on Recommender systems*, RecSys '09, páginas 53–60, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-435-5. 31

Jon Herlocker, Joseph A. Konstan, y John Riedl. An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Inf. Retr.*, 5(4):287–310, 2002. ISSN 1386-4564. 45, 49, 87, 88, 94

Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, y John Riedl. An algorithmic framework for performing collaborative filtering. En *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '99, páginas 230–237, New York, NY, USA, 1999. ACM. ISBN 1-58113-096-1. 26, 33, 47, 69, 75, 91, 121

Jonathan L. Herlocker, Joseph A. Konstan, y John Riedl. Explaining collaborative filtering recommendations. En *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, CSCW '00, páginas 241–250, New York, NY, USA, 2000. ACM. ISBN 1-58113-222-0. 36

Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, y John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, 2004. ISSN 1046-8188. 27, 34, 35, 78, 109

Will Hill, Larry Stead, Mark Rosenstein, y George Furnas. Recommending and evaluating choices in a virtual community of use. En *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '95, páginas 194–201, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co. ISBN 0-201-84705-1. 26

Thomas Hofmann y Jan Puzicha. Latent class models for collaborative filtering. En *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, IJCAI '99, páginas 688–693, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. ISBN 1-55860-613-0. 58

Zan Huang, Hsinchun Chen, y Daniel Zeng. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1): 116–142, 2004. ISSN 1046-8188. 42

Zan Huang, Daniel Zeng, y Hsinchun Chen. A comparison of collaborative-filtering recommendation algorithms for e-commerce. *IEEE Intelligent Systems*, 22(5):68–78, 2007. ISSN 1541-1672. 24, 100, 110

D.A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, Septiembre 1952. ISSN 0096-8390. 171

David Hull. Using statistical testing in the evaluation of retrieval experiments. En *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '93, páginas 329–338, New York, NY, USA, 1993. ACM. ISBN 0-89791-605-0. 111

Jing Jiang, Jie Lu, Guangquan Zhang, y Guodong Long. Scaling-up item-based collaborative filtering recommendation algorithm based on hadoop. En *Proceedings of the 2011 IEEE World Congress on Services*, SERVICES '11, páginas 490–497, Washington, DC, USA, Julio 2011. IEEE Computer Society. ISBN 978-0-7695-4461-8. 147

Rong Jin, Luo Si, y ChengXiang Zhai. Preference-based graphic models for collaborative filtering. En *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, UAI'03, páginas 329–336, San Francisco, CA, USA, 2003. Morgan Kaufmann Publishers Inc. ISBN 0-127-05664-5. 70

Rong Jin, Luo Si, y Chengxiang Zhai. A study of mixture models for collaborative filtering. *Inf. Retr.*, 9(3):357–382, Junio 2006. ISSN 1386-4564. 59

Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, y Geri Gay. Accurately interpreting clickthrough data as implicit feedback. En *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, páginas 154–161, New York, NY, USA, 2005. ACM. ISBN 1-59593-034-5. 24

Seikyung Jung, Jonathan L. Herlocker, y Janet Webster. Click data as implicit relevance feedback in web search. *Inf. Process. Manage.*, 43(3):791–807, 2007. ISSN 0306-4573. 24

George Karypis. Evaluation of item-based top-N recommendation algorithms. En *Proceedings of the tenth international conference on Information and knowledge management*, CIKM '01, páginas 247–254, New York, NY, USA, 2001. ACM. ISBN 1-58113-436-3. 100, 110

Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, y John Riedl. Grouplens: applying collaborative filtering to usenet news. *Commun. ACM*, 40(3):77–87, Marzo 1997. ISSN 0001-0782. 24

Joseph A. Konstan, Sean M. McNee, Cai-Nicolas Ziegler, Roberto Torres, Nishikant Kapoor, y John T. Riedl. Lessons on applying automated recommender systems to information-seeking tasks. En *Proceedings of the 21st national conference on Artificial intelligence - Volume 2*, AAAI'06, páginas 1630–1633. AAAI Press, 2006. ISBN 978-1-57735-281-5. 36

Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. En *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, páginas 426–434, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-193-4. 62, 63

B. Krulwich. Lifestyle finder: Intelligent user profiling using large-scale demographic data. *Artificial Intelligence Magazine*, 18(2):37–45, 1997. 30

Shyong K. Lam y John Riedl. Shilling recommender systems for fun and profit. En *Proceedings of the 13th international conference on World Wide Web*, WWW '04, páginas 393–402, New York, NY, USA, 2004. ACM. ISBN 1-58113-844-X. 42

Xuan Nhat Lam, Thuc Vu, Trong Duc Le, y Anh Duc Duong. Addressing cold-start problem in recommendation systems. En *Proceedings of the 2nd international conference on Ubiquitous information management and communication*, ICUIMC '08, páginas 208–211, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-993-7. 125

BIBLIOGRAFÍA

Tong Queue Lee, Young Park, y Yong-Tae Park. A time-based approach to effective recommender systems using implicit feedback. *Expert Syst. Appl.*, 34(4):3055–3062, 2008. ISSN 0957-4174. 24

Daniel Lemire. Scale and translation invariant collaborative filtering systems. *Information Retrieval*, 8:1–22, 2003. 146

Daniel Lemire y Anna Maclachlan. Slope one predictors for online rating-based collaborative filtering. En *Proceedings of SIAM Data Mining*, SDM'05, 2005. 53, 146

Henry Lieberman. Letizia: an agent that assists web browsing. En *Proceedings of the 14th international joint conference on Artificial intelligence - Volume 1*, IJCAI'95, páginas 924–929, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc. ISBN 1-55860-363-8, 978-1-558-60363-9. 29

Greg Linden, Brent Smith, y Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, Enero 2003. ISSN 1089-7801. 5, 24, 156

M. C Little. Javasim user's guide. public release 0.3, version 1.0. `http://javasim.codehaus.org/`, 1999. 187

Fabiana Lorenzi y Francesco Ricci. Case-based recommender systems: a unifying view. En *Proceedings of the 2003 international conference on Intelligent Techniques for Web Personalization*, ITWP'03, páginas 89–113, Berlin, Heidelberg, 2005. Springer-Verlag. ISBN 3-540-29846-0, 978-3-540-29846-5. 31

Christopher D. Manning, Prabhakar Raghavan, y Hinrich Schtze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. ISBN 0521865719, 9780521865715. 29, 109, 163

B. Marlin. Collaborative filtering: A machine learning perspective. Master's thesis, University of Toronto, 2004. 30, 44, 56

Matthew R. McLaughlin y Jonathan L. Herlocker. A collaborative filtering algorithm and evaluation metric that accurately model the user experience. En *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '04, páginas 329–336, New York, NY, USA, 2004. ACM. ISBN 1-58113-881-4. 100, 109

Sean M. McNee, John Riedl, y Joseph A. Konstan. Being accurate is not enough: how accuracy metrics have hurt recommender systems. En *CHI '06 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '06, páginas 1097–1101, New York, NY, USA, 2006. ACM. ISBN 1-59593-298-4. 34, 35

Prem Melville, Raymod J. Mooney, y Ramadass Nagarajan. Content-boosted collaborative filtering for improved recommendations. En *Eighteenth national conference on Artificial intelligence*, páginas 187–192, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence. ISBN 0-262-51129-0. 31, 125

Nader Mirzadeh, Francesco Ricci, y Mukesh Bansal. Feature selection methods for conversational recommender systems. En *Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'05) on e-Technology, e-Commerce*

and e-Service, EEE '05, páginas 772–777, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2274-2. 31

Dunja Mladenic. Text-learning and related intelligent agents: A survey. *IEEE Intelligent Systems*, 14(4):44–54, Julio 1999. ISSN 1541-1672. 29

Bamshad Mobasher, Robin Burke, Runa Bhaumik, y Chad Williams. Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness. *ACM Trans. Interet Technol.*, 7(4):23, 2007. ISSN 1533-5399. 42

Raymond J. Mooney y Loriene Roy. Content-based book recommending using learning for text categorization. En *Proceedings of the fifth ACM conference on Digital libraries*, DL '00, páginas 195–204, New York, NY, USA, 2000. ACM. ISBN 1-58113-231-X. 29

An-Te Nguyen, Nathalie Denos, y Catherine Berrut. Improving new user recommendations with rule-based induction on cold user data. En *Proceedings of the 2007 ACM conference on Recommender systems*, RecSys '07, páginas 121–128, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-730–8. 125

David M. Nichols. Implicit rating and filtering. En *Proceedings of the Fifth DELOS Workshop on Filtering and Collaborative Filtering*, páginas 31–36, 1998. 23

Rong Pan y Martin Scholz. Mind the gaps: weighting the unknown in large-scale one-class collaborative filtering. En *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, páginas 667–676, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-495-9. 101

Rong Pan, Yunhong Zhou, Bin Cao, Nathan N. Liu, Rajan Lukose, Martin Scholz, y Qiang Yang. One-class collaborative filtering. En *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, páginas 502–511, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3502-9. 24, 101

Manos Papagelis, Dimitris Plexousakis, y Themistoklis Kutsuras. Alleviating the sparsity problem of collaborative filtering using trust inferences. *Trust Management*, 3477: 224–239, 2005a. 125

Manos Papagelis, Ioannis Rousidis, Dimitris Plexousakis, y Elias Theoharopoulos. Incremental collaborative filtering for highly-scalable recommendation algorithms. En *Proceedings of the 15th international conference on Foundations of Intelligent Systems*, ISMIS'05, páginas 553–561, Berlin, Heidelberg, 2005b. Springer-Verlag. ISBN 3-540-25878-7, 978-3-540-25878-0. 147

Seung-Taek Park y Wei Chu. Pairwise preference regression for cold-start recommendation. En *Proceedings of the third ACM conference on Recommender systems*, RecSys '09, páginas 21–28, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-435-5. 124

Seung-Taek Park, David Pennock, Omid Madani, Nathan Good, y Dennis DeCoste. Naïve filterbots for robust cold-start recommendations. En *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, páginas 699–705, New York, NY, USA, 2006. ACM. ISBN 1-59593-339-5. 125

BIBLIOGRAFÍA

Arkadiusz Paterek. Improving regularized singular value decomposition for collaborative filtering. En *Proceedings of KDD Cup and Workshop*, KDDCup '07, páginas 39–42, San Jose, California, USA, 2007. ACM. 61

Michael J. Pazzani. A framework for collaborative, content-based and demographic filtering. *Artif. Intell. Rev.*, 13(5-6):393–408, Diciembre 1999. ISSN 0269-2821. 32

David Pennock, Eric Horvitz, Steve Lawrence, y C. Lee Giles. Collaborative filtering by personality diagnosis: A hybrid memory- and model-based approach. En *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence, UAI 2000*, páginas 473–480, Stanford, CA, 2000. 55

B. Piccart, J. Struyf, y H. Blockeel. Alleviating the sparsity problem in collaborative filtering by using an adapted distance and a graph-based method. En *SIAM International Conference on Data Mining*, SDM 2010, páginas 189–198, 2010. 126

Martin Piotte y Martin Chabbert. The pragmatic theory solution to the netflix grand prize. `http://www.netflixprize.com/assets/GrandPrize2009_BPC_PragmaticTheory.pdf`, 2009. 63

Yonggang Qiu y Hans-Peter Frei. Concept based query expansion. En *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '93, páginas 160–169, New York, NY, USA, 1993. ACM. ISBN 0-89791-605-0. 128

Al Mamunur Rashid, George Karypis, y John Riedl. Learning preferences of new users in recommender systems: an information theoretic approach. *SIGKDD Explor. Newsl.*, 10: 90–100, Diciembre 2008. ISSN 1931-0145. 126

Jennifer Redpath, DavidH. Glass, Sally McClean, y Luke Chen. Collaborative filtering: The aim of recommender systems and the significance of user ratings. En Cathal Gurrin, Yulan He, Gabriella Kazai, Udo Kruschwitz, Suzanne Little, Thomas Roelleke, Stefan Rüger, y Keith Rijsbergen, editores, *Advances in Information Retrieval*, volumen 5993 de *Lecture Notes in Computer Science*, páginas 394–406. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-12274-3. 110

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, y Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. En *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, páginas 452–461, Arlington, Virginia, United States, 2009. AUAI Press. ISBN 978-0-9749039-5-8. 24

Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, y John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. En *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, CSCW '94, páginas 175–186, New York, NY, USA, 1994. ACM. ISBN 0-89791-689-1. 26, 45, 46, 48, 69

Berthier A. Ribeiro-Neto y Ramurti A. Barbosa. Query performance for tightly coupled distributed digital libraries. En *Proceedings of the third ACM conference on Digital libraries*, DL '98, páginas 182–190, New York, NY, USA, 1998. ACM. ISBN 0-89791-965-3. 187

Francesco Ricci, Lior Rokach, y Bracha Shapira. Introduccion to recommender systems handbook. En Ricci et al. [2011b], páginas 1–35. ISBN 978-0-387-85819-7. 5, 6, 22, 29

Francesco Ricci, Lior Rokach, Bracha Shapira, y Paul B. Kantor, editores. *Recommender Systems Handbook*. Springer, 2011b. ISBN 978-0-387-85819-7. 220, 226, 227

Adam Richardson. From the information age to the recommendation age. `http://designmind.frogdesign.com/articles/early-articles/from-the-information-age-to-the-recommendation-age.html`, 2005. 4

J. Rocchio. Relevance feedback in information retrieval. En Gerard Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, páginas 313–323. Prentice-Hall, 1971. 127

G. Salton, A. Wong, y C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18:613–620, Noviembre 1975. ISSN 0001-0782. 149

J. J. Sandvig, Bamshad Mobasher, y Robin Burke. Robustness of collaborative recommendation based on association rule mining. En *Proceedings of the 2007 ACM conference on Recommender systems*, RecSys '07, páginas 105–112, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-730–8. 42

Badrul Sarwar, George Karypis, Joseph Konstan, y John Riedl. Analysis of recommendation algorithms for e-commerce. En *Proceedings of the 2nd ACM conference on Electronic commerce*, EC '00, páginas 158–167, New York, NY, USA, 2000a. ACM. ISBN 1-58113-272-7. 110

Badrul Sarwar, George Karypis, Joseph Konstan, y John Riedl. Item-based collaborative filtering recommendation algorithms. En *Proceedings of the 10th international conference on World Wide Web*, WWW '01, páginas 285–295, New York, NY, USA, 2001. ACM. ISBN 1-58113-348-0. 45, 50, 51

Badrul M. Sarwar, George Karypis, Joseph A. Konstan, y John T. Riedl. Application of dimensionality reduction in recommender systema – a case study. En *ACM WebKDD Workshop*, 2000b. 60

Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, y David M. Pennock. Methods and metrics for cold-start recommendations. En *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '02, páginas 253–260, New York, NY, USA, 2002. ACM. ISBN 1-58113-561-0. 6, 10, 31, 42, 125

Sebastian Schelter, Christoph Boden, y Volker Markl. Scalable similarity-based neighborhood methods with mapreduce. En *Proceedings of the sixth ACM conference on Recommender systems*, RecSys '12, páginas 163–170, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1270-7. 147

Barry Schwartz. *The paradox of choice: why more is less*. ECCO, New York, 2004. ISBN 0060005688. 20

Mahdi Shafiei y Evangelos Milios. Model-based Overlapping Co-Clustering. En *Proceedings of the Fourth Workshop on Text Mining, Sixth SIAM International Conference on Data Mining*, Bethesda, Maryland, Abril 2006. 56

Guy Shani y Asela Gunawardana. Evaluating recommendation systems. En Ricci et al. [2011b], páginas 257–297. ISBN 978-0-387-85819-7. 6, 33, 110

BIBLIOGRAFÍA

C. E. Shannon. A mathematical theory of communication. *Bell system technical journal*, 27, 1948. 164

Upendra Shardanand y Pattie Maes. Social information filtering: algorithms for automating "word of mouth". En *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '95, páginas 210–217, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co. ISBN 0-201-84705-1. 29, 30, 45, 46, 47, 48

Reza Shokri, Pedram Pedarsani, George Theodorakopoulos, y Jean-Pierre Hubaux. Preserving privacy in collaborative filtering through distributed aggregation of offline profiles. En *Proceedings of the third ACM conference on Recommender systems*, RecSys '09, páginas 157–164, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-435-5. 42

L. Si y R. Jin. Flexible mixture model for collaborative filtering. En *Proceedings of the Twentieth International Conference on Machine Learning*, ICML 2003, páginas 704–711. AAAI Press, 2003. 59

Fabrizio Silvestri. Sorting out the document identifier assignment problem. En *Proceedings of the 29th European conference on IR research*, ECIR'07, páginas 101–112, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 978-3-540-71494-1. 177

Fabrizio Silvestri, Salvatore Orlando, y Raffaele Perego. Assigning identifiers to documents to enhance the clustering property of fulltext indexes. En *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '04, páginas 305–312, New York, NY, USA, 2004. ACM. ISBN 1-58113-881-4. 176

B. Smith y P. Cotter. A personalized TV listings service for the digital TV age. *Knowledge-Based Systems*, 13:53–59, 2000. 32

Barry Smyth. A community-based approach to personalizing web search. *Computer*, 40(8): 42–50, 2007. ISSN 0018-9162. 100

E. Isaac Sparling y Shilad Sen. Rating: how difficult is it? En *Proceedings of the fifth ACM conference on Recommender systems*, RecSys '11, páginas 149–156, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0683-6. 24

Michael M. Stark y Richard F. Riesenfeld. Wordnet: An electronic lexical database. En *Proceedings of 11th Eurographics Workshop on Rendering*. MIT Press, 1998. 128

Xiaoyuan Su y Taghi M. Khoshgoftaar. A survey of collaborative filtering techniques. *Adv. in Artif. Intell.*, 2009:2–2, 2009. ISSN 1687-7470. 5

Cynthia A. Thompson, Mehmet H. Göker, y Pat Langley. A personalized system for conversational recommendations. *J. Artif. Int. Res.*, 21(1):393–428, Marzo 2004. ISSN 1076-9757. 30

Thomas Tran y Robin Cohen. Hybrid recommender systems for electronic commerce. En *Proceedings of the 17th National Conference on Artificial Intelligence*, AAAI, 2000. 32

Howard Turtle y James Flood. Query evaluation: strategies and optimizations. *Inf. Process. Manage.*, 31(6):831–850, Noviembre 1995. ISSN 0306-4573. 155

Andreas Töscher, Michael Jahrer, y Robert M. Bell. The BigChaos solution to the netflix grand prize. http://www.netflixprize.com/assets/GrandPrize2009_BPC_BigChaos.pdf, 2009. 63

C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 2nd edition, 1979. 109

Ellen M. Voorhees. The philosophy of information retrieval evaluation. En *Revised Papers from the Second Workshop of the Cross-Language Evaluation Forum on Evaluation of Cross-Language Information Retrieval Systems*, CLEF '01, páginas 355–370, London, UK, 2002. Springer-Verlag. ISBN 3-540-44042-9. 110

Slobodan Vucetic y Zoran Obradovic. A regression-based approach for scaling-up personalized recommender systems in e-commerce. En *ACM WebKDD Workshop*, 2000. 52, 53

Bing Wang, Zhaowen Tao, y Jun Hu. Improving the diversity of user-based top-N recommendation by cloud model. En *Computer Science and Education (ICCSE), 2010 5th International Conference on*, páginas 1323 –1327, Agosto 2010. 49

Jun Wang, Arjen P. de Vries, y Marcel J. T. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. En *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, páginas 501–508, New York, NY, USA, 2006a. ACM. ISBN 1-59593-369-7. 51

Jun Wang, Arjen P. de Vries, y Marcel J. T. Reinders. A user-item relevance model for log-based collaborative filtering. En *Proceedings of the 28th European conference on Advances in Information Retrieval*, ECIR'06, páginas 37–48, Berlin, Heidelberg, 2006b. Springer-Verlag. ISBN 3-540-33347-9, 978-3-540-33347-0. 24

Jun Wang, Johan Pouwelse, Reginald L. Lagendijk, y Marcel J. T. Reinders. Distributed collaborative filtering for peer-to-peer file sharing systems. En *Proceedings of the 2006 ACM symposium on Applied computing*, SAC '06, páginas 1026–1030, New York, NY, USA, 2006c. ACM. ISBN 1-59593-108-2. 147

Ian H. Witten, Alistair Moffat, y Timothy C. Bell. *Managing gigabytes (2nd ed.): compressing and indexing documents and images*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999. ISBN 1-55860-570-3. 161, 162, 164, 165

Liang Xiang y Qing Yang. Time-dependent models in collaborative filtering based recommender system. En *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Volume 01*, WI-IAT '09, páginas 450–457, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-0-7695-3801-3. 26

Gui-Rong Xue, Chenxi Lin, Qiang Yang, WenSi Xi, Hua-Jun Zeng, Yong Yu, y Zheng Chen. Scalable collaborative filtering using cluster-based smoothing. En *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, páginas 114–121, New York, NY, USA, 2005. ACM. ISBN 1-59593-034-5. 56

Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, y Georg Lausen. Improving recommendation lists through topic diversification. En *Proceedings of the 14th international conference on World Wide Web*, WWW '05, páginas 22–32, New York, NY, USA, 2005. ACM. ISBN 1-59593-046-9. 36