UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

TRABAJO FIN DE GRADO

Suite de algoritmos de recomendación en aplicaciones reales

Sofía Marina Pepa

Tutor: Pablo Castells Azpilicueta

Mayo 2014

Agradecimientos

En primer lugar, me gustaría agradecer a mi tutor Pablo Castells por brindarme la oportunidad de poder realizar este TFG con él y haberme hecho partícipe del grupo de investigación que dirige en la Escuela. Son pocos los tutores que se preocupan tanto por el progreso y aprendizaje de sus tutorandos, y tengo que decir que me siento muy afortunada de poder haberlo tenido a él como tutor para este trabajo, ya que Pablo es uno de ellos. Muchas gracias también a todos los compañeros del laboratorio por orientarme y ayudarme de buena voluntad siempre que lo he necesitado.

Gracias a mi familia por haberme apoyado a lo largo de toda la carrera y, por qué no, a lo largo de toda mi vida. He tenido la gran suerte de crecer en una familia muy unida y feliz, que me enseñó a mirar siempre el lado positivo de las cosas y a estar siempre agradecido por lo que uno tiene. Ellos han sido mi pilar principal que me ha mantenido con el ánimo siempre en lo más alto y gracias al cual he conseguido cumplir todas mis metas estos últimos años. Gracias por preocuparos por mí, gracias por traerme la cena las noches que tenía que entregar prácticas, gracias por desearme suerte para los exámenes en los que me sentía insegura, gracias por hacerme reír y hacerme olvidar mis pequeños problemas, gracias por todo eso y más.

Por último, me gustaría agradecer a todos aquellos que me han acompañado a lo largo de la carrera, sobre todo a esos compañeros que se conviertieron en grandes amigos. Desde el primer momento me he sentido como en casa, y estoy orgullosa de poder decir que todas las personas con las que me he encontrado a lo largo de esta carrera han sido maravillosas. Me gustaría destacar sobre todo el excepcional espíritu de compañerismo que todos han tenido desde el principio hasta el final. Nos convertimos en una gran familia que cuidaba unos de otros, ayudando con prácticas hasta las tantas de la madrugada, explicando dudas antes de los exámenes, o animándonos mutuamente con música hasta que salía el sol. De corazón, muchas gracias por haber corrido este gran maratón a mi lado, de principio a fin.

Termino ya una preciosa etapa llena de felicidad, *colacaos*, prácticas, Beatles, césped y mixtos con huevo que recordaré siempre con mucho cariño. Gracias a todos por haberme acompañado en mi paso de niña... a niña más grande.

Sofía Marina Pepa

Mayo 2014

Resumen

Internet irrumpió en las sociedades modernas como medio de comunicación y soporte a procesos de información y transacciones. El número de usuarios se extendió rápidamente y desde entonces no ha dejado de aumentar.

Debido a la cantidad masiva de información que Internet alberga, se vuelve muy complicado para los usuarios encontrar aquello que le interese entre toda la variedad disponible. Este problema motiva el desarrollo y despliegue de sistemas de recomendación, que estudian el perfil del usuario y le sugieren aquello que puede interesarle según sus gustos (vídeos de Youtube, contactos de LinkedIn o películas en Netflix). Existen diversos algoritmos de recomendación basados en diferentes formas de establecer el posible interés que el usuario pueda tener sobre cada "ítem", y a lo largo de la literatura se han ido investigando y desarrollando nuevos métodos fundamentados en diferentes bases.

El presente trabajo consiste en la realización de un estudio exhaustivo que compara algunos de los algoritmos más destacados de la literatura sobre diferentes conjuntos de datos obtenidos de aplicaciones con usuarios reales (como Twitter o Netflix). La forma de realizar esta comparación ha sido mediante métricas de precisión, novedad y diversidad, que son las que en años recientes han cobrado mayor interés en el campo de la recomendación frente a las tradicionales métricas de error. Asimismo, se analiza la eficiencia de los algoritmos en términos de costes computacionales.

Este trabajo incluye asimismo una puesta a punto de los algoritmos a estudiar, tanto configuraciones propias de cada uno como variaciones que mejoren su escalabilidad y eficiencia en el consumo de recursos, a fin de determinar qué elementos y parámetros de los algoritmos son determinantes en la efectividad de los mismos, con vistas a identificar configuraciones óptimas. Además, se han añadido librerías externas a las baterías de prueba, con implementaciones de estos y otros algoritmos en busca del mejor resultado posible para cada algoritmo y poder realizar la comparación entre sus versiones más óptimas (en cuanto a la métricas establecidas).

Como conclusión del presente trabajo, se puede decir que para las aplicaciones escogidas (MovieLens, Netflix, Last.fm, Blueknow y Twitter), la familia de algoritmos que mejor resultado ha obtenido en métricas de precisión, novedad y diversidad ha sido la del filtrado colaborativo basado en vecinos próximos. Sin embargo, en muchos casos estos algoritmos resultan inviables para su ejecución, donde toman la ventaja algoritmos basados en contenido como Rocchio o de factorización de matrices como pLSA.

Palabras clave

Sistemas de recomendación, filtrado colaborativo, métricas, acierto, novedad, diversidad, evaluación.

Abstract

Internet and the WWW rose in modern societies as a new communication channel and as a new media for information flow and transactions. The number of users grew dashingly and continuously since then.

Given the massive amount of information available on the WWW, it soon became very difficult for users to find what they may be interested in amidst billions of documents and digital objects. This problem motivates the development and deployment of so-called automatic recommendation technologies, which track the user's actions, identify potential user trends and preferences and, based on this, suggest items and choices the user may find interesting or useful (Youtube videos, LinkedIn contacts or Netflix movies). Many recommendation algorithms, techniques, and theories have been developed in this field, defining different approaches to predict the interest that users may have within huge spaces of available choices.

The present work undertakes an extensive study comparing some of the most important algorithms in the literature on different datasets obtained from real user applications (such as Twitter, Eurekakids and Netflix). The comparison is carried out in several dimensions, including accuracy, but also novelty and diversity, which in recent years have raised increasing in the recommender systems field. Furthermore, we study the efficiency of the algorithms in terms of computational cost.

This work also includes the development of recommendation algorithms, for some of which we study and compare arrays of configurations in order to understand which elements and parameters of the algorithms are crucial to their effectiveness, and to identify optimal configurations. External libraries implementing the same and different algorithms have also been added to our experimental testbed for comparison in order to observe and find optimal algorithms and versions thereof.

For the tested domains (MovieLens, Netflix, Last.fm, Twitter and Blueknow) the family of algorithms that obtained the best results in accuracy, novelty and diversity metrics has been by and large collaborative filtering based on nearest neighbors. However, in some cases these algorithms do not scale to real use because of the computional cost. Simpler content-based algorithms, or matrix factorization methods (as pLSA), take advantage over them in these cases.

Keywords

Recommender systems, collaborative filtering, metrics, accuracy, novelty, diversity, evaluation.

Índice

Índice de tablas

Índice de figuras

1. Introducción

1.1 Motivación

Los llamados sistemas de recomendación personalizada comenzaron a visualizarse como área específica de investigación e incipientes aplicaciones a principios de la última década del siglo pasado, y tuvieron un sostenido desarrollo que los ha llevado a crecer y hacerse presentes hoy día en aplicaciones de uso cotidiano de cualquier consumidor de tecnologías de la información. Hoy en día estamos ya acostumbrados a que Amazon trate de acertar con nuestras preferencias, que Spotify nos sugiera música adaptada a nuestros gustos, que Twitter, Facebook o LinkedIn nos recomienden posts y contactos de forma individualizada, que Youtube nos recomiende vídeos, que la publicidad *online* se intente adaptar a nosotros, o que nuestro móvil nos recomiende aplicaciones, y a nadie le resulta extraño hoy el concepto de recomendación personalizada.

En esta evolución ha tenido mucho que ver la explosión del uso de Internet (figura 1), tanto de forma profesional como doméstica, a lo que se suma actualmente el protagonismo de los dispositivos móviles. Estamos en continua conexión con una red cuyo tamaño y contenido han experimentado crecimientos de orden exponencial y cuya estructura se enriquece continuamente. Surgen escenarios donde la información útil para un usuario es una auténtica aguja en un astronómico pajar. Las tecnologías de recomendación cobran sentido y valor en este tipo de situaciones donde el espacio de búsqueda es inconmensurable a la capacidad de exploración del usuario, donde éste puede ni tan siguiera saber qué pregunta formular, y donde más aún, no sólo se trata de encontrar la aguja, sino de elegir una entre millones de agujas.

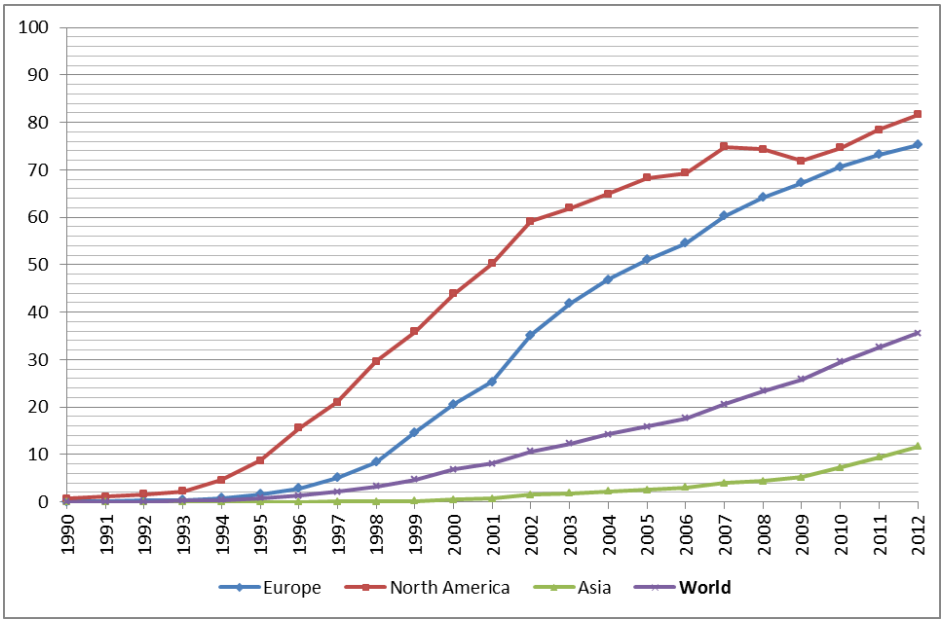

Figura 1. Porcentaje de usuarios de Internet desde 1990 hasta la actualidad.[1]

[1] Fuente: http://databank.worldbank.org/

La tarea de recomendar se basa en intentar predecir el interés que un usuario puede tener sobre un ítem (película, artista, libro, vídeo, etc.) que no conozca. Así, en lugar de que el usuario tenga que encontrar cosas que le gustan dentro de la inmensa cantidad de información de la que se dispone (todas las películas o libros existentes, todos los artistas a nivel mundial, etc.), el sistema filtra la información y selecciona aquellos ítems que cree que al usuario le van a interesar, generando así una recomendación.

A lo largo de las dos últimas décadas, el desarrollo de diferentes estrategias y algoritmos de recomendación ha sido continuo, pero, tal vez precisamente por ello, a día de hoy aún hay importantes cuestiones abiertas.

En particular, en años recientes se están revisando los métodos de evaluación (criterios con los que se valora la eficacia) de los sistemas de recomendación, reabriendo así espacio para el estudio y desarrollo de algoritmos más óptimos: aquellos optimizados en años pasados sobre determinadas metodologías y métricas resultan no ser los mejores en métricas nuevas.

Asimismo, está pendiente por abordar y revisar las comparativas de efectividad entre métodos de recomendación, teniendo en cuenta de forma medianamente sistemática multitud de detalles de implementación y configuración de los diferentes algoritmos.

No hay apenas en la literatura, hasta donde alcanza nuestro conocimiento, un estudio que compare de forma realmente exhaustiva algoritmos de diferentes familias en amplitud de configuraciones y bajo métricas del tipo que se vienen extendiendo en el área más recientemente (tales como la precisión en ránking, la novedad y la diversidad). En buena medida y relacionado con ello, sigue habiendo por tanto una pregunta abierta en el campo de la recomendación en cuanto a qué algoritmo es mejor.

El presente trabajo viene motivado por ese espacio abierto. Sin pretender naturalmente cubrirlo por completo, el trabajo persigue recopilar, implementar, ordenar y poner a punto una colección representativa de algoritmos de recomendación del estado del arte y llevar a cabo un estudio empírico de los mismos abarcando una amplitud significativa en la variedad de condiciones respecto a datos y configuración. Motiva así este trabajo contribuir a los avances hacia las cuestiones abiertas señaladas.

1.2 Objetivos

Las líneas generales a donde apunta el presente trabajo son, por un lado, seleccionar un conjunto de algoritmos de recomendación y buscar su configuración óptima; y por el otro, escoger y adaptar los conjuntos de datos a los que se les aplicarán los algoritmos. Se busca estudiar las recomendaciones obtenidas para cada caso con diferentes métricas de evaluación y compararlas, así como realizar las optimizaciones necesarias para tratar aquellos conjuntos de datos con tamaño masivo.

De forma concreta, los objetivos particulares que se abordan en este trabajo son los siguientes:

- Recopilación de una batería exhaustiva de implementaciones de los algoritmos de recomendación más relevantes del estado del arte (por sus buenos resultados reportados en la literatura, por su popularidad, etc.).
- Puesta a punto de las diferentes variantes de dichos algoritmos mediante la realización de estudios y barridos sobre los parámetros de configuración de los mismos.

- Adaptación de métodos de evaluación para datos con interacciones de diferente tipo (como click, visita, compra).
- Aplicación de los algoritmos a datos reales proporcionados por una empresa que realiza tareas de recomendación.
- Desarrollo de soluciones para problemas de cómputo para grandes conjuntos de datos, tales como la optimización del uso de los recursos o la escalabilidad de los algoritmos.

1.3 Estructura del trabajo

El presente trabajo se estructura en siete capítulos y un anexo, que se detallan a continuación:

- **Capítulo 1: Introducción.** Definición del contexto, motivación y objetivos del presente trabajo. Se detalla la estructura del mismo y se especifica la notación que se va a utilizar a lo largo de todo el trabajo.
- **Capítulo 2: Estado del arte.** Estudio sobre la literatura en el campo de los sistemas de recomendación, abordando de forma general las diferentes familias de algoritmos desarrolladas y el progreso de los métodos de evaluación a lo largo de los últimos 20 años.
- **Capítulo 3: Algoritmos de recomendación.** De forma más particular, se expone los algoritmos de recomendación incluidos en el presente trabajo, explicando en qué consisten e indicando la formulación utilizada para el estudio.
- **Capítulo 4: Métodos de evaluación.** Detalle de los conjuntos de datos con los que se trabaja, así como del tratamiento de los mismos para su uso como *input* para los algoritmos anteriormente descritos. Se incluye en este apartado puntos como el método de conversión de los datos implícitos a ratings o de partición de datos seguido para este trabajo. Finalmente en este capítulo se especifica el conjunto de métricas con las que se han evaluado los diferentes algoritmos.
- **Capítulo 5: Ingeniería.** Descripción de la ingeniería software seguida en el presente trabajo: metodología, implementación (diseño del software), tecnologías usadas (librerías externas y bases de datos) y las mejoras de escalabilidad realizadas sobre los algoritmos para poder trabajar con conjuntos de datos de gran envergadura.
- **Capítulo 6: Resultados experimentales.** Resultados obtenidos en el estudio, así como la interpretación de los mismos. Se incluye también el detalle de los barridos realizados sobre aquellos algoritmos con parámetros configurables para obtener su óptimo en los conjuntos de datos principales, así como el consumo de recursos (memoria y tiempo) que la variación de estos parámetros genera.
- **Capítulo 7: Conclusiones.** Resumen y contribuciones del estudio realizado en el presente trabajo, así como el trabajo futuro propuesto para la continuación del mismo.
- **Anexo I: Resultados completos para Blueknow, MovieLens 1M y Twitter.** Tablas de resultados de los algoritmos estudiados para estos tres conjuntos de datos con todas las métricas de evaluación, no sólo las principales que se muestran en el apartado 6.

1.4 Notación

A lo largo del presente trabajo se van a utilizar la siguiente notación y símbolos.

$\mathcal{U}$	Conjunto de usuarios existentes en el conjunto de datos con el que se está trabajando.
$\mathcal{I}$	Conjunto de ítems que se disponen para recomendar a los diferentes usuarios.
$\mathcal{Z}$	Conjunto de todos los aspectos considerados en las medidas de diversidad.
R	Ránking de los ítems a recomendar a un usuario en particular.
r	Matriz de ratings donde se encuentran las puntuaciones que los usuarios han dado a ciertos ítems (entrenamiento).
$\vec{u}$	Vector de valores, en este caso, vector de usuario *u*.
$r(u,i)$	Puntuación del usuario *u* al ítem *i*.
$\hat{r}(u,i)$	Puntuación del ítem *i* para el usuario *u* predicha por el recomendador correspondiente.
$\bar{r}(u)$	Puntuación promedio del usuario *u*.
$f(u,i)$	Función de ránking, calcula la puntuación del ítem *i* para el usuario *u*.
$sim(u,v)$	Similitud entre dos usuarios (o ítems) *u* y *v* calculada mediante la función de similitud correspondiente.
$d(u,v)$	Distancia entre dos ítems, dos usuarios o un usuario y un ítem.
$N_k(u)$	Conjunto de los *k* vecinos más similares a un usuario (o ítem) objetivo.
δ	Umbral de relevancia (valor a partir del cual un ítem se considera relevante).
$rel(i)$	Propiedad de relevancia para un ítem *i* (valor verdadero o falso).
$g(i)$	Grado de relevancia para un ítem *i* (valor comprendido entre un mínimo y un máximo estipulado).
$subtopics(i)$	Conjunto de aspectos de un ítem *i*.
$u[f]$	Valor del centroide de usuario para una característica *f*.

2. Estado del arte

El inicio de los sistemas de recomendación se suele situar a principios de los noventa (Goldberg 1992, Resnick 1994), como derivación del desarrollo de la inteligencia artificial y el modelado de usuario en décadas previas. El desarrollo de las tecnologías de recomendación ha recibido su principal impulso de la mano del crecimiento del comercio electrónico durante esa década, y en general de la exposición del consumidor a espacios de opciones cada vez más masivas en Internet y la WWW.

En ocasiones se describe la recomendación como el alter-ego de la búsqueda. A diferencia de un buscador, en la tarea de recomendación es el sistema el que toma la iniciativa, recuperando información y realizando recomendaciones sin que el usuario haga una consulta explícita. A partir de un modelo de usuario (históricos de interacción, edad y sexo, situación demográfica, etc.) y un conjunto de ítems a recomendar, se busca predecir la relevancia de los ítems para cada usuario y recomendarle a éste las opciones que más le puedan interesar.

El sistema observa la actividad de los usuarios en el sistema y aprende sus intereses, detectando patrones de comportamiento. Los intereses o preferencias se pueden inferir bien de forma implícita (como el número de veces que escucha una canción) o bien de forma explícita (como la puntuación que el usuario ha dado a una película).

- El input implícito tiene como ventaja que no infiere con la actividad del usuario, ya que el sistema infiere las preferencias midiendo la cantidad de interacción que éste tiene con los diferentes ítems. Sin embargo, la señal es más ruidosa y no se puede establecer de forma segura que el "uso" en que se basa la inferencia se deba a una preferencia real del usuario.
- El input explícito, por otra parte, proporciona una base más robusta para captar los intereses del usuario debido a que es él mismo el que expresa sus preferencias sobre algunos ítems. El principal inconveniente de esta opción es que requiere que el usuario se tome la molestia de introducir información, por lo que este tipo de dato suele estar disponible con muy baja densidad, además de presentar incoherencias debido al ruido natural (Amatriain 2009).

Las preferencias observadas de los usuarios se pueden representar de forma abstracta como una matriz de usuarios por ítems, donde unos pocos valores son conocidos (figura 2). El recomendador, entonces, intenta predecir los valores desconocidos (celdas vacías) y recomendar en consecuencia.

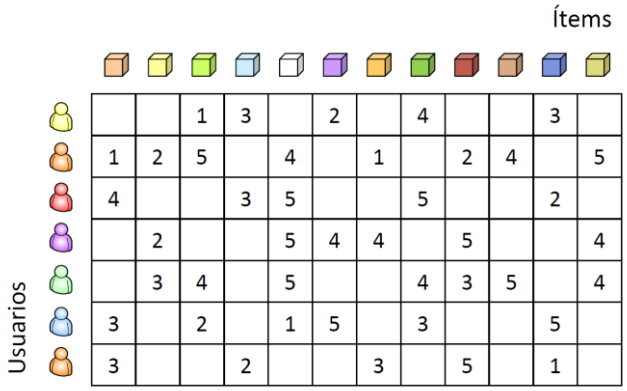

Figura 2. Ejemplo de tabla de puntuaciones de usuarios a ítems.

Existen múltiples formas de predecir unas preferencias a partir de otras. Una primera y típica subdivisión distingue entre los métodos de filtrado colaborativo, la recomendación basada en contenido y los algoritmos híbridos. En el primer apartado de esta sección se explicará en detalle cada una de estas familias.

Otro punto importante a tener en cuenta es la valoración y comparación de los resultados obtenidos por los diferentes algoritmos, lo que se conoce como evaluación. Las métricas de error han sido durante mucho tiempo la forma dominante de evaluar la efectividad de los algoritmos, pero en años más recientes se viene replanteando en la comunidad la representatividad de las métricas de error en cuanto a la utilidad práctica de las recomendaciones en términos de satisfacción del usuario y beneficios para el proveedor (rendimiento del negocio). Por ello, los métodos de evaluación (la forma de evaluar y las métricas) vienen evolucionando a favor de las métricas basadas en ránking (precisión, novedad y diversidad). De esto se hablará en el segundo apartado.

Ha destacado en esta evolución la popularidad que el llamado "premio Netflix" prestó al campo de la recomendación, y en muchos sentidos representó un hito en la evaluación y optimización de algoritmos de recomendación. Esta competición motivó la búsqueda de nuevos y mejores métodos, así como la optimización de los que ya existían.

Por último, en los siguientes apartados se presentarán las familias de algoritmos y las métricas principales utilizadas en el estudio, explicando ambos más detalladamente en el apartado 4 del presente trabajo.

2.1 Familias de algoritmos

En la literatura hay infinitud de algoritmos de recomendación que se basan en diferentes principios y técnicas (Adomavicius 2005, Ricci 2001) para la generación de recomendaciones. Todos parten de los mismos puntos y tienen el mismo objetivo, diferenciándose únicamente en la forma de elegir y ordenar los ítems a recomendar.

Recogen toda la actividad de los usuarios en forma de ratings (un valor para cada par usuario-ítem) o frecuencias (número de interacciones del usuario con el ítem) y utilizan estos datos para establecer la posible afinidad entre cada usuario y los ítems que éste desconoce, asignándole una puntuación. De esta forma, se genera una recomendación en forma de ránking donde los primeros ítems son aquellos con mayor puntuación y los últimos, los que tienen una menor.

Según la forma de predecir los ítems que le puedan interesar a cada usuario, los algoritmos pueden agruparse en tres familias: filtrado colaborativo, basados en contenido y algoritmos híbridos. A continuación vamos a mostrar qué algoritmos componen cada una de estas familias y las ventajas y los problemas que acarrean.

Filtrado colaborativo

Este tipo de algoritmos se caracteriza porque en la recomendación a un usuario se utiliza conocimiento sobre otros usuarios, de tal forma que los usuarios aprovechan la experiencia unos de otros. Dentro de esta familia se suelen diferenciar dos subgrupos: basados en memoria y basados en modelo (Adomavicius 2005).

- **Basados en memoria**: Se basan directamente en los datos observados, sobre los que aplican medidas de similitud, ya sea entre usuarios o entre ítems, a partir de las cuales se infieren predicciones de ratings. Los algoritmos basados en memoria se denominan también "basados en vecinos", pues infieren preferencias de un usuario basadas en las de usuarios (vecinos) similares, o bien infieren preferencias por ítems similares a los que se ha observado que el usuario prefiere, donde en todos los casos la similitud se mide en términos de la interacción entre usuarios e ítems. Describiremos con más detalle estos algoritmos en la sección 3.1.1, pues se han utilizado a fondo en el trabajo que aquí se presenta.
- **Basados en modelo**: El algoritmo genera una representación propia de los datos que se elabora a partir del formato original de los mismos. A esta representación se le denomina "modelo" y suele permitir, una vez construido, una generación relativamente rápida de recomendaciones. A este grupo pertenecen los algoritmos llamados de factorización de matrices, que explicaremos con más detalle en la sección 3.1.2.

Las ventajas del filtrado colaborativo son principalmente dos: buenos niveles de acierto (en términos de error de predicción), así como de novedad y de diversidad; y puede recomendar ítems de los que no se tenga ninguna descripción, sino sólo ratings de usuarios. Sin embargo, tiene la limitación del arranque en frío, es decir, tiene problemas para recomendar a usuarios poco activos o ítems poco conocidos.

Basado en contenido

Con esta denominación se suele hacer referencia a algoritmos que generan recomendaciones mediante la comparación del contenido que describe cada ítem y el contenido que le interesa al usuario objetivo (Adomavicius 2005).

Frente al filtrado colaborativo, los métodos basados en contenido pueden recomendar ítems nuevos o poco conocidos, aunque tienen en general la misma dificultad que el filtrado colaborativo para usuarios con poca actividad, ya que no se conoce lo suficiente sobre su perfil. La principal limitación de la recomendación basada en contenido es que, recomendando ítems parecidos al perfil del usuario, se puede producir un cierto efecto de "encasillamiento" para éste, con poca novedad y diversidad en la recomendación.

Algoritmos híbridos

Con el objetivo de mejorar la recomendación y cubrir las debilidades de estas dos familias, han surgido los algoritmos híbridos, que combinan algoritmos de filtrado colaborativo con basados en contenido (Adomavicius 2005).

Hay múltiples formas de realizar la combinación entre ambas familias, entre las que destacan las siguientes.

- **Monolítico (favoreciendo ciertas características)**: En este caso no se realiza ninguna combinación de resultados o algoritmos, sino que la combinación realizada es virtual. En otras palabras, se realiza una fase de pre-procesamiento para combinar el contenido de los ítems y sus correspondientes ratings (p.e. mediante pesos), y a partir de estos nuevos datos se ejecuta un único recomendador para generar el ránking de recomendación.
- **Paralelismo de varios sistemas**: Se realiza una recomendación por algoritmo y se combinan los resultados obtenidos en cada uno de ellos, posibilitando establecer diferentes pesos como en el caso anterior y dando lugar a un único ránking de ítems para cada usuario objetivo.

- **Implementación segmentada**: Un sistema de recomendación pre-procesa el input del siguiente algoritmo a ejecutar. Por ejemplo, el primer algoritmo excluye los ítems con puntuación nula, y el siguiente realiza el ránking descartando los ítems excluidos por el primero.

Las soluciones híbridas permiten, por ejemplo, paliar el problema del arranque en frío propio del filtrado colaborativo, o la recomendación de ítems sin descripción, propio de los algoritmos basados en contenido, gracias a que unos algoritmos respaldan las debilidades de los otros.

Para el presente trabajo se van a incluir los algoritmos más destacados de las dos primeras familias, filtrado colaborativo y basado en contenido, dejando como trabajo futuro incluir recomendaciones híbridas al estudio.

A lo largo de la literatura existen determinadas pruebas particulares en las que se ha observado el mejor o peor comportamiento de unos y otros algoritmos. En ellas, los algoritmos cuentan con diferencias de variantes, divergencias de métodos de evaluación y, en algunos casos, las pruebas son poco exhaustivas, por lo que no hay una perspectiva global clara de qué algoritmo es mejor que otro y para qué. Una respuesta completa a esa pregunta por supuesto excede el alcance de este estudio, pero el presente trabajo tiene por objetivo contribuir a esa dirección, abarcar un cierto ámbito y cubrirlo de forma sistemática y, en algunas variables, de forma exhaustiva.

2.2 Métodos de evaluación

Hay múltiples formas de medir la calidad de recomendación generada por un algoritmo, dependiendo de lo que se busque optimizar. Dentro de esta variedad es útil distinguir, en primer término, entre las métricas de error y las métricas de ránking.

Las métricas de error se centran en las puntuaciones que el recomendador predice, de forma que miden qué tan cerca ha estado la puntuación predicha respecto a la puntuación real que el usuario ha dado al ítem objetivo. Las métricas comunes de este conjunto son *Mean Absolute Error* (MAE), definida como la media absoluta de la diferencia entre los ratings predichos y los reales; y *Root Mean Squared Error* (RMSE), igual que la anterior pero tomando el cuadrado de las diferencias (para acentuar los valores más grandes) y la raíz cuadrada del promedio.

Por otro lado, las métricas de ránking sólo prestan atención al orden en el que los ítems han sido recomendados, independientemente de la puntuación que el algoritmo ha calculado. Para este fin se toman métricas que desde hace tiempo se vienen desarrollando en el campo de la Recuperación de Información para evaluar sistemas de búsqueda. Dentro de este conjunto cabe distinguir tres tipos principales de métricas, dependiendo del objetivo que se busque optimizar:

- **Precisión**: evalúan el grado de acierto de la recomendación realizada por el algoritmo. Observa los primeros ítems recomendados por el algoritmo y calculan la proporción de acierto. Entre éstas se encuentran métricas como *Precisión*, *Recall*, nDCG, entre muchas otras.
- **Diversidad**: busca que la recomendación cubra todos los intereses del usuario, basándose en características de los ítems y el perfil del usuario. Algunas métricas de diversidad son *ERR-IA* (Chapelle 2011), *Subtopic Recall* (Zhai 2003), o *Intra-List Dissimilarity* (Zhang 2008).

- **Novedad**: mide que el algoritmo recomiende ítems que es muy improbable que el usuario conozca. En esta línea se han propuesto métricas como *Expected Popularity Complement* (EPC) o *Expected Profile Distance* (EPD).

En el apartado 4 se detallarán éstas y el resto de métricas utilizadas, así como su formulación implementada para la evaluación de los algoritmos en el presente trabajo.

3. Algoritmos de recomendación aplicados

Existe un amplio abanico de algoritmos de recomendación, la mayoría de ellos con uno o varios parámetros y componentes configurables. Por ello, se ha hecho una selección de aquellos más representativos para realizar una comparación entre ellos sobre distintos conjuntos de datos con diferentes características.

A continuación se definen con detalle y por grupos los algoritmos que se han incluido en la comparativa realizada.

3.1 Filtrado colaborativo (FC)

Esta familia de recomendadores se caracteriza porque se utiliza información de unos usuarios para producir recomendaciones a otros. Así pues, en estos algoritmos los usuarios se benefician de la experiencia de otros usuarios. Este principio general se ha concretado en muy diversas formas, entre las cuales se distingue comúnmente entre los métodos basados en memoria y los basados en modelo. La diferencia radica en que en estos últimos el recomendador genera o aprende una representación propia de los datos (modelo), mientras que en los basados en memoria el algoritmo utiliza los datos en crudo en tiempo de recomendación. En el fondo todos los algoritmos realizan y almacenan parte de los cálculos en una fase online previa a la recomendación, por lo que la distinción entre estas dos vertientes del filtrado colaborativo no es una línea estricta.

Dentro de la amplia diversidad de algoritmos desarrollados en esta vertiente, las dos estrategias de filtrado colaborativo más extendidas y actualmente exitosas son, en cuanto a los métodos basados en memoria, los algoritmos de vecinos próximos, y entre los basados en modelo, la factorización de matrices.

3.1.1 Vecinos próximos (kNN)

Este método selecciona los *k* vecinos más similares al usuario o al ítem objetivo, de forma que mediante la combinación lineal del rating de los vecinos se realiza una predicción de rating. A partir de esta predicción, podemos sacar el ránking de ítems a recomendar, simplemente ordenándolos por orden descendente del rating predicho (Koren 2010).

Existe una gran cantidad de variaciones para estos algoritmos, ya que tienen muchos parámetros configurables que dan diferentes resultados dependiendo de los datos con los que se traten. El número *k* de vecinos, la similitud mínima a considerar como relevante, o el número de ítems en común para una similitud válida son algunos de los parámetros a establecer para estos algoritmos.

Los algoritmos de vecinos próximos se han desarrollado en dos perspectivas posibles: recomendación de vecinos próximos por usuario y por ítem.

Basado en usuario (User kNN)

Se recomiendan al usuario los ítems que han gustado a usuarios similares a éste. Su fórmula es la siguiente:

$$f(u,i) = \sum_{\substack{v \in N_k(u) \\ r(v,i) \neq \emptyset}} sim(u,v) * r(v,i)$$

Basado en ítem (Item kNN)

Se recomiendan al usuario los ítems que se parecen a ítems que le han gustado. Su función de ránking es muy parecida a la de User kNN:

$$f(u,i) = \sum_{\substack{j \in N_k(i) \\ r(u,j) \neq \emptyset}} sim(i,j) * r(u,j)$$

Normalmente, este algoritmo se utiliza con el número de vecinos igual al número total de ítems, es decir, que el vecindario de cada ítem son todos los demás existentes en el conjunto de datos. Sin embargo, en el presente trabajo se ha querido probar la efectividad del uso de vecindario, y por eso se ha implementado dicha versión, que ha resultado obtener buenos resultados, como se verá en el apartado 6 más adelante.

En cuanto a la función de similitud entre los usuarios o ítems, ésta se puede realizar mediante varios métodos. En las pruebas de este trabajo se ha incluido similitud por coseno, Jaccard y Pearson.

- **Coseno**: Mide el ángulo entre los vectores (ratings) de cada par de usuarios o ítems, de forma que son más similares aquellos cuyos vectores tienen la misma orientación.

$$sim(u,v) = \frac{\sum_{\substack{i:\, r(u,i) \neq \emptyset \\ r(v,i) \neq \emptyset}} r(u,i) * r(v,i)}{\sqrt{\sum_{i: r(u,i) \neq \emptyset} r(u,i)^2} \sqrt{\sum_{r(v,i) \neq \emptyset} r(v,i)^2}}$$

- **Jaccard**: Dos usuarios o ítems son más similares cuanto más parecida sea la intersección a la unión de ambos, es decir, cuantos más ratings en común tengan, independientemente del valor de rating.

$$sim(u,v) = \frac{|u \cap v|}{|u \cup v|} = \frac{|u \cap v|}{|u| + |v| - |u \cap v|}$$

- **Pearson**: Equivalente a la similitud por coseno, pero cada rating se centra en la puntuación promedio del usuario (o ítem) correspondiente. Para este caso existen dos versiones dependiendo de la forma de calcular el módulo de v: por intersección, donde se suman los ratings de los ítems que tiene en común con u; y total, donde también se incluyen los ratings de aquellos que u desconoce.

$$sim(u,v) = \frac{\sum_{\substack{i:\, r(u,i) \neq \emptyset \\ r(v,i) \neq \emptyset}} (r(u,i) - \bar{r}_u)\,(r(v,i) - \bar{r}_v)}{\sqrt{\sum_{i: r(u,i) \neq \emptyset} (r(u,i) - \bar{r}_u)^2} \sqrt{\sum_{i: r(v,i) \neq \emptyset} (r(v,i) - \bar{r}_v)^2}}$$

Esta fórmula representa la versión total de Pearson, pero la versión por intersección se diferencia de ésta en que, en el módulo de v que se calcula en el denominador, la condición del sumatorio no es $i: r(v,i) \neq \emptyset$ (todos los ítems de v) sino $i: r(v,i) \neq \emptyset, r(u,i) \neq \emptyset$ (los ítems en común de u y v).

Estos tres tipos de similitud tienen a su vez dos versiones, la normalizada y la no normalizada. La normalización se realiza para cada similitud entre usuario e ítem, y consiste en el cociente de la similitud concreta para ese ítem y la similitud total del usuario con todos sus ítems conocidos. De esta forma, la similitud obtenida para cada par usuario-ítem se encuentra en el rango de valores de preferencia utilizado, pudiendo calcular métricas de error como RMSE.

Los métodos de vecinos próximos son algoritmos de recomendación clásicos, que a pesar de su simplicidad, suelen ofrecer buenos resultados. Sin embargo, a partir del Premio Netflix, como se muestra en el artículo "The BellKor 2008 Solution to the Netflix Prize" (Koren 2008), se demostró que los modelos de factorización de matrices son superiores a los de kNN en términos de error de predicción (no así necesariamente, como veremos, en calidad de ránking).

3.1.2 Factorización de matrices

A los gustos de los usuarios y las características de los ítems subyace un "espacio no visible" que realmente determina por qué les gustan los ítems. Hay formalismos matemáticos que nos permiten hacer aflorar este tipo de espacio latente como unas cuantas dimensiones reducidas, sin tener que concretar qué son realmente esas dimensiones, y pudiendo sin embargo generar recomendaciones operando con ellas. La idea es representar tanto los ítems como los usuarios como vectores en un espacio de factores latentes, con una coordenada por factor que representa el grado de afinidad del usuario (o del ítem) hacia ese factor.

Este efecto se puede realizar mediante la factorización de matrices (figura 3), descomponiendo la matriz original de ratings en un producto de varias matrices, dos o tres dependiendo del algoritmo, obteniendo siempre una primera matriz de usuarios por factores y una última de factores por ítems.

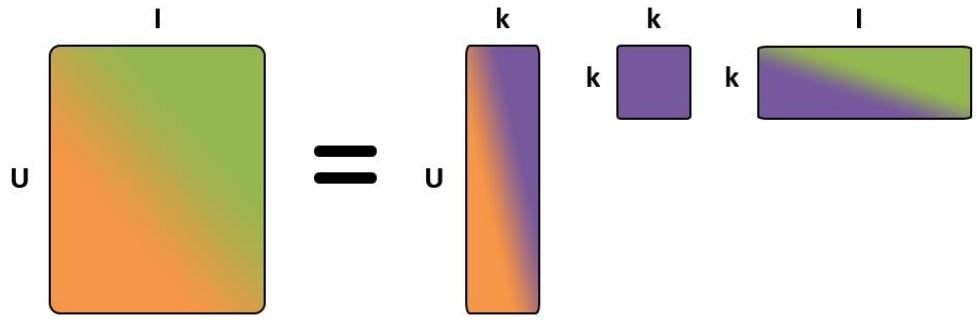

Figura 3. Descomposición de la matriz en el producto de otras tres.

De esta forma, se obtienen *k* factores latentes que establecen un espacio de características común, tanto para los usuarios como para los ítems, permitiendo la comparación directa entre ellos. Así, un usuario da ratings de acuerdo a sus factores latentes y a los factores latentes del ítem.

El número de factores latentes, *k*, es un número arbitrario, y podría corresponderse, por ejemplo, al grado de complejidad de los personajes o la cantidad de acción en una película.

A partir de aquí se ha desarrollado en el área gran cantidad de algoritmos para obtener la factorización de matrices, entre los que destacamos los siguientes.

pLSA (probabilistic Latent Semantic Analysis)

Divide la matriz de ratings en dos, usuarios por factores y factores por ítems, de forma que su función de ránking equivale a la probabilidad de que el usuario puntúe al ítem según el espacio de factores latentes (Hofmann 1999).

$$f(u,i) = p(i|u;\theta) = \sum_{f} p(i|f) * p(f|u)$$

SVD (Singular Value Decomposition)

La factorización de la matriz de ratings genera tres matrices, para lo que se tiene en cuenta todos los ratings, asignándoles un cero a aquellos no conocidos. Se obtienen los vectores de usuario e ítem mediante el producto de las matrices obtenidas según la fórmula que se indica a continuación, de forma que su función de ránking consiste en la multiplicación de los vectores de usuario e ítem, centrado en la media del usuario (Sarwar 2000).

$$r = W * S * F^T$$

$$f(u,i) = \bar{r}_u + W\sqrt{S}^T(u) * \sqrt{S}\,F^T(i)$$

SVDN (SVD no-empty entries)

Variante del anterior algoritmo en la que se obtienen dos matrices en lugar de tres, se tiene en cuenta únicamente los ratings conocidos, y su función de ránking no se centra en la media del usuario.

$$r = U_F * I_F$$

En este caso, entonces, el cálculo de dicha función es más inmediato, ya que los vectores de usuario e ítem se corresponden con la fila de la primera matriz y la columna de la segunda respectivamente, por lo que únicamente se multiplican la fila por la columna correspondiente (Koren 2009).

$$f(u,i) = U_F(u) * I_F(i)$$

HSVD (SVD with Hypergraph transformation)

Este método está dirigido a la recomendación de ítems a usuarios nuevos en el sistema, es decir, con poca actividad en el entrenamiento. Por ello, el primer paso del algoritmo consiste en dividir la matriz de ratings en tres: ratings de los usuarios que no están en test, ratings conocidos (de entrenamiento) de los usuarios que están en test, y ratings desconocidos de los usuarios que están en test (lo que se recomendará), como se muestra en la figura 4 a continuación.

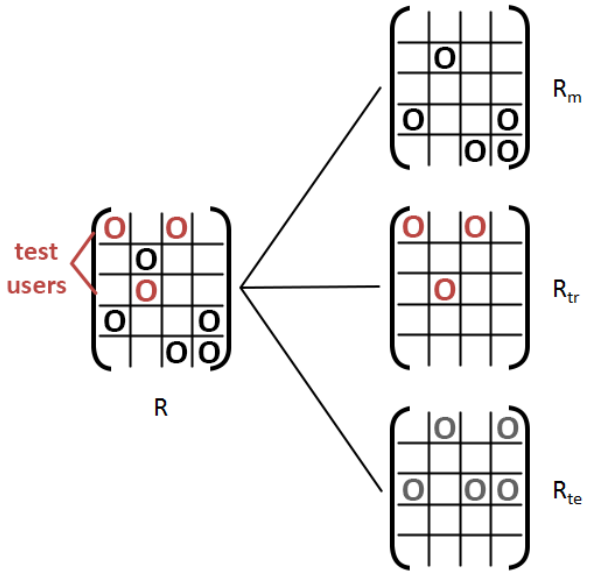

Figura 4. División de la matriz en otras tres según plantea el algoritmo HSVD.

Se realiza la factorización de matrices mediante los algoritmos anteriormente explicados de SVDR o SVDN, binarizando la matriz de ratings y normalizando cada columna.

$$X_m = (R_m > 0)$$

$$\bar{X}_m = D_e^{-\frac{1}{2}} * X_m * D_v^{-\frac{1}{2}}$$

$$SVD(\bar{X}_m) = W * S * F^T$$

Utilizando únicamente la matriz de la derecha (tercera o segunda dependiendo el método usado), se resuelve mediante *least-squares* la matriz de ratings estimados completa, a partir de la cual se realizan las recomendaciones.

$$Resolver\ F\theta = R_{tr}^T\ mediante\ mínimos\ cuadrados\ para\ obtener\ \theta$$

$$\hat{R}_{te} = \theta^T F^T$$

$$f(u,i) = \hat{R}_{te}(u,i)$$

Puede verse una descripción más detallada de este método en (Pu 2013).

ASVD (Assymetric SVD)

Se trata de la versión asimétrica de SVD, que solo usa la tercera matriz de la descomposición para realizar las recomendaciones. Sigue exactamente los mismos pasos que HSVD, con la diferencia de que no se binariza la matriz de ratings de los usuarios viejos, sino que se utiliza la original con los ratings numéricos. En otras palabras, el algoritmo es idéntico excepto que, según la figura 4, en lugar de usar X_m se usa R_m (Koren 2008).

3.2 Basados en contenido (BC)

Este grupo de algoritmos se basa en la utilización de la descripción de cada ítem para recomendar, sin utilizar información de otros usuarios para generar la recomendación al usuario objetivo. Por ejemplo, para recomendar *tweets* en Twitter se pueden considerar las palabras o *hashtags* como la descripción del contenido del tweet (ítem) objetivo; y el contenido de los tweets escritos y *retweeteados* por el usuario objetivo como base para inferir los gustos de éste, a comparar con el ítem objetivo. Así, estos algoritmos intentan recomendar ítems que son parecidos a los que le han gustado al usuario anteriormente.

De este grupo, los algoritmos representativos seleccionados en el presente trabajo para realizar la comparación han sido Rocchio y kNN, este último similar pero no idéntico al de FC (sólo los diferencia la función de similitud, se explicará en el punto correspondiente). Rocchio y kNN usan ratings como pesos en el cálculo de los centroides y en la suma de las similitudes, respectivamente.

Rocchio

Se basa, como ya se ha anticipado, en el cálculo de centroides para cada usuario, de forma que se obtenga un vector "representante" para cada uno. Estas clases se corresponderán con las características (*features*) de los ítems, por ejemplo, en Twitter, las palabras clave del contenido de los tweets. De esta forma, se obtiene para cada usuario un centroide que representa su relación con cada característica (término) (Adomavicius 2005).

La fórmula para el cálculo de los centroides es la siguiente:

$$u[f] = \frac{1}{|\mathbf{u}|} \sum_{i:r(u,i)\neq\emptyset} tfidf(f,i) * r(u,i), \text{ donde } \mathbf{u} = \{r(u,i) \neq \emptyset | i \in \mathcal{I}\}$$

Donde $u[f]$ denota el valor del centroide de usuario para una característica *f*, y $tfidf(f,i)$ es la importancia que tiene la característica en el ítem y en general, ya que también tiene en cuenta la "rareza" de la característica (una muy común tendrá menos importancia que otra más rara en el ítem).

Una vez se dispone de los centroides, el cálculo de la similitud de los usuarios con cada uno de los ítems se realiza mediante cualquiera de los métodos anteriormente descritos. En este caso he seguido la fórmula de similitud mediante coseno:

$$f(u,i) = sim(u,i) = \frac{\sum_f u[f] * tfidf(f,i)}{\sqrt{\sum_f u[f]^2}\ \sqrt{\sum_f tfidf(f,i)^2}}$$

Item kNN

La estructura de este algoritmo es idéntica a la de FC del mismo nombre, pero se diferencian en la forma de calcular la similitud entre los ítems. Mientras el de FC utiliza los ratings de otros usuarios, éste utiliza la descripción de los ítems. Por ejemplo, mientras el primero recomendaría películas siguiendo las puntuaciones de los usuarios, el segundo se basaría en, por ejemplo, el género, la sinopsis, el director, y/o el reparto de cada una de ellas (Adomavicius 2005).

A partir de aquí, las fórmulas se representan prácticamente de la misma forma, pero teniendo en cuenta la diferencia anteriormente descrita.

$$sim(i,j) = \frac{\sum_f tfidf(f,i) * tfidf(f,j)}{\sqrt{\sum_f tfidf(f,i)^2}\ \sqrt{\sum_f tfidf(f,j)^2}}$$

3.3 Algoritmos no personalizados

Estos algoritmos recomiendan ítems sin conocer ningún dato del usuario, de forma impersonal, como su propio nombre indica.

Estos algoritmos han sido implementados en el presente trabajo con el objetivo de dar un punto de referencia estándar de las métricas para cada uno de los conjuntos de datos utilizados. De esta forma, se establece un límite inferior sobre el que comparar los diferentes tipos de algoritmos más elaborados.

Popularidad

Recomienda los ítems por orden de popularidad y, por tanto, da exactamente el mismo ránking a cada uno de los usuarios. Por "popularidad de un ítem" se entiende el número de usuarios que han interactuado con el ítem (para ser más precisos, que el sistema ha observado interactuando).

$$f(u,i) = |\mathbf{i}|, \text{ donde } \mathbf{i} = \{r(v,i) \neq \emptyset \mid v \in U\}$$

Este método, que puede parecer trivial, es sin embargo uno de los más extendidos en escenarios reales. Es el que se nos muestra en las listas de "más vendidos", vídeos con más visitas en Youtube, tweets más retweeteados, programas de TV con más audiencia, ránking de taquilla en cine, *best-sellers*, etc.

Random

Recomienda ítems de forma aleatoria a cada usuario y su precisión está relacionada con la densidad del conjunto de datos.

$$sim(u,i) = random()$$

La efectividad de la recomendación aleatoria, medida con una cierta métrica, se puede interpretar como la esperanza del valor de la métrica sobre el conjunto de datos en el que se aplica, y depende generalmente de la densidad de los datos. Es decir, a mayor tasa de pares usuario-ítem con rating observado (en el conjunto de test, para ser más precisos), mayor es el valor de la métrica de la recomendación aleatoria.

4. Métodos de evaluación

Los recomendadores incluidos en este estudio se basan en el uso de ratings únicos, es decir, sólo un rating por cada par usuario-ítem. Sin embargo, no todos los conjuntos de datos con los que se ha trabajado están en este formato, por lo que se ha optado por realizar una conversión de frecuencias o tipos de eventos a ratings en los casos donde era necesario.

A continuación se describen los conjuntos de datos que se han utilizado para este estudio, su forma de partición en entrenamiento y test, el manejo de los tipos de interacción entre usuarios e ítems, y el mapping de frecuencias a ratings para los conjuntos de datos donde es necesaria la conversión. Finalmente, se detallarán las métricas que han sido utilizadas para realizar la evaluación y la comparación entre los diferentes recomendadores estudiados.

4.1 Configuración de experimentos

La configuración de los experimentos ha consistido en una fase de pre-procesamiento de cada conjunto de datos con el objetivo de que todos ellos sean lo más parecidos posibles para su correcta comparación.

Los algoritmos que se han decidido probar utilizan los datos en forma de ratings, puntuaciones de los usuarios a los ítems, únicas para cada par. Sin embargo, no todos los conjuntos de datos con los que se ha trabajado están en este formato, sino que en algunos casos están representados en forma de frecuencias y/o eventos de interacción entre usuarios e ítems (p.e. número de veces que un usuario ha escuchado una canción en Last.fm o ha visitado un producto en la tienda online gestionada por Blueknow).

Puesto que el objetivo que se persigue es la comparación entre los algoritmos en cada uno de los conjuntos de datos disponibles, se ha optado por convertir todos los datos a un formato común: ratings.

Para la conversión de frecuencias a ratings, se pueden obtener los ratings y realizar las particiones de entrenamiento y test de múltiples formas, que pueden influir en los resultados que se obtienen según el método utilizado. Estos son algunos de los parámetros de decisión en cuanto a los modos de partición de datos:

- **Split de datos antes/después de convertirlos a ratings**. En el caso de realizar la partición antes, hay que tener en cuenta que es muy probable que un mismo par de usuario-ítem se encuentre tanto en entrenamiento como en test, por lo que es necesario descartar uno y se está perdiendo información. Sin embargo, si se hace después esto no ocurre, ya que al convertirlo en ratings ya se está asegurando que los pares usuario-ítem son únicos, y se puede obtener particiones más comparables (con, aproximadamente, el mismo ratio de entrenamiento y test). Ello equivale a asegurar que todas las interacciones entre un par usuario-ítem dado queden completamente a un lado u otro de la partición (entrenamiento o test).
- **División de datos dependiendo del tipo de evento**. Otra posibilidad sería seleccionar uno o más tipos de evento para que formen parte del entrenamiento y otra del test. Por ejemplo, si se dispone de datos de visitas y compras, podría ser lógico poner las visitas en entrenamiento y las compras como test.

Por otro lado, respecto a la agregación de eventos estos son algunos de los parámetros de decisión planteados:

- **Ponderación de los eventos**. Cada tipo de evento puede ponderarse de forma diferente dependiendo la importancia que se le quiera atribuir a cada uno, estableciendo pesos para cada uno o utilizando simplemente el valor de su frecuencia.
- **Fusión de eventos para obtener ratings**. Una vez que se dispone de los datos ponderados, para unificar los tipos de evento de cada par usuario-ítem existen varias posibilidades, como elegir el máximo, realizar una suma ponderada normalizada (dividiendo entre la suma total) o sin normalizar.

Para este trabajo, en los casos en los que fue necesaria la conversión a ratings se ha realizado convirtiendo primero los datos a ratings (como se explicará de forma particular para cada conjunto de datos más adelante en la sección 4.3) y realizando la partición en entrenamiento y test posteriormente a la conversión de frecuencias a ratings.

La figura 5 muestra el mapa completo para la realización de los experimentos para cada algoritmo sobre cada conjunto de datos, incluyendo la recomendación y la evaluación.

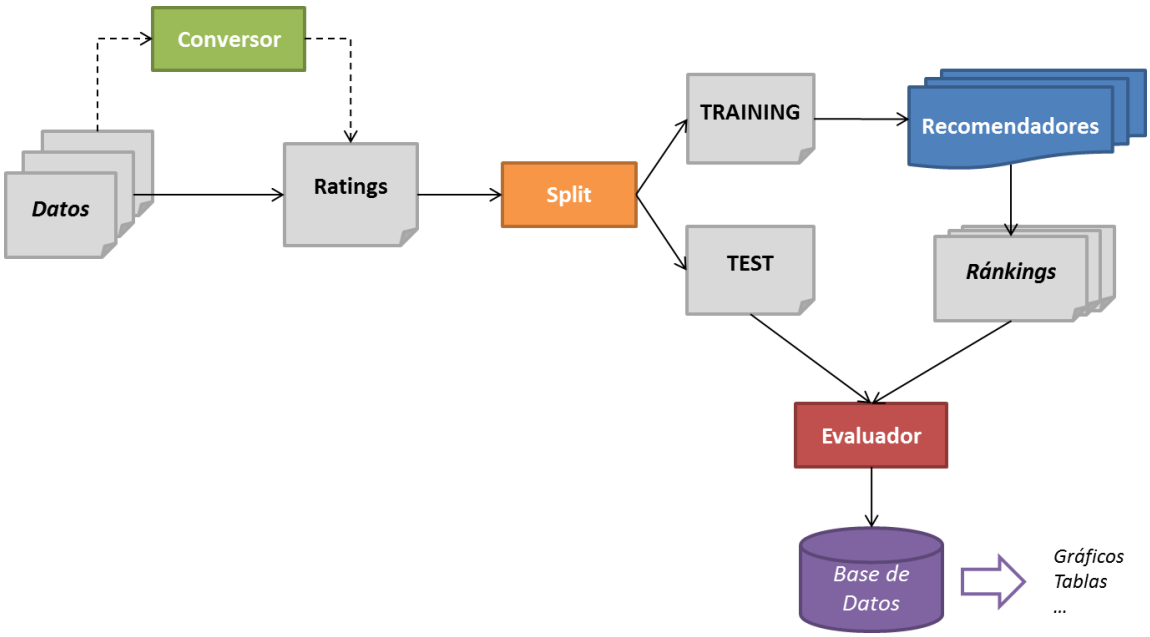

Figura 5. Mapa de la metodología seguida para la realización de experimentos.

4.2 Conjuntos de datos utilizados

Para la realización de este trabajo se han utilizado varios conjuntos de datos, cada uno de ellos con diferentes características, como la proporción de usuarios frente a ítems o la densidad de ratings. Las estadísticas de cada conjunto de datos se muestran en la tabla 1.

Conjuntos de datos	Usuarios	Ítems	Ratings	Densidad
MovieLens 100K	943	1.682	100.000	6,30%
MovieLens 1M	6.040	3.900	1.000.000	4,25%
MovieLens 10M	71.567	10.681	10.000.000	1,31%
Netflix	17.770	480.189	2.711.291	0,03%
Last.fm	992	176.948	896.653	0,51%
Blueknow[2]	57.888	6.087	597.328	0,17%
Twitter	10.028	2.023.133	2.139.284	0,01%

Tabla 1. Estadísticas de los conjuntos de datos utilizados para este estudio.

Cada uno de los conjuntos de datos tiene una razón por la cual se ha incluido en este estudio. A continuación se exponen las motivaciones junto con las características y variantes de cada uno de estos conjuntos de datos.

MovieLens

Se trata de datos para la recomendación de películas a usuarios. El conjunto de datos está formado por un fichero de ratings (puntuación de usuarios a películas) y otro de géneros (a los que pertenece cada una de las películas). Este último fichero da la posibilidad de calcular las métricas de novedad y diversidad que requiere el establecimiento de atributos para los ítems, en este caso identificados con los géneros de las películas.

Existen tres versiones de este conjunto de datos, de diferentes tamaños: con 100.000, un millón y 10 millones de ratings, de forma que su densidad, número de usuarios y número de ítems varía entre estas versiones. Las películas están puntuadas de 1 a 5, correspondiendo a valoraciones desde muy mala hasta muy buena, respectivamente.

Estos conjuntos de datos han sido distribuidos por el grupo de investigación GroupLens Research Lab de la Universidad de Minnesota, que ha recogido y preparado los conjuntos de datos desde la página de MovieLens[3]. Estos datos se recogieron en diferentes periodos para cada versión, y fueron sometidos a una limpieza, en la que se descartaron los usuarios con menos de 20 ratings o con datos insuficientes.

Netflix

Igual que MovieLens, se trata de datos para la recomendación de películas, donde los datos se encuentran también en forma de ratings (puntuación de usuarios a películas). En este caso, sin embargo, no se trabaja con ningún fichero de géneros, por lo que hemos descartado las métricas de novedad y diversidad para este conjunto de datos, pues no pueden todas ellas computarse.

Este conjunto de datos fue el distribuido para la realización del Netflix Prize[4], y consta de más de 100 millones de ratings a 17.000 películas por 480.000 usuarios anónimos escogidos al azar. Las puntuaciones están en una escala de 1 a 5 estrellas, al igual que MovieLens.

Este conjunto ha sido clave en la mejora de la escalabilidad y optimización de los algoritmos utilizados en este trabajo, debido principalmente al gran número de ratings del que dispone.

[2] Estos valores son los obtenidos tras la limpieza de usuarios con menos de 5 interacciones. Antes de la limpieza se disponía de 503.452 usuarios y 6.367 ítems.

[3] http://movielens.org

[4] http://www.netflixprize.com/

Last.fm

Este conjunto de datos incluye información para la recomendación de artistas (musicales) a usuarios. Los datos proceden de Last.fm[5], un servicio online de escucha y recomendación de música. El conjunto de datos con el que se ha trabajado en este caso ha sido recogido por Òscar Celma mediante la API de Last.fm, obteniendo los datos de los artistas más escuchados.

En la forma original del conjunto de datos existen dos tipos de ítems posibles: canciones y artistas (donde cada canción está asociada al artista o grupo que la interpreta). Los datos de interacción usuario-ítem se disponen de forma de registros de acceso con marca temporal correspondientes a los instantes en los que los usuarios escucharon una canción del catálogo, y no facilita ningún otro dato para establecer los atributos de los ítems necesarios para calcular las métricas de novedad y diversidad. Para la aplicación de los algoritmos de recomendación, optamos por tomar como ítems los artistas, y agregamos los registros de interacción contabilizando las frecuencias por cada par usuario-artista (esto es, número de veces que el usuario ha escuchado canciones del artista) como dato asociado al par.

Twitter

Este conjunto de datos se ha utilizado para la recomendación de tweets a usuarios. El conjunto de datos consta de eventos de interacción entre usuarios y tweets: los usuarios escriben, responden o hacen retweet de otros tweets. Como atributos para los ítems se han escogido los hashtags de los que cada ítem dispone (con la limitación de que en muchos casos el ítem no tendrá por tanto ningún atributo, puesto que muchos tweets no tienen tags). Se ha utilizado asimismo el texto de los tweets.

Se trata de otro caso real, esta vez facilitado por el Trabajo de Fin de Grado de Alfonso Alhambra Morón (Alhambra 2014), dentro del cual se han obtenido y tratado estos datos a través de la API de Twitter y su posterior almacenamiento y organización en una base de datos. El trabajo de A. Alhambra se centra en la recomendación de contactos en Twitter, mientras que en éste abordamos la recomendación de tweets.

El conjunto de datos incluye los 200 últimos tweets de 10.000 usuarios aleatorios, añadiendo también las interacciones de estos usuarios con los tweets seleccionados. El conjunto de datos incluye además las relaciones *follow* entre todos estos usuarios, que no utilizamos nosotros en el presente trabajo, pues nos ceñiremos a algoritmos de recomendación de tweets mediante filtrado colaborativo y algoritmos basados en contenido. Todos los datos de este conjunto están almacenados en una base de datos MySQL, a la que se accede directamente para las pruebas que documentamos en la sección 6.

Blueknow

Se trata de datos cedidos por la empresa Blueknow al grupo de investigación en el que se ha realizado el presente trabajo, que han sido recogidos a través de su servicio de recomendación en tiendas online, debidamente anonimizados. Para esta colaboración se han seleccionado datos de interacción y compra de una tienda de juguetes en el periodo de un mes. El conjunto de datos viene dispuesto en tres archivos que contienen los datos en forma de frecuencias de evento: número de veces que los usuarios han comprado (archivo de compras), visitado (archivo de visitas) o hecho click (archivo de clicks) a un producto. En el segundo se incluyen además diferentes categorías de los ítems, clasificándolos según su tipo (juguete, puericultura o libro), rango de edad, precio y marca.

[5] http://www.last.fm/

Blueknow[6] es una empresa española que proporciona una *suite* de personalización de *eCommerce* (compra y venta de productos por Internet) para web, dispositivos móviles y correo electrónico. Se trata de un caso real de datos proporcionados por esta empresa al grupo de investigación en el que he realizado este trabajo, con el objetivo de investigar diferentes algoritmos de recomendación sobre el conjunto de datos para la posible mejora de su sistema. El conjunto de datos utilizado se corresponde con los datos de tres meses de uso de un cliente de esta empresa, Eurekakids[7], que vende productos infantiles (juguetes, puericultura y libros).

Se identifican dos tipos de usuarios: anónimos y autenticados. Los primeros son aquellos usuarios que navegan por una página con recomendador Blueknow integrado y a cada uno se le asigna un identificador de usuario anónimo que se guarda en una *cookie*, de forma que se mantiene ese mismo identificador para esa y futuras sesiones, mientras no se elimine las cookies de su navegador. Por otro lado, los usuarios autenticados son aquellos que inician sesión en la página correspondiente, de forma que su perfil se mantiene aunque se cambie de navegador, ordenador o se borren las cookies. A efectos de los experimentos que se han realizado en el presente trabajo, se tratan estos tipos de usuarios indistintamente sin hacer diferencia entre ellos.

Se dispone de tres tipos de eventos: visitas ($\mathcal{V}$), clicks ($\mathcal{C}$) y compras ($\mathcal{P}$), y la tipología de eventos se produce como sigue. Se registra un evento de visita cuando el usuario accede a la descripción del ítem; uno de click cuando se visita un ítem de los que Blueknow recomienda; y uno de compra cuando se ha realizado la compra de los productos del carrito. Normalmente, para que un ítem haya sido comprado o hecho click tiene que haber sido visitado, pero en algunos casos no es así debido a la partición temporal realizada. La figura 6 muestra el solapamiento real (obtenido por cada par usuario-ítem en particular) de los diferentes tipos de eventos.

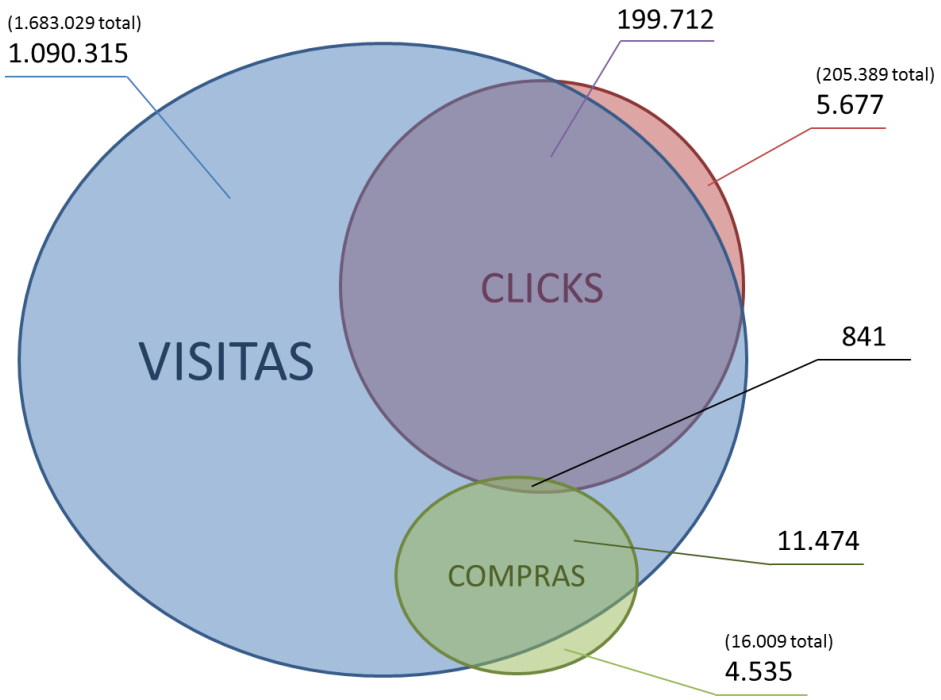

Figura 6. Relación entre los conjuntos de eventos de Blueknow. Se muestra el número de eventos en cada caso.

[6] http://www.Blueknow.com/
[7] http://www.eurekakids.es/

Los conjuntos de visitas y clicks dan lugar a varias interpretaciones, ya que su uso y significado están muy ligados entre sí: uno se realiza únicamente sobre los ítems recomendados por Blueknow (click), mientras que el otro sobre todos los ítems del conjunto de datos (visita). Así, los ítems recomendados por Blueknow cuentan con el doble de eventos (un evento de click conlleva asimismo uno de visita) que el resto, por lo que se ha realizado un estudio con tres interpretaciones, cambiando el uso de estos tipos de eventos para la posterior conversión a ratings:

- **Tratar como eventos independientes.** Si se tienen en cuenta los tres tipos de eventos tal cual se dispone de ellos, lo ítems recomendados por Blueknow salen favorecidos a la hora de la conversión a ratings. Esto se debe a que los eventos de click se suman con los de visita, por lo que una misma acción, acceder a la información de un ítem, puede dar el doble de importancia a uno frente a otro dependiendo si éste es recomendación de Blueknow o no.

$$events(u) = \{\mathcal{P}_u, \mathcal{C}_u, \mathcal{V}_u\}$$

- **Eliminar los eventos de click.** Una forma de reducir la ventaja anterior de los ítems recomendados por Blueknow frente al resto es tener en cuenta únicamente los eventos de visitas y compras, omitiendo los eventos de click.

$$events(u) = \{\mathcal{P}_u, \mathcal{V}_u\}$$

- **No tener en cuenta los ítems con clicks del usuario.** Para reducir más drásticamente la ventaja de éstos ítems, se puede tener en cuenta, para cada usuario, los ítems que ha visitado pero que no ha hecho click (es decir, tratar con el conjunto de visitas eliminando la intersección con el conjunto de clicks).

$$events(u) = \{\mathcal{P}_u, \mathcal{V}_u - (\mathcal{V}_u \cap \mathcal{C}_u)\}$$

Para escoger la interpretación más correcta, en el apartado 4.3 se muestra el estudio realizado donde cada una de estas interpretaciones se convierte en un conjunto de datos independiente y sobre el que se van a realizar pruebas de relevancia con algunos algoritmos de recomendación para escoger la mejor opción y realizar la comparación con el resto de conjuntos de datos y algoritmos.

Como extra, Blueknow puso a nuestra disposición un cuarto archivo extra con las recomendaciones generadas por su sistema. Cada línea del archivo se corresponde con una recomendación de cuatro ítems a un usuario, y se genera una recomendación cada vez que un usuario visita un ítem (para un mismo usuario hay varias recomendaciones, normalmente incluso con repetición de ítems recomendados). Este archivo se convirtió al formato del presente estudio (con un ránking por cada usuario de test) ordenando de forma descendente los ítems por número de veces que habían sido recomendados al usuario objetivo. Estas recomendaciones obtenidas nos pueden aportar una idea de los valores de relevancia que puede estar rondando su sistema de recomendación, aunque hay que matizar que estos valores son únicamente una aproximación.

Como se ha podido observar, sólo unos pocos de los múltiples conjuntos de datos tienen sus datos en forma de ratings. Por ello, a continuación se explicaremos la forma de tratar otros tipos de datos de interacción de los que se dispone en el resto de conjuntos de datos.

4.3 Utilización de ratings implícitos

La gran mayoría de los algoritmos documentados en el estado del arte de los sistemas de recomendación parten de la utilización de ratings explícitos, esto es, puntuaciones que los usuarios han hecho manifiestamente a un ítem (Adomavicius 2005). Sin embargo, otra forma de obtener información se puede realizar de forma implícita, es decir, sin que el usuario haga nada, únicamente observando su comportamiento o interacción con el sistema. Por ejemplo, si un usuario hace click en un objeto para acceder a sus características o comprarlo, esto puede ser indicio de que ese ítem le interesa al usuario objetivo. De hecho, ésta es la forma en que generalmente disponen de datos los sistemas de recomendación en aplicaciones reales, en las que los ratings explícitos no existen o son escasos en términos relativos.

Hay muchas formas de interpretar y tratar este *feedback* implícito: trabajando directamente con ellas, como se muestra en (Koren 2008), o convirtiendo los datos, como en (Celma 2008). En nuestro caso, optamos por la conversión de datos con frecuencias a ratings, en la línea de Celma. Esta opción simplifica el tratamiento de datos implícitos, pues permite la aplicación inmediata de la amplia batería de algoritmos de recomendación del estado del arte, mayoritariamente diseñados para datos explícitos.

En términos generales, la conversión consiste en asignar a cada tipo de evento un peso específico, y para unificar los diferentes eventos para un mismo par usuario-ítem, se realiza, por ejemplo, una suma ponderada. Esta adaptación se concretará de forma particular para cada conjunto de datos como se expondrá a continuación, ya que está en nuestras manos la elección de cómo manejar estos eventos, y ninguno se interpreta de la misma forma ni se le da la misma importancia a los diferentes eventos de los que se dispone en cada caso.

En los apartados que siguen explicamos cómo se ha materializado este tratamiento en cada caso, para los conjuntos de datos de Last.fm, Blueknow y Twitter.

Last.fm

Como hemos indicado anteriormente se dispone primariamente del número de veces que cada usuario ha escuchado una canción de un artista. En nuestros experimentos no se recomiendan canciones sino artistas, por lo que hay que hacer un paso intermedio, antes de pasar a ratings, que consiste en obtener las frecuencias de interacción usuario-ítem a nivel de artista. Hay varias formas de hacer esta conversión, pero en este caso se ha optado por la más simple e intuitiva: la frecuencia de cada par usuario-artista equivale a la suma de las veces que ese usuario ha escuchado cada una de las canciones de ese artista.

A partir de las tuplas usuario-artista-frecuencia, se realiza la conversión a rating. La conversión se lleva a cabo para cada usuario independientemente, ordenando los artistas y atribuyéndoles de forma equitativa pesos entre el máximo y mínimo rating especificado (esta vez sin tener en cuenta ningún tipo de evento). En definitiva, esto equivale a asignar una puntuación lineal por percentiles de frecuencia, de forma que, por ejemplo para un rango de rating 1 a 5, el artista menos escuchado recibe un 1, el más escuchado un 5, y los artistas intermedios reciben puntuaciones intermedias con separación uniforme sobre el intervalo [1,5]). El proceso completo se ilustra en la figura 7.

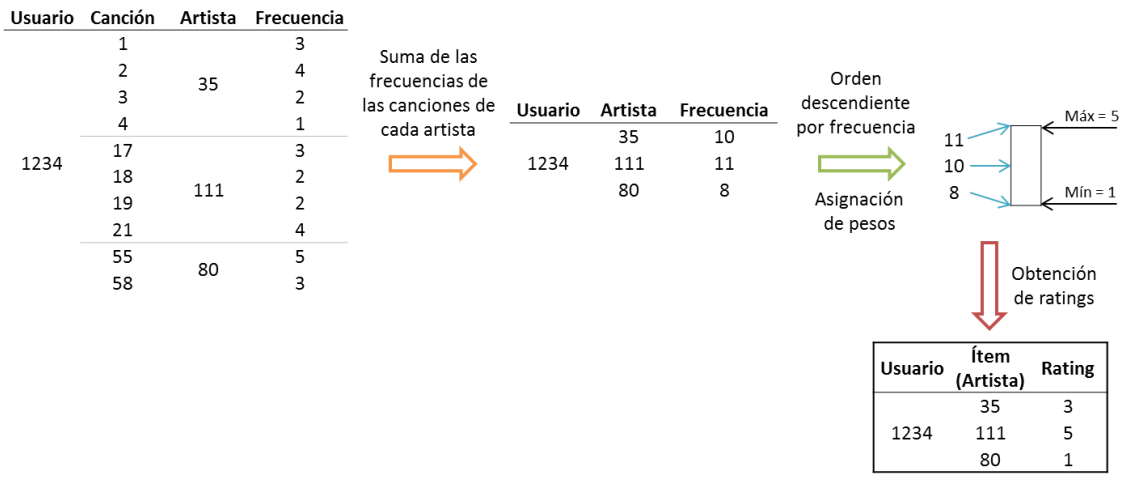

Figura 7. Proceso de conversión de eventos a puntuaciones para Last.fm.

Twitter

En el caso de Twitter las interacciones son únicas para cada par usuario-ítem: un usuario puede escribir, retweetear o responder un tweet (ítem), pero sólo una vez el mismo tweet.

Hemos optado por poner diferentes pesos a cada tipo de interacción, de forma que de mayor a menor importancia se consideran los tweets escritos por el usuario, los respondidos por éste y en último lugar, los retweets realizados (pesos con valor de 3, 2 y 1 respectivamente). El presente trabajo no se entra en un estudio en profundidad sobre qué pesos serían mejores, sino que se asignan con criterio informal para realizar experimentos con carácter exploratorio y de prueba de concepto, para una primera observación del comportamiento de diferentes algoritmos de recomendación, por lo que la asignación de pesos queda como elemento configurable.

Una vez se tienen los pares de usuario-ítem y se le ha asignado a cada tipo de interacción un peso, la conversión a ratings está realizada. El proceso se muestra en la figura 8.

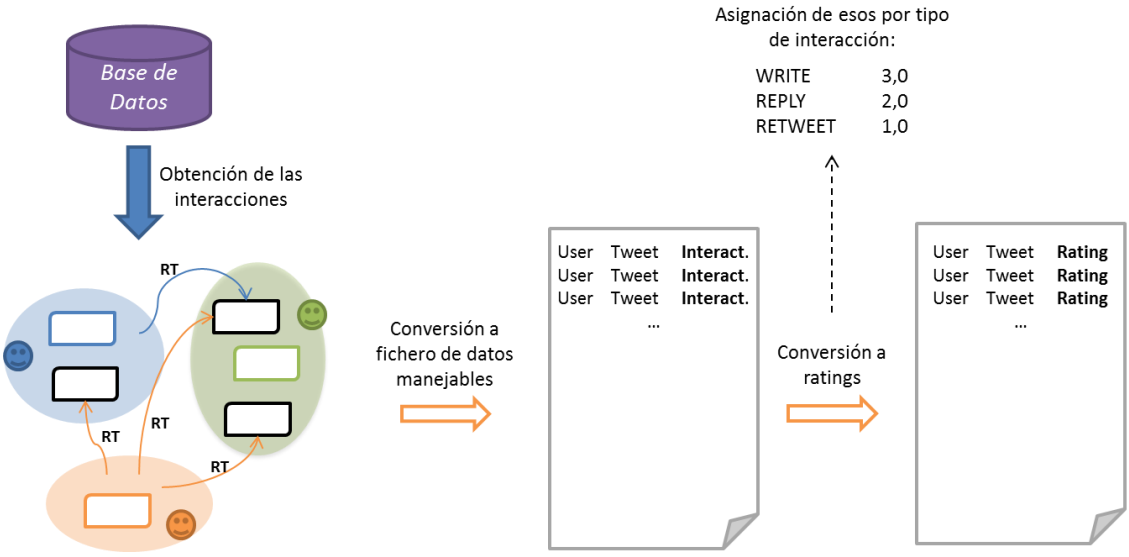

Figura 8. Proceso de conversión de eventos a puntuaciones para Twitter.

Blueknow

Para este conjunto de datos se dispone de distintos tipos de eventos (compras, visitas y clicks) pero, a diferencia de Last.fm, estas interacciones no son únicas para cada par usuario-ítem (como se ha explicado en la sección 4.2), por lo que además se dispone de frecuencias (número de veces que se realiza el evento para cada par usuario-ítem).

Por un lado, el manejo de tipos de evento se ha hecho de forma que tengan mayor peso las compras respecto a los clicks y visitas, pero no está clara la distinción entre estos dos eventos. Por ello, se ha realizado un estudio según las tres interpretaciones planteadas en el apartado 4.2, añadiendo además las recomendaciones realizadas por Blueknow según el archivo de recomendaciones obtenidas.

Primero se expondrá el estudio realizado sobre estas tres particiones, y a continuación se explicará de forma específica cómo se realizó la conversión de estos datos a ratings (son los pasos para la conversión en el caso de la primera interpretación, en la que se toman los tres eventos de forma independiente, pero la conversión en los otros casos se realiza de la misma forma).

En las tablas 2, 3 y 4 se muestran los resultados de los algoritmos pertenecientes al grupo IRG junto con los del archivo de recomendación de la propia empresa de Blueknow.

Blueknow										
Recomendador	Relevancia								Tiempos	
	P@10	P@20	nDCG-n@10	nDCG-n@20	nDCG-l@10	nDCG-l@20	nDCG-e@10	nDCG-e@20	Offline	Online
ItemKNN (cosine)	0,1349	0,0969	0,2270	0,2624	0,2261	0,2621	0,2259	0,2606	00:02:49,34	00:02:53,32
ItemKNN (jaccard)	0,1331	0,0958	0,2218	0,2569	0,2215	0,2572	0,2202	0,2545	00:01:00,95	00:02:56,72
Blueknow	0,1262	0,1240	0,1240	0,1833	0,1290	0,1871	0,1149	0,1737	-	-
UserKNN (cosine 400)	0,1210	0,0871	0,2063	0,2390	0,2057	0,2388	0,2053	0,2373	00:10:14,31	00:08:15,78
UserKNN (jaccard 250)	0,1204	0,0866	0,2024	0,2354	0,2027	0,2361	0,2005	0,2328	00:07:51,00	00:07:10,46
pLSA	0,0725	0,0569	0,1148	0,1421	0,1150	0,1424	0,1137	0,1406	00:05:23,54	00:02:21,33
Most Popular	0,0162	0,0138	0,0234	0,0304	0,0233	0,0305	0,0231	0,0299	00:00:00,00	00:00:12,42
Random	0,0012	0,0011	0,0013	0,0019	0,0014	0,0020	0,0012	0,0019	00:00:00,00	00:00:10,76

Tabla 2. Resultados de las métricas de precisión sobre el conjunto de datos de Blueknow, tomando las compras, clicks y visitas de forma independiente. No se dispone de datos de tiempo de ejecución del algoritmo de Blueknow.

Blueknow										
Recomendador	Relevancia								Tiempos	
	P@10	P@20	nDCG-n@10	nDCG-n@20	nDCG-l@10	nDCG-l@20	nDCG-e@10	nDCG-e@20	Offline	Online
ItemKNN (cosine)	0,1324	0,0962	0,2205	0,2572	0,2203	0,2577	0,2189	0,2546	00:07:49,67	00:02:34,06
ItemKNN (jaccard)	0,1308	0,0961	0,2186	0,2565	0,2187	0,2573	0,2167	0,2536	00:01:52,63	00:02:27,66
Blueknow	0,1243	0,1226	0,1056	0,1574	0,1163	0,1696	0,0922	0,1412	-	-
UserKNN (cosine 400)	0,1216	0,0878	0,2055	0,2395	0,2054	0,2398	0,2042	0,2374	00:09:40,10	00:07:09,59
UserKNN (jaccard 250)	0,1209	0,0872	0,2009	0,2348	0,2016	0,2359	0,1985	0,2318	00:13:33,65	00:05:53,00
pLSA	0,0635	0,0517	0,0990	0,1257	0,0993	0,1263	0,0980	0,1242	00:06:22,01	00:01:54,06
Most Popular	0,0150	0,0134	0,0206	0,0282	0,0209	0,0286	0,0202	0,0278	00:00:00,00	00:00:20,33
Random	0,0014	0,0014	0,0015	0,0022	0,0016	0,0024	0,0014	0,0021	00:00:00,00	00:00:24,63

Tabla 3. Resultados de las métricas de precisión sobre el conjunto de datos de Blueknow, tomando únicamente las compras y visitas (se omiten los eventos de clicks). No se dispone de datos de tiempo de ejecución del algoritmo de Blueknow.

Blueknow										
Recomendador	Relevancia								Tiempos	
	P@10	P@20	nDCG-n@10	nDCG-n@20	nDCG-l@10	nDCG-l@20	nDCG-e@10	nDCG-e@20	Offline	Online
ItemKNN (jaccard)	**0,1012**	**0,0774**	0,1601	0,1931	0,1608	0,1945	0,1583	0,1905	00:01:32,44	00:01:51,64
ItemKNN (cosine)	0,1006	0,0771	**0,1606**	**0,1938**	**0,1612**	**0,1949**	**0,1593**	**0,1915**	00:04:02,16	00:01:55,34
UserKNN (jaccard 250)	0,0918	0,0691	0,1454	0,1749	0,1463	0,1761	0,1436	0,1724	00:04:40,07	00:03:40,96
UserKNN (cosine 400)	0,0912	0,0689	0,1469	0,1769	0,1474	0,1776	0,1455	0,1751	00:05:40,73	00:04:37,15
Blueknow	0,0756	0,0751	0,0661	0,0990	0,0721	0,1058	0,0584	0,0896	-	-
pLSA	0,0445	0,0380	0,0655	0,0869	0,0660	0,0874	0,0648	0,0859	00:04:41,70	00:01:25,30
Most Popular	0,0120	0,0107	0,0161	0,0220	0,0164	0,0224	0,0156	0,0215	00:00:00,00	00:00:12,59
Random	0,0016	0,0015	0,0018	0,0025	0,0018	0,0026	0,0017	0,0024	00:00:00,00	00:00:16,64

Tabla 4. Resultados de las métricas de precisión sobre el conjunto de datos de Blueknow, tomando las compras y los eventos de visitas que no estén emparejados con un evento de click. No se dispone de datos de tiempo de ejecución del algoritmo de Blueknow.

Debido a que los resultados son parecidos en las tres interpretaciones, podemos considerar que la diferencia entre click y visita es irrelevante en cuanto a su significado como indicio de interés, por lo que en este caso no se hará distinción entre estos dos conjuntos y se unirán (i.e. es equivalente un ítem con dos visitas que uno con una visita y un click).

En resumen, mientras que en el caso anterior de Last.fm disponíamos de datos en forma de tuplas usuario-artista-frecuencia, en Blueknow se añade una cuarta variante, el tipo de evento, al que se le asigna más o menos importancia dependiendo de si es una compra, click o visita.

La conversión a ratings se ha realizado de forma independiente para cada uno de los usuarios. Si el ítem fue comprado por el usuario objetivo al menos una vez, se le asigna la puntuación máxima (aunque su frecuencia de clicks y visitas se tiene en cuenta para el cálculo del peso para el resto de ítems). En caso contrario, se han seguido los pasos que se explican a continuación. El proceso completo se ilustra en la figura 9.

- Se agrupan los ítems según su frecuencia de clicks y visitas (la suma de las frecuencias de ambos eventos) y se ordenan de forma decreciente.
- Se crea un mapa de frecuencia a su peso asociado (dos ítems con la misma frecuencia obtienen el mismo peso). La mayor frecuencia obtiene el máximo peso; la menor, el mínimo; y el resto de frecuencias intermedias se reparten equitativamente el peso entre el máximo y mínimo, como en Last.fm.
- El peso final para cada par usuario-ítem es el correspondiente al de su frecuencia asociada.

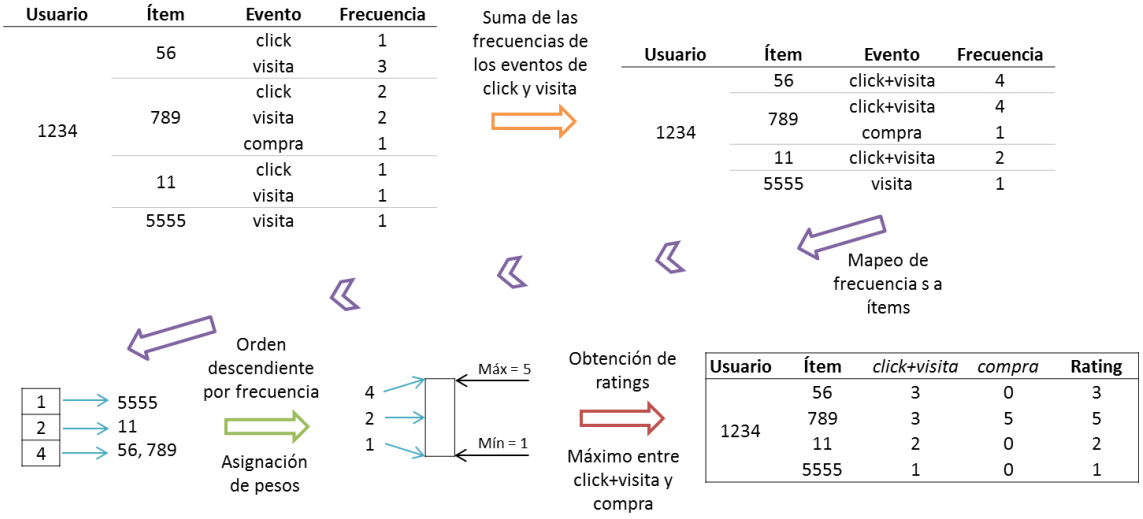

Figura 9. Proceso de conversión de eventos a puntuaciones para Blueknow.

De esta forma, para cada par usuario-ítem se obtiene un rating correspondiente a sus frecuencias y a los eventos que relacionan a ambos.

Una vez se tienen los datos en ratings para todos los conjuntos de datos, el siguiente paso consiste en realizar la partición de entrenamiento y test para poder realizar las pruebas de los algoritmos de recomendación sobre los conjuntos de datos. Lo describimos en la siguiente sección.

4.4 Partición entrenamiento-test

Los ratings de los conjuntos de datos se dividen en dos particiones independientes: entrenamiento y test. Como se indicó anteriormente en la figura 5, los datos de entrenamiento se le dan al recomendador como input para su algoritmo, en base al cual genera las recomendaciones; y los datos de test, ocultos al recomendador, se usan como *ground truth* para contrastar en qué medida las recomendaciones son consistentes con los ratings de test ocultados al algoritmo (en otras palabras, éstos se usan para calcular las métricas sobre el *output* de los recomendadores).

Partiendo de que se dispone de ratings usuario-ítem para todos los conjuntos de datos, la partición que se ha realizado es la misma en casi todos los casos. A partir de todos los ratings, se realiza una partición 80/20 de forma aleatoria, de forma que el 80% de los ratings se destina a entrenamiento y el 20% restante a test.

En Blueknow y MovieLens 100K se aplican no obstante particiones distintas, que describimos a continuación.

Smart MovieLens 100K

Para este conjunto se ha realizado una segunda partición, además de la estándar, específicamente destinada a evaluar el algoritmo HSVD. Éste es un algoritmo orientado a la recomendación a usuarios nuevos en el sistema, es decir, usuarios con pocos datos. Para poner a prueba esta capacidad específica del algoritmo, interesa disponer o simular un conjunto de tales usuarios nuevos, y centrar la evaluación en los mismos. Para ello, reproduciendo la metodología descrita en la propuesta original de HSVD (Pu 2013), formamos la partición seleccionando un conjunto de usuarios, que llamaremos “nuevos”, para los que separamos pocos datos de entrenamiento dejando el resto como test. Los datos del resto de usuarios, que llamaremos “viejos” se dejan enteramente para entrenamiento.

En concreto, hemos materializado esta partición seleccionando aleatoriamente un conjunto de usuarios nuevos y poniendo únicamente 5 de sus ratings en el entrenamiento y el resto en el test.

Blueknow

Al tener este conjunto de datos muchos usuarios con pocas interacciones, la partición aleatoria no funciona bien, ya que los usuarios de test no tienen apenas entrenamiento y no se puede realizar una buena recomendación debido al escaso input para los algoritmos.

Por esta razón, se optó por realizar una partición especial, seleccionando primero los usuarios con mayor actividad, eligiendo de éstos un porcentaje de test (que se ha fijado en un 20%) de forma aleatoria, y dejando los ratings del resto de usuarios enteramente para entrenamiento.

De esta forma, se recomienda a los usuarios seleccionados, sabiendo que van a tener suficiente entrenamiento y test, y el resto sirve únicamente para entrenar al sistema relacionando los ítems o usuarios entre sí.

Una vez se dispone de las particiones de entrenamiento y test, se ejecutan los algoritmos a evaluar y se obtienen los ficheros de recomendación de cada uno de ellos. Finalmente, se mide y compara la efectividad de las recomendaciones realizadas en cada caso mediante las métricas de evaluación que a continuación se explican.

4.5 Métricas de evaluación

La efectividad de una recomendación se ha medido tradicionalmente mediante la métrica RMSE (*Root-Mean Square Error*), que mide la diferencia entre los valores predichos por el algoritmo y los que actualmente se han observado. Esta métrica sólo es aplicable a aquellos algoritmos que realizan una estimación de rating, y en este trabajo muy pocos algoritmos cumplen con ese requisito. Por ello, se ha considerado otro enfoque a la hora de comparar los recomendadores: a través de métricas orientadas a calidad del ránking en base a relevancia, novedad y diversidad.

Todas estas métricas comparten una misma variante, con notación *M@n*, de forma que la métrica *M* se calcula para los *n* primeros ítems recomendados a cada usuario. A lo largo del documento se hará referencia a *n* como *cutoff*.

4.4.1 Relevancia

Este tipo de métricas tienen como objetivo medir el grado en que se satisfacen los gustos del usuario objetivo al realizar una recomendación. Hay que definir para cada conjunto de datos qué ítems se consideran relevantes (i.e. indican una opinión favorable del usuario hacia el ítem al que asigna el rating), y para ello se establece un *threshold* o umbral, a partir del cual el rating se considera positivo. Cada conjunto de datos puede tener un umbral diferente, dependiendo de la interpretación que proceda realizar sobre los datos. Así en MovieLens y Netflix, siguiendo la semántica sugerida por la aplicación[8], establecemos un umbral de cuatro, a fin de asegurar que una película puntuada con ese valor o más gustó al usuario, es decir, en terminología de Recuperación de Información, es relevante. Sin embargo, para el resto de conjuntos de datos en los que se trabaja con frecuencias el umbral es cero, considerando, para simplificar, que si un usuario ha interaccionado con un ítem es porque le ha resultado interesante y, por tanto, relevante a su vez.

A este grupo pertenecen las siguientes métricas, adaptadas a este contexto del campo de la Recuperación de Información:

- **Precisión**: fracción de ítems recomendados que han resultado ser relevantes para el usuario (Baeza 2011). Cuanto mayor sea su valor, mejor será el recomendador según esta métrica. Su valor equivale al porcentaje de ítems relevantes recomendados.

$$P = \frac{\{i \mid i \in R, rel(i)\}}{|R|}$$

- **nDCG (*Discounted Cumulative Gain*):** la relevancia de los ítems teniendo en cuenta su posición en el ránking, de forma que puede valorar más un ítem poco relevante en las primeras posiciones que uno con mayor relevancia que esté más abajo en el ránking. La métrica además admite la distinción de diferentes grados de relevancia (a diferencia de precisión, que considera relevancia binaria) (Jarvelin 2000).

[8] Ver por ejemplo http://www.movielens.org/images/star-legend.gif.

$$nDCG = \frac{DCG}{IDCG}$$

$$IDCG = \max DCG(R)$$

$$DCG = \sum_{i_k \in R} \frac{g(i_k)}{log_2(k+1)}$$

$$g(i) = \begin{cases} NONE = r(u,i) \\ LIN = r(u,i) - \delta + 1 \\ EXP = 2^{r(u,i)-\delta+1} - 1 \\ BIN = 1 \\ LOG = \log(r(u,i) - \delta + E) \end{cases}$$

Donde δ es el threshold, g es el grado de relevancia e IDCG es el valor DCG del mejor ránking posible. DCG suma grados de relevancia penalizados por lo tarde que aparezcan en el ránking, y la normalización por IDCG permite neutralizar la diferencia entre usuarios más fáciles o difíciles de satisfacer (es decir evitar que la métrica dependa del número grande o pequeño de ítems que gustan al usuario). Así pues, un valor alto de nDCG indica que hay ítems relevantes en las primeras posiciones. El grado de relevancia se obtiene de los ratings de test, mediante un mapeo de valores que se presta a diferentes variantes, tales como ninguno, lineal, exponencial, binario y logarítmico.

- **Cobertura:** ratio de usuarios a los que el algoritmo es capaz de realizar una recomendación (Baeza 2011). En los casos normales, esta métrica estará siempre en 1 o casi (i.e. cobertura total), pero conviene monitorizar esta métrica ya que ciertos algoritmos presentan deficiencias de cobertura, pudiendo distorsionar el valor de otras métricas (p.e. si sólo se recomienda a los usuarios más "fáciles", puede salir una precisión muy alta que no es la que realmente le corresponde al algoritmo en global).

$$Cobertura = \frac{\{u \in \mathcal{U} \mid R(u) \neq \emptyset\}}{|\mathcal{U}|}$$

- **RMSE (*Root-Mean Square Error*):** anteriormente explicado, es la diferencia entre los valores predichos por el recomendador y los ratings reales de los usuarios a los ítems correspondientes. Esta métrica sólo se ha aplicado en los casos en los que se realizaba predicción de rating (Adomavicius 2005).

$$RMSE = \sqrt{\frac{1}{|test|} \sum_{(u,i) \in test} \left(\hat{r}(u,i) - r(u,i)\right)^2}$$

Las métricas de relevancia son importantes debido a que el objetivo principal es que al usuario le gusten los ítems que se le recomiendan. Sin embargo, es interesante para éste que no sólo se acierten sus gustos: por ejemplo, si a un usuario le gustan mucho las películas de acción, se le recomendará más películas de acción que no haya visto. Sin embargo, lo que se está consiguiendo así es encasillar al usuario, mostrándole siempre películas del mismo género. Para que esto no suceda, se motiva la evaluación de otras dimensiones de calidad además de la relevancia: novedad y diversidad. Éstas métricas permiten comparar los diferentes algoritmos para ver en qué medida hacen que el usuario pueda ampliar sus gustos y añadir variedad a sus experiencia.

4.4.2 Novedad

Esta familia de métricas valora la cantidad de novedad que se le aporta a un usuario objetivo en la recomendación que se le realiza. En otras palabras, se mide qué tan diferente son los ítems recomendados frente a los que han sido observados por el usuario anteriormente.

En este caso, las métricas utilizadas son las siguientes:

- **EPC (*Expected Popularity Complement*):** Representa la probabilidad promedio de que un usuario aleatorio no haya puntuado los ítems recomendados (Vargas 2011).

$$EPC = \frac{1}{|R|}\sum_{i \in R}\big(1 - p(known|i)\big)$$

$$p(known|i) \sim \frac{|U_i|}{|\mathcal{U}|} = \frac{\{u \in \mathcal{U} \mid r(u,i) \neq \emptyset\}}{|\mathcal{U}|}$$

- **EFN (*Expected Free Novelty*):** Es muy parecida a la anterior métrica, pero se diferencia en dos puntos clave. Por un lado, se aplica un logaritmo para crear más diferencia entre los valores; y por el otro, la probabilidad no se divide entre el número de usuarios sino entre el número de ratings. La suma de todas las probabilidades de todos los ítems debe ser uno (Vargas 2011).

$$EFN = -\frac{1}{|R|}\sum_{i \in R}\log p(i)$$

$$p(i) = \frac{|U_i|}{|\mathbf{r}|} = \frac{\{u \in \mathcal{U} \mid r(u,i) \neq \emptyset\}}{\{(u,j) \in \mathcal{U} \times \mathcal{I} \mid r(u,j) \neq \emptyset\}}$$

- **EPD (*Expected Profile Distance*):** Se basa en el cálculo de la distancia media entre los ítems recomendados y los puntuados por el usuario, de forma que su cercanía o lejanía son indicadores de novedad para dicho usuario. La similitud entre dos ítems se calcula normalmente con un enfoque basado en contenido, es decir, teniendo en cuenta los géneros de las películas, por ejemplo (Vargas 2011).

$$EPD = \frac{1}{|\mathcal{U}||R|}\sum_{u \in \mathcal{U}}\sum_{i \in R} d(i,u)$$

$$d(i,u) = \frac{1}{|\mathbf{u}|}\sum_{j : r(u,j) \neq \emptyset} d(i,j)$$

$$d(i,j) = 1 - sim(i,j)$$

Estas métricas se basan únicamente en el aporte de novedad intrínseco a los ítems del ránking. Existen versiones que tienen en cuenta la posición en el ránking y la relevancia de los ítems (Ricci 2011), pero para este trabajo optamos por esta versión más sencilla y estándar.

4.4.3 Diversidad

Finalmente, este conjunto de métricas tiene como objeto de valoración que la recomendación que se realice al usuario sea variada, es decir, busca que los ítems recomendados sean diferentes entre sí.

En el caso anterior de EPD hablamos de similitud entre ítems según su contenido, y esto mismo se va a tratar aún con más importancia con los algoritmos de diversidad. Los ítems se van a agrupar en *subtopics*, denominados de múltiples formas dependiendo el contexto y que en este caso el término que más se adecúa es "aspectos" que los ítems tienen en común (como los géneros de las películas). Las métricas y conceptos de diversidad que aplicaremos se basan en las propuestas documentadas en (Clarke 2008, Vargas 2011). Resumimos a continuación las métricas que vamos a utilizar pertenecientes a este grupo:

- **EILD (*Expected Intra-List Dissimilarity*):** De la misma forma que en el caso de EPD se observa la distancia entre los ítems observados por el usuario y los recomendados, en este caso se realiza el mismo cálculo pero entre los propios ítems recomendados, unos con otros. No se tiene en cuenta ni la relevancia de los ítems del ránking ni su posición en el mismo, y el sumatorio se realiza de forma triangular, haciendo que la constante quede multiplicada por dos y ahorrando el cálculo de distancias repetidas (Zhang 2008).

$$EILD = \frac{2}{|R|(|R|-1)} \sum_{\substack{i,j \in R \\ i<j}} d(i,j)$$

- **ERR-IA (*Intent-Aware Expected Reciprocal Rank*):** Utiliza la posición de los ítems recomendados y su grado de relevancia para comprobar que aquellos más relevantes estén en las posiciones más altas del ránking, pero al mismo tiempo penalizando la repetición (redundancia) de subtopics que ya han aparecido en posiciones anteriores a cada ítem del ránking. Esto se realiza para cada uno de los aspectos de forma independiente, realizando una media final (Chapelle 2011).

$$ERR\text{-}IA = \sum_{z \in \mathcal{Z}} ERR(z)\, p(z|q)$$

$$ERR(z) = \sum_{i_k \in R} \frac{1}{k} p(r|i_k) \prod_{j<k} (1 - p(rel|i_j))$$

$$p(r|i) \sim \frac{2^{g(i)} - 1}{2^{g_{max}}}$$

  Y $p(z|q)$ se suele tomar uniforme $p(z|q) \sim 1/|\mathcal{Z}|$ (siendo $\mathcal{Z}$ el conjunto de todos los aspectos posibles) a falta de información particular sobre qué aspectos son más o menos probables para un usuario.

- **α-nDCG:** Es una forma muy parecida de calcular la métrica anterior, pero añade una penalización por posición logarítmica y un factor de peso $\alpha \in [0,1]$ que controla la tolerancia a la redundancia, cuyo valor suele ser 0'5.

$$\alpha\text{-}nDCG = \frac{1}{\alpha\text{-}IDCG} \sum_{z \in \mathcal{Z}} \sum_{i_k \in R} \frac{r(i_k)}{\log_2(k+1)} \prod_{j<k} \left(1 - \alpha\, r(i_j)\right)$$

  $\alpha\text{-}IDCG$ representa el máximo valor posible de $\alpha\text{-}DCG$ que se obtendría con un ránking ideal, y se divide por tanto para normalizar. Se utiliza un algoritmo avaro para estimar este ránking ideal en el que se busca tener en el top del ránking los ítems más relevantes, pero de diferentes aspectos, ya que un cálculo exacto sería un problema NP (Clarke 2008).

- **S-Recall (*Subtopic Recall*):** Representa el porcentaje de aspectos cubiertos por al menos uno de los ítems del ránking. Sólo los ítems relevantes son tenidos en cuenta en este caso (Zhai 2003).

$$S\text{-}Recall = \frac{1}{|\mathcal{Z}|}|\{z \in subtopics(i) | i \in R, rel(i,u)\}| = \frac{1}{|\mathcal{Z}|}\left|\bigcup_{\substack{i \in R \\ rel(i,u)}} subtopics(i)\right|$$

- **S-Recall-NOR:** Una variante de la métrica anterior, en la que todos los ítems, tanto relevantes como no relevantes, son contabilizados (Zhai 2003).

$$S\text{-}Recall\text{-}NOR = \frac{1}{|\mathcal{Z}|}|\{z \in subtopics(i) | i \in R\}| = \frac{1}{|\mathcal{Z}|}\left|\bigcup_{i \in R} subtopics(i)\right|$$

Una vez establecidas las diferentes métricas con las que se van a evaluar los algoritmos, antes de describir las pruebas realizadas y los resultados obtenidos explicamos en el siguiente capítulo los aspectos más técnicos del desarrollo del proyecto.

5. Ingeniería y desarrollo

En este capítulo se describen los aspectos técnicos del proyecto en lo que se refiere al desarrollo del software. El presente trabajo involucra el estudio de algoritmia avanzada, la integración de librerías externas, el tratamiento de la escalabilidad, y la ejecución de una amplia batería de prueba, así como el tratamiento de los datos resultantes.

A continuación se detallan: la implementación realizada, las tecnologías usadas y las soluciones orientadas a la escalabilidad y eficiencia que fueron necesarias para el trabajo con los conjuntos de datos más grandes.

5.1 Metodología

A nivel general, ha cobrado especial importancia en el desarrollo técnico de este trabajo la orientación a objetos, la separación por módulos y el uso de clases e interfaces para su desarrollo, a fin de maximizar la facilidad de extensión de código para integrar nuevos algoritmos y variaciones de sus componentes, así como la inclusión de librerías externas en el proyecto allí donde se requiere su uso. La mayoría de las librerías con las que se ha trabajado están disponibles en Java, por lo que, el uso de este lenguaje en el proyecto facilita la adaptación y modificación del código de estas librerías acorde a las necesidades y objetivos del proyecto.

En cuanto a la metodología seguida, en toda implementación de nuevos algoritmos y la variación de otros ya existentes se ha seguido el esquema clásico de desarrollo en cascada, aceptando iteraciones con la fase anterior inmediata:

- **Análisis de requisitos**. Se realiza un análisis teórico del algoritmo a implementar mediante el estudio en detalle de la documentación (en nuestro caso generalmente artículos científicos) donde está expuesto, así como la búsqueda de información y aclaraciones extra en el caso en que sea necesario.
- **Implementación**. Una vez comprendido el algoritmo a implementar, se realiza la correspondiente conversión a código a partir del modelo teórico anteriormente estudiado, siguiendo la estructura modular del proyecto.
- **Validación**. Para realizar la validación de la implementación del algoritmo, se realizan pruebas con un conjunto de datos pequeño. De esta forma, su ejecución es muy rápida y se puede depurar el código en ciclos mucho más cortos.
- **Evaluación y comparativas**. Aunque no forme parte del esquema clásico, es necesario añadir esta fase extra propia de este tipo de estudio. Una vez validado el algoritmo, se evalúa sobre conjuntos de datos reales y se realiza una comparación con los resultados obtenidos para otros algoritmos.
- **Optimización**. Cuando el algoritmo implica altos costes y/o se aplica en conjuntos de datos de mayor escala, es necesario realizar una fase de optimización de memoria y/o de cómputo según se requiera. Si se da el caso, se revisa la implementación en busca de mejoras, se realizan pruebas con conjuntos pequeños de datos para revalidar el algoritmo, y finalmente se vuelve a probar con el conjunto de datos problemático.

El diagrama correspondiente al esquema de desarrollo seguido con estas fases se muestra en la figura 10.

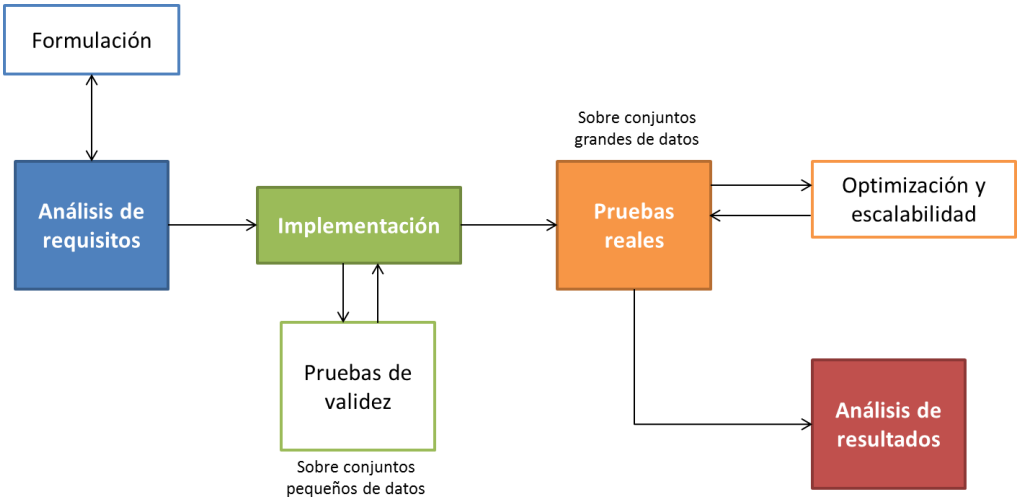

Figura 10. Esquema de desarrollo seguido en el presente trabajo.

5.2 Implementación

La amplitud del presente estudio ha aconsejado la organización del software en varios paquetes con diferentes objetivos, que conceptualmente se subdividen en módulos coincidentes con proyectos NetBeans definidos.

Gran parte del desarrollo se basa en un proyecto principal que se encarga de realizar las recomendaciones y obtener los resultados de las métricas correspondientes a las mismas. Además, este proyecto incluye clases para la realización de barridos de parámetros (variación de dos parámetros en un algoritmo con el objetivo de observar la tendencia que éstos causan en los resultados), partición de datos, compresión y descompresión de archivos, etc. La figura 11 muestra la jerarquía de los tipos de algoritmos.

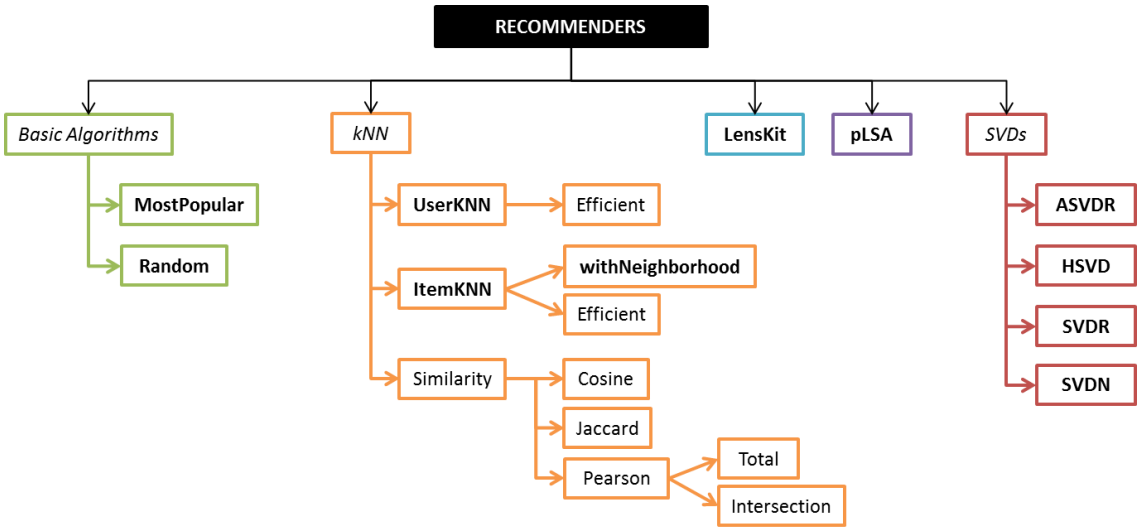

Figura 11. Jerarquía de los tipos de algoritmos.

Para el tratamiento de los conjuntos de datos, en el caso de MovieLens y Netflix ha bastado con crear dos clases en el proyecto principal que se encargan de realizar una partición sobre sus datos con un ratio dado (que se ha fijado a 80/20% en nuestros experimentos). Sin embargo, para los conjuntos cuyos datos se encuentran en forma de frecuencias o eventos (Blueknow, Twitter y Last.fm), es necesaria la conversión a ratings para obtener las particiones como son requeridas.

Como se ha mencionado en capítulos anteriores, el trabajo realizado ha conllevado el uso e integración de librerías externas, que se explicarán con detalle en el primer apartado de la sección 6.3, y cuyo uso ha supuesto, en algún caso, la separación en un módulo aparte para adaptar los algoritmos de la librería a la estructura utilizada (es el caso de LensKit, que además he requerido una clase en el proyecto principal a modo de *Object Adapter* para su uso).

En cuanto a la estructura interna y la codificación, es necesario apuntar que para la implementación de nuevos algoritmos o variaciones de otros ya existentes se ha seguido una estructura basada en el diseño de las clases e interfaces de la librería Mahout, descrita en la figura 12. El proceso sigue los siguientes pasos:

- Primero se calculan las similitudes mediante un objeto SimilarityCalculator y se obtiene el fichero de *similarities* (en caso de que el algoritmo lo requiera).
- Una clase ExampleRecommenderGenerator utiliza el fichero de similitudes, el training y el test para llamar a la función de *getRecommender()*. Ésta crea un objeto ExampleRecommender que implementa Recommender de Mahout.
- Esta misma clase, ExampleRecommenderGenerator, realiza la función de *generateRecommendations()*, que genera recomendaciones a cada usuario de test, obteniendo la similitud entre éste y cada ítem mediante el método de *estimatePreference(u,i)* del objeto ExampleRecommender.
- Se obtiene, entonces, un ránking para cada usuario de test y se genera el fichero de *rankings*, que serán las recomendaciones del algoritmo que posteriormente se someterán al estudio de las métricas.

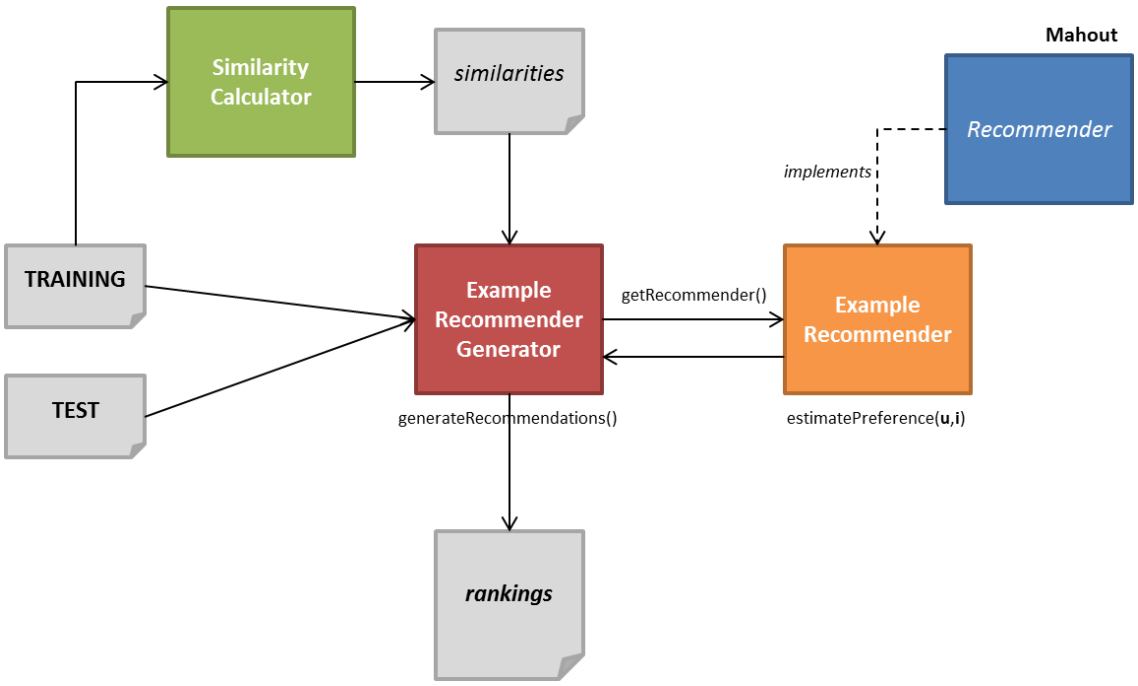

Figura 12. Diseño y dependencias de clases e interfaces que sigue Mahout.

5.3 Tecnologías utilizadas

Para este proyecto se han utilizado varios recursos que han facilitado en gran medida el desarrollo del mismo, tanto en la ejecución de algoritmos (uso de librerías externas) como en el tratamiento de los resultados obtenidos (uso de bases de datos y tablas dinámicas). A continuación se detallan las tecnologías que han tenido mayor relevancia en el desarrollo del trabajo.

5.3.1 Librerías externas

Junto con el software desarrollado expresamente para este trabajo, se han utilizado diversas librerías software externas, tanto para cubrir tareas que estaban fuera del ámbito de la investigación (como operaciones matriciales) como para realizar una comparación entre los algoritmos propios con las implementaciones de los mismos en dichas librerías.

5.3.1.1 Librerías de sistema de recomendación

Con el objetivo de observar la competitividad de los algoritmos de recomendación implementados con los mismos en otras librerías, se seleccionaron para realizar dicha comparación aquellas con las mejores implementaciones disponibles a día de hoy en el dominio público:

- **MyMediaLite**[9]: Librería *open source* en C#. Implementa varios algoritmos de filtrado colaborativo, como kNN (ítem y usuario), algoritmos no personalizados (Random, Popularidad) o de factorización de matrices (WRMF y BPRMF, entre otros).
- **LensKit**[10]: Librería *open source* realizada por el grupo de investigación de GroupLens. Está escrita en Java y actualmente se encuentra en desarrollo. Implementa pocos algoritmos hasta el momento, todos ellos de filtrado colaborativo, entre los que se encuentran los kNN (ítem y usuario), SlopeOne y FunkSVD, este último de factorización de matrices.
- **IRG projects**[11] **(*Information Retrieval Group*)**: he realizado este trabajo con el grupo de investigación IRG de la Universidad Autónoma de Madrid, que me ha permitido utilizar las librerías e implementaciones de los algoritmos de los que disponen actualmente con el objetivo de realizar pruebas sobre ellas y añadir algoritmos nuevos a la librería. Este software del grupo está codificado en Java y el proyecto principal de este trabajo ha mantenido la estructura en la que se basan estas librerías.

Se han evaluado otras librerías tales como Apache Mahout[12] y Graphlab[13], pero éstas no se incluyen en el estudio debido a sus limitaciones. Por un lado, las implementaciones de los algoritmos de recomendación de Mahout son muy primitivas algorítmicamente y rígidas en su diseño, de forma que es complicado variar sus parámetros. Por otro lado, los algoritmos de FC de los que dispone la librería de Graphlab no sirven para realizar el estudio, ya que no implementan los algoritmos más comunes en el campo de los sistemas de recomendación.

[9] http://www.mymedialite.net/
[10] http://lenskit.grouplens.org/
[11] http://ir.ii.uam.es/
[12] https://mahout.apache.org/
[13] http://graphlab.org/

5.3.1.2 Librerías de operaciones matriciales

El tamaño de los conjuntos de datos utilizados en este trabajo plantean retos de escalabilidad para la implementación del algoritmo de HSVD anteriormente nombrado, debido a las costosas operaciones matriciales que este algoritmo realiza, tanto las estándar (producto, inverso) como de descomposición (QR, Singular Value). Para este fin, se han probado y evaluado diversas librerías *open source* en Java que cumplen con los requisitos establecidos (no sólo se requieren para realizar operaciones matriciales, sino que también éstas se realicen de forma eficiente en cuanto al consumo de tiempo y memoria).

- **EJML**[14] **(*Efficient Java Matrix Library*)**: Se encuentra actualmente en desarrollo, ya que se está realizando la versión *multi-thread* de la librería. Tiene como objetivo realizar operaciones con matrices de gran tamaño y que además sea eficiente en cuanto al cómputo y la memoria.
- **Colt**[15]: Librería científica muy amplia que incluye tanto matrices multidimensionales como álgebra lineal o histogramas. Es sencilla de usar y nuevamente dispone de todas las operaciones que se necesitan para la tarea a realizar.
- **Jama**[16] **(*Java Matrix Package*)**: Es un paquete Java de álgebra lineal básico. Es muy sencillo de utilizar y sólo dispone de seis clases que permiten manipular, operar y descomponer matrices.
- **Apache Math Commons**[17]: Se trata de una librería muy conocida y utilizada. Dispone de operaciones con matrices y vectores, tanto básicas (suma, producto, etc.) como de descomposición.

De estas cuatro librerías, la que mejor resultado ha dado en cuanto a los requisitos establecidos (eficiencia, sencillez, documentación, etc.) fue EJML, por lo que ha sido seleccionada para la implementación del algoritmo de HSVD.

5.3.1.3 Integración de librerías

Debido a la variedad de librerías incluidas en este estudio, se ha establecido una estructura común para que el cálculo de las métricas de sus algoritmos se realice de la misma forma para todas. Esto se ha realizado mediante el encapsulamiento de los recomendadores, de forma que usando el conjunto de datos de entrenamiento y los usuarios a recomendar, se genera un ránking con el formato de fichero estándar. Una vez se obtienen los ficheros de recomendación, se pueden someter a las métricas deseadas, sin importar el origen de dicho algoritmo.

A continuación, se explica más concretamente cómo se han adaptado las librerías al proyecto principal del estudio y cómo éstas han sido utilizadas.

My Media Lite

Esta librería está implementada en C#, por lo que su uso ha sido externo al proyecto (sólo su ejecución, ya que como se ha dicho antes, las métricas y los datos son iguales para todos), realizando una integración a nivel de formato de datos de entrada y salida (i.e. formato de los ficheros de input y output).

[14] https://code.google.com/p/efficient-java-matrix-library/
[15] http://acs.lbl.gov/software/colt/
[16] http://math.nist.gov/javanumerics/jama/
[17] http://commons.apache.org/proper/commons-math/

Para realizar una buena comparación, se ha buscado establecer los parámetros más parecidos a los de los algoritmos propios, de forma que la diferencia resida en el propio algoritmo o la forma de implementarlo y no en la variación de algún parámetro común.

LensKit

Se ha creado un proyecto aparte, de forma que tenga diferentes clases que implementasen los recomendadores utilizando las funciones de esta librería (figura 13).

Asimismo, en el proyecto principal se ha creado una clase, LenskitRecommender, que implementa el Recommender de Mahout como se expuso antes. Esta clase es un *Object Adapter* del proyecto donde se implementan los algoritmos de LensKit.

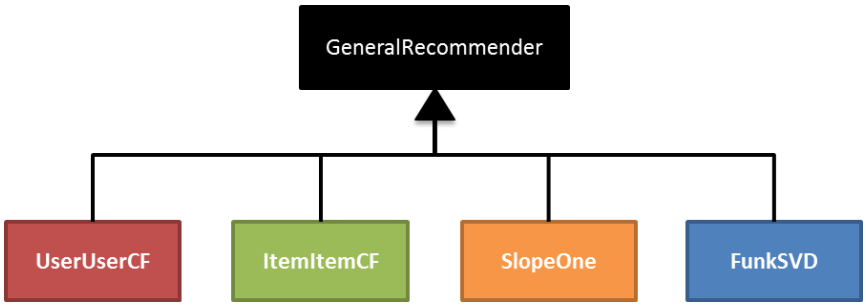

Figura 13. Diagrama UML de las clases del módulo de LensKit.

Librerías matriciales

Para probar y seleccionar una librería entras las citadas en el apartado anterior (EJML, Colt, Jama y Apache), se ha creado para cada una de ellas una clase que implementa el algoritmo HSVD utilizando sus propias funciones, y en las que se controla el tiempo de ejecución en cada operación costosa a realizar para comparar la eficiencia entre ellas.

Como se ha indicado antes, se ha seleccionado EJML, la cual ha sido muy fácil de incluir en el desarrollo. Puesto que su API tiene un uso sencillo e intuitivo, se añadió al proyecto y se utilizó directamente en los algoritmos, sin ser necesaria ninguna clase intermedia. Además, las pequeñas modificaciones de eficiencia realizadas sobre esta librería han sido introducidas directamente en su código fuente y siguiendo su estructura, ya que ésta es clara.

5.3.2 Bases de datos

Una de las características principales de este trabajo es la gran cantidad de resultados producidos a raíz de la realización de recomendaciones de todos los algoritmos, variantes y combinaciones de parámetros para cada uno de los conjuntos de datos seleccionados, y diversidad de métricas, dando lugar a una amplia combinatoria de ejecución de pruebas, mediciones y comparativas. Por esta razón, se ha optado por el almacenamiento de todos los resultados en tablas de una base de datos, la cual se ha estructurado como se indica en la figura 14 a continuación.

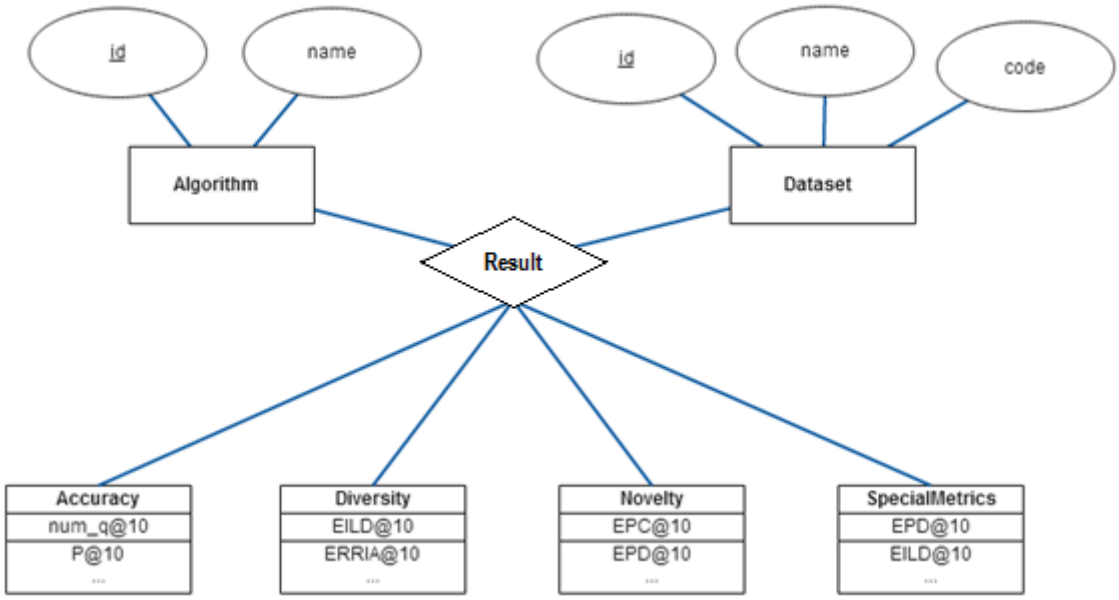

Figura 14. Diagrama UML de la base de datos utilizada.

Asimismo, el objetivo que se persigue en este trabajo es la comparación de los resultados, por lo que el uso de la base de datos permite la creación de tablas dinámicas en Excel, haciendo mucho más sencilla la comparación y el estudio de los resultados.

De esta forma, se ha facilitado en gran medida la comparación de los resultados de las diferentes métricas y ganado organización y disponibilidad de los datos, consiguiendo además un ahorro considerable de espacio de almacenamiento de ficheros de resultados en disco.

5.4 Mejoras de escalabilidad y eficiencia

Tras la implementación de los algoritmos y su correspondiente validación, éstos se utilizan para la recomendación en cada conjunto de datos seleccionado para el estudio. Para todos ellos, se ha realizado una distinción en lo que llamamos cálculo *offline* y cálculo *online*.

El cálculo offline incluye todo aquello que se base en el estudio y procesamiento de los datos de entrenamiento. Estas tareas dependen del algoritmo que se vaya a utilizar, encontrándose tareas como las siguientes:

- Lectura de todos los ratings y obtención de la matriz usuario-ítem (común para todos los algoritmos).
- Cálculo de similitudes entre usuarios o entre ítems, así como del vecindario de cada uno (algoritmos kNN).
- Obtención de preferencias de los ítems, como la popularidad de cada uno (algoritmos no personalizados).
- Cómputo de los factores de usuario e ítem (algoritmos de factorización de matrices).

Por otro lado, el cálculo online tiene como objetivo realizar las recomendaciones partiendo de los datos obtenidos en el cálculo offline. Las operaciones que se realizan en este caso son las siguientes:

- Lectura de las similitudes, vecindarios o matrices (dependiendo del algoritmo).
- A partir de un usuario, buscar el top-N de ítems más relevantes para el mismo siguiendo el algoritmo elegido.

De esta forma, el procesamiento complejo y costoso se realiza previamente y la selección de los ítems más relevantes para el usuario se realiza en el momento en el que se necesita y sólo para aquellos que lo requieren.

Sin embargo, en algunos casos esto no ha sido suficiente y las ejecuciones de algunos algoritmos sobre los conjuntos de datos más grandes resultan demasiado costosas, bien por uso de memoria o por tiempo de cómputo. Por esta razón, ha sido necesaria la optimización de alguno de los algoritmos en busca de la mejora de estos dos factores, y el desarrollo de soluciones específicas para hacer viable la ejecución de los algoritmos.

User kNN

El algoritmo de User kNN para un conjunto de datos como el de Blueknow, formado por gran número de usuarios, presenta un problema de escalabilidad. Este algoritmo crea una matriz triangular de similitud $|\mathcal{U}| \times |\mathcal{U}|$ donde en este conjunto de datos $|\mathcal{U}| = 500.000$, sólo en la estructura para mantener la matriz de similitud (*float*[][]) se necesita casi 1TB de RAM ($|\mathcal{U}|\ x\ |\mathcal{U}|\ x\ 32bits$) . Por ellos, su procesamiento así como su almacenamiento en memoria es muy costoso, volviéndose inviable para esta clase de conjunto de datos.

Para poder realizar las recomendaciones de este algoritmo sobre Blueknow, se transforma la matriz de similitud en una sub-matriz trapezoidal de similitud, en la que se reduce el número de filas al número de usuarios de test. Esto se puede realizar debido a que lo único que se hace es evitar los cálculos innecesarios (sólo se van a estudiar las recomendaciones realizadas a los usuarios de test), sin intervenir en la efectividad del propio algoritmo. La figura 15 muestra de forma gráfica la optimización realizada.

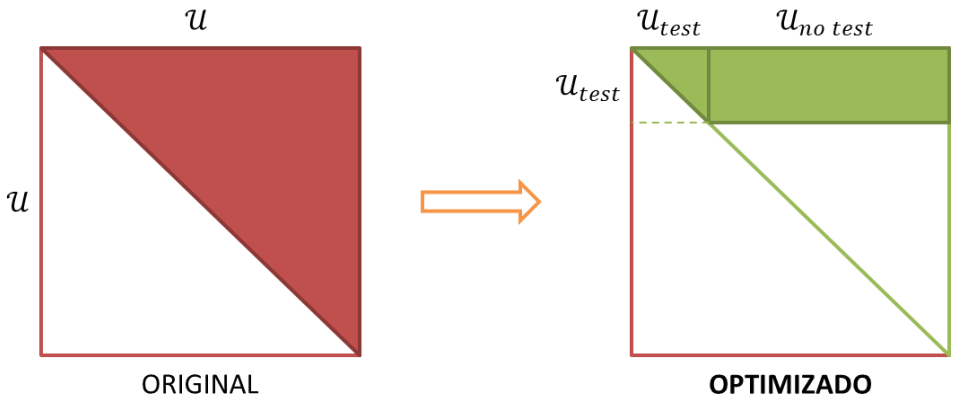

Figura 15. Optimización de la matriz de similitud de usuarios.

De esta forma y sobre todo en los casos en los que se realiza un sub-muestreo de usuarios, se consigue optimizar en gran medida tanto el uso de memoria como el cálculo de las similitudes, ya que sólo se calculan aquellas que se van a requerir en las pruebas de test.

Hay que decir que en el caso del otro algoritmo de kNN, Item kNN, no es posible realizar ninguna optimización de este tipo para conjuntos de datos con muchos ítems, ya que todos ellos pueden ser objeto de recomendación ante algún usuario de test, y evitar ciertos ítems sería dar una ventaja "injusta" a favor del recomendador.

Clases optimizadas

Como se muestra en el diagrama lógico del proyecto principal (figura 11), se han creado dos variaciones extras para los dos algoritmos de la familia de kNN, User e Item (también sobre la versión basada en contenido). Esto ha sido necesario debido a que el tratamiento masivo de datos en conjuntos como Netflix (muchos usuarios) o Twitter (muchos ítems) resulta completamente inviable a menos que se apliquen este tipo de soluciones, debido tanto al tiempo de procesamiento como al uso de memoria.

Para su optimización se han creado dos variaciones basadas en el mismo principio, aplicada a usuarios o ítems según el caso. En ambas versiones, el cuello de botella se encuentra en el cálculo y almacenamiento en memoria de las similitudes.

El algoritmo optimizado seguido mantiene en memoria un *heap* de los *n* elementos más similares para cada elemento (usuario o ítem), donde *n* es un número de vecinos que estrictamente se van a necesitar al final, evitando utilizar memoria para almacenar innecesariamente la similitud con *todos* los elementos. Se mantiene un heap para cada elemento, donde la clave es la similitud de los elementos del heap con el elemento en cuestión. Cada vez que se compara el elemento con un nuevo usuario o ítem, sólo se inserta éste en el heap si su similitud es mayor que el usuario o ítem de la cima (el que tiene la similitud menor de todos los usuarios o ítems del heap).

Sólo se calcula la similitud para cada par de elementos una vez (el bucle consiste en obtener la similitud de cada elemento con los menores a él, considerando "menores" a aquellos elementos con un identificador menor numéricamente), de forma que una vez realizado este cálculo, se introduce su valor en ambos heaps.

Además, a este algoritmo se le puede pasar como argumento la sección triangular a realizar, pudiendo partir la matriz triangular en otras más pequeñas y, así, paralelizar el proceso (figura 16). Una vez se obtiene un heap por cada triángulo calculado, se ha creado una clase para realizar la unión de éstos y obtener un único heap que contiene los *n* elementos más similares.

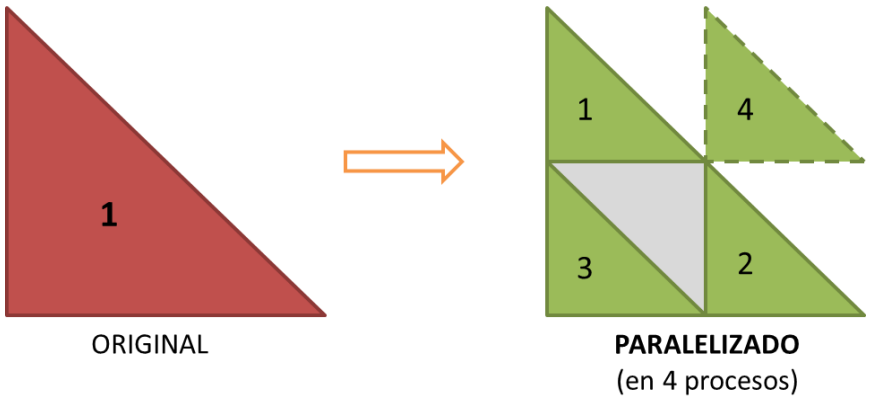

Figura 16. Ejemplo de paralelización de la matriz triangular de similitud.

Así, aunque sigue siendo necesaria la costosa comparación entre todos los elementos entre sí (cosa que no se puede ni se debe evitar), se consigue mejorar en gran medida la eficiencia de estos algoritmos y, por tanto, su nivel de escalabilidad.

6. Resultados experimentales

Tras la realización de la batería de pruebas se ha obtenido una gran cantidad de resultados. Mostramos y analizamos a continuación los más relevantes respecto al objetivo y motivación principal del presente trabajo.

En primer lugar especificaremos la configuración experimental de la que se han obtenido los resultados. A continuación, mostraremos los resultados generales de los conjuntos de datos incluidos en el estudio con los algoritmos de recomendación anteriormente descritos (las versiones óptimas para cada conjunto), agrupados por la librería con la que cada uno se obtiene. Realizaremos además el análisis de los resultados obtenidos en cada caso y de forma global.

Una vez vistos los resultados globales, expondremos las pruebas realizadas sobre los algoritmos configurables, en este caso User e Item kNN, para llegar a su correspondiente óptimo, variando la función de similitud aplicada y los valores de sus parámetros más significativos. Se incluye también un estudio de la correlación entre precisión, tiempo y memoria dependiendo del tamaño del vecindario.

Finalmente, se muestran los resultados obtenidos para aquellos algoritmos de recomendación basados en la descomposición de matrices (*Singular Value Decomposition*) que requieren un conjunto de datos distribuido de una forma específica (distinción entre usuarios "nuevos" a los que recomendar y usuarios "antiguos" que sólo sirven para que el algoritmo se entrene).

6.1 Configuración experimental

Antes de mostrar los resultados de los estudios realizados, es necesario especificar la configuración que se ha utilizado en los mismos, ya que su variación podría implicar resultados diferentes.

En cuanto al entorno de trabajo, las pruebas se han realizado en dos terminales: el ordenador local (pruebas pequeñas y algoritmos de MyMediaLite) y el servidor del laboratorio de investigación (el resto de pruebas). El primero dispone de 32GB de RAM y un procesador AMD i5, mientras que el segundo consta de 512GB de RAM y 64 procesadores de 1.4GHz.

Por otro lado, en cuanto a la configuración de las propias pruebas realizadas hay varios puntos a tener en cuenta, los cuales se detallan a continuación.

Las métricas de novedad y diversidad se han aplicado únicamente a los conjuntos de datos de los que se disponía atributos de ítems: MovieLens 1M, Blueknow y Twitter. Para estos casos se muestra una versión reducida de la tabla, disponiendo de los resultados completos en el Anexo 1.

Se han utilizado espacios de atributos diferentes en cada uno de estos conjuntos de datos para las métricas de diversidad, que se basan en atributos de los ítems (ERR-IA, S-Recall, EILD,…): en MovieLens se han seleccionado los géneros de las películas; para Blueknow, el tipo (libro, puericultura o juguete), rango de edad y marca de los productos, teniendo así tres atributos para cada ítem; y para Twitter, los hashtags de cada uno de los tweets, si tienen.

La librería de LensKit sólo se aplica en el conjunto de datos de MovieLens 1M; debido a su poca precisión y el alto tiempo de ejecución que requiere, se optó por no incluir en el resto de conjuntos. Por otro lado, debido a que MyMediaLite sólo dispone de una versión para Windows, no pudo ejecutarse en el servidor del que se ha dispuesto en este trabajo. Debido a ello no se incluyen aquí resultados para los conjuntos más grandes: MovieLens 10M, Netflix, Twitter y Last.fm, que resultan fuera del alcance de la capacidad de los recursos de cómputo del equipo Windows disponible para el presente trabajo.

6.2 Resultados generales

Las tablas 5 a 11 recogen los resultados globales obtenidos en el estudio. Para cada conjunto de datos se muestra el resultado de las versiones óptimas de cada algoritmo (la configuración se indica entre paréntesis: método y vecindario); y se separan en grupos las filas que corresponden a una u otra librería.

El formato de las tablas es el siguiente: cada fila es un algoritmo (agrupados por la librería a la que pertenecen) y cada columna una métrica (agrupadas por precisión, novedad o diversidad). Los valores se muestran con un gradiente de calor por métrica (i.e. columna), donde los valores con fondo verde son los más altos (el máximo se resalta en letra negrita de color blanco) y los rojos, los más bajos. Los algoritmos (filas) de cada bloque (librerías IRG, MyMediaLite, LensKit) de las tablas están ordenados de forma descendiente por la métrica de precisión (con cutoff de 20).

La métrica de cobertura se ha aplicado en todos los casos, pero en todos los conjuntos de datos (sin excepción) los algoritmos tienen cobertura completa, es decir, que recomiendan a todos los usuarios de test, por lo que se ha optado por no incluir esta columna en las tablas de resultados.

En algunos conjuntos de datos no se disponen de resultados para determinados algoritmos de recomendación, como por ejemplo User kNN en Netflix, o Item kNN por coseno en Twitter. Esto es debido a que estos algoritmos en los casos particulares de los conjuntos de datos han resultado inviables en el tiempo de realización del presente trabajo, incluso aplicando las optimizaciones de cómputo explicadas en el apartado 5.4.

Se incluye además una tabla-resumen (tabla 12) que recoge los resultados de los mejores algoritmos con una de las métricas de relevancia (se ha escogido precisión @20, aunque todas siguen la misma tendencia) para todos los conjuntos de datos, de forma que se dispone de una visión transversal para poder observar y analizar mejor en el apartado 6.2.1 las posibles diferencias de comportamiento de relevancia entre los conjuntos de datos.

	MovieLens 100K									
	Recomendador	Relevancia							Tiempos	
		P@10	P@20	nDCG-n@10	nDCG-n@20	nDCG-l@10	nDCG-l@20	nDCG-e@10	nDCG-e@20	Offline	Online
IRG	ItemKNN (jaccard)	0,1927	0,1494	0,2577	0,2761	0,2400	0,2651	0,2288	0,2573	00:00:01,69	00:00:19,02
	ItemKNN (cosine)	0,2105	0,1617	0,2861	0,3030	0,2689	0,2936	0,2580	0,2869	00:00:12,41	00:00:19,00
	pLSA	0,2295	0,1813	0,3033	0,3292	0,2869	0,3189	0,2764	0,3116	00:00:28,90	00:00:08,55
	UserKNN (jaccard 60)	0,2335	0,1807	0,3194	0,3406	0,3025	0,3310	0,2917	0,3242	00:00:07,86	00:00:07,52
	UserKNN (cosine 60)	**0,2424**	**0,1837**	**0,3278**	**0,3449**	**0,3114**	**0,3356**	**0,3010**	**0,3290**	00:00:08,75	00:00:07,64
MyMediaLite	WRMF	0,2121	0,1659	0,2847	0,3058	0,2735	0,3012	0,2667	0,2981	-	00:00:02,94
	UserKNN	0,2042	0,1573	0,2766	0,2948	0,2655	0,2901	0,2587	0,2869	-	00:00:32,62
	MultiCoreBPRMF	0,1780	0,1443	0,2280	0,2518	0,2161	0,2460	0,2085	0,2420	-	00:00:02,00
	LeastSquareSLIM	0,1761	0,1406	0,2427	0,2629	0,2331	0,2586	0,2273	0,2558	-	00:02:33,60
	BPRSLIM	0,1759	0,1361	0,2377	0,2580	0,2273	0,2531	0,2207	0,2496	-	00:00:11,78
	BPRMF	0,1726	0,1400	0,2309	0,2549	0,2197	0,2494	0,2127	0,2455	-	00:00:04,94
	ItemKNN	0,1597	0,1396	0,1763	0,2082	0,1650	0,2026	0,1581	0,1986	-	00:00:50,16
	MostPopular	0,1162	0,0966	0,1471	0,1621	0,1391	0,1587	0,1342	0,1564	-	00:00:01,59
	WeightedBPRMF	0,1080	0,0925	0,1380	0,1596	0,1295	0,1542	0,1239	0,1503	-	00:00:03,63
	SoftMarginRankingMF	0,1060	0,0948	0,1287	0,1527	0,1202	0,1478	0,1148	0,1443	-	00:00:04,33
	Zero	0,0285	0,0270	0,0273	0,0343	0,0240	0,0320	0,0218	0,0303	-	00:00:01,05
	Random	0,0090	0,0085	0,0097	0,0113	0,0084	0,0104	0,0076	0,0098	-	00:00:01,96

Tabla 5. Métricas de precisión para MovieLens 100K. Se incluyen tanto los resultados de los algoritmos implementados del grupo IRG (parte superior) como los de la librería externa de MyMediaLite (parte inferior). A la derecha se muestran los tiempos de ejecución offline y online para cada uno de estos algoritmos (MyMediaLite no tiene tiempo offline porque la librería realizaba los cálculos y recomendaciones en un único proceso).

	MovieLens 1M							
	Recomendador	Relevancia		Diversidad			Novedad		Tiempo (offline + online)
		P@20	nDCG-e@20	EILD@20	ERR-IA@20	S-Recall@20	EPC@20	EPD@20	
IRG	ItemKNN (jaccard 70)	0,1998	0,2714	0,7058	0,1242	0,3643	0,8331	0,7704	00:02:57,80
	UserKNN (jaccard 60)	0,2080	0,3053	0,7234	0,1400	0,3871	0,8257	0,7798	00:04:38,30
	UserKNN (cosine 60)	0,2095	**0,3104**	0,7274	0,1424	**0,3901**	0,8231	0,7792	00:04:22,46
	ItemKNN (cosine 150)	0,2115	0,2916	0,6871	0,1313	0,3679	0,8512	0,7576	00:04:17,61
	pLSA	**0,2122**	0,2879	0,7014	**0,2162**	0,3745	0,8608	0,7633	00:06:35,67
MyMediaLite	WRMF	0,1826	0,2607	0,7116	0,2062	0,3463	0,8184	0,7683	00:00:25,79
	UserKNN (cosine 80)	0,1815	0,2742	0,7339	0,1262	0,3621	0,8026	0,7824	00:10:38,20
	UserKNN (jaccard 50)	0,1813	0,2725	0,7296	0,1251	0,3605	0,8116	0,7792	00:08:20,00
	ItemKNN (jaccard 190)	0,1743	0,2276	0,7182	0,1880	0,3319	0,8728	0,7573	00:29:04,64
	MultiCoreBPRMF	0,1644	0,2275	0,6806	0,0952	0,3267	0,8792	0,7496	00:00:20,08
	BPRSLIM	0,1616	0,2350	0,7086	0,1807	0,3175	0,8264	0,7723	00:03:15,07
	BPRMF	0,1613	0,2238	0,7109	0,1751	0,3209	0,8390	0,7688	00:00:35,97
	ItemKNN (cosine 80)	0,1561	0,1948	0,6609	0,0835	0,2949	0,9076	0,7414	00:13:15,04
	LeastSquareSLIM	0,1515	0,2200	0,7286	0,1754	0,3224	0,8038	0,7858	01:21:24,07
	MostPopular	0,1093	0,1465	0,7589	0,1002	0,2559	0,7374	0,8333	00:00:05,35
	SoftMarginRankingMF	0,1015	0,1267	0,7408	0,1049	0,2368	0,8672	0,7819	00:00:34,33
	WeightedBPRMF	0,0806	0,1029	0,6288	0,0811	0,1761	0,9511	0,7393	00:00:32,02
	Zero	0,0310	0,0332	0,7266	0,0272	0,1002	0,8963	0,8406	00:00:02,44
	Random	0,0057	0,0052	**0,7810**	0,0045	0,0176	0,9797	0,8304	00:00:05,76
LensKit	UserKNN	0,0219	0,0184	0,7131	0,0046	0,0541	0,9686	0,8372	01:15:46,98
	ItemKNN	0,0213	0,0204	0,6970	0,0049	0,0488	0,9820	0,8251	00:28:29,83
	FunkSVD	0,0092	0,0087	0,6462	0,0033	0,0208	0,9943	**0,8564**	00:07:01,58
	SlopeOne	0,0011	0,0009	0,6196	0,0002	0,0026	**0,9973**	0,8367	00:25:13,33

Tabla 6. Métricas de precisión, diversidad y novedad para MovieLens 1M. Se incluyen los resultados de los algoritmos del grupo IRG (superior), MyMediaLite (central) y LensKit (inferior). A la derecha se muestran los tiempos totales de ejecución de los algoritmos.

	MovieLens 10M									
	Recomendador	Relevancia							Tiempos	
		P@10	P@20	nDCG-n@10	nDCG-n@20	nDCG-l@10	nDCG-l@20	nDCG-e@10	nDCG-e@20	Offline	Online
IRG	UserKNN (cosine 60)	0,2455	0,1843	0,3370	0,3527	0,3188	0,3410	0,3069	0,3327	17:45:40,12	03:15:59,71
IRG	UserKNN (jaccard 60)	0,2387	0,1797	0,3272	0,3434	0,3088	0,3314	0,2968	0,3228	10:44:50,44	04:54:34,96
IRG	pLSA	0,2160	0,1677	0,2828	0,3006	0,2650	0,2892	0,2535	0,2811	01:24:50,82	01:21:24,07
IRG	ItemKNN (cosine)	0,1918	0,1464	0,2590	0,2738	0,2415	0,2623	0,2301	0,2541	03:52:08,62	*08:57:26,00*
IRG	ItemKNN (jaccard)	0,1736	0,1337	0,2333	0,2505	0,2152	0,2378	0,2033	0,2287	00:19:40,57	05:17:29,06
IRG	MostPopular	0,1120	0,0875	0,1479	0,1576	0,1389	0,1525	0,1332	0,1490	00:00:00,00	00:23:38,71
IRG	Random	0,0015	0,0014	0,0015	0,0018	0,0012	0,0016	0,0011	0,0015	00:00:00,00	00:27:38,14

Tabla 7. Métricas de precisión para MovieLens 10M. Se incluyen tanto los resultados de los algoritmos implementados del grupo IRG como el tiempo de ejecución en cada caso.

	Netflix									
	Recomendador	Relevancia							Tiempos	
		P@10	P@20	nDCG-n@10	nDCG-n@20	nDCG-l@10	nDCG-l@20	nDCG-e@10	nDCG-e@20	Offline	Online
IRG	pLSA	0,2310	0,1922	0,2590	0,2593	0,2367	0,2434	0,2234	0,2333	0d 20:14:13,00	1d 11:55:24,12
IRG	Random	0,0018	0,0019	0,0018	0,0019	0,0016	0,0018	0,0015	0,0016	0d 00:00:00,00	0d 06:39:25,45
IRG	MostPopular	0,1020	0,0913	0,1099	0,1150	0,0955	0,1036	0,0870	0,0964	0d 00:00:00,00	0d 14:03:55,32
IRG	ItemKNN (jaccard)	0,1526	0,1273	0,1764	0,1785	0,1586	0,1656	0,1481	0,1576	0d 07:11:33,27	5d 15:02:00,80
IRG	ItemKNN (cosine)	0,1736	0,1416	0,2009	0,1983	0,1826	0,1852	0,1717	0,1771	0d 05:45:21,51	6d 19:24:30,50

Tabla 8. Métricas de precisión para Netflix. Se incluyen tanto los resultados de los algoritmos implementados del grupo IRG como el tiempo de ejecución en cada caso.

	Last.fm									
	Recomendador	Relevancia							Tiempos	
		P@10	P@20	nDCG-n@10	nDCG-n@20	nDCG-l@10	nDCG-l@20	nDCG-e@10	nDCG-e@20	Offline	Online
IRG	ItemKNN (cosine)	0,0779	0,0665	0,0540	0,0517	0,0599	0,0565	0,0404	0,0421	05:28:48,93	02:34:52,72
IRG	ItemKNN (jaccard)	0,0755	0,0639	0,0483	0,0467	0,0545	0,0518	0,0348	0,0368	03:41:28,93	02:26:50,37
IRG	UserKNN (cosine 60)	0,0755	0,0654	0,0514	0,0497	0,0568	0,0543	0,0380	0,0403	00:01:06,84	00:26:10,23
IRG	UserKNN (jaccard 60)	0,0723	0,0625	0,0484	0,0466	0,0537	0,0512	0,0354	0,0375	00:00:35,91	00:28:55,48
IRG	MostPopular	0,0497	0,0421	0,0282	0,0277	0,0329	0,0316	0,0185	0,0201	00:00:00,00	00:17:26,17
IRG	pLSA	0,0040	0,0027	0,0015	0,0013	0,0020	0,0017	0,0008	0,0008	00:09:21,65	00:39:48,56
IRG	Random	0,0005	0,0006	0,0001	0,0002	0,0002	0,0003	0,0000	0,0001	00:00:00,00	00:26:21,36

Tabla 9. Métricas de precisión para Last.fm. Se incluyen tanto los resultados de los algoritmos implementados del grupo IRG como el tiempo de ejecución en cada caso.

	Twitter								
	Recomendador	Relevancia		Diversidad			Novedad		Tiempos	
		P@20	nDCG-e@20	EILD@20*	ERR-IA@20	S-Recall@20	EPC@20	EPD@20*	Offline	Online
IRG	Rocchio (CB)	0,0497	0,2283	0,9154	0,0486	0,0071	0,9999	0,9937	0d 00:20:05,53	0d 00:48:21,55
IRG	ItemKNN (CB)	0,0116	0,0556	0,9287	0,0110	0,0016	0,9999	0,9965	11d 01:47:58,14 (*)	0d 02:38:52,01
IRG	ItemKNN (jaccard)	0,0049	0,0166	0,9430	0,0042	0,0011	0,9999	0,9992	15d 06:03:42,86 (*)	0d 00:17:08,38
IRG	UserKNN (jaccard 60)	0,0018	0,0055	0,9420	0,0021	0,0005	0,9999	0,9992	0d 00:21:37,00	0d 00:09:28,00
IRG	UserKNN (cosine 60)	0,0018	0,0053	0,9420	0,0021	0,0005	0,9999	0,9992	0d 00:21:33,00	0d 00:08:02,00
IRG	MostPopular	0,0007	0,0022	0,9479	0,0001	0,0002	0,9981	0,9995	0d 00:00:00,00	0d 00:05:23,70
IRG	ItemKNN (CB) (norm)	0,0003	0,0011	0,9445	0,0002	0,0001	0,9999	0,9985	11d 01:47:58,14 (*)	0d 02:41:24,23
IRG	pLSA	0,0002	0,0005	0,9460	0,0001	0,0001	0,9998	0,9993	0d 00:35:36,41	0d 00:29:02,99
IRG	Random	0,0001	0,0002	0,9494	0,0000	0,0000	0,9999	0,9994	0d 00:00:00,00	0d 00:04:13,67

Tabla 10. Métricas de precisión, diversidad y novedad para Twitter. Se incluyen tanto los resultados de los algoritmos implementados del grupo IRG como el tiempo de ejecución en cada caso. Los algoritmos con "CB" son basados en contenido, y los que tienen "norm" han sido normalizados. Los tiempos con (*) han sido paralelizados manualmente, por lo que el tiempo de ejecución offline es orientativo.

	Blueknow								
	Recomendador	Relevancia		Diversidad			Novedad		Tiempos	
		P@20	nDCG-e@20	EILD@20	ERR-IA@20	S-Recall@20	EPC@20	EPD@20	Offline	Online
IRG	pLSA	0,0569	0,1406	0,6211	0,0082	0,1474	0,9915	0,6980	00:05:23,54	00:02:21,33
	UserKNN (jaccard 250)	0,0866	0,2328	0,5537	0,0163	0,2087	0,9920	0,6517	00:07:51,00	00:07:10,46
	UserKNN (cosine 400)	0,0871	0,2373	0,5571	0,0169	0,2081	0,9919	0,6547	00:10:14,31	00:08:15,78
	ItemKNN (jaccard)	0,0958	0,2545	0,5494	0,0178	0,2237	0,9934	0,6451	00:01:00,95	00:02:56,72
	ItemKNN (cosine)	**0,0969**	**0,2606**	0,5534	**0,0187**	**0,2261**	0,9936	0,6451	00:02:49,34	00:02:53,32
MyMediaLite	UserKNN (cosine 280)	0,0884	0,2344	0,5520	0,0160	0,2107	0,9921	0,6508	06:53:21,42	00:36:43,31
	BPRSLIM	0,0889	0,2336	0,5852	0,0157	0,2161	0,9918	0,6504	-	00:01:48,12
	UserKNN (jaccard 230)	0,0873	0,2332	0,5482	0,0160	0,2094	0,9921	0,6502	07:13:59,30	00:30:29,97
	ItemKNN (jaccard 10)	0,0902	0,2237	0,5923	0,0149	0,2199	0,9958	0,6460	00:12:20,12	00:02:31,52
	ItemKNN (cosine 10)	0,0883	0,2217	0,5884	0,0149	0,2168	0,9958	0,6468	00:11:40,96	00:02:01,58
	WRMF	0,0480	0,1155	0,5866	0,0075	0,1387	0,9923	0,6699	-	00:00:28,75
	MultiCoreBPRMF	0,0490	0,1157	0,5818	0,0067	0,1199	0,9883	0,6866	-	00:00:18,80
	WeightedBPRMF	0,0430	0,0993	0,5907	0,0060	0,1270	0,9961	0,6722	-	00:00:42,73
	BPRMF	0,0419	0,0970	0,5936	0,0055	0,1236	0,9903	0,6889	-	00:00:45,22
	SoftMarginRankingMF	0,0207	0,0407	0,6532	0,0021	0,0755	0,9944	0,7014	-	00:00:41,59
	MostPopular	0,0138	0,0299	0,5431	0,0018	0,0376	0,9782	0,7784	-	00:00:12,42
	LeastSquareSLIM	0,0051	0,0150	0,7430	0,0010	0,0168	0,9934	0,8096	-	02:01:07,12
	Zero	0,0039	0,0074	0,7435	0,0003	0,0124	0,9935	0,8111	-	00:00:07,30
	Random	0,0011	0,0019	**0,7865**	0,0001	0,0037	**0,9984**	**0,8312**	-	00:00:10,76

Tabla 11. Métricas de precisión, diversidad y novedad para Blueknow. Se incluyen tanto los resultados de los algoritmos implementados del grupo IRG (parte superior) como los de la librería externa de MyMediaLite (parte inferior). A la derecha se muestran los tiempos de ejecución offline y online para cada uno de estos algoritmos. Los algoritmos de MyMediaLite se ejecutan en un solo paso, por lo que no tienen tiempo offline, aunque en este caso se ha modificado este programa para que genere ficheros intermedios de similitud en los algoritmos de kNN.

	Precisión @ 20						
	Recomendador	Conjunto de datos					
		ML 100K	ML 1M	ML 10M	Netflix	Last.fm	Twitter	Blueknow
IRG	UserKNN (cosine)	**0,1837**	0,2095	**0,1843**	-	0,0654	0,0018	0,0871
	UserKNN (jaccard)	0,1807	0,2080	0,1797	-	0,0625	0,0018	0,0866
	ItemKNN (cosine)	0,1617	0,2115	0,1464	0,1416	**0,0665**	-	**0,0969**
	ItemKNN (jaccard)	0,1494	0,1998	0,1337	0,1273	0,0639	**0,0049**	0,0958
	pLSA	0,1813	**0,2122**	0,1677	**0,1922**	0,0027	0,0002	0,0569
	MostPopular	0,0966	0,1093	0,0875	0,0913	0,0421	0,0007	0,0138
	Random	0,0085	0,0057	0,0014	0,0019	0,0006	0,0001	0,0011

Tabla 12. Resultados de precisión (de los primeros 20 ítems recomendados) para los algoritmos de IRG en los conjuntos de datos estudiados (incluyendo Random y Most Popular para visión global). Las casillas que no tienen valor son aquellas en las que el algoritmo no ha podido ejecutarse sobre el conjunto de datos debido a conllevar unos tiempos de ejecución inviables en el plazo de realización del presente trabajo.

6.3 Análisis de resultados

De forma general se observa que, en cuanto a métricas de relevancia, los algoritmos kNN toman la delantera en la mayoría de conjuntos de datos. Sin embargo, aunque tengan buenos índices de precisión estos algoritmos van muy ligados al número de usuarios o ítems presentes en el conjunto, ya que el cálculo de las similitudes puede hacer que se vuelva inviable para ciertos conjuntos de datos (es el caso de User kNN para Netflix, por ejemplo).

Para realizar la comparación entre los resultados obtenidos para los algoritmos en los distintos conjuntos de datos, se procederá de la forma siguiente. En primer lugar, se va a analizar la tabla 12, que aporta una visión transversal sobre una de las métricas de relevancia (en este caso, precisión con cutoff de 20) de los algoritmos que mejor funcionan. A continuación, se van a analizar los resultados de novedad y diversidad para los conjuntos de datos de MovieLens 1M, Twitter y Blueknow; y finalmente se hablará de algunos casos particulares de los resultados de las librerías externas.

En cuanto a los resultados de relevancia resumidos en la tabla 12, se pueden observar dos tendencias principales: el de los conjuntos de MovieLens y Netflix frente al resto.

En los conjuntos de datos de MovieLens y Netflix, los algoritmos que mejores resultados obtienen son User kNN y pLSA. Es posible que esto se deba, por ejemplo, a que aunque una persona tenga un género de películas favorito, vea y le gusten otras que se salgan del mismo, bien porque sea una producción novedosa tecnológicamente, participe un actor preferido, o que la persona se sienta identificada o ligada personalmente a ella. En otras palabras, quizás sea común que a una persona le guste más una película fuera del género preferido pero que haya recibido muy buenas críticas, que una que no ha recibido críticas o que no hayan sido favorables y que sí sea del género favorito.

Sin embargo, en Last.fm, Twitter y Blueknow los algoritmos de Item kNN destacan sobre el resto de algoritmos de recomendación. Quizás en estos ámbitos es más común que cada persona se “encasille” en un mismo conjunto de ítems. En el caso de la música podría ocurrir que a pesar de que las personas escuchen canciones de diferentes géneros, sus favoritas pertenezcan la mayoría al mismo género. Asimismo, podría ser común que en las redes sociales cada usuario explote ciertos temas u opiniones frente a otros que raramente aborde. También es probable que en el ámbito de la compra de productos sea frecuente que si se compra un objeto, se compre también accesorios u otros objetos relacionados con el primero.

Cabe destacar de forma particular que en el conjunto de datos de Twitter los algoritmos basados en contenido obtienen valores de precisión muy altos en comparación con el resto. Se observa pues que en este dominio el “encasillamiento” del que anteriormente se ha hablado no parece ser contraproducente sino por el contrario, efectivo: si se recomiendan tweets cuyo contenido es muy parecido a los que forman el perfil del usuario, es muy probable que éstos también le interesen.

Dejando atrás las métricas de relevancia, en cuanto a la diversidad podemos decir que se observan dos tendencias en sus métricas: la de EILD y la de ERR-IA y S-Recall (estas dos últimas son bastante similares, por lo que era de esperar que siguiesen aproximadamente la misma tendencia).

En la primera destaca el algoritmo Random, que obtiene muy buenos valores siempre debido a que selecciona ítems aleatorios y esta métrica mide la distancia entre los ítems recomendados sin importar su grado de relevancia ni posición en el ránking.

Sin embargo, la segunda tendencia ha resultado ser bastante contraria a la de EILD, destacando los algoritmos de kNN y pLSA en MovieLens 1M y Blueknow, y Rocchio en Twitter. En estas métricas lo que se analiza es que los ítems relevantes del ránking cubran la mayor cantidad de suptopics posibles, teniendo en cuenta en el caso de ERR-IA, además, su posición en el ránking. Así, dependiendo del conjunto de datos unos algoritmos conseguirán mayor diversidad que otros.

Por otro lado, en cuanto a las métricas de novedad en estos conjuntos de datos se obtienen resultados muy parecidos. La métrica de EPC calcula la probabilidad de que un usuario no haya puntuado los ítems recomendados, mientras que EPD calcula la distancia entre los ítems recomendados y los conocidos por el usuario objetivo. Así, destaca el algoritmo Random de la misma forma que lo hizo en el caso de EILD, ya que ésta es en gran medida muy parecida a la métrica de EPD; y destaca negativamente el algoritmo de Most Popular, que es normal que ocurra puesto que EPC es una métrica que busca precisamente la anti-popularidad. En cuanto al resto de algoritmos, pLSA obtiene valores bastante buenos, y los algoritmos de kNN, en MovieLens 1M y Twitter.

Finalmente, llama la atención los resultados de la librería de LensKit en MovieLens 1M. Destaca muy notablemente en cuanto a novedad, pero obtiene valores muy bajos en cuanto a diversidad y precisión, además de que sus tiempos de ejecución son demasiado altos para plantear siquiera su posible puesta a punto en conjuntos de datos más grandes y complejos.

En resumen, dependiendo del tipo de datos con el que se esté tratando, el tipo de evaluación que se pretenda optimizar, y las características del propio conjunto de datos (como la densidad o el número de usuarios e ítems disponibles), puede ser mejor o peor utilizar ciertos algoritmos de recomendación. Por ello, si se busca el mejor algoritmo para utilizar en un determinado conjunto de datos, es recomendable seleccionar un subconjunto que lo represente y probar los principales algoritmos para, posteriormente, ver cuales dan mejores resultados y entender la lógica que está detrás de ello y poder mejorar al máximo la eficiencia de las recomendaciones del sistema.

6.4 Resultados de los algoritmos kNN

De forma más concreta, en los siguientes apartados muestran las pruebas realizadas para establecer la configuración óptima, tanto de User kNN como de Item kNN, que maximiza en cada caso la precisión para los conjuntos de datos estudiados.

En primer lugar, vamos a estudiar las diferentes funciones de similitud (coseno, Jaccard y Pearson en sus dos versiones), así como la utilización de la normalización de las similitudes; y a continuación veremos la sensibilidad de la precisión de estos algoritmos según se modifique el tamaño de vecindario. Además, en este último punto también se estudia la variación en cuanto al consumo de recursos según este mismo parámetro, el vecindario.

6.4.1 Pruebas de similitudes

Los algoritmos de kNN dependen, como ya se ha expuesto anteriormente, de una función de similitud que relaciona usuarios o ítems dependiendo del algoritmo. Las funciones de similitud probadas han sido coseno, Jaccard y Pearson (con sus dos variantes, por intersección y total, descritas en la sección 3.1.1).

Asimismo, para cada una de ellas se ha probado normalizar o no normalizar la similitud obtenida para cada par (nuevamente explicado en la sección 3.1.1). Realizando la normalización se obtiene un ránking cuyos valores están en el rango de los posibles ratings de usuario, por lo que se puede valorar también en términos de RMSE.

Se han realizado para cada una de las posibles combinaciones (función de similitud y normalización) unos pequeños barridos sobre el número de vecinos y el número mínimo de vecinos que tiene que tener un usuario o ítem para realizar la recomendación, obteniendo unos óptimos orientativos que son los que se muestran en las ilustraciones que siguen.

Las pruebas se han realizado sobre los conjuntos de datos de MovieLens 1M y Blueknow, tanto para User kNN como para Item kNN.

6.4.1.1 User kNN

Como se puede observar en las tablas 13 y 14, los resultados son muy parecidos en ambos conjuntos de datos, siendo claramente los mejores las versiones sin normalizar de coseno y Jaccard, quedándose atrás las versiones centradas en la media del usuario (Pearson).

Las versiones que sí utilizan normalización perjudican en gran medida el nivel de acierto del algoritmo, a pesar de que sus valores de RMSE son buenos.

6.4.1.2 Item kNN

Por otro lado, los resultados obtenidos para este otro algoritmo, Item kNN, no difieren mucho de los obtenidos para el algoritmo anterior, como se puede observar en las tablas 15 y 16.

Como ya se ha adelantado, este algoritmo comúnmente se utiliza sin vecindario, es decir, con un vecindario igual al número de ítems total. Sin embargo, en este trabajo también se ha explorado el uso de vecindario de la misma forma que en el caso anterior de User kNN. Por esta razón, para ello se ha realizado un pequeño barrido para cada versión de Item kNN sobre los dos mismos parámetros anteriormente nombrados, igual que en User kNN, y se han obtenido sus óptimos, que son los que se incluyen en las tablas resultantes.

Para ambos conjuntos de datos los resultados vuelven a ser muy similares para cada versión de cada algoritmo, destacando la relevancia de las similitudes de coseno y Jaccard, como en el caso de User kNN. Asimismo, el uso de normalización en las funciones de similitud de nuevo empeora los resultados, obteniendo relevancias incluso menores que las del algoritmo aleatorio (Random).

En cuanto al uso de vecindario, la versión óptima de cada función de similitud es mejor que su versión sin vecindario en el caso de MovieLens 1M, mientras que éstas dan resultados muy similares en Blueknow, por lo que en este caso la diferencia primordial reside en el tiempo de procesamiento de cada una.

Para conjuntos de datos con muchos ítems (Blueknow en este caso), la creación de vecindario implica un tiempo de procesamiento extra muy grande, y el tiempo para realizar las recomendaciones no mejora respecto a la versión sin vecindario. Por el contrario, para conjuntos de datos con un número menor de ítems (p.e. MovieLens 1M), la creación de vecindario no implica gran diferencia en el procesamiento extra, pero sí que resulta mejor respecto a la precisión obtenida.

UserKNN - MovieLens 1M											
Recomendador	Relevancia								Tiempos		RMSE
	P@10	P@20	nDCG-n@10	nDCG-n@20	nDCG-l@10	nDCG-l@20	nDCG-e@10	nDCG-e@20	Offline	Online	
cosine (60/1)	0,2678	0,2095	0,3248	0,3276	0,3054	0,3171	0,2939	0,3104	00:08:32,00	00:04:22,46	-
jaccard (60/1)	0,2650	0,2080	0,3209	0,3242	0,3004	0,3127	0,2882	0,3053	00:05:03,12	00:04:38,30	-
total-pearson (60/14)	0,2154	0,1640	0,2563	0,2488	0,2401	0,2409	0,2307	0,2362	00:08:54,39	00:04:39,60	-
cosine (norm) (40/14)	0,2053	0,1627	0,2504	0,2534	0,2402	0,2493	0,2343	0,2469	00:08:58,89	00:03:47,15	0,9221
jaccard (norm) (40/14)	0,2015	0,1583	0,2439	0,2430	0,2330	0,2382	0,2266	0,2352	00:05:11,75	00:04:03,56	0,9420
total-pearson (norm) (60/14)	0,1636	0,1357	0,1855	0,1934	0,1783	0,1910	0,1742	0,1898	00:08:19,94	00:04:29,04	0,8961
intersec-pearson (400/1)	0,1197	0,1002	0,1348	0,1395	0,1257	0,1352	0,1205	0,1328	00:05:02,44	00:08:49,82	-
intersec-pearson (norm) (200/14)	0,0956	0,0836	0,1007	0,1054	0,0947	0,1029	0,0913	0,1016	00:05:13,15	00:06:33,88	0,9260

Tabla 13. Resultados de precisión en MovieLens 1M de los métodos de similitud para User kNN. En cada caso se ha seleccionado el vecindario que alcanza la precisión más alta. Los métodos "(norm)" están normalizados, y son los únicos que pueden evaluarse por RMSE.

UserKNN - Blueknow											
Recomendador	Relevancia								Tiempos		RMSE
	P@10	P@20	nDCG-n@10	nDCG-n@20	nDCG-l@10	nDCG-l@20	nDCG-e@10	nDCG-e@20	Offline	Online	
cosine (400/1)	0,1210	0,0871	0,2063	0,2390	0,2057	0,2388	0,2053	0,2373	00:10:14,31	00:08:15,78	-
jaccard (250/1)	0,1204	0,0866	0,2024	0,2354	0,2027	0,2361	0,2005	0,2328	00:07:51,00	00:07:10,46	-
total-pearson (380/1)	0,1020	0,0727	0,1753	0,2025	0,1744	0,2020	0,1745	0,2012	00:13:02,65	00:07:03,65	-
cosine (norm) (220/10)	0,1015	0,0792	0,1495	0,1866	0,1487	0,1862	0,1490	0,1853	00:09:54,51	00:07:37,93	0,7634
jaccard (norm) (220/10)	0,0938	0,0778	0,1372	0,1780	0,1369	0,1783	0,1365	0,1762	00:08:02,86	00:07:08,23	0,7642
intersec-pearson (400/1)	0,0876	0,0651	0,1461	0,1749	0,1464	0,1755	0,1446	0,1730	00:10:54,55	00:06:59,53	-
total-pearson (norm) (120/6)	0,0817	0,0665	0,1180	0,1525	0,1176	0,1524	0,1174	0,1512	00:11:28,55	00:05:56,74	0,7760
intersec-pearson (norm) (280/14)	0,0647	0,0526	0,0965	0,1243	0,0962	0,1246	0,0959	0,1230	00:11:04,63	00:06:28,66	0,7650

Tabla 14. Resultados de precisión en Blueknow para los diferentes métodos de similitud del algoritmo User kNN. En cada caso se ha seleccionado el vecindario que alcanza la precisión más alta. Los métodos "(norm)" están normalizados, y éstos son los únicos que pueden evaluarse por RMSE.

	ItemKNN - MovieLens 1M										
	Recomendador	Relevancia								Tiempos		RMSE
		P@10	P@20	nDCG-n@10	nDCG-n@20	nDCG-l@10	nDCG-l@20	nDCG-e@10	nDCG-e@20	Offline	Online	
ItemKNN **sin** vecindario	cosine (norm)	0,0001	0,0002	0,0001	0,0002	0,0001	0,0002	0,0001	0,0002	00:03:39,39	00:09:11,74	1,0018
	jaccard (norm)	0,0002	0,0004	0,0002	0,0003	0,0001	0,0003	0,0001	0,0003	00:00:59,60	00:09:01,81	0,9903
	intersec-pearson (norm)	0,0003	0,0012	0,0002	0,0010	0,0002	0,0010	0,0002	0,0010	00:00:53,71	00:09:19,20	0,9665
	total-pearson (norm)	0,0005	0,0026	0,0004	0,0020	0,0004	0,0019	0,0004	0,0019	00:06:09,59	00:09:33,61	0,9390
	intersec-pearson	0,0011	0,0012	0,0011	0,0012	0,0009	0,0010	0,0008	0,0009	00:00:53,71	00:09:28,29	-
	total-pearson	0,1202	0,1000	0,1346	0,1418	0,1181	0,1299	0,1081	0,1221	00:06:09,59	00:09:19,76	-
	jaccard	0,1996	0,1608	0,2319	0,2388	0,2093	0,2241	0,1958	0,2145	00:00:59,60	00:09:05,25	-
	cosine	0,2284	0,1754	0,2701	0,2670	0,2483	0,2550	0,2354	0,2474	00:03:39,39	00:09:30,87	-
ItemKNN **con** vecindario	cosine (200/1)	**0,2665**	**0,2111**	**0,3119**	**0,3142**	**0,2884**	**0,3008**	**0,2744**	**0,2922**	00:04:19,56	00:04:36,75	-
	jaccard (120/1)	0,2543	0,1996	0,2933	0,2947	0,2694	0,2809	0,2550	0,2720	00:01:17,99	00:03:34,73	-
	total-pearson (40/1)	0,1975	0,1601	0,2244	0,2282	0,2019	0,2139	0,1885	0,2045	00:03:50,61	00:05:02,65	-
	cosine (norm) (20/10)	0,1363	0,1122	0,1440	0,1382	0,1321	0,1316	0,1251	0,1275	00:03:55,76	00:06:44,73	1,0018
	total-pearson (norm) (20/6)	0,1351	0,1102	0,1459	0,1399	0,1354	0,1345	0,1291	0,1310	00:04:13,13	00:04:57,20	0,9390
	jaccard (norm) (20/6)	0,1327	0,1209	0,1416	0,1574	0,1328	0,1525	0,1276	0,1493	00:01:29,43	00:06:15,15	0,9903
	intersec-pearson (norm) (400/6)	0,0167	0,0168	0,0163	0,0186	0,0152	0,0182	0,0145	0,0179	00:02:36,74	00:10:17,91	0,9665
	intersec-pearson (20/14)	0,0141	0,0117	0,0190	0,0178	0,0163	0,0160	0,0147	0,0148	00:01:45,17	00:05:40,61	-
	Random	0,0055	0,0057	0,0054	0,0063	0,0046	0,0056	0,0041	0,0052	00:00:00,00	00:00:05,76	-

Tabla 15. Resultados de precisión en MovieLens 1M para los diferentes métodos de similitud del algoritmo Item kNN. En los casos con vecindario se ha seleccionado el que alcanza la precisión más alta. Los métodos con "(norm)" han sido normalizados, y éstos son los únicos que pueden evaluarse por RMSE. Se ha añadido el algoritmo "Random" para tener un límite inferior de valores a partir del cual se puede considerar un algoritmo como "no válido". El grupo superior (sin vecindario) está ordenado de forma ascendente por precisión @20, y el inferior (con vecindario), de forma descendente.

	ItemKNN - Blueknow										
	Recomendador	Relevancia								Tiempos		RMSE
		P@10	P@20	nDCG-n@10	nDCG-n@20	nDCG-l@10	nDCG-l@20	nDCG-e@10	nDCG-e@20	Offline	Online	
ItemKNN **sin** vecindario	cosine (norm)	0,0009	0,0011	0,0012	0,0020	0,0012	0,0020	0,0012	0,0020	00:02:49,34	00:03:01,36	0,7100
	jaccard (norm)	0,0009	0,0009	0,0012	0,0018	0,0012	0,0018	0,0012	0,0018	00:01:00,95	00:03:03,71	0,7078
	intersec-pearson (norm)	0,0009	0,0009	0,0012	0,0018	0,0012	0,0018	0,0012	0,0018	00:01:05,04	00:02:58,86	0,7241
	total-pearson (norm)	0,0012	0,0013	0,0016	0,0025	0,0016	0,0025	0,0016	0,0025	00:04:03,75	00:03:00,43	0,7405
	intersec-pearson	0,0094	0,0097	0,0112	0,0171	0,0117	0,0178	0,0106	0,0163	00:01:05,04	00:02:53,14	-
	total-pearson	0,0807	0,0622	0,1290	0,1579	0,1295	0,1586	0,1280	0,1566	00:04:03,75	00:02:58,22	-
	jaccard	0,1331	0,0958	0,2218	0,2569	0,2215	0,2572	0,2202	0,2545	00:01:00,95	00:02:56,72	-
	cosine	**0,1349**	0,0969	**0,2270**	**0,2624**	**0,2261**	**0,2621**	**0,2259**	**0,2606**	00:02:49,34	00:02:53,32	-
ItemKNN **con** vecindario	cosine (300/2)	0,1341	**0,0969**	0,2259	0,2617	0,2251	0,2615	0,2250	0,2599	00:16:35,53	00:06:35,32	-
	jaccard (120/2)	0,1313	0,0945	0,2191	0,2537	0,2188	0,2542	0,2175	0,2511	00:15:17,59	00:05:14,72	-
	total-pearson (40/2)	0,0891	0,0672	0,1410	0,1698	0,1418	0,1710	0,1396	0,1677	00:15:50,00	00:03:23,15	-
	jaccard (norm) (20/2)	0,0732	0,0665	0,1023	0,1444	0,1023	0,1448	0,1016	0,1429	00:13:33,89	00:03:24,62	0,7078
	cosine (norm) (20/2)	0,0728	0,0657	0,1001	0,1410	0,1001	0,1414	0,0992	0,1394	00:16:04,68	00:03:26,82	0,7100
	total-pearson (norm) (20/2)	0,0636	0,0533	0,0911	0,1206	0,0911	0,1211	0,0905	0,1192	00:17:06,09	00:03:05,76	0,7405
	intersec-pearson (norm) (400/10)	0,0178	0,0148	0,0266	0,0344	0,0271	0,0351	0,0260	0,0335	00:14:32,53	00:05:47,60	0,7241
	intersec-pearson (240/6)	0,0121	0,0127	0,0146	0,0235	0,0151	0,0241	0,0140	0,0228	00:14:08,74	00:05:11,97	-
	Random	0,0012	0,0011	0,0013	0,0019	0,0014	0,0020	0,0012	0,0019	00:00:00,00	00:00:10,76	-

Tabla 16. Resultados de precisión en Blueknow para los diferentes métodos de similitud del algoritmo Item kNN. En los casos con vecindario se ha seleccionado el que alcanza la precisión más alta. Los métodos con "(norm)" han sido normalizados, y éstos son los únicos que pueden evaluarse por RMSE. Se ha añadido el algoritmo "Random" para tener un límite inferior de valores a partir del cual se puede considerar un algoritmo como "no válido". El grupo superior (sin vecindario) está ordenado de forma ascendente por precisión @20, y el inferior (con vecindario), de forma descendente.

6.4.2 Sensibilidad de kNN al tamaño de vecindario

Los algoritmos de kNN disponen de una gran cantidad de parámetros configurables, de forma que un mismo algoritmo en un conjunto de datos puede resultar muy bueno o muy malo dependiendo de la configuración que se le establezca.

En este caso, los analizamos la sensibilidad y puntos óptimos de los algoritmos respecto a dos de los parámetros más significativos: el tamaño del vecindario y el número mínimo de vecinos que debe tener un usuario o ítem para poder realizar una recomendación. Realizamos el análisis observando cómo varía la precisión de los algoritmos sobre los barridos combinados (i.e. bidimensionales) de estos dos parámetros. Variamos el tamaño de vecindario entre 20 y 3000 vecinos (de 20 a 400 de diez en diez y de 500 a 3000 de 250 en 250), y el mínimo número de vecinos empezando en 1 y luego de 2 a 30 (de dos en dos).

Otros parámetros que en este caso se han dejado fijos son, entre otros, el número de ítems en común para una similitud valida (fijado en dos), parámetro de "*significance weighting*" (fijado en uno), o la similitud mínima a considerar (fijada en cero).

Los algoritmos sometidos al barrido de estos dos parámetros son User kNN e Item kNN, con coseno y Jaccard, ambos sin normalizar, como funciones de similitud para ambos casos, y se ha utilizado los conjuntos de datos de tanto de MovieLens 1M como Blueknow. Además, para medir los resultados se ha utilizado la métrica de precisión para los 20 primeros ítems del ránking.

6.4.2.1 Barridos de User kNN

La figura 17 muestra dos tablas de precisión @20 para los barridos realizados de User kNN sobre los conjuntos de Blueknow y MovieLens 1M. El rango de variación que producen estos parámetros sobre esta métrica en concreto se encuentra entre 0.0016 y 0.0866 para Blueknow, y entre 0.012 y 0.208 para MovieLens 1M. En la figura correspondiente se ha optado por dejar sólo el gradiente de color, ya que lo que interesa es ver la evolución más que los valores exactos en cada caso, sabiendo los valores mínimos y máximos de la escala que hemos mencionado en cada caso.

En el caso de Blueknow se observa una tendencia diagonal monótona, de forma que el tamaño de vecindario es directamente proporcional a la precisión obtenida. Este fenómeno es debido a que el conjunto de datos dispone de un número muy grande de usuarios, por lo que es muy probable que se encuentren usuarios altamente similares al objetivo, y en consecuencia, un número alto de vecinos genera recomendaciones mejores (en este caso la máxima precisión se alcanza con 200 vecinos).

En cuanto a MovieLens 1M, la tendencia es parecida a la de Blueknow pero al disponer de muchos menos usuarios, la diagonal es mucho más cerrada. Esto se debe a que hay menos usuarios que sean realmente similares al usuario objetivo, por lo que un aumento excesivo del vecindario hace que se tenga en cuenta usuarios que no compartan realmente gustos con el usuarios objetivo, creando peores recomendaciones. De esta forma, se alcanza el máximo mucho antes que en el caso anterior, en torno a un vecindario de 60.

En ambos casos, el valor del parámetro de mínimo vecindario no ayuda, pues hace decrecer la precisión de forma monótona, encontrando su valor óptimo en uno, que equivale a que no se tenga en cuenta ningún umbral de vecindario mínimo.

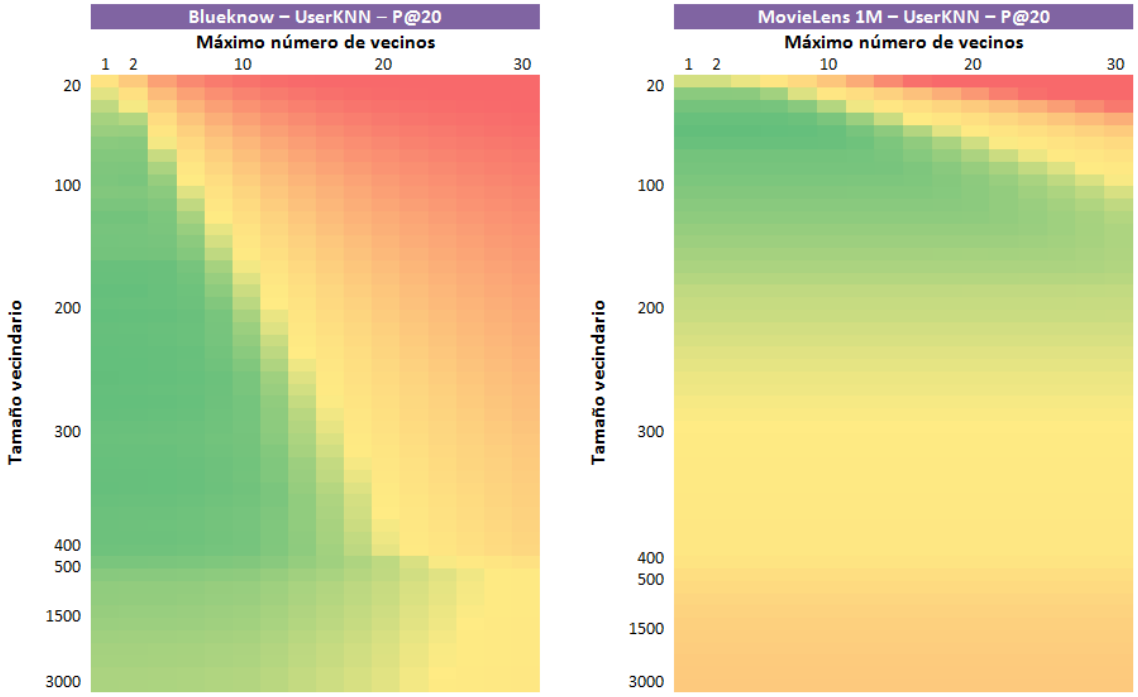

Figura 17. Tendencia de Precisión @20 para User kNN Jaccard en Blueknow a la izquierda y MovieLens 1M a la derecha. La escala del eje horizontal (máximo número de vecinos) comienza en 1 y de 2 a 30 tiene un incremento de 2 en cada fila; y la del eje vertical (tamaño del vecindario), de 20 a 400 con un incremento de 10, y de 500 a 300 con un incremento de 250.

6.4.2.2 Barridos de Item kNN

La figura 18 muestra otras dos tablas de precisión @20 para los barridos realizados de Item kNN sobre Blueknow y MovieLens 1M. En este caso, el rango de variación que producen estos parámetros sobre esta métrica en concreto se encuentra entre 0.0016 y 0.0958 para Blueknow, y entre 0.012 y 0.1998 para MovieLens 1M.

La tendencia que se puede observar en el conjunto de Blueknow no es diagonal como en el caso anterior sino más atenuante con el aumento del vecindario. En torno a los 1000 vecinos, la precisión sigue aumentando con el tamaño del vecindario pero lo hace de forma muy leve, variando únicamente en milésimas. Así, el uso de vecindario en este caso es beneficioso como se había deducido en el apartado anterior y compensa en gran medida su uso, teniendo en cuenta que la precisión no aumenta de forma significativa, pero sin embargo el coste de ejecución crece con el vecindario, como veremos en la sección 6.4.3 a continuación.

Es posible que el vecindario óptimo crezca con el tamaño del conjunto de datos (en este caso, el número de ítems) y que si siguiésemos aumentándolo en Blueknow podría llegarse a un punto de cambio de tendencia. Sin embargo, el coste de la prueba requiere dedicar un trabajo expreso para seguir explorando más allá, por lo que se entiende que está fuera de los límites del presente trabajo.

Para MovieLens 1M, sin embargo, se percibe una tendencia más parecida a User kNN que a la de este mismo algoritmo sobre Blueknow, ya que alcanza el máximo valor esta vez en torno a un vecindario de 200 vecinos, pero a partir de este punto disminuye notablemente su precisión.

Nuevamente, en ambos casos el valor del parámetro de mínimo vecindario empeora la precisión, siendo óptimo no tener umbral de vecindario mínimo (valor igual a uno).

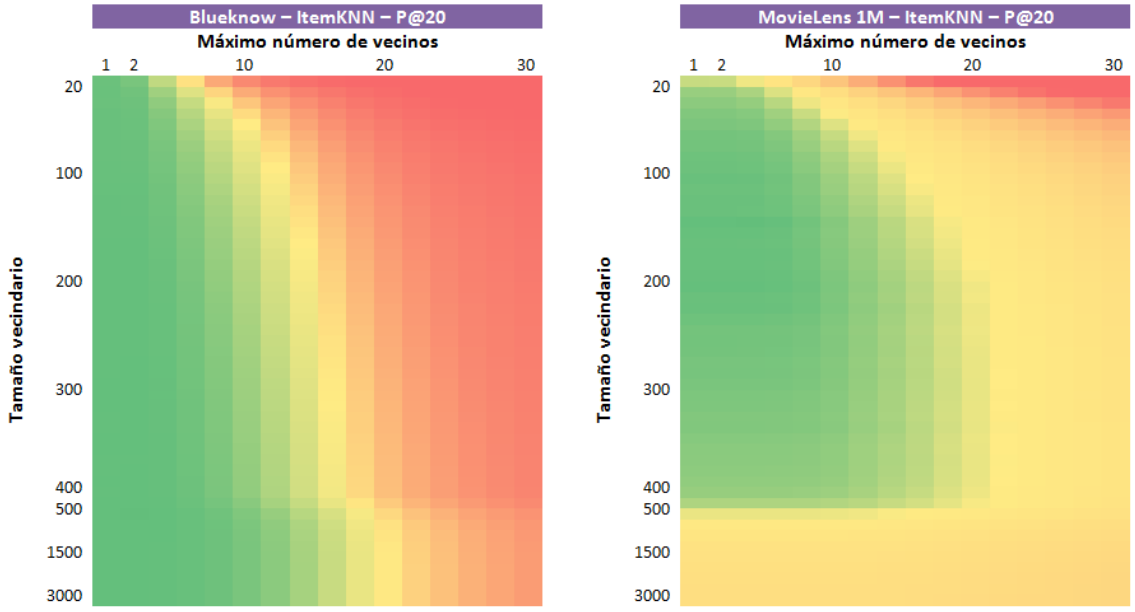

Figura 18. Tendencia de Precisión @20 para Item kNN Jaccard en Blueknow a la izquierda y MovieLens 1M a la derecha. La escala del eje horizontal (máximo número de vecinos) comienza en 1 y de 2 a 30 tiene un incremento de 2 en cada fila; y la del eje vertical (tamaño del vecindario), de 20 a 400 con un incremento de 10, y de 500 a 300 con un incremento de 250.

6.4.2.3 Consumo de recursos según el vecindario

Es importante estudiar la correlación entre el tamaño del vecindario y el tiempo de ejecución y memoria que requiere, ya que un algoritmo con buena precisión puede ser inviable en un caso real si se tiene en cuenta el consumo de recursos que requiere.

En las gráficas siguientes (figuras 19 y 20) se muestra la variación del consumo de recursos de tiempo y memoria dependiendo del tamaño del vecindario, y para tener una visión completa, se muestra además la variación de la precisión respecto al vecindario. Se realiza sobre una sección de los barridos anteriormente realizados, fijando el valor del parámetro del número mínimo de usuarios en uno, el óptimo para todos los casos.

Los resultados obtenidos para las versiones de coseno y Jaccard en los dos algoritmos kNN son equivalentes en coste y orden de complejidad, por lo que para obtener una visión más clara de la tendencia que cada uno sigue, sólo mostramos la tendencia para la función de similitud de Jaccard.

En cuanto a la precisión, en el apartado anterior se ha observado que se obtienen resultados diferentes entre los dos conjuntos de datos. Mientras que en Blueknow se alcanza el máximo muy pronto y, a partir de ese punto, la precisión se mantiene según se aumenta el vecindario; en MovieLens, una vez alcanzado el óptimo, la precisión baja drásticamente conforme se va aumentando el vecindario.

Respecto a la variación de tiempo se observa un fenómeno parecido al anterior. En Blueknow, Item kNN aumenta el tiempo de ejecución hasta un vecindario de 500 vecinos, manteniéndose constante a partir de entonces independientemente del tamaño del vecindario. Esto se debe a que los ítems cuya similitud con el ítem objetivo son cero se invalidan, ya que no aportan nada. Así, por ejemplo, aunque el vecindario establecido

sea de 3000, si el ítem objetivo sólo tiene similitud válida con 100, sólo se tendrán en cuenta éstos, ahorrando así muchos cálculos innecesarios que aumentan la eficiencia temporal. Dicho de otro modo, aunque se establece un determinado tamaño (grande) de vecindario, en la práctica no se encuentran tantos vecinos y los vecindarios efectivos son más pequeños.

Sin embargo, User kNN parece tener la misma tendencia, pero mucho más tardía en llegar al tiempo constante. Esto es debido a que el número de usuarios es mucho mayor al de ítems, por lo que, si la estabilización del tiempo de ejecución se alcanza en torno al 10% de los elementos, para los ítems se alcanzará mucho antes que para los usuarios.

Por el contrario, en el caso de MovieLens los resultados son completamente diferentes, ya que la variación de tiempo respecto al tamaño del vecindario aumenta de forma lineal dentro de los límites de los vecindarios trabajados.

Finalmente, en cuanto a la memoria se observa que en el caso de Blueknow el consumo de memoria se mantiene mayormente constante en ambos algoritmos, mientras que en MovieLens 1M el consumo aumenta. Esto podría ser debido a que en el caso de Blueknow, al tener tan poca densidad, el cálculo de similitud entre usuario e ítem no es problema debido al poco solapamiento, sino que lo más costoso es la lectura de similitudes desde fichero, lo que es independiente del tamaño de vecindario. Sin embargo, en MovieLens 1M ocurriría lo contrario, el consumo de memoria se dispararía al realizar las recomendaciones y la lectura de los ficheros de similitudes no serían relevantes frente a ello.

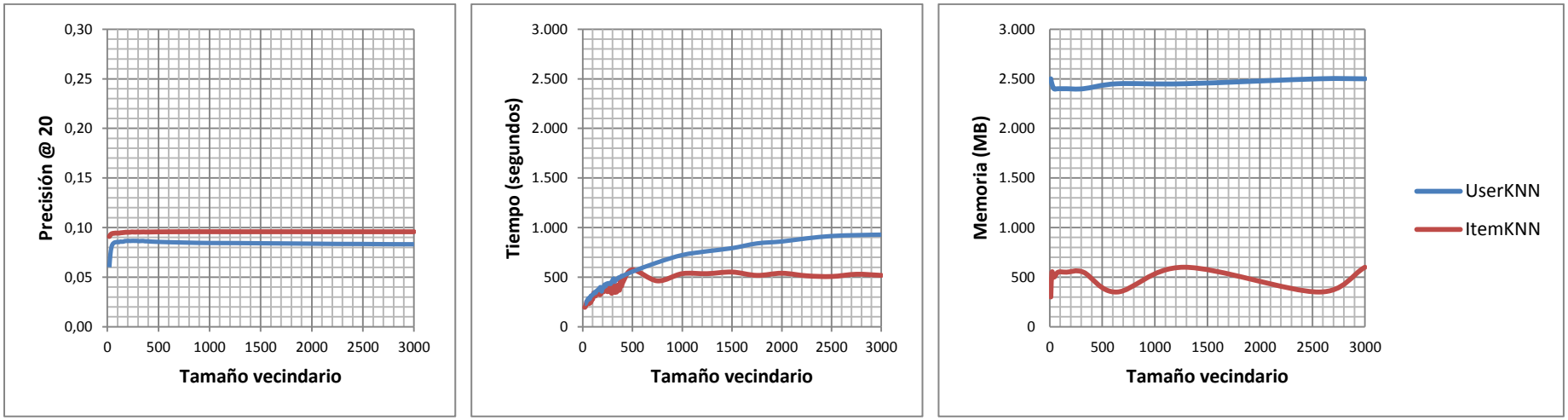

Figura 19. Variación de precisión, tiempo y memoria de User kNN e Item kNN según el vecindario para Blueknow.

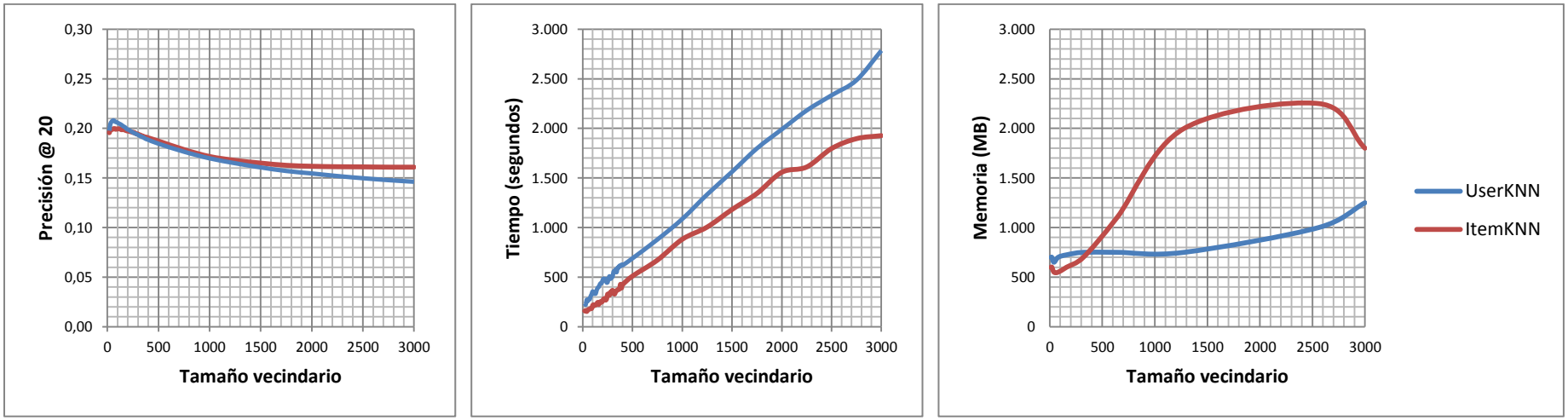

Figura 20. Variación de precisión, tiempo y memoria de User kNN e Item kNN según el vecindario para MovieLens 1M.

6.5 Resultados de los algoritmos de descomposición de matrices (SVD)

Los algoritmos de descomposición de matrices (SVDs) incluidos en el trabajo requieren un gran uso de recursos que aumenta cuanto mayor es el conjunto de datos al que se aplica. Por ello, sólo ha resultado viable probarlos en determinados conjuntos, ya que en otros la ejecución se extiende durante semanas, a pesar de los recursos disponibles, o requiere más memoria de la disponible para estas pruebas. Así, los conjuntos de datos sobre los que se ha podido aplicar estos algoritmos son MovieLens 1M, MovieLens 100K, una partición especial de este último (seleccionando un conjunto de usuarios "nuevos" a los que recomendar) y Blueknow.

Algunos de estos algoritmos están orientados a la recomendación de ítems a usuarios nuevos en el sistema, de forma que es mejor que los conjuntos de datos tengan separación de usuarios nuevos (aquellos que aparecen en entrenamiento y en test) y usuarios viejos (los que sólo aparecen en el entrenamiento). En los conjuntos de datos sobre los que se ha probado cumplen esta condición sólo dos, Blueknow y la partición especial de MovieLens 100K.

Los resultados obtenidos para las métricas de relevancia se muestran en las tablas 17 a 20, una para cada conjunto de datos estudiado para este caso.

Los métodos que mostramos para ser comparados en cada caso, además de los propios SVDs a estudiar, son:

- **Algoritmo con mejores resultados:** para establecer el límite superior, o ver si se mejoran los resultados obtenidos.
- **pLSA:** éste pertenece a la familia de factorización de matrices y vale la pena comprobar si se obtienen mejoras dentro de la misma familia de algoritmos.
- **Random** y **Most Popular:** para establecer el límite inferior, ya que un algoritmo no se considera relevante si no supera estos dos.

MovieLens 100K									
Recomendador	Relevancia								Tiempo
	P@10	P@20	nDCG-n@10	nDCG-n@20	nDCG-l@10	nDCG-l@20	nDCG-e@10	nDCG-e@20	
ASVDR	**0,2586**	**0,2002**	**0,3423**	**0,3641**	**0,3251**	**0,3547**	**0,3142**	**0,3480**	00:00:33,01
UserKNN (cosine)	0,2424	0,1837	0,3278	0,3449	0,3114	0,3356	0,3010	0,3290	00:00:21,18
pLSA	0,2312	0,1787	0,3052	0,3260	0,2876	0,3155	0,2763	0,3081	00:00:39,04
SVDR	0,2115	0,1628	0,2889	0,3113	0,2726	0,3009	0,2621	0,2933	00:02:38,24
MostPopular	0,1361	0,1061	0,1694	0,1798	0,1572	0,1726	0,1493	0,1676	00:00:06,02
SVDN	0,0326	0,0300	0,0369	0,0413	0,0343	0,0401	0,0328	0,0393	00:51:43,47
Random	0,0107	0,0101	0,0113	0,0127	0,0101	0,0118	0,0093	0,0111	00:00:07,95

Tabla 17. Resultados de precisión de los algoritmos matriciales para MovieLens 100K. Los tiempos que se muestran para cada algoritmo son offline y online conjuntamente. Ninguna de las dos versiones de HSVD puede realizarse porque la partición de entrenamiento y test tienen los mismos usuarios.

MovieLens 1M									
Recomendador	Relevancia								Tiempo
	P@10	P@20	nDCG-n@10	nDCG-n@20	nDCG-l@10	nDCG-l@20	nDCG-e@10	nDCG-e@20	
ASVDR	**0,2823**	**0,2241**	**0,3305**	**0,3342**	**0,3079**	**0,3216**	**0,2945**	**0,3135**	00:22:48,17
UserKNN (cosine)	0,2678	0,2095	0,3248	0,3276	0,3054	0,3171	0,2939	0,3104	00:04:22,46
pLSA	0,2641	0,2122	0,3040	0,3103	0,2808	0,2967	0,2670	0,2879	00:06:35,67
MostPopular	0,1312	0,1093	0,1483	0,1529	0,1393	0,1489	0,1341	0,1465	00:00:05,35
SVDR	0,0984	0,0780	0,1122	0,1099	0,1026	0,1045	0,0970	0,1011	02:19:02,84
HSVD (SVDR)	0,0843	0,0659	0,0940	0,0926	0,0848	0,0874	0,0794	0,0841	00:05:18,68
HSVD (SVDN)	0,0457	0,0413	0,0459	0,0519	0,0403	0,0481	0,0370	0,0457	00:05:42,10
SVDN	0,0085	0,0093	0,0085	0,0106	0,0075	0,0097	0,0069	0,0092	04:36:46,10
Random	0,0055	0,0057	0,0054	0,0063	0,0046	0,0056	0,0041	0,0052	00:00:05,76

Tabla 18. Resultados de precisión de los algoritmos matriciales para MovieLens 1M. Los tiempos que se muestran para cada algoritmo son offline y online conjuntamente.

MovieLens 100K (partición especial)									
Recomendador	Relevancia								Tiempo
	P@10	P@20	nDCG-n@10	nDCG-n@20	nDCG-l@10	nDCG-l@20	nDCG-e@10	nDCG-e@20	
UserKNN (cosine)	**0,4227**	**0,3676**	**0,4277**	**0,3912**	**0,3740**	**0,3509**	**0,3432**	**0,3270**	00:00:27,40
HSVD (SVDR)	0,3568	0,3186	0,3450	0,3256	0,3004	0,2903	0,2750	0,2695	00:00:16,58
pLSA	0,3497	0,2959	0,3411	0,3102	0,2901	0,2727	0,2606	0,2504	00:00:42,01
MostPopular	0,3022	0,2489	0,3117	0,2717	0,2695	0,2398	0,2452	0,2209	00:00:01,49
HSVD (SVDN)	0,2741	0,2322	0,2650	0,2410	0,2271	0,2120	0,2056	0,1952	00:05:16,90
SVDR	0,1859	0,1622	0,1818	0,1655	0,1564	0,1438	0,1423	0,1316	00:00:36,25
ASVDR	0,1541	0,1332	0,1503	0,1365	0,1239	0,1167	0,1088	0,1050	00:00:24,05
SVDN	0,0454	0,0459	0,0388	0,0402	0,0319	0,0330	0,0280	0,0289	00:50:20,55
Random	0,0259	0,0311	0,0203	0,0249	0,0156	0,0201	0,0129	0,0173	00:00:00,99

Tabla 19. Resultados de precisión de los algoritmos matriciales para la partición específica de MovieLens 100K. Los tiempos que se muestran para cada algoritmo son offline y online conjuntamente.

Blueknow									
Recomendador	Relevancia								Tiempo
	P@10	P@20	nDCG-n@10	nDCG-n@20	nDCG-l@10	nDCG-l@20	nDCG-e@10	nDCG-e@20	
ItemKNN (cosine)	**0,1349**	**0,0969**	**0,2270**	**0,2624**	**0,2261**	**0,2621**	**0,2259**	**0,2606**	00:05:43,06
pLSA	0,0725	0,0569	0,1148	0,1421	0,1150	0,1424	0,1137	0,1406	00:07:45,27
HSVD (SVDN)	0,0187	0,0161	0,0248	0,0335	0,0252	0,0341	0,0241	0,0327	04:08:17,24
MostPopular	0,0162	0,0138	0,0234	0,0304	0,0233	0,0305	0,0231	0,0299	00:00:12,42
HSVD (SVDR)	0,0133	0,0110	0,0187	0,0246	0,0189	0,0247	0,0186	0,0245	08:56:06,02
SVDR	0,0069	0,0060	0,0098	0,0129	0,0100	0,0130	0,0096	0,0125	11:05:36,81
ASVDR	0,0057	0,0049	0,0083	0,0107	0,0083	0,0108	0,0082	0,0106	06:19:29,30
Random	0,0012	0,0011	0,0013	0,0019	0,0014	0,0020	0,0012	0,0019	00:00:10,76
SVDN	0,0005	0,0006	0,0007	0,0012	0,0006	0,0011	0,0008	0,0013	09:48:24,68

Tabla 20. Resultados de precisión de los algoritmos matriciales para Blueknow. Los tiempos que se muestran para cada algoritmo son offline y online conjuntamente.

Los algoritmos de SVDN y SVDR no dan buenos resultados para ninguno de los conjuntos de datos donde se ha probado, ni tampoco consiguen mejorar los resultados para su familia de algoritmos.

ASVDR, aunque está orientado a la recomendación de usuarios nuevos, obtiene un nivel muy alto de relevancia en aquellos conjuntos de datos donde apenas dispone de entrenamiento por parte de usuarios viejos (conjuntos donde no hay separación específica de usuarios, MovieLens 1M y 100K).

Sin embargo, los algoritmos de HSVD centrados en la recomendación a usuarios nuevos en el sistema sí que obtienen resultados muy positivos para aquellos conjuntos de datos que disponen de separación de usuarios nuevos y viejos (los dos últimos, Blueknow y la partición especial de MovieLens 100K).

Se confirma la idea con la que se propusieron estos últimos algoritmos en el artículo en el que nos hemos basado (Pu 2013). Sin embargo, para este estudio se buscan algoritmos no tan centrados en el caso particular de usuarios nuevos y viejos, sino algoritmos que se puedan aplicar de forma general.

En conclusión, no se puede establecer un algoritmo claro que mejore los resultados de pLSA, ya que aunque en algunos casos lo superan, a nivel global éste es mejor para todos los conjuntos de datos.

7. Conclusiones

El presente trabajo aborda la implementación y puesta a punto de una colección de algoritmos de recomendación, así como su ejecución y comparación a lo largo de un amplio espectro de conjuntos de datos de distinta naturaleza, métricas, así como opciones de configuración de los algoritmos.

7.1 Resumen y contribuciones

La realización de este trabajo ha pasado por varias fases para su completo desarrollo. Lo primero fue establecer los algoritmos y métricas a ser estudiadas. En el primer caso se eligieron aquellos algoritmos de recomendación más importantes, bien por sus buenos resultados en la literatura, como aquellos más populares y clásicos, o bien por su carácter innovador. Con respecto a las métricas, se escogieron aquellas más utilizadas en el estado del arte, relacionadas con el acierto y la precisión. Pero también se han estudiado otras métricas menos tradicionales que han cobrado importancia más recientemente en el área, como son las orientadas a valorar la novedad y diversidad de las recomendaciones.

El ámbito del estudio ha incluido tanto algoritmos de diferentes familias (filtrado colaborativo, basados en contenido) como conjuntos de datos varios, encontrándose tanto casos de estudio (competición de Netflix) como casos reales (Blueknow o Twitter), tanto datos explícitos (puntuaciones de usuarios a ítems) como implícitos (eventos de clicks, visitas, compras), y tanto tareas de recomendación tradicionales (recomendación de películas basadas en ratings) como escenarios más novedosos (redes sociales en Twitter).

A continuación, se realizaron múltiples barridos y pruebas sobre los algoritmos de recomendación con el objetivo de encontrar sus valores óptimos en los conjuntos de datos para las métricas elegidas. Paralelamente, se estudió cómo afectan ciertos parámetros de los algoritmos (p.e. tamaño del vecindario) respecto al consumo de memoria o al tiempo de ejecución.

Finalmente, se estudiaron los resultados obtenidos para cada conjunto de datos y se ha observado que, aunque los algoritmos kNN habían perdido cierto protagonismo en la comunidad a favor de la factorización de matrices, debido al menor acierto comparativo de los métodos kNN en las métricas de error (RMSE), estos algoritmos son realmente competitivos en cuanto a métricas de precisión. Sin embargo, un punto muy importante a tener en cuenta para los algoritmos que utilizan vecindario es la eficiencia en cuanto a tiempo y memoria, ya que dependiendo de la cantidad de usuarios o ítems, el algoritmo kNN puede dejar de ser viable, siendo éste el punto a partir del cual otros algoritmos como pLSA toman la delantera.

Con este trabajo se ha conseguido una visión panorámica de todos los algoritmos de recomendación más importantes de la literatura hasta el momento. Se ha realizado una comparación general, destacando las métricas orientadas a ránking (precisión, novedad y diversidad), sobre diferentes conjuntos de datos.

7.2 Trabajo futuro

Son múltiples y amplias las posibilidades de continuación del trabajo realizado hasta aquí.

Si bien el trabajo tiene vocación de amplitud en el ámbito de opciones y configuraciones estudiadas, está obviamente lejos de agotar las infinitas posibilidades, y es relevante seguir ampliando el estudio en diversas direcciones relevantes. Así, por ejemplo, en el presente trabajo se han realizado barridos sobre el tamaño del vecindario únicamente para obtener el valor óptimo respecto a la precisión del recomendador. Sin embargo, sería interesante también ver en qué medida afecta este parámetro en métricas de novedad y diversidad, haciendo especial hincapié en las versiones normalizadas de los algoritmos, ya que aunque sus valores de precisión no son buenos (comparados con las versiones no normalizadas), es posible que los de novedad y diversidad sí lo sean.

En esta misma línea, también podrían realizarse los mismos barridos hechos sobre MovieLens 1M, esta vez sobre las otras dos variaciones del mismo conjunto de datos, MovieLens 100K y 10M. Esto tendría el objetivo de observar la tendencia de la precisión que adquiere según la variación de la cantidad de datos con la que se trabaja.

Asimismo, como ampliación podría añadirse un parámetro más de estudio para estos barridos realizados, como por ejemplo, centrar la puntuación en la media. En el presente trabajo se ha realizado en la medida de similitud mediante el método de Pearson, pero sería también interesante centrar la puntuación en la media en la combinación lineal posterior. Además, podría incluirse en el estudio un método de recomendación híbrida.

Por otro lado, dejando atrás los barridos y más orientado al caso particular de Blueknow, prevemos simular el algoritmo que utiliza esta empresa para generar sus recomendaciones y comparar los resultados obtenidos para las diferentes métricas con los algoritmos estudiados en el presente trabajo.

Finalmente, más allá de la experimentación comparativa, el trabajo aborda puntos que merecen amplia atención por sí mismos. En particular, se proyecta profundizar en la recomendación en redes sociales hacia la recomendación de usuarios, de URLs mediante tweets, etc. En este contexto, es interesante definir, medir y optimizar la novedad y diversidad en las redes sociales, cosa que no se tiene en cuenta en los algoritmos documentados que hoy en día se han empezado a proponer en las mismas.

Referencias

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

A. Alhambra. Recomendación de contactos en Twitter. Trabajo Fin de Grado, Escuela Politécnica Superior, Universidad Autónoma de Madrid, June 2014.

X. Amatriain, J. M. Pujol, N. Oliver. I like it... I like it not: Evaluating user ratings noise in recommender systems. User Modeling, Adaptation, and Personalization 5535(21), June 2009, pp. 247-258.

R. Baeza-Yates, B. Ribeiro-Neto. Modern Information Retrieval - the concepts and technology behind search, 2nd Edition. Pearson Education Ltd., 2011.

Ò. Celma, P. Herrera. A new approach to evaluating novel recommendations. 2nd ACM Conference on Recommender Systems (RecSys 2008). Lausanne, Switzerland, October 2008, pp. 179-186.

O. Chapelle, S. Ji, C. Liao, E. Velipasaoglu, L. Lai, S. L. Wu. Intent-based diversification of web search results: Metrics and algorithms. Information Retrieval Journal, 14(6), December 2011, pp. 572–592.

C. L. A. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, I. MacKinnon. Novelty and Diversity in Information Retrieval Evaluation. 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2008). Singapore, July 2008, pp. 659-666.

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

T. Hofmann. Probabilistic latent semantic indexing. 22nd Annual International ACM SIGIR conference on Research and development in information retrieval (SIGIR 1999). Berkeley, CA, August 1999, pp. 50-57.

K. Järvelin, J. Kekäläinen. IR evaluation methods for retrieving highly relevant documents. 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2000). Athens, Greece, July 2000, pp. 41-48.

Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2008). Las Vegas, NV, August 2008, pp. 426-434.

Y. Koren, R. Bell, C. Volinsky. Matrix Factorization Techniques for Recommender Systems. IEEE Computer 42(8), August 2009, pp. 30-37.

Y. Koren, P. Cremonesi, R. Turrin. Performance of recommender algorithms on top-N recommendation tasks. 4th ACM Conference on Recommender Systems (RecSys 2010). Barcelona, Spain, September 2010, pp. 39-46.

L. Pu, B. Faltings. Understanding and improving relational matrix factorization in recommender systems. 7th ACM Conference on Recommender Systems (RecSys 2013). Hong Kong, China, October 2013, pp. 41-48.

P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, J. Riedl. GroupLens: an open architecture for collaborative filtering of netnews. ACM Conference on Computer Supported Cooperative Work (CSCW 1994). Chapel Hill, NC, October 1994, pp. 175-186.

F. Ricci, L. Rokach, L., B. Shapira, P. B. Kantor (Eds.). Recommender Systems Handbook, 1st Edition. Springer, 2011.

B. Sarwar, G. Karypis, J. Konstan, J. Riedl. Application of dimensionality reduction in recommender system-a case study. ACM WebKDD Web Mining for E-Commerce Workshop. Boston, MA, August 2000.

S. Vargas, P. Castells. Rank and Relevance in Novelty and Diversity Metrics for Recommender Systems. 5th ACM Conference on Recommender Systems (RecSys 2011). Chicago, Illinois, October 2011, pp. 109-116.

C. X. Zhai, W. W. Cohen, J. Lafferty. Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2003). Toronto, Canada, July 2003, pp. 10-17.

M. Zhang, N. Hurley. Avoiding Monotony: Improving the Diversity of Recommendation Lists. 2nd International ACM Conference in Recommender Systems (RecSys 2008). Lau-sanne, Switzerland, October 2008, pp. 123-130.

Anexo I. Resultados completos para Blueknow, MovieLens 1M y Twitter

En el apartado 6.2 se muestran las tablas con los resultados generales para las métricas de precisión para todos los conjuntos de datos. En los casos de Blueknow, MovieLens 1M y Twitter, donde también se calculan las métricas de novedad y diversidad, se exponen tablas reducidas que contienen las métricas más importantes de precisión, novedad y diversidad.

En este anexo se añaden las tablas completas de precisión, novedad y diversidad con todas las métricas calculadas. Las tablas 21 a 23 muestran los resultados para el conjunto de datos de MovieLens 1M; las tablas 24 a 26, para Blueknow; y por último, las tablas 27 a 29 contienen los resultados para el caso de Twitter.

Como puede observarse, los resultados de todas las métricas son las esperadas sin ninguna particularidad importante, lo que ha permitido la reducción de las tablas para el apartado 6 de resultados sin perder información relevante.

	MovieLens 1M								
	Method	Accuracy							RMSE
		P@10	P@20	nDCG-n@10	nDCG-n@20	nDCG-l@10	nDCG-l@20	nDCG-e@10	nDCG-e@20	
IRG	ItemKNN (jaccard 70)	0,2520	0,1998	0,2903	0,2936	0,2670	0,2801	0,2530	0,2714	-
	pLSA	0,2641	**0,2122**	0,3040	0,3103	0,2808	0,2967	0,2670	0,2879	-
	ItemKNN (cosine 150)	0,2672	0,2115	0,3123	0,3144	0,2884	0,3005	0,2741	0,2916	-
	UserKNN (jaccard 60)	0,2650	0,2080	0,3209	0,3242	0,3004	0,3127	0,2882	0,3053	-
	UserKNN (cosine 60)	**0,2678**	0,2095	**0,3248**	**0,3276**	**0,3054**	**0,3171**	**0,2939**	**0,3104**	-
MyMediaLite	UserKNN (cosine 80)	0,2322	0,1815	0,2817	0,2842	0,2672	0,2780	0,2588	0,2742	-
	UserKNN (jaccard 50)	0,2300	0,1813	0,2780	0,2827	0,2635	0,2763	0,2551	0,2725	-
	WRMF	0,2281	0,1826	0,2689	0,2713	0,2538	0,2646	0,2450	0,2607	-
	ItemKNN (jaccard 190)	0,2064	0,1743	0,2278	0,2428	0,2114	0,2335	0,2016	0,2276	-
	MultiCoreBPRMF	0,2008	0,1644	0,2317	0,2370	0,2181	0,2310	0,2104	0,2275	-
	BPRSLIM	0,2000	0,1616	0,2398	0,2481	0,2246	0,2401	0,2156	0,2350	-
	BPRMF	0,1965	0,1613	0,2264	0,2338	0,2131	0,2276	0,2055	0,2238	-
	LeastSquareSLIM	0,1926	0,1515	0,2295	0,2296	0,2159	0,2236	0,2080	0,2200	-
	ItemKNN (cosine 80)	0,1835	0,1561	0,1999	0,2119	0,1835	0,2016	0,1737	0,1948	-
	MostPopular	0,1312	0,1093	0,1483	0,1529	0,1393	0,1489	0,1341	0,1465	-
	SoftMarginRankingMF	0,1171	0,1015	0,1293	0,1377	0,1180	0,1310	0,1113	0,1267	-
	WeightedBPRMF	0,0922	0,0806	0,1017	0,1118	0,0929	0,1064	0,0877	0,1029	-
	Zero	0,0272	0,0310	0,0287	0,0378	0,0249	0,0351	0,0226	0,0332	-
	Random	0,0055	0,0057	0,0054	0,0063	0,0046	0,0056	0,0041	0,0052	-
LensKit	UserKNN	0,0174	0,0219	0,0129	0,0187	0,0122	0,0185	0,0118	0,0184	0,8958
	ItemKNN	0,0095	0,0213	0,0073	0,0195	0,0073	0,0200	0,0073	0,0204	0,8505
	FunkSVD	0,0032	0,0092	0,0045	0,0086	0,0044	0,0087	0,0043	0,0087	0,8599
	SlopeOne	0,0001	0,0011	0,0001	0,0009	0,0001	0,0009	0,0001	0,0009	0,9068

Tabla 21. Resultados completos de las métricas de precisión para el conjunto de datos de MovieLens 1M.

	MovieLens 1M											
	Method	nDCG-e@20	Diversity									
			EILD@10	EILD@20	ERR-IA@10	ERR-IA@20	a-nDCG@10	a-nDCG@20	S-Recall@10	S-Recall@20	S-Recall-NOR@10	S-Recall-NOR@20
IRG	ItemKNN (jaccard 70)	0,2714	0,6407	0,7058	0,1177	0,1242	0,2804	0,3234	0,2765	0,3643	0,6361	0,7911
IRG	pLSA	0,2879	0,6486	0,7014	**0,2080**	**0,2162**	0,2884	0,3322	0,2865	0,3745	0,6308	0,7764
IRG	ItemKNN (cosine 150)	0,2916	0,6295	0,6871	0,1250	0,1313	0,2950	0,3366	0,2829	0,3679	0,6194	0,7728
IRG	UserKNN (jaccard 60)	0,3053	0,6694	0,7234	0,1335	0,1400	0,3116	0,3553	0,2999	0,3871	0,6601	0,8078
IRG	UserKNN (cosine 60)	**0,3104**	0,6734	0,7274	0,1360	0,1424	**0,3151**	**0,3585**	**0,3033**	**0,3901**	0,6643	0,8143
MyMediaLite	UserKNN (cosine 80)	0,2742	0,6753	0,7339	0,1199	0,1262	0,2742	0,3164	0,2752	0,3621	0,6645	0,8155
MyMediaLite	UserKNN (jaccard 50)	0,2725	0,6725	0,7296	0,1186	0,1251	0,2714	0,3144	0,2728	0,3605	0,6624	0,8123
MyMediaLite	WRMF	0,2607	0,6527	0,7116	0,1981	0,2062	0,2565	0,2980	0,2595	0,3463	0,6428	0,7942
MyMediaLite	BPRSLIM	0,2350	0,6693	0,7182	0,1796	0,1880	0,2368	0,2785	0,2461	0,3319	0,6655	0,8166
MyMediaLite	ItemKNN (jaccard 190)	0,2276	0,6216	0,6806	0,0883	0,0952	0,2160	0,2617	0,2332	0,3267	0,5912	0,7536
MyMediaLite	MultiCoreBPRMF	0,2275	0,6496	0,7086	0,1724	0,1807	0,2225	0,2628	0,2325	0,3175	0,6325	0,7791
MyMediaLite	BPRMF	0,2238	0,6557	0,7109	0,1667	0,1751	0,2200	0,2615	0,2346	0,3209	0,6354	0,7814
MyMediaLite	LeastSquareSLIM	0,2200	0,6695	0,7286	0,1674	0,1754	0,2285	0,2668	0,2380	0,3224	0,6582	0,8173
MyMediaLite	ItemKNN (cosine 80)	0,1948	0,6061	0,6609	0,0773	0,0835	0,1895	0,2303	0,2095	0,2949	0,5765	0,7302
MyMediaLite	MostPopular	0,1465	0,6948	0,7589	0,0914	0,1002	0,1372	0,1760	0,1648	0,2559	0,6213	0,8182
MyMediaLite	SoftMarginRankingMF	0,1267	0,6993	0,7408	0,0970	0,1049	0,1365	0,1697	0,1598	0,2368	**0,6992**	**0,8417**
MyMediaLite	WeightedBPRMF	0,1029	0,5780	0,6288	0,0742	0,0811	0,0970	0,1236	0,1151	0,1761	0,5050	0,6404
MyMediaLite	Zero	0,0332	0,6920	0,7266	0,0222	0,0272	0,0369	0,0561	0,0479	0,1002	0,6477	0,7826
MyMediaLite	Random	0,0052	**0,7379**	**0,7810**	0,0036	0,0045	0,0056	0,0086	0,0086	0,0176	0,5923	0,7671
LensKit	ItemKNN	0,0204	0,6444	0,7131	0,0019	0,0046	0,0066	0,0221	0,0129	0,0541	0,3285	0,5978
LensKit	UserKNN	0,0184	0,5906	0,6970	0,0031	0,0049	0,0104	0,0205	0,0209	0,0488	0,3181	0,5250
LensKit	FunkSVD	0,0087	0,6850	0,6462	0,0022	0,0033	0,0040	0,0100	0,0038	0,0208	0,2517	0,3455
LensKit	SlopeOne	0,0009	0,5465	0,6196	0,0000	0,0002	0,0000	0,0009	0,0001	0,0026	0,2214	0,3892

Tabla 22. Resultados completos de las métricas de diversidad para el conjunto de datos de MovieLens 1M. Los algoritmos de IRG (superior) están ordenados por nDCG ascendente y los de MyMediaLite (central) y LensKit (inferior), por orden descendiente.

	MovieLens 1M							
	Method	nDCG-e@20	Novelty					
			EPC@10	EPC@20	EPD@10	EPD@20	EFN@10	EFN@20
IRG	ItemKNN (jaccard 70)	0,2714	0,8229	0,8331	0,7623	0,7704	2,7500	2,8515
	pLSA	0,2879	0,8420	0,8608	0,7603	0,7633	2,9390	3,1571
	ItemKNN (cosine 150)	0,2916	0,8382	0,8512	0,7519	0,7576	2,9080	3,0537
	UserKNN (jaccard 60)	0,3053	0,8015	0,8257	0,7754	0,7798	2,5098	2,7463
	UserKNN (cosine 60)	**0,3104**	0,7999	0,8231	0,7743	0,7792	2,4903	2,7133
MyMediaLite	UserKNN (cosine 80)	0,2742	0,7802	0,8026	0,7762	0,7824	2,3057	2,4946
	UserKNN (jaccard 50)	0,2725	0,7886	0,8116	0,7738	0,7792	2,3842	2,5900
	WRMF	0,2607	0,7961	0,8184	0,7615	0,7683	2,4184	2,6119
	BPRSLIM	0,2350	0,8597	0,8728	0,7510	0,7573	3,1970	3,3649
	ItemKNN (jaccard 190)	0,2276	0,8798	0,8792	0,7430	0,7496	3,8154	3,6390
	MultiCoreBPRMF	0,2275	0,8095	0,8264	0,7664	0,7723	2,6663	2,8243
	BPRMF	0,2238	0,8220	0,8390	0,7634	0,7688	2,8258	2,9862
	LeastSquareSLIM	0,2200	0,7786	0,8038	0,7814	0,7858	2,2663	2,4648
	ItemKNN (cosine 80)	0,1948	0,9076	0,9076	0,7361	0,7414	4,1094	4,0292
	MostPopular	0,1465	0,7074	0,7374	0,8415	0,8333	1,7839	1,9485
	SoftMarginRankingMF	0,1267	0,8546	0,8672	0,7804	0,7819	3,1937	3,3552
	WeightedBPRMF	0,1029	0,9483	0,9511	0,7350	0,7393	5,3400	5,4145
	Zero	0,0332	0,9033	0,8963	0,8405	0,8406	3,8158	3,8217
	Random	0,0052	0,9797	0,9797	0,8305	0,8304	7,3859	7,3876
LensKit	ItemKNN	0,0204	0,9868	0,9686	0,8478	0,8372	10,6384	9,0474
	UserKNN	0,0184	0,9893	0,9820	0,8132	0,8251	11,2199	10,4818
	FunkSVD	0,0087	0,9984	0,9943	**0,8746**	**0,8564**	**12,2385**	**11,3563**
	SlopeOne	0,0009	**0,9996**	**0,9973**	0,8311	0,8367	11,8195	11,2117

Tabla 23. Resultados completos de las métricas de novedad para el conjunto de datos de Blueknow. Los algoritmos de IRG (superior) están ordenados por nDCG ascendente y los de MyMediaLite (central) y LensKit (inferior), por orden descendiente.

	Blueknow							
	Method	Accuracy						
		P@10	P@20	nDCG-n@10	nDCG-n@20	nDCG-l@10	nDCG-l@20	nDCG-e@10	nDCG-e@20
IRG	pLSA	0,0725	0,0569	0,1148	0,1421	0,1150	0,1424	0,1137	0,1406
	UserKNN (jaccard 250)	0,1204	0,0866	0,2024	0,2354	0,2027	0,2361	0,2005	0,2328
	UserKNN (cosine 400)	0,1210	0,0871	0,2063	0,2390	0,2057	0,2388	0,2053	0,2373
	ItemKNN jaccard	0,1331	0,0958	0,2218	0,2569	0,2215	0,2572	0,2202	0,2545
	ItemKNN cosine	**0,1349**	**0,0969**	**0,2270**	**0,2624**	**0,2261**	**0,2621**	**0,2259**	**0,2606**
MyMediaLite	UserKNN (cosine 280)	0,1225	0,0884	0,2037	0,2372	0,2042	0,2381	0,2014	0,2344
	BPRSLIM	0,1222	0,0889	0,2014	0,2361	0,2016	0,2370	0,1996	0,2336
	UserKNN (jaccard 230)	0,1213	0,0873	0,2031	0,2361	0,2037	0,2371	0,2010	0,2332
	ItemKNN (jaccard 10)	0,1197	0,0902	0,1880	0,2264	0,1889	0,2277	0,1860	0,2237
	ItemKNN (cosine 10)	0,1187	0,0883	0,1884	0,2244	0,1892	0,2257	0,1865	0,2217
	WRMF	0,0619	0,0480	0,0962	0,1178	0,0971	0,1189	0,0945	0,1155
	MultiCoreBPRMF	0,0608	0,0490	0,0927	0,1175	0,0938	0,1189	0,0913	0,1157
	WeightedBPRMF	0,0535	0,0430	0,0798	0,1016	0,0812	0,1034	0,0780	0,0993
	BPRMF	0,0511	0,0419	0,0768	0,0992	0,0780	0,1006	0,0750	0,0970
	SoftMarginRankingMF	0,0228	0,0207	0,0299	0,0420	0,0309	0,0431	0,0288	0,0407
	MostPopular	0,0162	0,0138	0,0234	0,0304	0,0233	0,0305	0,0231	0,0299
	LeastSquareSLIM	0,0056	0,0051	0,0120	0,0150	0,0119	0,0149	0,0121	0,0150
	Zero	0,0032	0,0039	0,0044	0,0076	0,0045	0,0077	0,0043	0,0074
	Random	0,0012	0,0011	0,0013	0,0019	0,0014	0,0020	0,0012	0,0019

Tabla 24. Resultados completos de las métricas de precisión para el conjunto de datos de Blueknow.

	Blueknow										
	Method	nDCG-e@20	Diversity								
			EILD@10	EILD@20	ERR-IA@10	ERR-IA@20	a-nDCG@10	a-nDCG@20	S-Recall@10	S-Recall@20	S-Recall-NOR@10	S-Recall-NOR@20
IRG	pLSA	0,1406	0,5653	0,6211	0,0075	0,0082	0,1431	0,1737	0,1037	0,1474	0,4263	0,5289
IRG	UserKNN (jaccard 250)	0,2328	0,4958	0,5537	0,0155	0,0163	0,2456	0,2808	0,1651	0,2087	0,4420	0,5381
IRG	UserKNN (cosine 400)	0,2373	0,4988	0,5571	0,0161	0,0169	0,2471	0,2819	0,1650	0,2081	0,4415	0,5369
IRG	ItemKNN jaccard	0,2545	0,4867	0,5494	0,0170	0,0178	0,2653	0,3026	0,1785	0,2237	0,4487	0,5504
IRG	ItemKNN cosine	**0,2606**	0,4922	0,5534	**0,0178**	**0,0187**	**0,2696**	**0,3072**	**0,1799**	**0,2261**	0,4535	0,5562
MyMediaLite	UserKNN (cosine 280)	0,2344	0,4955	0,5520	0,0151	0,0160	0,2479	0,2833	0,1667	0,2107	0,4425	0,5378
MyMediaLite	BPRSLIM	0,2336	0,5380	0,5852	0,0149	0,0157	0,2455	0,2826	0,1709	0,2161	0,4787	0,5882
MyMediaLite	UserKNN (jaccard 230)	0,2332	0,4918	0,5482	0,0152	0,0160	0,2467	0,2821	0,1657	0,2094	0,4389	0,5327
MyMediaLite	ItemKNN (jaccard 10)	0,2237	0,5247	0,5923	0,0139	0,0149	0,2310	0,2719	0,1687	0,2199	**0,4824**	**0,6148**
MyMediaLite	ItemKNN (cosine 10)	0,2217	0,5186	0,5884	0,0141	0,0149	0,2309	0,2698	0,1679	0,2168	0,4734	0,6017
MyMediaLite	MultiCoreBPRMF	0,1157	0,5415	0,5866	0,0069	0,0075	0,1213	0,1502	0,0984	0,1387	0,4368	0,5291
MyMediaLite	WRMF	0,1155	0,5305	0,5818	0,0062	0,0067	0,1230	0,1481	0,0855	0,1199	0,4131	0,5060
MyMediaLite	WeightedBPRMF	0,0993	0,5433	0,5907	0,0055	0,0060	0,1056	0,1312	0,0900	0,1270	0,4383	0,5296
MyMediaLite	BPRMF	0,0970	0,5446	0,5936	0,0049	0,0055	0,1030	0,1295	0,0854	0,1236	0,4140	0,5131
MyMediaLite	SoftMarginRankingMF	0,0407	0,6101	0,6532	0,0018	0,0021	0,0452	0,0619	0,0470	0,0755	0,4568	0,5567
MyMediaLite	MostPopular	0,0299	0,5529	0,5431	0,0016	0,0018	0,0303	0,0395	0,0246	0,0376	0,3053	0,3473
MyMediaLite	LeastSquareSLIM	0,0150	0,6579	0,7430	0,0010	0,0010	0,0159	0,0205	0,0086	0,0168	0,2656	0,4034
MyMediaLite	Zero	0,0074	0,6556	0,7435	0,0002	0,0003	0,0062	0,0109	0,0041	0,0124	0,2553	0,3977
MyMediaLite	Random	0,0019	**0,7400**	**0,7865**	0,0001	0,0001	0,0021	0,0031	0,0017	0,0037	0,3622	0,4460

Tabla 25. Resultados completos de las métricas de diversidad para el conjunto de datos de Blueknow. Los algoritmos de IRG (grupo superior) están ordenados por nDCG ascendente y los de MyMediaLite (grupo inferior), por orden descendiente.

	Blueknow							
	Method	nDCG-e@20	Novelty					
			EPC@10	EPC@20	EPD@10	EPD@20	EFN@10	EFN@20
IRG	pLSA	0,1406	0,9903	0,9915	0,6966	0,6980	6,9281	7,1698
	UserKNN (jaccard 250)	0,2328	0,9914	0,9920	0,6451	0,6517	7,3068	7,4358
	UserKNN (cosine 400)	0,2373	0,9913	0,9919	0,6482	0,6547	7,2798	7,4020
	ItemKNN jaccard	0,2545	0,9930	0,9934	0,6371	0,6451	7,7151	7,7957
	ItemKNN cosine	0,2606	0,9932	0,9936	0,6375	0,6451	7,7878	7,8755
MyMediaLite	UserKNN (cosine 280)	0,2344	0,9915	0,9921	0,6441	0,6508	7,3121	7,4546
	BPRSLIM	0,2336	0,9911	0,9918	0,6429	0,6504	7,2070	7,3412
	UserKNN (jaccard 230)	0,2332	0,9915	0,9921	0,6432	0,6502	7,3299	7,4645
	ItemKNN (jaccard 10)	0,2237	0,9956	0,9958	0,6353	0,6460	9,1551	9,1724
	ItemKNN (cosine 10)	0,2217	0,9956	0,9958	0,6359	0,6468	9,1236	9,2563
	MultiCoreBPRMF	0,1157	0,9919	0,9923	0,6655	0,6699	7,3982	7,4716
	WRMF	0,1155	0,9869	0,9883	0,6852	0,6866	6,3710	6,5633
	WeightedBPRMF	0,0993	0,9959	0,9961	0,6676	0,6722	8,3804	8,4963
	BPRMF	0,0970	0,9899	0,9903	0,6849	0,6889	6,9726	7,0492
	SoftMarginRankingMF	0,0407	0,9944	0,9944	0,6972	0,7014	7,8722	7,9139
	MostPopular	0,0299	0,9758	0,9782	0,7869	0,7784	5,3802	5,5319
	LeastSquareSLIM	0,0150	0,9941	0,9934	0,8223	0,8096	7,8207	7,6287
	Zero	0,0074	0,9943	0,9935	0,8253	0,8111	7,8424	7,6396
	Random	0,0019	0,9984	0,9984	0,8312	0,8312	10,8246	10,8274

Tabla 26. Resultados completos de las métricas de novedad para el conjunto de datos de Blueknow. Los algoritmos de IRG (grupo superior) están ordenados por nDCG ascendente y los de MyMediaLite (grupo inferior), por orden descendiente.

	Twitter							
	Method	Accuracy						
		P@10	P@20	nDCG-n@10	nDCG-n@20	nDCG-l@10	nDCG-l@20	nDCG-e@10	nDCG-e@20
IRG	Rocchio (CB)	0,0791	0,0497	0,2092	0,2282	0,2094	0,2279	0,2090	0,2283
	ItemKNN (CB)	0,0211	0,0116	0,0549	0,0563	0,0564	0,0577	0,0542	0,0556
	ItemKNN (jaccard)	0,0085	0,0049	0,0165	0,0168	0,0169	0,0172	0,0162	0,0166
	UserKNN (jaccard 60)	0,0030	0,0018	0,0054	0,0056	0,0056	0,0058	0,0053	0,0055
	UserKNN (cosine 60)	0,0030	0,0018	0,0053	0,0055	0,0055	0,0057	0,0052	0,0053
	MostPopular	0,0008	0,0007	0,0017	0,0022	0,0018	0,0023	0,0017	0,0022
	ItemKNN (CB) (norm)	0,0004	0,0003	0,0010	0,0011	0,0010	0,0011	0,0010	0,0011
	pLSA	0,0003	0,0002	0,0004	0,0005	0,0004	0,0005	0,0004	0,0005
	Random	0,0001	0,0001	0,0001	0,0002	0,0001	0,0002	0,0001	0,0002

Tabla 27. Resultados completos de las métricas de precisión para el conjunto de datos de Twitter.

Twitter													
Method	nDCG-e @20	Diversity											
		EILD @10	EILD @20	EILD* @10	EILD* @20	ERR-IA @10	ERR-IA @20	a-nDCG @10	a-nDCG @20	S-Recall @10	S-Recall @20	S-Rec-N @10	S-Rec-N @20
Rocchio (CB)	**0,2283**	0,4730	0,7390	0,8576	0,9154	**0,0468**	**0,0486**	**0,1657**	**0,1801**	**0,0059**	**0,0071**	**0,0104**	**0,0140**
ItemKNN (CB)	0,0556	0,4282	0,7144	0,8696	0,9287	0,0109	0,0110	0,0411	0,0421	0,0014	0,0016	0,0037	0,0050
ItemKNN (jaccard)	0,0166	0,7057	0,8327	0,8966	0,9430	0,0041	0,0042	0,0203	0,0215	0,0009	0,0011	0,0054	0,0065
UserKNN (jaccard 60)	0,0055	0,7442	**0,8533**	0,8960	0,9420	0,0021	0,0021	0,0096	0,0101	0,0005	0,0005	0,0023	0,0031
UserKNN (cosine 60)	0,0053	**0,7442**	0,8532	0,8960	0,9420	0,0020	0,0021	0,0092	0,0099	0,0005	0,0005	0,0023	0,0031
MostPopular	0,0022	0,0001	0,3743	**0,9000**	0,9479	0,0000	0,0001	0,0003	0,0012	0,0000	0,0002	0,0008	0,0010
ItemKNN (CB) (norm)	0,0011	0,6214	0,8221	0,8925	0,9445	0,0001	0,0002	0,0008	0,0011	0,0000	0,0001	0,0009	0,0018
pLSA	0,0005	0,5251	0,7817	0,8953	0,9460	0,0001	0,0001	0,0006	0,0007	0,0001	0,0001	0,0006	0,0014
Random	0,0002	0,4960	0,7635	0,8995	**0,9494**	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0005	0,0011

Tabla 28. Resultados completos de las métricas de diversidad para el conjunto de datos de Twitter. Los algoritmos de IRG (todos los que se muestran) están ordenados por nDCG de forma descendiente.

	Twitter									
	Method	nDCG-e@20	Novelty							
			EPC@10	EPC@20	EPD@10	EPD@20	EPD*@10	EPD*@20	EFN@10	EFN@20
IRG	Rocchio (CB)	**0,2283**	0,9999	0,9999	0,9235	0,9820	0,9928	0,9937	12,9863	13,0115
	ItemKNN (CB)	0,0556	0,9999	0,9999	0,9051	0,9837	0,9958	0,9965	13,0148	13,0149
	ItemKNN (jaccard)	0,0166	0,9998	0,9999	0,9625	0,9789	0,9993	0,9992	13,0070	13,0923
	UserKNN (jaccard 60)	0,0055	0,9999	0,9999	0,9869	0,9901	0,9994	0,9992	13,2145	13,2450
	UserKNN (cosine 60)	0,0053	0,9999	0,9999	0,9871	0,9900	0,9994	0,9992	13,2147	**13,2450**
	MostPopular	0,0022	0,9977	0,9981	**0,9883**	0,9913	**0,9994**	**0,9995**	8,7958	9,0596
	ItemKNN (CB) (norm)	0,0011	**0,9999**	**0,9999**	0,9753	0,9916	0,9985	0,9985	**13,2383**	13,2237
	pLSA	0,0005	0,9997	0,9998	0,8964	**0,9923**	0,9993	0,9993	12,5692	12,8747
	Random	0,0002	0,9999	0,9999	0,9448	0,9893	0,9994	0,9994	13,0805	13,0781

Tabla 29. Resultados completos de las métricas de novedad para el conjunto de datos de Twitter. Los algoritmos de IRG (todos los que se muestran) están ordenados por nDCG de forma descendiente.