Universidad Autónoma de Madrid
Escuela Politécnica Superior


Trabajo de Fin de Máster

# EVALUACIÓN E INTEGRACIÓN DE TOMA DE DECISIONES EN SISTEMAS DE RECOMENDACIÓN

Máster en Ingeniería Informática

Rus María Mesas Jávega
Junio 2017

# EVALUACIÓN E INTEGRACIÓN DE TOMA DE DECISIONES EN SISTEMAS DE RECOMENDACIÓN

AUTOR: Rus María Mesas Jávega
TUTOR: Alejandro Bellogín Kouki
PONENTE: Pablo Castells Azpilicueta

Grupo de la EPS: Information Retrieval Group (IRG)
Dpto. de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio 2017

# Resumen


## Resumen

Los sistemas de recomendación tienen por objeto sugerir elementos relevantes a los usuarios, aunque recientemente se han considerado otros objetivos como incrementar la diversidad o la novedad de las recomendaciones, aumentar la confianza en el sistema, etc.

En este trabajo se realiza una investigación sobre la confianza pero desde la perspectiva del sistema: cuál es la confianza que el sistema tiene en sus propias recomendaciones; en concreto, se centra en diferentes métodos para introducir una cierta toma de decisión en algoritmos de recomendación para decidir cuándo se debería recomendar un ítem. Nuestra hipótesis es que, en ocasiones, es mejor no recomendar que fallar, pues fallar puede disminuir la confianza del usuario en el sistema. Con la toma de decisiones se puede aumentar el rendimiento en términos de precisión y confianza del usuario en el sistema, a costa de, probablemente, reducir otras métricas como la cobertura.

A diferencia de otros trabajos de la literatura, nuestra aproximación no explora o analiza los datos de entrada sino que considera aspectos intrínsecos del algoritmo o de los componentes usados durante la predicción. Se propone una taxonomía de técnicas que pueden ser aplicadas a algunas familias de recomendadores para poder incluir mecanismos con los que decidir cuándo recomendar. En particular, se explora la incertidumbre en la predicción para un algoritmo de factorización de matrices probabilístico y la familia de $K$ Vecinos Próximos, el soporte de la predicción del algoritmo de $K$ Vecinos Próximos, y un método independiente del algoritmo.

Por otro lado, se estudia cómo evoluciona el rendimiento de un algoritmo cuando decide no recomendar en algunas situaciones. Si la decisión de evitar una recomendación es razonable, es decir, no es aleatoria sino que está relacionada con el usuario o ítem, se espera que la precisión mejore a costa del descenso de otras dimensiones como la cobertura, la novedad o la diversidad.

El equilibrio entre precisión y cobertura es fundamental, ya que es posible lograr una precisión muy alta recomendando sólo un elemento a un único usuario, lo cual no sería una recomendación muy útil. Debido a esto, se exploran también a lo largo de este proyecto algunas técnicas para combinar métricas de precisión y cobertura, un problema abierto en el área. Por otro lado, se propone también una familia de métricas (*correctness*) basadas en la asunción de que es mejor evitar una recomendación que recomendar un ítem no relevante.

En resumen, las contribuciones de este trabajo son dos: una taxonomía de técnicas que pueden aplicarse a algunas familias de recomendadores que permiten incluir mecanismos para decidir si una recomendación debe ser generada, y una primera exploración a la combinación de métricas de evaluación, principalmente centrado en medidas de precisión y cobertura.



## Palabras Clave

Recomendación, Toma de decisión, Evaluacion, Precisión, Cobertura, Novedad, Diversidad

## Abstract

Recommender Systems aim at suggesting as many relevant items to the users as possible, although other goals are being considered recently: increasing the diversity or novelty of the recommendations, increasing the confidence of the user in the system, etc.

In this work, we investigate about confidence but from the perspective of the system: what is the confidence a system has in its own recommendations; more specifically, we focus on different methods to embed awareness into the recommendation algorithms about deciding whether an item should be suggested. Sometimes it is better not to recommend that fail because failure can decrease user confidence in the system. In this way, we hypothesise the system would only show the more reliable suggestions, hence, increasing the performance of such recommendations, at the expense of, presumably, reducing the number of potential recommendations.

Different from other works in the literature, our approaches do not exploit or analyse the input data but intrinsic aspects of the recommendation algorithms or of the components used during prediction are considered. We propose a taxonomy of techniques that can be applied to some families of recommender systems allowing to include mechanisms to decide if a recommendation should be generated. More specifically, we exploit the uncertainty in the prediction score for a probabilistic matrix factorisation algorithm and the family of nearest-neighbour algorithms, the support of the prediction score for nearest-neighbour algorithms, and a method independent of the algorithm.

We study how the performance of a recommendation algorithm evolves when it decides not to recommend in some situations. If the decision of avoiding a recommendation is sensible – i.e., not random but related to the information available to the system about the target user or item –, the performance is expected to improve at the expense of other quality dimensions such as coverage, novelty, or diversity. This balance is critical, since it is possible to achieve a very high precision recommending only one item to a unique user, which would not be a very useful recommender. Because of this, on the one hand, we explore some techniques to combine precision and coverage metrics, an open problem in the area. On the other hand, a family of metrics (correctness) based on the assumption that it is better to avoid a recommendation rather than providing a bad recommendation has been proposed.

In summary, the contributions of this paper are twofold: a taxonomy of techniques that can be applied to some families of recommender systems allowing to include mechanisms to decide if a recommendation should be generated, and a first exploration to the combination of evaluation metrics, mostly focused on measures for precision and coverage.



## Key words

Recommendation, Decision-aware, Evaluation, Accuracy, Coverage, Novelty, Diversity

# Agradecimientos

A mi madre y mejor amiga, ejemplo de lucha, valentía y fortaleza. Allá donde estés espero que te sientas orgullosa.

# Índice general

# Índice de figuras

# Índice de tablas

# 1 Introducción

A lo largo de este primer capítulo se detallará la motivación de este proyecto y sus principales objetivos. Finalmente, se describirá la estructura del presente documento.

## 1.1. Motivación

En los últimos años, el incremento masivo de la información en la Web y el crecimiento de cada vez más servicios online han provocado que surjan nuevas necesidades como la recomendación de productos. Muchas páginas web tienen un catálogo inmenso de productos ofertados que hacen que la elección de uno de ellos para el cliente sea una ardua tarea. Este problema es extensible a cualquier sistema, como una red social, en la que el usuario puede elegir a quién seguir, o un videoclub, donde el cliente desea elegir una película de acuerdo a sus gustos. Por ello, a pesar de que el área de los Sistemas de Recomendación es relativamente reciente, estos sistemas están cada vez más presentes en entornos familiares. Éstos tienen como objetivo sugerir nuevos ítems a los usuarios de un entorno basándose en las interacciones que dichos usuarios tuvieron con él en el pasado. Con ello, no sólo ayudan al usuario en su elección sino que mejoran la accesibilidad del sistema, aumentan la satisfacción del cliente y sirven como herramienta para intentar que todos los ítems del catálogo sean visibles al menos por los usuarios más potenciales, incrementando, probablemente, los beneficios económicos.

En la actualidad es un área abierta a la investigación con algoritmos cada vez más sofisticados capaces de manejar grandes volúmenes de datos de manera rápida y eficiente. Su principal objetivo es sugerir ítems relevantes a los usuarios, aunque otras dimensiones se estudian también habitualmente, como la novedad, la diversidad, la confianza, etc. En este proyecto, con el objetivo de incrementar la cantidad de ítems relevantes presentados al usuario, se analizará cómo el sistema puede medir la confianza en sus propias recomendaciones para, de este modo, tener la capacidad de tomar decisiones sobre si un ítem debe ser o no recomendado. Nuestra hipótesis es que, en ocasiones, es mejor no recomendar que fallar, pues fallar puede disminuir la confianza del usuario en las recomendaciones del sistema. El objetivo de la toma de decisión propuesta es que sólo se recomienden ítems basados en datos consistentes y de confianza.

El concepto de confianza en recomendación ha sido aplicado en diferentes aspectos del campo, por ejemplo basado en los datos de entrada del algoritmo. En este trabajo se estudia la

confianza pero desde el punto de vista del sistema considerando aspectos intrínsecos del algoritmo, una de las principales aportaciones del proyecto. Así, se propondrá una taxonomía de técnicas de toma de decisión que pueden ser aplicadas a algunas familias de recomendadores. En concreto, se explorará la incertidumbre de la predicción en un algoritmo de factorización de matrices y en la familia de $K$ Vecinos Próximos (KNN), el soporte de la predicción de esta última familia citada y un método independiene del algoritmo.

El objetivo de que el sistema muestre sólo aquellas recomendaciones generadas a partir de datos fiables y consistentes puede conllevar un aumento de la cantidad de ítems relevantes mostrados al usuario y un aumento de su confianza en las recomendaciones pero también puede influir negativamente en otras dimensiones. Como consecuencia, el segundo objetivo de este proyecto es estudiar cómo influye esto en distintas métricas de evaluación como métricas de precisión, cobertura, diversidad o novedad, métricas que serán estudiadas y explicadas en profundidad también a lo largo de este proyecto.

Al retornar sólo recomendaciones fiables, de confianza, se puede aumentar la precisión del recomendador disminuyendo el número de usuarios a los que se les recomienda y/o el número de ítems recomendados, siendo difícil elegir el recomendador que compense ambas características. Así, otra de las novedades de este trabajo es el estudio y desarrollo de nuevas técnicas y métricas de evaluación para una evaluación más completa de estos nuevos sistemas.

## 1.2. Objetivos

Como ya se ha introducido anteriormente, el proyecto tiene tres objetivos principales:

1. Dotar a los sistemas de recomendación de la capacidad de tomar decisiones sobre sus recomendaciones. Para ello se propondrán diversas estrategias para que el sistema retorne sólo aquellas sugerencias basadas en datos fiables.

2. Estudiar la influencia de la incorporación de la toma de decisiones a un sistema de recomendación en términos de precisión, cobertura, popularidad o diversidad.

3. Propuesta de nuevas métricas que ayuden a evaluar los nuevos algoritmos propuestos.

Con este proyecto se pretende poner en práctica muchos de los conocimientos aprendidos a lo largo de este máster y, en particular, desarrollar y profundizar sobre otros nuevos conocimientos de un área no estudiada a lo largo de mi carrera y máster universitario, el área de los Sistemas de Recomendación.

## 1.3. Estructura del trabajo

El presente documento está dividido en 6 capítulos, los cuales se detallan a continuación:

- **Capítulo 1. Introducción:** Descripción de la motivación y objetivos principales del proyecto.

- **Capítulo 2. Estado del Arte:** Repaso de los principales conceptos del área de los Sistemas de Recomendación y estudio de trabajos realizados hasta el momento en este área. Se hace hincapié principalmente en los algoritmos de filtrado colaborativo y en métricas de evaluación de precisión, cobertura y brevemente sobre diversidad y novedad.

- **Capítulo 3. Toma de Decisiones en Algoritmos de Recomendación:** Detalle de las diversas estrategias seguidas para dotar a distintos algoritmos de recomendación de la posibilidad de decidir cuándo recomendar un ítem a un usuario.
- **Capítulo 4. Evaluación de Sistemas de Recomendación que Toman Decisiones:** Estudio de cómo crear nuevas métricas que evaluen los algoritmos propuestos en el capítulo anterior. Métricas que combinen precisión y cobertura o que premien no recomendar a recomendar un ítem no relevante.
- **Capítulo 5. Experimentos:** En este capítulo se analiza y comparan los efectos que la toma de decisiones tiene en términos de precisión, cobertura, novedad y diversidad. Se realizarán experimentos con distintos conjuntos de datos para poder obtener conclusiones consistentes y objetivas que no dependan de los datos utilizados.
- **Capítulo 6. Conclusiones y Trabajo Futuro:** En este último capítulo se resumirán las contribuciones de este proyecto y se propondrán nuevas líneas de investigación con las que continuar en el futuro.

# 2
# Estado del Arte

A lo largo de este capítulo se desarrollará la base teórica sobre la que se sustenta este trabajo. En concreto, se explicará en primer lugar en qué consisten los sistemas de recomendación y algunos de los algoritmos más conocidos y utilizados. Para finalizar, se estudiará cómo evaluar dichos sistemas para poder seleccionar aquel algoritmo que mejor se ajuste al entorno en el que se quiere aplicar.

## 2.1. Introducción a los sistemas de recomendación

Los sistemas de recomendación tienen como objetivo sugerir nuevos ítems a los usuarios de un entorno basándose en las interacciones que dichos usuarios tuvieron con él en el pasado. Se considera ítem cualquier objeto a recomendar: una película, una canción, un viaje, una bicicleta e incluso otro usuario. En otras palabras, son herramientas software y técnicas que proporcionan sugerencias personalizadas sobre qué ítems podrían ser útiles para un usuario [26]. Con estos sistemas se ayuda al usuario a tomar decisiones como cuál será la siguiente canción que quiere escuchar, qué producto le interesa comprar o a qué usuario de una red social le gustaría seguir, entre otros muchos ejemplos.

El área de los Sistemas de Recomendación lleva desarrollándose desde principio de los 90 y, actualmente, continúa siendo un área abierta a la investigación, con algoritmos cada vez más sofisticados y eficientes, capaces de manejar grandes volúmenes de datos. Estos algoritmos pueden dividirse principalmente en dos categorías: no personalizados, como por ejemplo recomendar los ítems más populares del sistema, o personalizados. En general, los algoritmos que personalizan las recomendaciones a cada usuario siguen dos estrategias:

- **Filtrado colaborativo**: Es una de las estrategias más populares para recomendar. Estos algoritmos se basan en las puntuaciones (*ratings*) o interacciones que los usuarios han mostrado con los ítems del sistema. Su idea es que el rating que un usuario daría a un nuevo ítem probablemente sea similar al de otro usuario que haya dado ratings a ítems de manera similar [23, 18].
- **Basado en contenido:** Estos sistemas aprenden a recomendar ítems similares a otros que gustaron al usuario en el pasado. Para medir la similitud de dos ítems se utilizan los

atributos de los ítems. Por ejemplo, podría considerarse que una película es similar a otra si son del mismo género cinematográfico y están dirigidas por el mismo director [10].

Aunque estas dos estrategias son las más extendidas, existen otros tipos de recomendadores, como los demográficos, los sociales, los basados en conocimiento y otros que combinan diversas técnicas y estrategias, conocidos como híbridos [26].

En general, con respecto a los datos de entrada, los algoritmos de recomendación parten de la información referente a tres tipos de objetos:

- **Usuarios** que reciben la recomendación. La información necesaria de los usuarios puede ser más o menos detallada según el tipo de algoritmo. Existen algoritmos sencillos que sólo se basan en qué interacción tuvo el usuario con qué ítems y otros que basan su recomendación en sus características, por ejemplo en su género, edad, localización, etc.
- **Ítems** u objetos con los que interaccionan los usuarios y que, generalmente, es lo que se recomiendan al usuario. Al igual que en el caso de los usuarios, la información necesaria referente a los ítems del sistema es más o menos exhaustiva según el algoritmo y el tipo de ítem. Por ejemplo, un recomendador de películas podría utilizar el título, género cinematográfico, fecha del estreno, director, actores, etc.
- **Interacciones** de los usuarios con los ítems del sistema. En muchos casos estas interacciones se representan como el rating o puntuación que un usuario dio a un ítem pero también pueden ser implícitos, donde no se tiene constancia de las preferencias negativas (por ejemplo: el número de veces que ha escuchado una canción), o unarios, donde sólo están marcados los ítems observados por los usuarios, sin que se puedan distinguir qué ítems gustan más que otros (por ejemplo, los *likes* de Facebook).

Debido a que el proyecto se centra en métodos de filtrado colaborativo en la siguiente sección se estudiarán en detalle los aspectos y algoritmos más importantes de este tipo.

## 2.2. Filtrado colaborativo

Como ya se ha mencionado anteriormente, esta popular estrategia se basa en la idea intuitiva de que si dos usuarios votaron de manera similar a varios ítems es probable que opinen parecido sobre un nuevo ítem, es decir, se basan en los ratings que los usuarios han dado a los ítems del sistema.

Para comprender de manera más sencilla e intuitiva cuál es la información de la que se parte en estos algoritmos y cuál es el objetivo, en la Figura 2.1 se muestra un ejemplo sencillo de la interacción de 7 usuarios con 9 ítems. La imagen recoge lo que se conoce como matriz de ratings, $R$, una matriz $V \times J$ donde $V$ es el número de usuarios y $J$ el número de ítems, en la que el valor $r_{ui}$ es el rating o voto que el usuario $u$-ésimo dio al ítem $i$-ésimo. En este caso, los usuarios podían votar entre 1 y 5, siendo 1 la peor votación y 5 la mejor. Se puede ver, por ejemplo, que el usuario 3 puntuó con un 2 al ítem 2 y que no ha puntuado todavía al ítem 1. El objetivo de los sistemas de recomendación es predecir los valores ausentes de esta matriz, es decir, estimar qué rating daría un usuario a un ítem al que todavía no ha votado. De este modo, tras estimar todos los valores se podrá hacer un ranking de estas predicciones y recomendar los ítems que han recibido ratings más altos a lo largo de la predicción.

Los dos tipos de algoritmos más utilizados dentro del filtrado colaborativo son los algoritmos de $K$ Vecinos Próximos (KNN: *K Nearest Neighbours*) y los de factorización de matrices, ambos se explicarán en detalle a lo largo de esta sección.

| | I1 | I2 | I3 | I4 | I5 | I6 | I7 | I8 | I9 |
|---|---|---|---|---|---|---|---|---|---|
| U1 | 3 | | | | 5 | | | | |
| U2 | | | 5 | 1 | 4 | | | | |
| U3 | | 2 | | | | | 4 | 5 | |
| U4 | | | | 4 | 3 | 2 | | | |
| U5 | | | 5 | | | | | | |
| U6 | | | | | | | | | 1 |
| U7 | | | | | | | 3 | 4 | 5 |

Figura 2.1: Ejemplo de una matriz $R$ de ratings.

### 2.2.1. $K$ Vecinos Próximos ($K$ Nearest Neighbours)

Los algoritmos de $K$ Vecinos Próximos basan su estimación directamente en los ratings de los usuarios siguiendo dos posibles estrategias: basado en usuario y basado en ítem.

#### 2.2.1.1. Basado en usuario

Para calcular el rating $r_{ui}$ que el usuario $u$ dará al ítem $i$ estos algoritmos utilizan los ratings de los $k$ usuarios más similares a $u$. El conjunto de $k$ usuarios similares a dicho usuario que votaron al ítem se denomina como vecindario de $u$, $N_i(u)$. Para calcular cómo de similares son dos usuarios es necesaria una función de similitud $w_{uv} : \mathcal{U} \mapsto \mathcal{U}$ que, dados dos usuarios $u$ y $v$, calcule la similitud entre ellos. Así, una vez calculadas todas las similitudes del usuario $u$ con el resto de usuarios, se pueden seleccionar los $k$ usuarios con similitudes más altas y calcular el nuevo rating. Este cálculo se realiza típicamente a través de una media ponderada de los ratings de los usuarios del vecindario usando las similitudes $w_{uv}$ como pesos [23], tal y como muestra la Ecuación 2.1.

$$\hat{r}_{ui} = \frac{\sum_{v \epsilon N_i(u)} w_{uv} r_{vi}}{\sum_{v \epsilon N_i(u)} |w_{uv}|} \tag{2.1}$$

Debido a que cada usuario tiene su propia forma de puntuar (algunos usuarios tienden a dar puntuaciones más altas que otros) se suelen normalizar los ratings antes de utilizarlos en los cálculos centrando la media o usando la normalización Z-score [23].

#### 2.2.1.2. Basado en ítems

Normalmente el número de usuarios es mayor que el de ítems y crece mucho más rápido haciendo peligrar la escalabilidad del filtrado colaborativo basado en usuarios. Para solventar este problema surge el filtrado colaborativo basado en ítems, uno de los más extendidos actualmente, principalmente en el comercio electrónico [20]. Fue descrito en primer lugar por Sarwar et al. [29] y Karypis [16] aunque la primera versión fue usada por Amazon.com [20]. La idea es basarse en las puntuaciones que el usuario $u$ dio a ítems parecidos al ítem $i$ para estimar el rating $r_{ui}$. Por ello, de nuevo es necesaria una función de similitud que, dados dos ítems $i$ y $j$ retorne la similitud de ambos, $w_{ij}$. Dos ítems son parecidos si muchos usuarios del sistema los han puntuado de manera similar. Un ejemplo sencillo: se quiere conocer la puntuación que $u$ daría al ítem $i$ y se observa que los ítems $j$ y $l$, que ya puntuó $u$, han sido puntuados de forma similar a $i$ por otros usuarios. Se puede suponer que el rating de $i$ será similar al rating que $u$ dio a los ítems $j$ y $l$. Tras calcular el vecindario del ítem $i$ se puede estimar el rating $r_{ui}$ utilizando la Ecuación 2.2 [23].

$$\hat{r}_{ui} = \frac{\sum_{j \epsilon N_u(i)} w_{ij} r_{uj}}{\sum_{i \epsilon N_u(i)} |w_{ij}|} \tag{2.2}$$

De nuevo se pueden utilizar las normalizaciones mencionadas en el KNN basado en usuarios.

#### 2.2.1.3. Algunas métricas de similitud

En el algoritmo de $K$ Vecinos Próximos, tanto el basado en usuarios como el basado en ítems, la elección de la función de similitud es crítica ya que es utilizada para elegir el vecindario y forma parte del cálculo de la predicción. A continuación se citarán algunas de las similitudes más utilizadas:

- **Similitud Coseno:** Cada usuario es representado por un vector de dimensión $J$ (número de ítems) en el que la componente $i$ es el rating que el usuario dio al $i$-ésimo ítem, o 0 si no lo ha puntuado. Así, la similitud de dos usuarios se mide como la distancia coseno entre los dos vectores que los representan. Del mismo modo, si se aplica a ítems, cada ítem será un vector de longitud $V$ (número de usuarios) donde la componente $u$ representa el rating que el usuario $u$-ésimo dio a dicho ítem y 0 si aún no lo ha puntuado [23, 11]. La Ecuación 2.3 muestra cómo se calcula la similitud coseno entre dos usuarios $u$ y $v$:

$$cos(u, v) = \frac{u^\top v}{\|u\|\|v\|} = \frac{\sum_{i \epsilon \mathcal{J}_u} r_{ui} r_{vi}}{\sqrt{\sum_{i \epsilon \mathcal{J}_u} r_{ui}^2} \sqrt{\sum_{j \epsilon \mathcal{J}_v} r_{vj}^2}} \tag{2.3}$$

  donde $u = (r_{u1}, r_{u2}, ..., r_{uJ})$, $v = (r_{v1}, r_{v2}, ..., r_{vJ})$, $\mathcal{J}_u$ representa el conjunto de ítems puntuados por $u$, $\mathcal{J}_v$ los puntuados por $v$ y $\mathcal{J}_{uv}$ los puntuados por ambos.

  La misma idea se aplica para calcular la similitud entre dos ítems $i$ y $j$ en la Ecuación 2.4:

$$cos(i, j) = \frac{i^\top j}{\|i\|\|j\|} = \frac{\sum_{u \epsilon \mathcal{U}_i} r_{ui} r_{uj}}{\sqrt{\sum_{u \epsilon \mathcal{U}_i} r_{ui}^2} \sqrt{\sum_{v \epsilon \mathcal{U}_j} r_{vj}^2}} \tag{2.4}$$

  con $i = (r_{1i}, r_{2i}, ..., r_{Ii})$, $j = (r_{1j}, r_{2j}, ..., r_{Ij})$, $\mathcal{U}_i$ representa el conjunto de usuarios que han puntuado al ítem $i$, $\mathcal{U}_j$ los que han puntuado a $j$ y $\mathcal{U}_{ij}$ los que han puntuado a ambos.

- **Correlación de Pearson:** Esta similitud mide, en el caso de la técnica basada en ítems, la tendencia de los usuarios a puntuar dos ítems de manera similar, mientras que, en el caso de basado en usuarios, mide la tendencia de que los ítems sean puntuados de manera similar por dos usuarios [23, 11]. Utilizando la misma notación utilizada en la explicación de la similitud coseno, la Ecuación 2.5 y la Ecuación 2.6 muestran cómo se realiza el cálculo para la técnica basada en usuarios y basada en ítems, respectivamente, donde $\overline{r}_u$ y $\overline{r}_i$ denotan la puntuación promedio de un usuario $u$ y de un ítem $i$.

$$PC(u, v) = \frac{\sum_{i \epsilon \mathcal{J}_{uv}} (r_{uv} - \overline{r}_u)(r_{vi} - \overline{r}_v)}{\sqrt{\sum_{i \epsilon \mathcal{J}_{uv}} (r_{ui} - \overline{r}_u)^2 \sum_{i \epsilon \mathcal{J}_{uv}} (r_{vi} - \overline{r}_v)^2}} \tag{2.5}$$

$$PC(i, j) = \frac{\sum_{u \epsilon \mathcal{U}_{ij}} (r_{ui} - \overline{r}_i)(r_{uj} - \overline{r}_j)}{\sqrt{\sum_{i \epsilon \mathcal{U}_{ij}} (r_{ui} - \overline{r}_i)^2 \sum_{u \epsilon \mathcal{U}_{ij}} (r_{uj} - \overline{r}_j)^2}} \tag{2.6}$$

### 2.2.2. Factorización de matrices

Tras explicar los métodos basados en $K$ Vecinos Próximos se va a explicar otra familia importante de algoritmos dentro del filtrado colaborativo: los métodos de factores latentes. La principal idea de estos últimos es que tanto los usuarios como los ítems pueden ser representados por un conjunto pequeño de factores no observados o latentes.

La factorización de matrices es uno de los métodos más populares para el cálculo de factores latentes. Estos métodos transforman tanto a usuarios como ítems a un espacio de factores latentes de dimensión $D$ y las interacciones usuario-item (ratings en el caso de los recomendadores) se modelan como el producto escalar en ese espacio [17, 22]. En otras palabras, el objetivo es descomponer la matriz inicial de ratings $R$ en el producto:

$$R \approx R' = UI^\top \tag{2.7}$$

donde $U$ es una matriz de tamaño $V \times D$ (con $D << V$) de coeficientes de los usuarios e $I$ la matriz $J \times D$ ($D << J$) de coeficientes de los ítems. Así, el usuario $u$ está representado por la fila $u$-ésima de la matriz $U$, el ítem $i$ está representado por la fila $i$-ésima de la matriz $I$ y el rating que un usuario $u$ da al ítem $i$ está modelado por el producto de estos dos vectores, como se puede ver en la Ecuación 2.8.

$$r_{ui} = u^\top \cdot i \tag{2.8}$$

El principal problema para estos modelos es encontrar el mejor ranking de los $D$ factores que hagan que $R'$ sea lo más similar posible a la matriz $R$ real. Esto se podría traducir en un problema de minimización de la función objetivo de la Ecuación 2.9 [19].

$$F(U, I) = \sum_{u,i} (u^\top \cdot i - r_{ui})^2 \tag{2.9}$$

Cabe tener en cuenta que no se conocen todas las entradas de la matriz $R$, ya que no todos los usuarios han puntuado todos los ítems, y si así fuera no tendría sentido nuestro objetivo de predecir ratings.

La descomposición en valores singulares (SVD: *Singular Value Descomposition*) es uno de los algoritmos para la identificación de factores latentes más utilizados tanto en Recuperación de Información como en Sistemas de Recomendación [17]. El problema que presenta dentro del área de Recomendación, y en concreto en el filtrado colaborativo, es que la matriz a factorizar es la matriz de ratings, una matriz muy dispersa. La gran cantidad de valores ausentes incrementa las dificultades para la aplicación de este algoritmo y su aplicación sobre sólo los valores conocidos aumenta la posibilidad de sobreajuste [18, 19]

Existen diferentes formas de reconstruir la matriz y evitar que existan valores ausentes. Una de ellas la proponen Srebro y Jaakkola en [31], los cuales se basan en el algoritmo EM y rellena los valores ausentes con las predicciones de la reconstrucción de menor rango de la iteración anterior.

Jin y Whye proponen en [19] su técnica de Inferencia Bayesiana Variacional (*Variational Bayesian Inference*) para evitar el sobreajuste de SVD. En este método los prioris son introducidos y todos los parámetros son estimados usando inferencia variacional. Debido a que este algoritmo es uno de los utilizados y modificados en el proyecto, en la siguiente subsección se va a explicar con detalle.

Una versión probabilística, PMF (*Probabilistic Matrix Factorization*), propuesto por Ruslan Salakhutdinov y Andriy Mnih [28], es otro método de factorización de matrices basado en un modelo lineal probabilístico con ruido Gaussiano y el método MAP. Su variante, Bayesian Probabilistic Matrix Factorization o BPMF, propuesto por los mismos autores, utiliza métodos de MCMC (*Markov chain Monte Carlo*) para llevar a cabo la inferencia aproximada [27].

#### 2.2.2.1. Variational Bayesian

Dada la matriz de ratings $R$ de dimensión $V \times J$ ($V$ es el número de usuarios y $J$ el número de ítems), en la que el valor $r_{ui}$ es el rating o voto que el usuario $u$ ha dado al ítem $i$, se desea encontrar la descomposición:

$$R = UI^\top \tag{2.10}$$

tal que $U \in \mathbb{R}^{V \times D}$ (con $D << V$) es la matriz de factores latentes de los usuarios e $I \in \mathbb{R}^{J \times D}$ (con $D << J$) es la matriz de factores latentes de los ítems.

El objetivo es minimizar la función objetivo de la Ecuación 2.9 usando un modelo probabilistico donde $R$ son las observaciones, $U$ e $I$ los parámetros y se introduce una cierta regularización a través de prioris de estos parámetros.

En este caso, $u$ corresponde a la $u$-ésima fila de la matriz $U$ e $i$ a la $i$-ésima fila de la matriz $I$. A partir de ahora, a pesar de ser las filas de una matriz, se considerará que los vectores son vectores columna.

La primera hipótesis del modelo es que la probabilidad de la puntuación o rating, dadas las matrices $U$ e $I$, sigue una normal de media $u^\top i$ y varianza $\tau^2$:

$$P(r_{ui}|U, I) = N(u^\top \cdot i, \tau^2) \tag{2.11}$$

Además, las entradas de las matrices $U$ e $I$ siguen distribuciones independientes, en concreto, $u_l \sim N(0, \sigma_l^2)$ y $i_l \sim N(0, \rho_l^2)$. Como consecuencia, las funciones de densidad de $U$ e $I$ vienen dadas por la Ecuación 2.12.

$$P(U) = \prod_{u=1}^{V} \prod_{l=1}^{D} N(u_l|0, \sigma_l^2) \qquad P(I) = \prod_{i=1}^{J} \prod_{l=1}^{D} N(i_l|0, \rho_l^2) \tag{2.12}$$

El objetivo es calcular o aproximar la probabilidad a posteriori $P(U, I \mid R)$, para, de este modo, poder calcular una distribución predictiva de los ratings dada la matriz de observaciones, Ecuación 2.13

$$P(\hat{r}_{ui}|R) = \int P(\hat{r}_{ui}|u, i, \tau^2) P(U, I \mid R) \tag{2.13}$$

Ya que calcular la probabilidad a posteriori, $P(U, I \mid R)$, es inviable, el algoritmo que se propone tiene como objetivo aproximarla utilizando inferencia bayesiana, más concretamente el método conocido como *Mean-Field Variational Inference*.

Antes de continuar con el algoritmo, a lo largo de la siguiente subsección se va a explicar brevemente en qué consiste la inferencia bayesiana, ya que es la base de este método.

***Inferencia Bayesiana***

La inferencia bayesiana es un método que aproxima una distribución a posteriori de un conjunto de variables no observadas $Z = \{Z_1, \cdots, Z_D\}$ conociendo un conjunto de información $R$, $P(Z \mid R)$, a través de una distribución variacional $Q(Z)$ [1]. Para ello, busca la función $Q(Z)$

que minimiza la función de disimilitud $d(Q, P)$. En el caso concreto del método *Mean-Field Variational Inference*, dicha función de disimilitud viene dada por la divergencia Kullback-Leibler entre $P$ y $Q$ [3]:

$$D_{KL}(Q \mid\mid P) = \sum_{Z} Q(Z) \log \frac{Q(Z)}{P(Z \mid R)} = \sum_{Z} Q(Z) \log \frac{Q(Z)}{P(Z, R)} + \log P(R) \tag{2.14}$$

donde se ha usado que $P(Z \mid R) = P(Z, R)/P(R)$.

Despejando $\log P(R)$ de la Ecuación (2.14) se llega a la Ecuación 2.15

$$\log P(R) = D_{KL}(Q \mid\mid P) - \sum_{Z} Q(Z) \log \frac{Q(Z)}{P(Z, R)} = D_{KL}(Q \mid\mid P) + \mathcal{F}(Q) \tag{2.15}$$

Dejando fija la cantidad $\log P(R)$ en (2.15) basta con maximizar $\mathcal{F}(Q)$ para minimizar $D_{KL}(Q \mid\mid P)$.

$\mathcal{F}(Q)$ es denominada como *(negative) variational free energy*, ya que se puede escribir como la suma de la entropía de Q y una energía (Ecuación 2.16).

$$\begin{aligned} \mathcal{F}(Q) &= -\sum_{Z} Q(Z) \log Q(Z) + \sum_{Z} Q(Z) \log P(Z, R) \\ &= H(Q) + \mathbb{E}[\log P(Z, R)] = \mathbb{E}_{Q(z)}[\log P(Z, R) - \log Q(Z)] \end{aligned} \tag{2.16}$$

***Inferencia Bayesiana en el algoritmo Variational Bayesian***

Aplicando el método de *Mean-Field Variational Inference*, explicado en la sección anterior, con $Z = \{U, I\}$ las variables no observadas y la matriz de ratings $R$ la información conocida, el objetivo es estimar una función de distribución variacional $Q(U, I)$ que aproxime la distribución $P(U, I \mid R)$.

Para ello, el objetivo es maximizar la conocida *(negative) variational free energy* dada por la Ecuación 2.17.

$$\begin{aligned} \mathcal{F}(Q(U, I)) &= \mathbb{E}_{Q(U,I)}[\log P(U, I, R) - \log Q(U, I)] \\ &= -\sum Q(U, I)(\log Q(U, I) - \log P(U, I, R)) \end{aligned} \tag{2.17}$$

En la práctica es intratable maximizar $F(U, I)$ y, por ello, este algoritmo aplica una nueva simplificación considerando que $Q(U, I) = Q(U)Q(I)$. Con esto se llega a la ecuación definitiva de $\mathcal{F}(Q(U)Q(I))$, la Ecuación 2.18, tal y como se describe en [19].

$$\begin{aligned} \mathcal{F}(Q(U)Q(I)) &= \mathbb{E}_{Q(U)Q(I)}[\log P(U, I, R) - \log Q(U, I)] \\ &= \mathbb{E}_{Q(U)Q(I)} \left[\log \frac{P(R \mid U, I)}{P(U, I)}\right] + H\left(Q(U, I)\right) \\ &= \mathbb{E}_{Q(U)Q(I)} \left[\log P(R \mid U, I) - \log P(U) - \log P(I)\right] + H(Q(U, I)) \\ &= -\frac{K}{2} \log(2\pi\tau^2) - \frac{1}{2} \sum_{(u,i)} \frac{\mathbb{E}_{Q(U)Q(I)}[(r_{ui} - u^\top i)^2]}{\tau^2} \\ &- \frac{V}{2} \sum_{l=1}^{D} \log(2\pi\sigma_l^2) - \frac{1}{2} \sum_{l=1}^{D} \frac{\sum_{u=1}^{V} \mathbb{E}_{Q(U)}[u_l^2]}{\sigma_l^2} \\ &- \frac{J}{2} \sum_{l=1}^{D} \log(2\pi\rho_l^2) - \frac{1}{2} \sum_{l=1}^{D} \frac{\sum_{i=1}^{J} \mathbb{E}_{Q(I)}[i_l^2]}{\rho_l^2} \\ &+ H(Q(U, I)) \end{aligned} \tag{2.18}$$

donde $K$ es el número de entradas observadas en $R$, es decir, el número de ratings o interacciones conocidas en el sistema en el momento de entrenar el modelo.

Maximizando $\mathcal{F}(Q(U)Q(I)$ respecto a $Q(U)$ manteniendo fija $Q(I)$ y viceversa se llega a las distribuciones de los ítems y de los usuarios. Las Ecuaciones (2.19) y (2.20) describen la matriz de covarianzas y de la media del usuario $u$ en $Q(U)$, respectivamente.

$$\phi_u = \left( \begin{pmatrix} \frac{1}{\sigma_1^2} & & 0 \\ & \ddots & \\ 0 & & \frac{1}{\sigma_D^2} \end{pmatrix} + \sum_{i \in N(u)} \frac{\psi_i + \bar{i}^\top \bar{i}}{\tau^2} \right)^{-1} \tag{2.19}$$

$$\overline{u} = \phi_u \left( \sum_{i \in N(u)} \frac{r_{ui} \bar{i}}{\tau^2} \right) \tag{2.20}$$

donde $N(u)$ son los ítems observados por el usuario $u$, es decir, los identificadores $i$ correspondientes tal que $r_{ui}$ se ha observado en la matriz.

Y, del mismo modo, las Ecuaciones (2.21) y (2.22), son las ecuaciones de la matriz de covarianzas y de la media del ítem $i$ en $Q(I)$, respectivamente

$$\psi_i = \left( \begin{pmatrix} \frac{1}{\rho_1^2} & & 0 \\ & \ddots & \\ 0 & & \frac{1}{\rho_D^2} \end{pmatrix} + \sum_{u \in N(i)} \frac{\phi_u + \overline{u}^\top \overline{u}}{\tau^2} \right)^{-1} \tag{2.21}$$

$$\bar{i} = \psi_i \left( \sum_{u \in N(i)} \frac{r_{ui} \overline{u}}{\tau^2} \right) \tag{2.22}$$

donde $N(i)$ se toma como el conjunto de usuarios que han puntuado el ítem $i$, es decir, los identificadores $u$ tal que $r_{ui}$ se ha observado.

Las desviaciones $\sigma_l$, $\rho_l$ y $\tau$ también pueden ser estimadas y aprendidas por el modelo. Diferenciando la ecuación (2.18) respecto de $\sigma_l$, $\rho_l$ y $\tau$ se obtienen las ecuaciones (2.23), (2.24) y (2.25).

$$\sigma_l^2 = \frac{1}{V-1} \sum_{u=1}^{V} (\phi_u)_{ll} + \overline{u_l}^2 \tag{2.23}$$

$$\rho_l^2 = \frac{1}{J-1} \sum_{i=1}^{J} (\psi_i)_{ll} + \overline{v_l}^2 \tag{2.24}$$

$$\tau^2 = \frac{1}{K-1} \sum_{(u,i)} r_{ui}^2 - 2 r_{ui} \overline{u}^\top \bar{i} + trace[(\phi_u + \overline{u}^\top \overline{u})(\psi_i + \bar{i}^\top \bar{i})] \tag{2.25}$$

***Predicción de ratings***

Una vez calculada la aproximación óptima de $P(U, I \mid R)$, es decir, $Q(U, I)$, ésta se puede utilizar para predecir las puntuaciones futuras de los usuarios. Para ello, dada la matriz $R$ tenemos la distribución del nuevo rating dada por la Ecuación 2.26.

$$P(\hat{r}_{ui}|R) \sim \int P(r_{ui}|u^\top i, \tau) Q(U, I) dU dI = \int N(\hat{r}_{ui}|u^\top i, \tau^2) Q(U, I) dU dI \tag{2.26}$$

Así, podemos obtener su media y predicción final del rating:

$$\mathbb{E}(\hat{r}_{ui}|R) = \int\int \hat{r}_{ui} N(\hat{r}_{ui}|u^\top i, \tau^2) Q(U, I) dU dI d\hat{r}_{ui} \tag{2.27}$$

Aplicando el teorema de Fubini, tenemos:

$$\mathbb{E}(\hat{r}_{ui}|R) = \int \underbrace{\int \hat{r}_{ui} N(\hat{r}_{ui}|u^\top i, \tau^2) d\hat{r}_{ui}}_{u^\top i} Q(U, I) dU dI = \mathbb{E}(u^\top i) = \overline{u}^\top \overline{i} \tag{2.28}$$

## 2.3. Evaluación de sistemas de recomendación

Como se ha podido ver en secciones anteriores, existen multitud de algoritmos y técnicas en los sistemas de recomendación y, como consecuencia, la elección de aquel que mejor se adapte a las necesidades del entorno se convierte en un problema a resolver. Para seleccionar el algoritmo, además, es necesario focalizar las propiedades que se desean medir y sobre las que basar la elección.

La evaluación de sistemas de recomendación comenzó basándose en cómo de precisas eran las predicciones de las elecciones del usuario. Pero, al igual que los algoritmos, los métodos de evaluación también evolucionan y, actualmente, no sólo se basan en medir la precisión de los sistemas sino que también existen métodos para medir gran variedad de propiedades que los usuarios valoran en un recomendador. Algunas de estas propiedades son: el descubrimiento de ítems novedosos, diversidad en las recomendaciones, la rapidez de la recomendación, entre otras muchas [12].

A lo largo de esta sección se estudiarán las técnicas de evaluación más populares y utilizadas dentro del área de Recomendación. Se comenzará con el estudio de las dos estrategias de evaluación: online y offline, y se continuará con la descripción de algunas propiedades importantes de estos sistemas junto con posibles métricas para su evaluación.

### 2.3.1. Estrategias de evaluación

La evaluación de un sistema de recomendación, en general, se puede realizar de manera *online* u *offline*. En este proyecto se utilizarán sólo evaluaciones del segundo tipo y, por esta razón, esta estrategia será estudiada con mayor detalle en este apartado.

En los experimentos **online** los usuarios interactúan con un sistema de recomendación en ejecución mientras se recoge su feedback, bien preguntándoles directamente o bien observándoles.

Los experimentos **offline** utilizan un conjunto de datos previamente recolectado en el que se recoge la información sobre los usuarios, los ítems y las interacciones entre ellos, los ratings. El conjunto de ratings observados, $\mathcal{T}$, se divide en dos: el conjunto de entrenamiento, $Tr$, y el conjunto de test, $Te$. El objetivo es predecir los ratings de test a partir de los datos de entrenamiento para, tras esto, evaluar [12].

Como se detallará más adelante, existen multitud de métricas de evaluación y muchas de ellas evalúan para cada usuario listas de ítems ordenadas. Como estas van a cobrar un papel importante dentro de este proyecto se van a estudiar a continuación las diferentes metodologías o estrategias para seleccionar los ítems que conformarán dichas listas durante la evaluación.

En estas métricas, se estiman los ratings $\hat{r}_{ui}$ para todo ítem $i$ que no ha puntuado un usuario $u$ en el conjunto de entrenamiento, a partir de la información de dicho conjunto. Tras esto, se

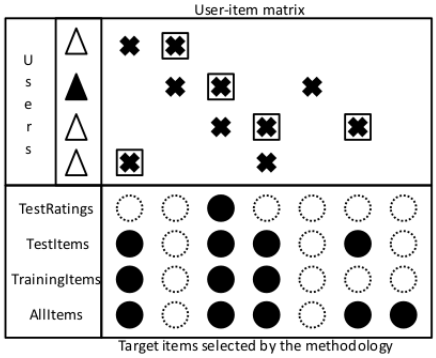


Figura 2.2: Metodologías para evaluar listas de ítems ordenados.

ordenan los ítems de mayor a menor según el rating estimado y se obtiene la lista $L_u$, que puede estar restringida a aquellos ítems devueltos por el sistema en el top-$N$. El problema reside en qué ítems forman dicha lista. Surgen así diferentes metodologías que influyen directamente en los resultados finales de las métricas [6]:

- ***TestRatings***: En esta metodología la lista $L_u$ se forma sólo con ítems puntuados por $u$ y que se encuentran en el conjunto de test.

$$L_u = Te_u \tag{2.29}$$

- ***TestItems***: En este caso, además de los ítems puntuados por $u$ del conjunto de test, se incluyen ítems no puntuados por $u$ pero sí por otro usuario en test y, por tanto, considerados no relevantes para $u$. Dicho de otro modo, todos los ítems en test menos los ítems puntuados por $u$ en el conjunto de entrenamiento.

$$L_u = \cup_v Te_v \setminus Tr_u \tag{2.30}$$

- ***TrainingItems***: Se trata de una adaptación de la metodología anterior. En este caso, se seleccionan todos los ítems puntuados por algún usuario en entrenamiento, es decir, en tiempo de recomendación, excepto los puntuados por el usuario $u$ en el conjunto de entrenamiento.

$$L_u = \cup_{v \neq u} Tr_v \tag{2.31}$$

- ***AllItems***: Esta metodología selecciona todos los ítems, hayan sido o no puntuados por algún usuario del sistema, menos los ítems puntuados por $u$ en el conjunto de entrenamiento.

$$L_u = \mathcal{T} \setminus Tr_u \tag{2.32}$$

En la Figura 2.2 se muestra un diagrama, recogido de [6], que resume las metodologías anteriormente citadas. En él se muestra una matriz de ratings en el que las cruces recuadradas marcan los ratings que forman parte del conjunto de test, las cruces simples los que forman parte del conjunto de entrenamiento y los espacios vacíos ratings desconocidos por el sistema. El usuario en estudio es el usuario representado por la segunda fila.

### 2.3.2. Dimensiones y métricas de evaluación

La elección de un recomendador es una tarea compleja pues no se trata de un problema con una única solución, un recomendador puede dar mejores resultados que otros en una dimensión pero empeorar cuando la evaluación se focaliza en otra diferente. A continuación se citarán y describirán algunas de las dimensiones más evaluadas de los sistemas de recomendación.

#### 2.3.2.1. Métricas de precisión

Las métricas de precisión miden cómo de bien se ha estimado el rating comparado con el rating real y, también, pueden medir cómo de bien ordena las recomendaciones [12]. La precisión, por tanto, puede ser entendida y medida de múltiples formas.

En algunos sistemas de recomendación el objetivo es estimar el rating que el usuario daría a un ítem, por ejemplo de 1 a 5. En estos casos la precisión de estas estimaciones se mide con métricas como las siguientes [11, 12]:

- **Error cuadrático medio (RMSE):** Uno de las métricas más populares. El sistema sólo estima ratings para aquellos pares $(u, i)$ del conjunto de test $Te$ para los que el rating verdadero existe. Así, la métrica se calcula con la Ecuación 2.33.

$$RMSE = \sqrt{\frac{1}{|Te|} \sum_{(u,i) \in Te} (\hat{r}_{ui} - r_{ui})^2} \tag{2.33}$$

- **Error absoluto medio (MAE):** Es la alternativa a RMSE. Como se ve en su Ecuación 2.34, en lugar de tomar la raíz cuadrada se utiliza el valor absoluto. La diferencia entre ambos es que RMSE penaliza más los errores más grandes tomando el cuadrado de los errores individuales.

$$MAE = \frac{1}{|Te|} \sum_{(u,i) \in Te} |\hat{r}_{ui} - r_{ui}| \tag{2.34}$$

  Tanto RMSE como MAE tienen su versión normalizada, NRMSE y NMAE respectivamente, donde simplemente se normaliza por la diferencia del valor máximo y el valor mínimo $(r_{max} - r_{min})$

Existen otras métricas de precisión que no miden cómo de bien ha predicho un rating el sistema sino si lo que ha recomendado es útil o no para el usuario. Para ello, se considera que una recomendación es útil o relevante si el usuario la ha utilizado o, en algunos casos, si lo ha puntuado además por encima de un umbral. Así, en los experimentos *offline*, tras proporcionar una lista de $N$ recomendaciones a cada usuario del conjunto de test, $Te$, se evalúa si dichos ítems fueron puntuados por el usuario de manera relevante, es decir, si en el conjunto de test está dicho rating, y, si así fuera, se consideraría que la recomendación habría sido útil para el usuario. Siguiendo esta idea y teniendo en cuenta la Tabla 2.1, pueden aplicarse las métricas de precisión y *recall* ampliamente conocidas en el área de Aprendizaje Automático [11, 12]:

- **Precisión (*at N*):** Dada una lista de $N$ recomendaciones retornadas a un usuario $u$, mide la proporción de ítems relevantes de dicha lista sobre las $N$ retornadas (ver Ecuación 2.35). La precisión total del sistema será la media de las precisiones de los usuarios con recomendación.

$$P@N = \frac{\#TP}{\#TP + \#FP} = \frac{\#TP}{N} \tag{2.35}$$

- ***Recall (at N)***: Dada una lista de $N$ recomendaciones retornadas a un usuario $u$ mide la proporción de ítems relevantes retornados sobre el total de ítems relevantes de $u$ en el conjunto de test (ver Ecuación 2.36). Al igual que en la precisión, el *recall* del sistema se calcula a partir de la media de los *recalls* obtenidos de los usuarios con recomendación.

| | Recomendado | No Recomendado |
|---|---|---|
| Relevante | Positivo Verdadero (TP) | Falso Negativo (FN) |
| No Relevante | Falso Positivo (FP) | Negativo Verdadero (TN) |

Tabla 2.1: Notación utilizada en algunas métricas de evaluación.

$$R@N = \frac{\#TP}{\#TP + \#FN} \tag{2.36}$$

En muchas ocasiones las recomendaciones se presentan como una lista ordenada y, en estos casos, lo importante no es cómo de bien se ha predicho el rating exacto o simplemente cuántos ítems realmente relevantes contiene la lista, sino que dichos ítems aparezcan en las primeras posiciones. Con ello se evita que el usuario tenga que recorrer toda la lista hasta encontrar algo que le guste. Es aquí donde cobran sentido las métricas de ranking. Muchas de estas métricas son conocidas en el área de Recuperación de Información, como *Discounted Cumulative Gain* (NDCG) [15] o *Mean Reciprocal Rank* (MRR) [9], entre otras.

#### 2.3.2.2. Otras Propiedades a Evaluar

Como ya se ha mencionado, dependiendo de qué tipo de recomendador se desea obtener, la evaluación se puede centrar en unas propiedades u otras. A lo largo de este apartado se van a citar brevemente algunas de las propiedades más importantes y qué métricas pueden utilizarse para su medición.

***Cobertura***

Es muy común que en ciertos algoritmos se consigan precisiones muy altas pero sólo generando recomendaciones para un conjunto pequeño de usuarios o de ítems de los que se tiene mucha información. En muchos entornos existe mucha información sobre pocos usuarios y poca información sobre la gran mayoría. Por ello, es importante no sólo tener en cuenta medidas de precisión sino también de cobertura como las siguientes [12]:

- **Cobertura de Usuario** (*User Space Coverage (USC)*): Proporción de usuarios para los que se ha recomendado algún ítem.
- **Cobertura de Ítem o de Catálogo** (*Item Space Coverage (ISC@N)*): Proporción de ítems que han sido recomendados a, al menos, un usuario, donde cada usuario recibe el top $N$ de sus recomendaciones. Esta métrica también puede considerarse una métrica de diversidad [12].

Uno de los problemas más estudiados en el área de Recomendación es el denominado *Cold Start Problem*, es decir, qué hacer cuando entra en el sistema un nuevo usuario o un nuevo ítem del que no se conoce todavía ningún tipo de información.

***Diversidad***

En muchos casos recomendar cosas muy similares puede no ser útil para el usuario. Así, Gunawardana y Shani definen la diversidad en [12] como lo opuesto a la similitud. Dos métricas de diversidad populares son la diversidad agregada (Aggdiv) y el Índice de Gini. Aggdiv (*Aggregate Diversity*), propuesta por Adomavicius y Kwon en [5], es la versión no normalizada de la cobertura de ítem $ISC$ y mide el número total de ítems recomendados por el sistema. Con

ella se puede medir qué cantidad de ítems del sistema están siendo mostrados a alguno de los usuarios.

$$\text{Aggdiv} = \mid \bigcup_{u \epsilon \mathcal{U}} L_u \mid \tag{2.37}$$

donde $L_u$ representa el top-$N$ de los ítems recomendados al usuario $u$.

Por otro lado, el Índice de Gini mide la dispersión estadística de $L_u$, no sólo tiene en cuenta que un ítem haya sido recomendado a alguien sino también cuántos usuarios y con que distribución. Viene dado por la Ecuación 2.38 [12, 8].

$$\text{Gini} = \frac{1}{\mid \mathcal{J} \mid -1} \sum_{k=1}^{|\mathcal{J}|} (2k - N - 1) p(i_k \mid s) \tag{2.38}$$

donde $p(i_k \mid s)$ es la probabilidad de que el $k$-ésimo ítem menos recomendado sea extraído de una lista de recomendaciones generada por el sistema $s$:

$$p(i \mid s) = \frac{|\{u \in \mathcal{U} \mid i \in L_u\}|}{\sum_{j \in \mathcal{J}} |\{u \in \mathcal{U} \mid j \in L_u\}|} \tag{2.39}$$

***Novedad***

Las métricas de novedad miden la capacidad de un sistema de recomendar ítems que para el usuario son novedosos, es decir, no conocidos. Podría considerarse que un ítem es novedoso para el usuario si no ha sido puntuado por él. El problema es que los usuarios no reportan qué ítems conocen del pasado y podría no haberlo puntuado a pesar de conocerlo [12]. Así, como los autores definen en [8], la novedad puede ser entendida como la diferencia entre el pasado y el presente.

En este proyecto han sido utilizadas dos métricas de novedad: EPC (*Expected Popularity Complement*) y EFD (*Expected Free Discovery*). Ambas proporcionan una medida de disponibilidad del sistema a recomendar ítems denominados *long tail*, es decir, poco populares [32].

# 3

# Toma de Decisiones en Algoritmos de Recomendación

El principal objetivo de los sistemas de recomendación es ofrecer ítems relevantes a los usuarios, aunque recientemente otros objetivos también se están teniendo en cuenta [12]: aumentar la diversidad o la novedad de las recomendaciones, sugerir ítems de manera que se pueda explicar de dónde viene dicha recomendación, aumentar la confianza del usuario en el sistema, etc. Este Trabajo de Fin de Máster toma este último punto y le da la vuelta, estudiando la confianza que tiene el sistema en sus recomendaciones; en particular, nos centramos en **el estudio de distintos métodos para dotar a los sistemas de recomendación de la capacidad para tomar decisiones sobre sus recomendaciones**. De esta forma, se espera que el sistema sólo muestre aquellas sugerencias que se han generado a partir de datos más fiables, obteniendo recomendaciones mejores a costa de, previsiblemente, reducir el número de las mismas.

Hay que tener en cuenta que el concepto de confianza en recomendación se ha aplicado a distintos aspectos. Por un lado, se define la confianza en los datos de entrada, donde algunos autores han estudiado qué combinaciones de usuarios o de pares (usuario, ítem) permiten generar mejores recomendaciones, véase el *profile-level trust* y el *item-level trust* de [24], o como en [14], donde se utiliza para interpretar la confianza al convertir datos implícitos (frecuencias) en explícitos. Por otro lado, se pueden encontrar distintos mecanismos orientados a contextualizar las predicciones realizadas por los algoritmos de recomendación. De esta forma, en [13] se propone un peso (*significance weighting*) que se combina con la similitud entre usuarios para devaluar aquellos casos donde dicha similitud se haya calculado con pocos datos. De una manera similar, en [4] se propone un método para filtrar recomendaciones atendiendo a la desviación de los ratings recibidos por los ítems. En paralelo, en [21] se ha estudiado cómo introducir medidas de confianza a la hora de mostrar las recomendaciones, es decir, en la interfaz con la que interactúa el usuario. En este trabajo la métrica de confianza es muy simple (número de *ratings* conocidos por el sistema para un ítem) pero es suficiente como para incrementar el nivel de satisfacción del usuario.

Debido a que existen algoritmos muy diferentes, no va a ser posible definir de una manera genérica y universal la toma de decisiones sobre la confianza de una recomendación, sino que dependerá del algoritmo, de sus características e hipótesis de partida. A lo largo de este trabajo se han analizado algunos algoritmos importantes dentro del área de Recomendación para poder comprender en qué partes del algoritmo y a partir de qué datos el sistema podría decidir si una recomendación es o no fiable y si, por tanto, debería ser mostrada, o no, al usuario. A diferencia de lo que se ha estudiado en el área, los métodos que se proponen en este trabajo no

se centran en los datos de entrada (confiabilidad del *rating* como en [21, 14]), sino que analizan aspectos intrínsecos de los algoritmos de recomendación o de las componentes utilizadas durante la predicción (similar a los ajustes de los valores de similitud propuestos en [13]). En concreto, se ha profundizado sobre algoritmos de filtrado colaborativo tanto basado en $K$ Vecinos Próximos como en factorización de matrices.

A lo largo de las subsecciones de este capítulo se explicarán cada una de las estrategias seguidas para dotar a diferentes algoritmos de recomendación de la posibilidad de decidir cuándo un ítem debe ser recomendado a un usuario. En el siguiente capítulo se descubrirá qué implicaciones tienen estas estrategias en la evaluación de las recomendaciones y, en el Capítulo 5, se explicarán los experimentos realizados sobre ellos y los resultados y conclusiones obtenidas.

## 3.1. Independiente del modelo

Las interacciones que un usuario tiene con los ítems del sistema pueden ser dados a través de una puntuación o rating. Estas puntuaciones pueden tener diferentes valores según el entorno, por ejemplo, en algunas redes sociales sólo se indica si una foto o comentario gusta o no, mientras que en sistemas como Netflix se puede votar con 1, 2, 3, 4 o 5 estrellas una película, indicando con 1 que no le ha gustado nada al usuario y con un 5 que le ha encantado.

Como se vio anteriormente, el objetivo del algoritmo es estimar el rating o puntuación que el usuario daría a un ítem que aún no ha puntuado, realizar un ranking de estas estimaciones y recomendar aquellos ítems con puntuaciones más altas. De este modo, el algoritmo puede retornar ítems con estimaciones bajas que deberían considerarse no relevantes para el usuario. Así surge la primera estrategia de decisión: **El sistema sólo confía en recomendar ítems para los que la puntuación final estimada supera un umbral de confianza**.

En un entorno en el que los valores de los ratings son entre 1 y 5, podría considerarse que los valores por debajo de 3 indican que el ítem no es relevante para el usuario y que, por tanto, no debe recomendarse aunque sean aquellos con valores más altos estimados por el algoritmo. Es decir, tras realizar todas las predicciones el algoritmo decidiría no confiar en aquellos ítems con valores considerados no relevantes para el usuario, es decir, por debajo de un umbral.

Nótese que el mismo mecanismo se podría aplicar cuando los ratings no están disponibles, siempre y cuando se pueda aplicar algún umbral de relevancia a la salida de los algoritmos; por ejemplo, un algoritmo de factorización de matrices que predice un valor de 0,2 en un escenario de una única clase (rango de las salidas entre 0 y 1) se podría considerar como no relevante y, por tanto, no ser recomendado.

De hecho, en Recuperación de Información es habitual utilizar algoritmos para encontrar y optimizar este umbral, sobre todo en sistemas de filtrado basados en contenido [33].

## 3.2. Según el soporte de la predicción

Como ya se explicó en detalle en la Sección 2.2.1, los algoritmos de $K$ Vecinos Próximos basan su estimación directamente en los ratings de los usuarios, siguiendo una estrategia basada en usuario o bien basada en ítem. La primera estima los ratings de un usuario a partir de los ratings que han dado los $k$ usuarios más parecidos a él. Mientras que la segunda estrategia estima el rating que un usuario dará a un ítem basándose en los ítems que han recibido votaciones similares a ese ítem.

La pregunta que surge es: ¿Se debería confiar igual en un ítem votado por gran parte del vecindario que en uno votado, por ejemplo, por uno o dos vecinos? Tiene sentido que, al igual

que en la vida real, cuantas más opiniones se conocen sobre un ítem más confianza genera la estimación final. Así surge la segunda estrategia de decisión dentro del algoritmo KNN basado en usuario:

**Si un ítem no ha sido votado por al menos $n$ de los $k$ vecinos del usuario el sistema decide no confiar en la estimación final obtenida**.

Dicho de otro modo, el rating que se estima que daría el usuario a un determinado ítem votado por menos de $n$ de sus vecinos se ignoraría. Esta idea es parecida a la propuesta en [13] y denominada *significance weighting*, donde se introducen pesos a la hora de calcular similitudes entre usuarios para devaluar aquellos casos que se han calculado con pocos datos.

## 3.3. Según la incertidumbre de la predicción

En muchos métodos y algoritmos de recomendación la estimación final de un rating se calcula a partir de una media, $\mu$. Ejemplo de ello son los algoritmos de $K$ Vecinos Próximos, los cuales estiman el rating a partir de una media ponderada, o algoritmos de factorización de matrices probabilísticos, como el estudiado *Variational Bayesian*, que estiman la distribución del rating de la que se puede obtener la media.

Al igual que se calcula la media, puede obtenerse la desviación típica de la distribución del rating. De este modo, esta desviación típica puede ser interpretada como un parámetro de confianza del recomendador sobre el rating: a mayor desviación típica más incertidumbre sobre la predicción del rating y por tanto menos confianza sobre él. Así, surge la tercera idea de este proyecto, donde se puede utilizar la desviación típica como el valor en el que el recomendador basa su decisión de confiar o no en la estimación obtenida, de tal forma que **el recomendador decide no recomendar aquellos ratings cuya desviación típica excede un determinado umbral.**

Por otro lado, conociendo la media $\mu$ y la desviación típica $\sigma$ se puede cambiar la idea de que la estimación final sea únicamente la media y aplicar la Ecuación 3.1.

$$r_{ui} = \mu_{r_{ui}} + \lambda \cdot \sigma_{r_{ui}}, \text{ donde } \begin{cases} \text{si } \lambda < 0 & \text{subestimación} \\ \text{si } \lambda = 0 & \text{estimación usual} \\ \text{si } \lambda > 0 & \text{sobreestimación} \end{cases} \tag{3.1}$$

La idea de la variación del factor $\lambda$ se basa en el concepto que se muestra en la Figura 3.1. Si la distribución de predicción de cada rating se aproximara a una normal se sabe que alrededor del 68 % de los datos de una distribución de ese tipo están dentro del intervalo $(\mu - \sigma, \mu + \sigma)$ y que en torno al 95 % viven en el intervalo $(\mu - 2\sigma, \mu + 2\sigma)$. En el caso del algoritmo *Variational Bayesian*, la distribución del rating se aproxima bastante a esta distribución.

De esta forma, dependiendo del valor de $\lambda$ que se tome, estaríamos estimando el rating como el punto medio de estos intervalos ($\lambda = 0$) o como uno de los extremos. Lo interesante es que esta transformación se aplica a todos los ítems recomendados a cada usuario, de manera que, cuando se presenten ordenados, aquel ítem que se hubiera predicho con un valor muy alto ($\mu_{r_{ui}}$) pero tuviera una incertidumbre también alta ($\sigma_{r_{ui}}$) podría llegar a aparecer después en el ranking (si $\lambda < 0$) que un ítem con una predicción agregada menor pero con menos dispersión. El factor $\lambda$ atenúa o aumenta este efecto.

A continuación se explicará cómo aplicar estas ideas en algoritmos de $K$ Vecinos Próximos y en el algoritmo de factorización de matrices *Variational Bayesian*.

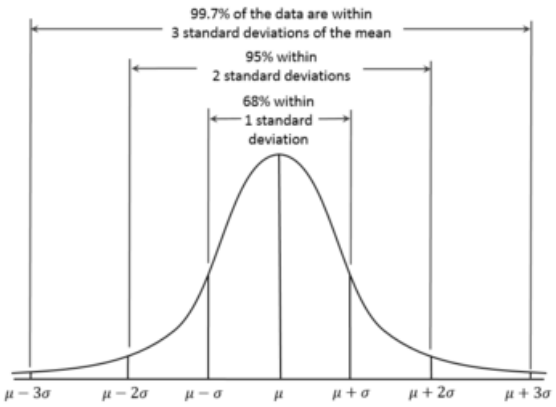


Figura 3.1: Distribución de probabilidad alrededor de la media en una distribución $N(\mu, \sigma^2)$.

### 3.3.1. Aplicación a factorización de matrices probabilístico

La aproximación bayesiana de factorización de matrices que Yew Jin Lim y Yee Whye Teh proponen para recomendación en [19] ha sido explicada en profundidad en la Sección 2.2.2 con el objetivo de comprender la modificación que en este proyecto se propone.

En este algoritmo de factorización de matrices el objetivo es calcular una distribución predictiva para las puntuaciones estimadas. Así, para cada rating hay una distribución aproximada con su correspondiente media y desviación típica, o incertidumbre sobre la predicción. Esta distribución viene dada por la Ecuación 2.26. Los autores de [19] no tratan cómo calcular la desviación típica de esta distribución ya que no la necesitan, pero es necesario obtener la ecuación exacta para poder utilizarla y aplicarla en el contexto de este trabajo.

En la Ecuación 3.2 se muestra la fórmula de la varianza. De ésta se conoce, por la Ecuación 2.28, que $\mathbb{E}(\hat{r}_{ui}|R)^2 = (\overline{u}^\top \overline{i})^2 = \overline{i}^\top \overline{uu}^\top \overline{i}$.

$$Var(\hat{r}_{ui}|R) = \mathbb{E}(\hat{r}_{ui}^2|R) - \mathbb{E}(\hat{r}_{ui}|R)^2 \tag{3.2}$$

$$\begin{aligned}
\mathbb{E}(\hat{r}_{ui}^2|R) &= \int\int \hat{r}_{ui}^2 N(\hat{r}_{ui}|u^\top i, \tau^2) Q(U,I) dU dI d\hat{r}_{ui} \\
&= \int \underbrace{\int \hat{r}_{ui}^2 N(\hat{r}_{ui}|u^\top i, \tau^2) d\hat{r}_{ui}}_{\mathbb{E}(r_{ui}^2)=\tau^2-\mathbb{E}(r_{ui})^2=\tau^2+i^\top uu^\top i} Q(U,I) dU dV \\
&= \mathbb{E}(\tau^2 + i^\top uu^\top i) \\
&= \tau^2 + \mathbb{E}(i^\top uu^\top i) \\
&= \tau^2 + trace((\phi_u - \overline{uu}^\top)(\psi_i - \overline{ii}^\top))
\end{aligned} \tag{3.3}$$

$$Var(\hat{r}_{ui}|M) = \tau^2 + trace((\phi_u + \overline{uu}^\top)(\psi_i + \overline{ii}^\top)) + \overline{i}^\top \overline{uu}^\top \overline{i} \tag{3.4}$$

Por tanto, la ecuación de la desviación tipica $\sigma$, necesaria para la Ecuación 3.1, viene dada por la raíz cuadrada de la Ecuación 3.4. Con ella, en el Capítulo 5, se estudiará en diferentes experimentos cómo influye en la calidad del recomendador tanto la estimación de la predicción a partir de la Ecuación 3.1 y con diferentes valores de $\lambda$, como el filtrado de ratings a partir del establecimiento de umbrales usando la incertidumbre de la predicción $\sigma$.

### 3.3.2. Aplicación a Vecinos Próximos

En el algoritmo KNN el rating se estima a partir de una media ponderada (Ecuación 2.1). La idea propuesta es calcular también la desviación típica ponderada y poder así aplicar la Ecuación 3.1 y la toma de decisión según la incertidumbre de la predicción. El problema de esta propuesta es que, al contrario que en el caso de *Variational Bayesian*, el algoritmo no proporciona una fórmula explícita con la que calcular la desviación típica de la estimación del rating y, además, la distribución de ésta no tiene por qué ser normal.

La varianza muestral ponderada viene dada por la Ecuación 3.5 [2].

$$\begin{aligned}
\sigma^2 &= \frac{\sum_{i=1}^{N} w_i (r_i - \overline{r})^2}{V_1} \\
&= \frac{\sum_{i=1}^{N} w_i (r_i^2 - 2 r_i \overline{r} + \overline{r}^2)}{V_1} \\
&= \frac{\sum_{i=1}^{N} w_i r_i^2}{V_1} + -2\overline{r} \left( \frac{\sum_{i=1}^{N} w_i r_i}{V_1} \right) + \frac{\overline{r}^2 \sum_{i=1}^{N} w_i}{V_1} \\
&= \frac{\sum_{i=1}^{N} w_i r_i^2}{V_1} - 2\overline{r}^2 + \overline{r}^2 \frac{V_1}{V_1} \\
&= \frac{\sum_{i=1}^{N} w_i r_i^2}{V_1} - \overline{r}^2
\end{aligned} \tag{3.5}$$

donde $V_1 = \sum_{i=1}^{N} w_i$. Tal y como se explica en [2], cuando los pesos son pesos de confianza la varianza ponderada insesgada se calcula como se muestra en la Ecuación 3.6.

$$\sigma^2 = \frac{\sum_{i=1}^{N} w_i \mathbb{E}[(r_i - \hat{r})^2]}{V_1} = \mathbb{E}[(R - E[R])^2] - \frac{V_2}{V_1^2} \mathbb{E}[(R - E[R])^2] \tag{3.6}$$

donde $V_2 = \sum_{i=1}^{N} w_i^2$ y $(1 - \frac{V_2}{V_1^2})$ es el sesgo. Análogo al sesgo $\frac{(N-1)}{N}$ en la versión no ponderada.

La Ecuación 3.7 muestra la fórmula de la varianza ponderada muestral. Su raíz cuadrada será el valor que tomará $\sigma$ en la Ecuación 3.1.

$$\sigma^2 = \frac{\sum_{i=1}^{N} w_i (r_i - \overline{r})^2}{V_1 - (V_2/V_1)} = \frac{\sum_{i=1}^{N} w_i r_i^2 - V_1 \overline{r}^2}{V_1 - (V_2/V_1)} \tag{3.7}$$

donde $\overline{r}$ representa la media ponderada.

Puede ocurrir que el denominador de la Ecuación 3.7 sea 0. Tal caso sólo puede darse si todos los pesos son iguales, en cuyo caso calcular la desviación insesgada ponderada no cobra sentido y se podría calcular la versión no ponderada, o si $N = 1$, donde por definición la desviación de un solo elemento es 0.

De este modo, la idea que se propone es aplicar la Ecuación 3.7 para calcular la varianza del rating calculado en el algoritmo de $K$ Vecinos Próximos. Así, la ecuación de esta varianza $\sigma^2_{r_{ui}}$ aplicada al caso de un KNN basado en usuarios vendría dada por la Ecuación 3.8.

$$\sigma^2_{r_{ui}} = \frac{\sum_{v \in N_i(u)} w_{uv} r_{vi}^2 - V_1 \hat{r}_{ui}^2}{V_1 - (V_2/V_1)} \tag{3.8}$$

con $V_1 = \sum_{v \in N_i(u)} w_{uv}$ y $V_2 = \sum_{v \in N_i(u)} w_{uv}^2$.

# 4

# Evaluación de Sistemas de Recomendación que Toman Decisiones

A lo largo del Capítulo 3 se han propuesto estrategias para incorporar la toma de decisiones en una serie de algoritmos de recomendación y, como consecuencia, surge la necesidad de encontrar métricas con las que evaluar el rendimiento y los resultados obtenidos con estos nuevos algoritmos.

Cuando un recomendador decide no recomendar ciertos ítems que han recibido una estimación no confiable puede ocurrir que tanto la cobertura de usuario como la cobertura de ítem desciendan, a pesar de que la precisión y otras métricas como NDCG aumenten. Por ejemplo, se puede conseguir una precisión muy alta y una cobertura baja recomendando a un único usuario una lista de un elemento relevante. Por ello, surge el objetivo de estudiar métricas que combinen, de alguna forma, la cobertura y la precisión.

Además, otro objetivo de este trabajo es proponer una métrica que favorezca aquellos casos donde un recomendador es consciente de cuándo no recomendar. Como ya se ha explicado, en ocasiones es mejor que el recomendador devuelva menos resultados a que retorne los $N$ resultados solicitados pero con muchos de ellos irrelevantes. En este último caso, el usuario podría comenzar a no confiar en las sugerencias del recomendador. Sin embargo, la mayoría de las métricas de evaluación (tanto las de Recuperación de Información como las que se usan en los Sistemas de Recomendación) no son capaces de distinguir estos casos, por lo que, en la práctica, se tiende a evitar tomar dichas decisiones y devolver las listas de recomendación tan completas como sea posible.

A lo largo de este capítulo se explicarán nuevas métricas para conseguir los objetivos anteriormente citados y la idea en la que se ha basado su creación y desarrollo.

## 4.1. Precisión vs. cobertura

Como ya se ha citado anteriormente, en ocasiones los sistemas de recomendación no son capaces de dar $N$ recomendaciones a cada usuario del sistema. En estos casos no es suficiente con medir únicamente la precisión ($P@N$), es también importante medir la cobertura de usuario, y en ocasiones la de ítem. Esta sección se va a centrar en la cobertura de usuario ya que la de ítem

también puede tomarse como una medida de diversidad [12].

Gunawardana y Shani hablan sobre este problema en [12] y lo dejan como un problema abierto a la investigación. Los autores únicamente proponen diseñar un experimento en el que se comparen las precisiones de dos recomendadores tras realizar sus correspondientes filtrados. En este experimento proponen calcular una curva variando los umbrales de filtrado.

En este proyecto se propone combinar ambas métricas a través de varias estrategias y evaluar las ventajas y desventajas de cada una de ellas a partir de los resultados obtenidos con diferentes experimentos que se estudiarán y analizarán en detalle en el Capítulo 5.

### 4.1.1. F-score o media armónica

El valor F o F1-score es una medida de evaluación muy utilizada en áreas como el Aprendizaje Automático para combinar precisión y recall. Se trata de la media armónica de ambas métricas y toma valores entre 0 y 1, siendo 0 el peor valor y 1 el óptimo.

Basándonos en esta idea surge la primera estrategia: **combinar precisión y cobertura a través de la media armónica** tal y como se muestra en la Ecuación 4.1.

$$F_1 = 2 \cdot \frac{1}{\frac{1}{P@N} + \frac{1}{USC}} = 2 \cdot \frac{P@N \cdot USC}{P@N + USC} \tag{4.1}$$

La ecuación general viene dada por la Ecuación 4.2. Si $\beta$ es igual a 1 se da la misma importancia a ambas métricas, si $\beta$ es menor que 1 se da mayor importancia a la precisión y si es mayor que 1, a la cobertura.

$$F_\beta = (1 + \beta^2) \cdot \frac{P@N \cdot USC}{\beta^2 \cdot P@N + USC} \tag{4.2}$$

### 4.1.2. G-score o media geométrica

La segunda propuesta es combinar precisión y cobertura a través de una media geométrica, también conocida como G-score, tal y como se muestra en la Ecuación 4.3.

$$G = (P@N \cdot USC)^{\frac{1}{2}} \tag{4.3}$$

Si se desea ponderar las métricas se puede utilizar la Ecuación 4.4.

$$G_{\alpha_1,\alpha_2} = (P@N^{\alpha_1} \cdot USC^{\alpha_2})^{1/(\alpha_1+\alpha_2)} \tag{4.4}$$

## 4.2. Métricas de Exactitud (Correctness)

### 4.2.1. Idea base de las nuevas métricas

Anselmo Peñas y Álvaro Rodrigo en su artículo «A Simple Measure to Assess Non-response» [25] sobre *Question Answering* proponen una medida que recompensa a los sistemas que mantienen el número de respuestas correctas y reducen el número de respuestas incorrectas dejando algunas no contestadas. Su idea se basa en que en un examen puede valorarse más a un alumno que asume que no sabe la respuesta que a uno que contesta y falla.

Imaginemos que el escenario es un test en el que se suponen las siguientes hipótesis:

| | Correct ($C$) | Incorrect ($\neg C$) |
|---|---|---|
| Answered ($A$) | $n_{ac}$ | $n_{aw}$ |
| Unanswered ($\neg A$) | $n_u$ | |

Tabla 4.1: Notación utilizada en [25] para el número de respuestas correctas e incorrectas respondidas y no respondidas.

1. El test tiene varias preguntas
2. Cada pregunta tiene varias opciones
3. Sólo una de las opciones es correcta

La Tabla 4.1 muestra la notación que se utiliza en este escenario. Debido a que el objetivo es premiar de algún modo las respuestas no contestadas frente a las respuestas incorrectas, la idea inicial en el artículo de Peñas y Rodrigo [25] es que las respuestas no contestadas tengan el mismo valor que si una proporción de ellas hubieran sido contestadas correctamente. Esto puede expresarse matemáticamente tal y como se muestra en la Ecuación 4.5.

$$\begin{aligned} P(C) &= P(C \cap A) + P(C \cap \neg A) \\ &= P(C \cap A) + P(C \mid \neg A)P(\neg A) \end{aligned} \tag{4.5}$$

donde P(C) nos indica el porcentaje de corrección del test, mientras que $P(\neg A)$ la proporción de preguntas no contestadas. Cabe destacar que el concepto de corrección o exactitud del test no sólo tiene en cuenta la proporción de respuestas correctas sino también la proporción de respuestas no contestadas.

Por tanto, de la Ecuación 4.5 se conocen los siguientes términos:

$$P(C \cap A) = \frac{n_{ac}}{n} \qquad P(\neg A) = \frac{n_u}{n} \tag{4.6}$$

Sin embargo, se desconoce el valor de $P(C \mid \neg A)$. Con el objetivo de estimar esta probabilidad, Peñas y Rodrigo proponen una serie de aproximaciones y llegan a la conclusión, tras varios experimentos y tests, que la mejor aproximación es aquella que supone que la probabilidad de que sea correcto sabiendo que no ha contestado, $P(C \mid \neg A)$, se estime como la probabilidad de que sea correcto y que haya contestado, $P(C \cap A)$. Así, la métrica final, denominada correctness ($c$@1), queda como sigue:

$$\begin{aligned} P(C) &= P(C \cap A) + P(C \mid \neg A)P(\neg A) \\ &= \frac{n_{ac}}{n} + \frac{n_{ac}}{n}\frac{n_u}{n} \end{aligned} \tag{4.7}$$

Esta métrica cumple las siguientes propiedades básicas:

1. Un sistema que responde a todas las preguntas recibe un *score* final igual que si se hubiera usado la precisión tradicional ($n_{ac}/n$).
2. Un sistema que no responde a ninguna pregunta ($n_{ac} = 0$) recibe como score final un 0.
3. Las respuestas no contestadas añaden valor a la métrica, en la misma proporción que las respuestas contestadas y correctas; es decir, las respuestas no contestadas de un sistema que acierta poco no añaden tanto valor a la métrica como las de un sistema que acierte mucho.

| | Relevant ($C$) | Not Relevant ($\neg C$) |
|---|---|---|
| Recommended ($A$) | $TP$ | $FP$ |
| Not Recommended ($\neg A$) | $FN$ | $TN$ |

Tabla 4.2: Adaptación de la notación de la Tabla 4.1 para la creación de las métricas *Correctness*.

### 4.2.2. Métricas de exactitud (correctness metrics)

La idea presentada en la sección anterior puede utilizarse en el área de los Sistemas de Recomendación para evaluar cómo de correctas están siendo las respuestas/recomendaciones, premiando a aquellos algoritmos que deciden no contestar – es decir, no recomendar – frente a los que recomiendan items no relevantes, o dicho de otro modo, responden incorrectamente. En la Tabla 4.2 se incluye la adaptación de la Tabla 4.1 al contexto de los sistemas de recomendación, distinguiendo entre ítems relevantes y no relevantes cuando no se recomienda.

Se considera que cada usuario recibe una lista de $\hat{N} \leq N$ recomendaciones del sistema $r$, $L_r^N = (i_1, ..., i_{\hat{N}})$.

Dicha lista está formada por:

- **Ítems relevantes** para el usuario $u$, considerando como relevantes aquellos que fueron puntuados por él y están en el conjunto de test, $Te(u)$, opcionalmente filtrando por el valor que aparezca en el test para considerar como relevantes sólo aquellos por encima de un umbral:

$$TP = \sum_{i \in L_r^N(u)} \mathbb{1}(i \in Te(u)) = \sum_{i \in L_r^N(u)} \mathbb{1}_{Te(u)}(i) \tag{4.8}$$

- **Ítems no relevantes** para el usuario $u$, es decir, aquellos que no fueron puntuados por $u$ en el conjunto de test:

$$FP = \sum_{i \in L_r^N(u)} (1 - \mathbb{1}_{Te(u)}(i)) \leq N \tag{4.9}$$

- El usuario puede recibir menos de $N$ recomendaciones denominando $\neg A = NR$ a los «huecos» que hay en la lista retornada o recomendaciones sin responder.

$$NR = N - (TP + FP) \tag{4.10}$$

  con $0 \leq NR \leq N$. Así, si $\hat{N} = N$ entonces $NR = 0$.

El objetivo es premiar de algún modo no recomendar frente a recomendar algo no relevante. Así surge la primera métrica: User Correctness (UC) dada por la Ecuación 4.11.

$$\text{UserCorrectness}(u, r, N) = UC(u, r, N) = \frac{1}{N}\left(TP + TP\frac{NR}{N}\right) \in [0, 1] \tag{4.11}$$

Para comprender mejor esta métrica, en la Tabla 4.3 se muestra un ejemplo sencillo donde se muestran los valores explícitos de la métrica para seis listas distintas de recomendaciones. Además, se muestra la comparativa con la precisión. De todas las listas la que mejor puntuación recibe, según $UC@5$, es la lista (f) ya que los dos ítems retornados son relevantes para el usuario $u$. Sin embargo, la precisión da la misma puntuación a la lista (a) y a la lista (f) ya que esta métrica no tiene en cuenta que la primera lista contiene 3 ítems no relevantes.

| Listas de Recomendación | Métricas de evaluación P@N | UC |
|---|---|---|
| (a) ❶❷❸❹❺ | **0,40** | 0,40 |
| (b) ❶❷❸ | 0,20 | 0,28 |
| (c) ❶ | 0,20 | 0,36 |
| (d) ❷ | 0,00 | 0,00 |
| (e) | 0,00 | 0,00 |
| (f) ❶❺ | **0,40** | **0,64** |

Tabla 4.3: Ejemplo sencillo para comparar $P@N$ y $UC@N$. En negrita se señalan los mejores resultados de cada una de las métricas. Las listas de recomendación (a)-(f) están formadas por 5 elementos ($N = 5$), dos de ellos relevantes (❶ y ❺).

Puesto que en $UC$ se calcula un valor por cada usuario, para combinar todos estos valores y resumirlos en un sólo valor se podría realizar una média aritmética de todos los valores que han obtenido los usuarios que han recibido alguna recomendación, tal y como se hace con $P@N$ y otras métricas, como se muestra en la Ecuación 4.12.

$$\text{MeanUserCorrectness}(r, N) = \text{MeanUC}(r, N) = \frac{\sum_{u \in \text{UsrRec}(r)} UC(u, r, N)}{|\text{UsrRec}(r)|} \tag{4.12}$$

donde $\text{UsrRec}(r)$ es el conjunto de usuarios con alguna recomendación del sistema $r$.

En este caso no se tiene en cuenta la existencia de usuarios sin recomendación. Al igual que la idea base de esta métrica es premiar que el sistema no responda antes que responder de forma incorrecta, también hay que tener en cuenta que no responder a nada merece algún tipo de penalización. Esta penalización consiste en restar a la media de la Ecuación 4.12 una parte proporcional de los usuarios sin recomendación. Para ello, se aplicará una idea parecida a la que se utilizó en la creación de $UC(u, r, N)$ y que se muestra en la Ecuación 4.13.

$$\begin{aligned} \text{UserCorrectness}(r, N) &= \text{MeanUC}(r, N) - \text{MeanUC}(r, N) \cdot (1 - USC) \\ &= \text{MeanUC}(r, N) \cdot USC \\ &= \frac{\sum_{u \in \text{UsrRec}(r)} UC(u, r, N)}{V_r} \end{aligned} \tag{4.13}$$

donde $USC$ es la cobertura de usuario (*User Space Coverage*) y $V_r$ es el número de usuarios del sistema de recomendación $r$.

En resumen, para combinar los valores de $UC$ en una métrica final se realiza una media aritmética tomando por base el total de usuarios del sistema. Con ello, se consigue tener en cuenta también la cobertura de usuario.

Imaginemos ahora que dos usuarios, $u_1$ y $u_2$, reciben sólo dos ítems como recomendación en un sistema que retorna listas de hasta 5 elementos. En ambos casos el sistema ha dejado sin contestar 3. No obstante, mientras que $u_1$ sólo tenía en test dichos ítems como relevantes, $u_2$ tenía 10 ítems maś en test relevantes que podría haber recibido. A pesar de esta diferencia, según $UC@5$ ambos reciben un 0,64. Basada en esta idea surge la segunda métrica propuesta: *Recall User Correctness*. Puesto que es menos correcto dejar de contestar cuando se sabe que existen ítems relevantes en test que no han sido retornados, puede introducirse en la métrica anterior el concepto de *recall*, obteniendo así la Ecuación 4.14.

$$\text{RecallUserCorrectness}(u, r, N) = RUC(u, r, N) = \frac{1}{N}\left(TP + \frac{TP}{Rel}NR\right) \in [0, 1] \tag{4.14}$$

donde $Rel = |Te(u)|$, es decir, el número de ítems relevantes disponibles en el conjunto de test para el usuario $u$.

Retomando el ejemplo, con $RUC@5$ el usuario $u_1$ recibiría un 1,0 mientras que para $u_2$ este valor sólo sería de 0,5.

De nuevo, para combinar todos los valores de $RUC@N$ se realiza una media aritmética tomando por base el total de usuarios del sistema. De este modo, se consigue combinar en una misma métrica precisión, recall y cobertura de usuario.

La siguiente pregunta que se planteó en este proyecto es si se podría utilizar esta idea para combinar otras métricas, en concreto precisión y cobertura de ítem ($ISC@N$) con el objetivo de obtener una métrica que ayude a decidir cuándo el aumento de precisión compensa el descenso de la diversidad. Como se analizará en los experimentos del Capítulo 5, en muchos casos la toma de decisiones provoca no sólo un descenso en la cobertura de usuario sino también en la cobertura de ítem. Así surge la versión derivada de $UC@N$ basada en ítem: $IC@N$, y la de $RUC@N$: $RIC@N$.

Para esto, usamos las listas de recomendaciones $L_r^N(u)$ de todos los usuarios de test para crear los siguientes conjuntos:

$$S(i, r, N) = \{u : i \in L_r^N(u)\} \tag{4.15}$$

es decir, $S(i, r, N)$ es el conjunto de todos los usuarios que tienen al ítem $i$ en su lista de recomendación (para un algoritmo $r$ y tomando $N$ como tamaño de dichas listas de recomendación).

Usando el *ground truth* se pueden crear los siguientes conjuntos:

$$T(i) = \{u : i \in Te(u)\} \tag{4.16}$$

o dicho de otro modo, los usuarios de test para los que el ítem $i$ es relevante.

Con ello, para cada ítem se tiene:

- Usuarios que han recibido a $i$ en la lista y para los que además es relevante:

$$TP = S(i) \cap T(i) \tag{4.17}$$

- Usuarios con $i$ en la lista siendo $i$ no relevante:

$$FP = S(i) \cap \overline{T(i)} \tag{4.18}$$

- Usuarios sin $i$ en la lista y para los que $i$ era relevante:

$$FN = \overline{S(i)} \cap T(i) \tag{4.19}$$

- Usuarios sin $i$ en la lista y para los que $i$ no era relevante:

$$TN = \overline{S(i)} \cap \overline{T(i)} \tag{4.20}$$

Con las ecuaciones anteriores se puede calcular la información con el número de usuarios a los que no se le ha recomendado el ítem $i$ (Ecuación 4.21) y el número de usuarios para los que el ítem $i$ es relevante (Ecuación 4.22).

$$NR = TN + FN \tag{4.21}$$

$$Rel = TP + FN = |T(i)| \tag{4.22}$$

Una vez definidos estos conjuntos, se propone las métricas ItemCorrectness o $IC@N$ (Ecuación 4.23) y RecallItemCorrectness o $RIC@N$ (Ecuación 4.24).

$$\text{ItemCorrectness}(i, r, N) = IC(i, r, N) = \frac{1}{|U|}\left(TP + \frac{TP}{|U|}NR\right) \tag{4.23}$$

$$\text{RecallItemCorrectness}(i, r, N) = RIC(i, r, N) = \frac{1}{|U|}\left(TP + \frac{TP}{Rel}NR\right) \tag{4.24}$$

Al igual que en las métricas anteriores, para obtener un resultado que resuma todos los valores adjudicados a los ítems del sistema, se calcula la media aritmética de todos ellos tomando como base el total de ítems que lo componen. De este modo, entra en juego la cobertura de ítem en la métrica, ya que, cuando un ítem no se ha recomendado a ningún usuario toma como valor en ambas métricas 0 y, al dividir por el total de ítems, la métrica global desciende cuando existen muchos ítems sin recomendar. Además, debido al término $\frac{TP}{|U|}$ de las Ecuaciones 4.23 y 4.24, cuando la cobertura de usuario, $USC$, o la precisión descienden, este término también lo hace indirectamente y, como consecuencia, ambas métricas se ven afectadas por ello. Por tanto, no sólo se tiene en cuenta la cobertura de ítem sino también indirectamente la precisión y la cobertura de usuario.

Una de las desventajas de la métrica $IC@N$, que se verá en el Capítulo 5, es que, al tener como denominador el total de usuarios del sistema, retorna valores muy bajos. En el caso de $RIC@N$ se tiene en cuenta la proporción de ítems relevantes que se han recomendado de tal modo que si se ha recomendado el ítem $i$ a la mayoría de usuarios para los que era relevante se premia más el no recomendar al resto de usuarios. Por ello, esta última métrica retorna valores más altos.

# 5

# Experimentos Realizados y Resultados

A lo largo de este capítulo se detalla cómo se realizaron los experimentos y se analizan los resultados obtenidos para, finalmente, poder obtener conclusiones sobre los mismos.

## 5.1. Entorno

Para realizar los experimentos que se detallan en la próxima sección, se han utilizado diferentes *frameworks* en diversos lenguajes de programación, principalmente Java y Python. Además se han probado varios *datasets* de diferentes dominios para poder obtener conclusiones objetivas y consistentes.

### 5.1.1. Implementación de algoritmos y métricas de evaluación

Parte de la implementación del proyecto se ha apoyado en RankSys[1]. Se trata de un *framework* para la implementación y evaluación de algoritmos de recomendación programado en Java 8. Éste utiliza muchos de los avances de la última versión de Java haciendo que su rendimiendo sea óptimo, algo esencial para manejar grandes *datasets* en algoritmos complejos y/o costosos. En concreto se ha utilizado para ejecutar los algoritmos de $K$ Vecinos Proximos (KNN) con las modificaciones propuestas en el Capítulo 3. Además de implementar las modificaciones de estos algoritmos sobre dicho framework, también se implementaron en Python para, de este modo, poder profundizar en detalle en ellos y comprenderlos mejor.

El algoritmo de *Variational Bayesian*, explicado exhaustivamente en la Sección 2.2.2.1 y sus modificaciones, explicadas en la Sección 3.3.1, se implementaron por completo en Python utilizando paquetes importantes de este lenguaje como *pandas*[2] y *numpy*[3]. Inicialmente, debido a las cuentas matriciales que conlleva este algoritmo, se implementó una primera versión en Matlab que finalmente se reescribió en Python.

En cuanto a las métricas de evaluación se combinaron dos *frameworks*: el ya citado RankSys

[1] https://github.com/RankSys/RankSys
[2] http://pandas.pydata.org/
[3] http://www.numpy.org/

| Dataset | Users | Items | Ratings | Density | Rating Scale |
|---|---|---|---|---|---|
| ML100K | 943 | 1.681 | 100.000 | 1,33 % | [1,5] |
| ML1M | 6.040 | 3.883 | 1.000.209 | 4,26 % | [1,5] |
| Jester | 59.132 | 150 | 1.710.677 | 24,28 % | [-10,10] |

Tabla 5.1: Características de los *datasets* utilizados a lo largo del proyecto.

y RiVal[4]. Éste último es un *framework*, también implementado sobre Java, orientado al procesamiento de datos de preferencias de usuario (*splitting*) y la evaluación de sistemas de recomendación. Han sido utilizadas algunas de sus métricas ya implementadas como precisión, *recall* y NDCG. Sobre este sistema se añadieron nuevas métricas:

- **Cobertura de Usuario** (*User Space Coverage*, $USC$): Proporción de usuarios que han recibido alguna recomendación.
- **Cobertura de Usuario@**$N$ (*User Space Coverage@*$N$, $USC@N$): Proporción de usuarios que han recibido en su lista de recomendación por lo menos $N$ ítems.
- **Cobertura de Ítem@**$N$ (*Item Space Coverage@*$N$, $ISC@N$): Proporción de ítems que han sido recomendados a algún usuario en un sistema que retorna, como máximo, $N$ ítems a cada usuario. A pesar de que RankSys propociona una implementación de Aggrdiv (versión no normalizada de $ISC@N$), se decidió implementar $ISC@N$ porque cuando la cobertura no es total, es decir, cuando el sistema retorna menos de $N$ ítems a un usuario, esta métrica, tal cual está implementada en RankSys, comienza a comportarse de forma distinta. Por ello, a pesar de haber obtenido las dos en todos los experimentos, se decidió reportar sólo $ISC@N$.
- **Métricas de exactitud** (*Correctness Metrics*): Métricas propuestas en este trabajo y explicadas en la Sección 4.2.2: $UC@N$, $RUC@N$, $IC@N$ y $RIC@N$.

Además, se utilizó RankSys para incorporar al estudio la evaluación de la novedad del sistema, utilizando las métricas EPC y EFD, y la diversidad, utilizando el Índice de Gini, mencionadas en la Sección 2.3.2.2.

Por último, se utilizó el *framework* Rival para realizar la división de conjunto de entrenamiento y test con diferentes *folds*. En concreto, se utilizó la estrategia *TestItems* explicada a lo largo de la Sección 2.3.1.

### 5.1.2. Datasets utilizados

El conjunto de datos es una de las partes fundamentales en la realización de experimentos. En este proyecto se han utilizado diferentes conjuntos de distintos dominios para poder obtener conclusiones consistentes de los algoritmos sin depender del conjunto que se haya usado. En la Tabla 5.1.2 se muestran los *datasets* o conjuntos de datos utilizados y sus principales características. Las dos versiones de MovieLens (ML)[5] (ML100K y ML1M) son *datasets* sobre películas mientras que Jester[6] es de bromas.

En el caso de MovieLens se ha utilizado tanto la versión reducida de 100K ratings como la versión de 1M. La razón de esto es que para poder probar la correcta implementación de los

[4]https://github.com/recommenders/rival
[5]https://grouplens.org/datasets/movielens/
[6]http://eigentaste.berkeley.edu/dataset/

distintos algoritmos y métricas con ejecuciones más rápidas se utilizaba la versión reducida. En el caso de Jester, se transformaron los ratings del rango $[-10, 10]$ a un rango de valores positivos $[0, 20]$ ya que hay algunos algoritmos que no soportan ratings negativos.

## 5.2. Análisis de resultados

Como ya se ha mencionado, a lo largo de esta sección se analizarán y explicarán los experimentos realizados a lo largo de este proyecto. Se seguirá una estructura similar a la utilizada en el Capítulo 3, comenzando con la toma de decisiones independiente del modelo, a continuación se explicarán los experimentos realizados con los algoritmos de toma de decisiones según el soporte de la predicción y, para finalizar, se estudiarán los relacionados con la incertidumbre de la predicción. En estos experimentos se hará uso tanto de métricas de evaluación estándar en el área (Sección 2.3) como de las métricas propuestas en el Capítulo 4.

### 5.2.1. Independiente del modelo

Una primera idea sencilla para incorporar la toma de decisiones a un algoritmo de recomendación es recomendar sólo aquellos ítems que obtuvieron una estimación del rating superior a un umbral. Por ejemplo, en un sistema en el que los ratings varían de 1 a 5, donde 1 significa que no le ha gustado y 5 que le ha encantado, podría considerarse que son ítems relevantes aquellos con un rating igual a 4 o 5 y que sólo aquellas estimaciones con un valor mayor o igual a 4 son de confianza a la hora de recomendar.

En las siguientes subsecciones se muestra cómo influye la variación de este umbral, denominado Umbral de Confianza o *Confidence Threshold* y representado por la letra griega $\gamma$, en métricas de precisión, cobertura, diversidad y novedad. Nótese que esta estrategia se podría utilizar con cualquier técnica de recomendación que genere predicciones en el rango de los ratings del sistema, sin embargo, por consistencia con el resto de experimentos, sólo lo vamos a aplicar a los algoritmos de $K$ Vecinos Próximos y a la versión probabilística de factorización de matrices mencionada anteriormente, *Variational Bayesian*.

#### 5.2.1.1. $K$ Vecinos Próximos

La Tabla 5.2 muestra los resultados obtenidos respecto a métricas de precisión y cobertura, tras variar el umbral de confianza en el algoritmo de $K$ Vecinos Próximos basado en usuarios con $k = 10$, similitud coseno y el *dataset* ML100K. Se puede ver cómo la precisión y los dos tipos de cobertura (USC e ISC) se mantienen hasta $\gamma = 4$ y $\gamma = 5$ donde comienzan a bajar. En este escenario, las métricas propuestas en el Capítulo 4 no son necesarias pues precisión y cobertura alcanzan su máximo para el mismo valor pero se incluyen en la Tabla 5.2 para mostrar su consistencia.

| $\gamma$ | P@10 | *USC* | *ISC* | $F_1$ | $F_2$ | $F_{0,5}$ | $G_{1,1}$ | $G_{1,2}$ | $G_{2,1}$ | *UC* | *RUC* | *IC* | *RIC* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | **0,037** | **100,0** | **62,1** | **0,071** | **0,159** | **0,045** | **0,191** | **0,332** | **0,110** | **0,037** | **0,037** | **0,000** | **0,015** |
| 2 | 0,037 | 100,0 | 62,1 | 0,071 | 0,159 | 0,045 | 0,191 | 0,332 | 0,110 | 0,037 | 0,037 | 0,000 | 0,015 |
| 3 | 0,037 | 100,0 | 62,1 | 0,071 | 0,159 | 0,045 | 0,191 | 0,332 | 0,110 | 0,037 | 0,037 | 0,000 | 0,015 |
| 4 | 0,036 | 100,0 | 62,1 | 0,070 | 0,159 | 0,045 | 0,191 | 0,331 | 0,110 | 0,036 | 0,036 | 0,000 | 0,015 |
| 5 | 0,020 | 99,2 | 60,0 | 0,040 | 0,094 | 0,025 | 0,142 | 0,272 | 0,074 | 0,022 | 0,022 | 0,000 | 0,011 |

Tabla 5.2: Comparación de medidas de evaluación (@10) usando como estrategia de decisión el **umbral de confianza** ($\gamma$) para el algoritmo de KNN con $k = 10$ y similitud coseno en el *dataset* ML100K.

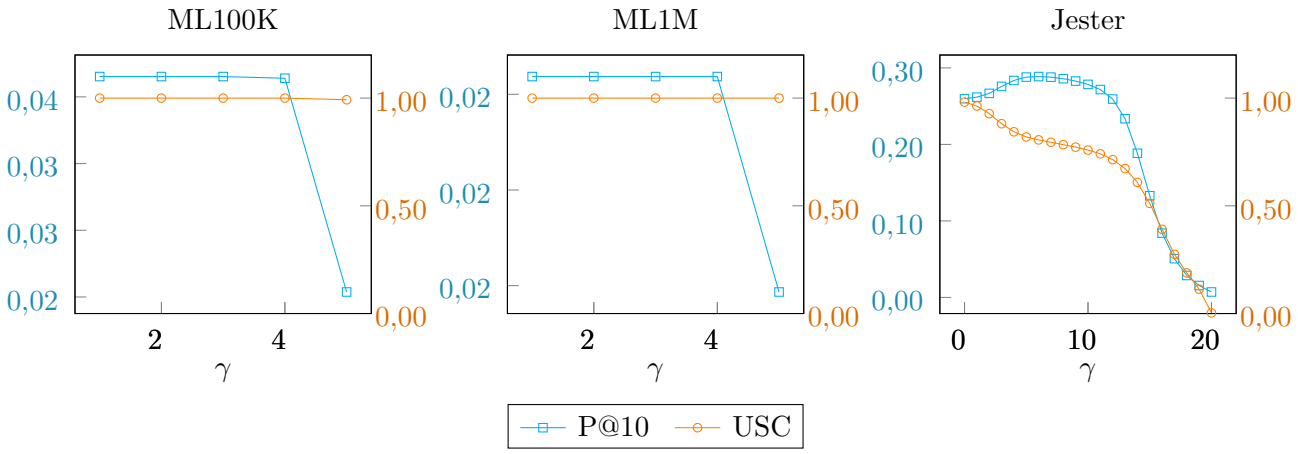


Figura 5.1: Precisión ($P@10$) vs. cobertura de usuario ($USC$) usando como estrategia de decisión el **umbral de confianza** ($\gamma$) para el algoritmo de KNN con $k = 10$ y similitud coseno en tres *datasets* distintos: ML100K, ML1M y Jester.

Se realizó este experimento, además de con ML100K, con los otros dos *datasets*: ML1M y Jester. Los resultados se recogen en la Figura 5.1. En el caso de los dos *datasets* de MovieLens, tanto la precisión como la cobertura de usuario se mantienen estables hasta que dicho umbral se vuelve muy restrictivo y empeoran considerablemente. En cambio, en el caso de Jester la cobertura de usuario empeora a medida que aumenta la restricción sobre el umbral pero la precisión comienza mejorando levemente hasta que alcanza su máximo con $\gamma = 8$ y comienza a empeorar. Esto puede deberse a cómo es la distribución de los ratings en estos *datasets*, mostradas en la Figura 5.2. Mientras que en los conjuntos de MovieLens los ratings se concentran en valores entorno a 3 y, principalmente, 4, en Jester están distribuídos entre más valores ya que éste tiene un mayor rango de ratings. Así, en el caso de MovieLens cuando se filtra por valores bajos del umbral $\gamma$ no afecta a los resultados finales ya que lo más posible es que las estimaciones eliminadas no formaran parte de las listas de recomendación de los usuarios. En cambio, cuando el umbral es muy alto, se eliminan la mayor parte de las estimaciones haciendo que tanto las coberturas de usuario e ítem como la precisión bajen. En el caso de Jester, estas métricas empeoran cuando se comienza a establecer valores de $\gamma$ mayores de 10, valores en los que se concentran más ratings.

Aunque precisión y cobertura de usuario son dos métricas fundamentales en la evaluación de los algoritmos de recomendación, es interesante también estudiar qué ocurre con otras dimensiones como la diversidad y la novedad. Como ya se ha mencionado en capítulos anteriores, la cobertura de ítem ($ISC$) puede considerarse una medida de diversidad pues reporta el porcenta-

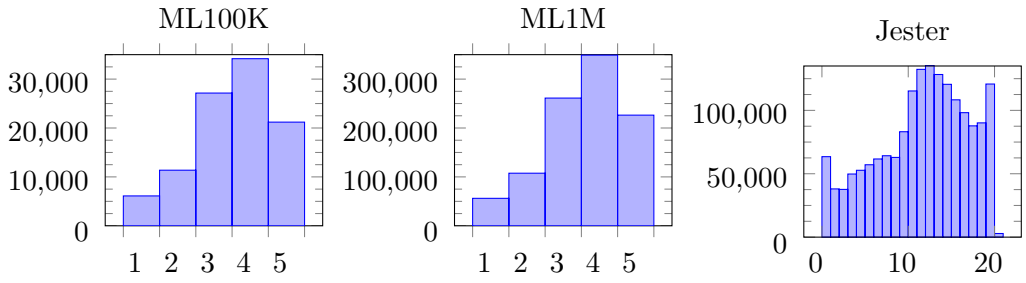


Figura 5.2: Distribución de ratings en cada *dataset*: ML100K, ML1M y Jester.

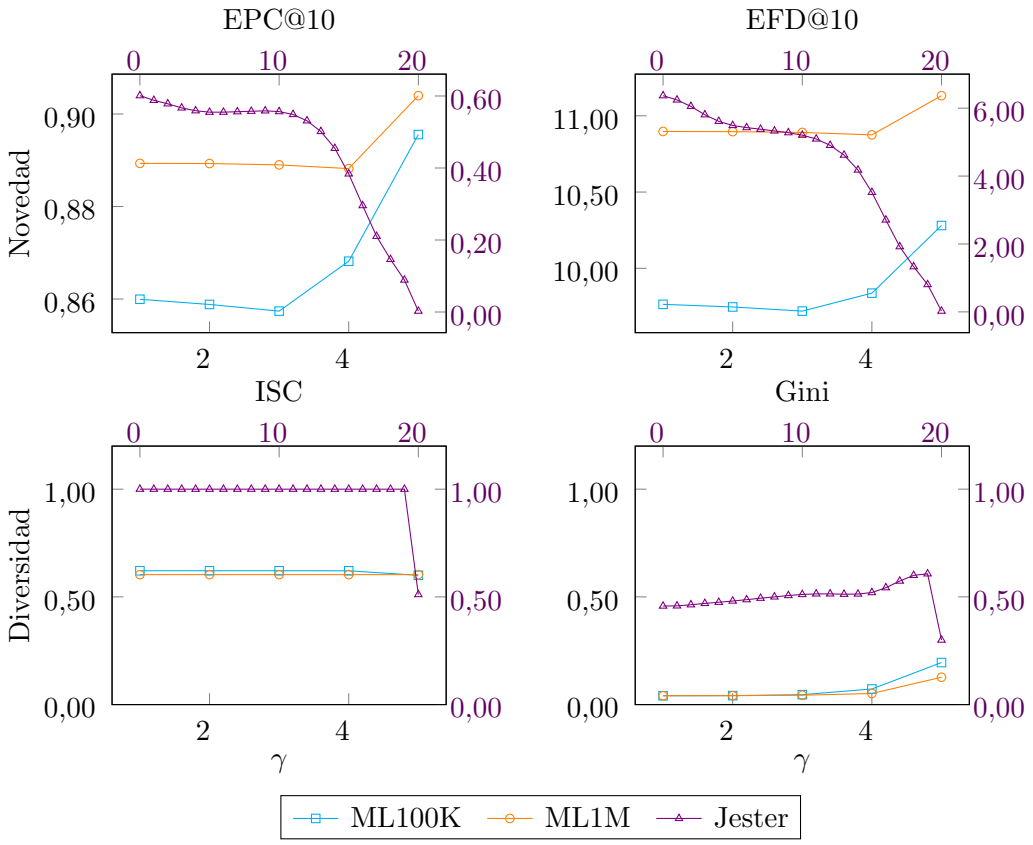


Figura 5.3: Métricas de novedad y diversidad usando como estrategia de decisión el **umbral de confianza** ($\gamma$) para el algoritmo de KNN con $k = 10$ y similitud coseno en tres *datasets* distintos: ML100K, ML1M y Jester.

je de ítems del sistema que han sido recomendados a algún usuario. En muchos casos en los que el catálogo de ítems es muy grande es importante que la mayor parte de ellos sean recomendados a al menos un usuario para poder hacer todos visible, este es el caso de Amazon. Como se ve en la Tabla 5.2 y en la Figura 5.3, en este experimento la diversidad presenta un empeoramiento cuando $\gamma$ es muy restrictivo ya que se eliminan más estimaciones y la probabilidad de eliminar por completo un ítem de las recomendaciones es mayor.

Otra dimensión interesante es la novedad, medida equivalente al inverso de la popularidad. Para conocer si la estrategia de decisión conlleva un aumento o descenso de la novedad se han utilizado dos medidas: $EPC@N$ y $EFD@N$, ambas mencionadas en el Capítulo 2. En la Figura 5.3 se ve cómo, en el caso de MovieLens se mantienen más o menos constantes, mientras que en el caso de Jester la novedad decrece a medida que la restricción aumenta. Por tanto, en este último caso, cuando se aumenta la restricción se tiende a recomendar ítems más populares.

#### 5.2.1.2. Variational Bayesian

En esta sección se van a analizar los resultados obtenidos al realizar el mismo experimento que en la sección anterior pero aplicado al algoritmo de factorización de matrices *Variational Bayesian*, es decir, el objetivo es analizar cómo influye la variación del umbral de confianza ($\gamma$) en diferentes métricas al aplicarlo en este algoritmo.

| $\gamma$ | P@10 | *USC* | *ISC* | $F_1$ | $F_2$ | $F_{0,5}$ | $G_{1,1}$ | $G_{1,2}$ | $G_{2,1}$ | *UC* | *RUC* | *IC* | *RIC* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | **0,093** | **100,0** | **22,7** | **0,170** | **0,338** | **0,113** | **0,304** | **0,453** | **0,205** | **0,093** | **0,093** | 0,001 | 0,009 |
| 2 | 0,093 | 100,0 | 22,7 | 0,170 | 0,338 | 0,113 | 0,304 | 0,453 | 0,205 | 0,093 | 0,093 | 0,001 | 0,009 |
| 3 | 0,093 | 99,9 | 22,7 | 0,170 | 0,338 | 0,113 | 0,304 | 0,452 | 0,205 | 0,093 | 0,093 | **0,001** | **0,009** |
| 4 | 0,086 | 97,8 | 22,3 | 0,158 | 0,317 | 0,105 | 0,290 | 0,434 | 0,193 | 0,086 | 0,085 | 0,001 | 0,007 |
| 5 | 0,024 | 59,0 | 15,5 | 0,047 | 0,104 | 0,030 | 0,120 | 0,204 | 0,070 | 0,018 | 0,015 | 0,000 | 0,002 |

Tabla 5.3: Comparación de medidas de evaluación (@10) usando como estrategia de decisión el **umbral de confianza** ($\gamma$) para el algoritmo de Variational Bayesian en el *dataset* ML100K.

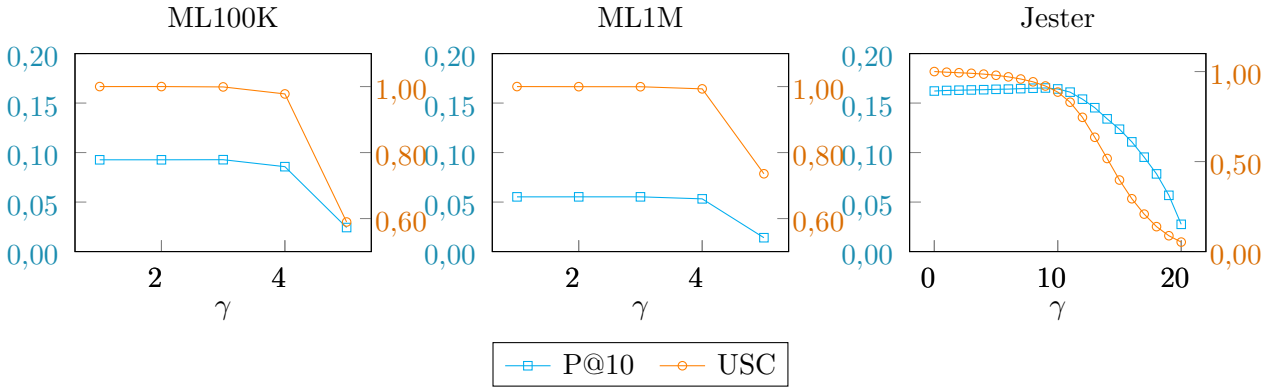


Figura 5.4: Precisión ($P$@10) vs. cobertura de usuario ($USC$) usando como estrategia de decisión el **umbral de confianza** ($\gamma$) para el algoritmo de factorización de matrices Variational Bayesian en tres *datasets* distintos: ML100K, ML1M y Jester.

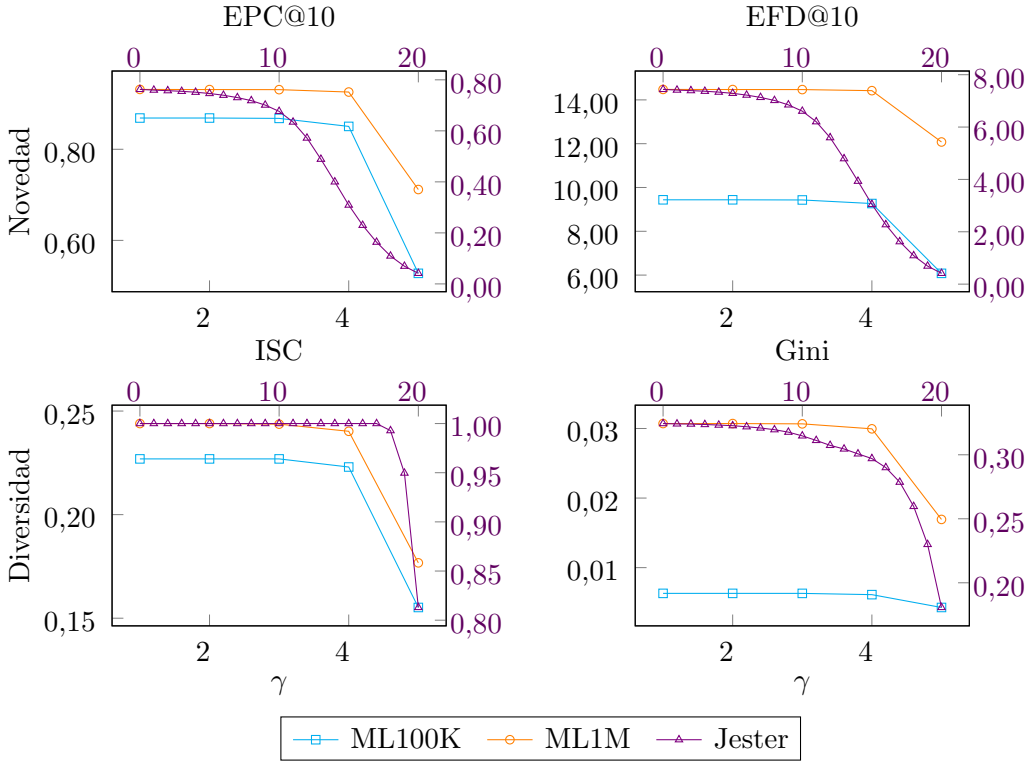


Figura 5.5: Métricas de novedad y diversidad usando como estrategia de decisión el **umbral de confianza** ($\gamma$) para el algoritmo de factorización de matrices Variational Bayesian en tres *datasets* distintos: ML100K, ML1M y Jester.

La Tabla 5.3 recoge los resultados de métricas de precisión y cobertura del experimento aplicado al *dataset* ML100K. Al igual que en el caso de KNN tanto la precisión como la cobertura se mantienen estables hasta que el umbral es demasiado restrictivo y descienden considerablemente.

La Figura 5.4 recoge cómo se comporta P@10 y USC cuando varía el umbral aplicado con los tres *datasets*. En los tres casos las métricas se mantienen estables hasta que llegados a un umbral demasiado restrictivo comienzan a empeorar, tal y como pasaba en el experimento con KNN de la Sección 5.2.1.1.

Si se analiza qué ocurre con las dimensiones de diversidad, representada por $ISC$ y Gini, y novedad, representada por $EPC$ y $EFD$, el comportamiento es similar al observado en el experimento con KNN. La Figura 5.5 muestra los resultados para las cuatro métricas en los experimentos con tres conjuntos de datos distintos. De nuevo, la novedad empeora en el caso de Jester mientras que para los *datasets* de MovieLens se mantiene más o menos constante. En el caso de la diversidad, la cobertura de ítem se mantiene constante excepto cuando $\gamma$ alcanza el máximo valor, que empeora.

Se puede concluir que ambos algoritmos, KNN y VB, se comportan de forma similar cuando se introduce la estrategia de decisión en estudio. En ambos experimentos no se observan mejoras en las métricas pero podría realizarse un experimento con usuarios reales para poder ver si ellos perciben algún cambio y qué recomendaciones consideran más útiles.

### 5.2.2. Según el soporte de la predicción

En este apartado se analizan los resultados de los experimentos realizados aplicando la toma de decisiones según el soporte de la predicción, es decir, se estudia cómo influye en diferentes métricas la modificación realizada al algoritmo KNN basado en usuarios explicada en detalle en la Sección 3.2. Para ello, se aplicó el algoritmo con $k = 10$ variando el umbral mínimo de vecinos necesarios para considerar la estimación de confianza ($n$) desde $n = 1$, es decir sin restricción, hasta $n = 8$. De este modo, si un rating ha sido calculado a partir de las votaciones de $l$ vecinos con $l < n$ se considera que la estimación no ha sido calculada con suficientes datos como para confiar en ella y considerarla consistente; por ello, se eliminaría de la lista de recomendación.

La Tabla 5.4 recoge los resultados obtenidos utilizando en el experimento el *dataset* de ML100K. Se puede observar que a medida que se aumenta el umbral $n$ aumenta $P$@10 hasta alcanzar un máximo, consiguiendo una mejora de hasta un 562,1 %. A partir de ese punto comienza a disminuir. Esto puede deberse a que cuando la restricción es muy estricta comienza a perderse demasiada información haciendo empeorar dicha métrica. Desde el punto de vista de la cobertura de usuario ($USC$), a medida que aumenta la restricción existen menos usuarios que la puedan cumplir y, como consecuencia, disminuye el número de usuarios que reciben recomendaciones. En cuanto a la cobertura de ítem, al igual que la cobertura de usuario, disminuye cuando aumenta la restricción. En concreto, se consigue una mejora de $P$@10 de un 562,1 %, aumentando de un 0,037, que tiene el algoritmo de KNN basado en usuarios sin restricción, a un 0,245, con $n = 5$ en un vecindario de $k = 10$. A cambio disminuye la cobertura de usuario de un 100 % a un 99,7 % y la de ítem de un 74, 9 % a un 45, 3 %, empeoramiento de un 39, 5 %. Así, se puede ver en la Tabla 5.4 que las métricas propuestas para combinar precisión y cobertura de usuario ($F_\beta$, $G_{\alpha_1,\alpha_2}$, $UC$@10 y $RUC$@10) y precisión y cobertura de ítem ($IC$@10 y $RIC$@10) coinciden en elegir $n = 5$ como la mejor opción. En el caso de precisión y cobertura de usuario, para $n = 5$ la precisión es máxima y la cobertura de usuario apenas varía. Sin embargo, la cobertura de ítem empeora aunque sólo un 39, 5 %, empeoramiento compensado con la mejora del 560 % de $P$@10.

| $n$ | P@10 | $USC$ | $ISC$ | $F_1$ | $F_2$ | $F_{0,5}$ | $G_{1,1}$ | $G_{1,2}$ | $G_{2,1}$ | $UC$ | $RUC$ | $IC$ | $RIC$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0,037 | **100,0** | **62,1** | 0,070 | 0,159 | 0,045 | 0,191 | 0,332 | 0,110 | 0,037 | 0,037 | 0,000 | 0,015 |
| 2 | 0,133 | 100,0 | 46,9 | 0,234 | 0,433 | 0,160 | 0,364 | 0,510 | 0,260 | 0,133 | 0,133 | 0,002 | 0,021 |
| 3 | 0,188 | 100,0 | 39,5 | 0,317 | 0,537 | 0,225 | 0,434 | 0,573 | 0,329 | 0,189 | 0,189 | 0,002 | 0,026 |
| 4 | 0,230 | 100,0 | 35,1 | 0,374 | 0,599 | 0,272 | 0,480 | 0,613 | 0,376 | 0,234 | 0,236 | 0,003 | 0,029 |
| 5 | **0,245** | 99,7 | 32,3 | **0,393** | **0,618** | **0,288** | **0,494** | **0,624** | **0,391** | **0,259** | **0,266** | **0,003** | **0,029** |
| 6 | 0,241 | 96,4 | 28,5 | 0,386 | 0,603 | 0,284 | 0,482 | 0,607 | 0,383 | 0,257 | 0,263 | 0,003 | 0,026 |
| 7 | 0,237 | 85,9 | 24,8 | 0,371 | 0,563 | 0,277 | 0,451 | 0,559 | 0,364 | 0,231 | 0,231 | 0,002 | 0,023 |
| 8 | 0,226 | 66,9 | 21,7 | 0,338 | 0,480 | 0,260 | 0,389 | 0,466 | 0,324 | 0,180 | 0,171 | 0,002 | 0,018 |

Tabla 5.4: Comparación de medidas de evaluación cuando se usa la estrategia de toma de decisión basada en el **soporte de la predicción** para el algoritmo de KNN con $k = 10$ y usando el *dataset* ML100K.

Este experimento no solo se aplicó sobre ML100K sino que también se utilizaron ML1M y Jester, todos ellos con $k = 10$. En la Figura 5.6 se muestra el impacto de la variación del umbral $n$ en la precisión y la cobertura de usuario en los 3 *datasets*. En todos ellos se mejora la precisión con $n = 5$, o lo que es lo mismo, exigiendo que al menos la mitad del vecindario haya votado el ítem para considerar la estimación segura o de confianza, y empeora levemente la cobertura de usuario hasta ese punto. Con umbrales mayores la cobertura decrece demasiado como para considerarlo útil.

Hasta este punto se ha analizado cómo influye la toma de decisiones según el soporte de la predicción en métricas de precisión y cobertura pero, como en otros experimentos, es interesante también conocer cómo influye en otras dimensiones del sistema. En la Figura 5.7 se muestra la influencia de la variación del umbral $n$ en métricas de novedad, EPC y EFD, y de diversidad, $ISC$ y Gini. En estas gráficas el eje de abscisas representa el umbral $n$, el eje de ordenadas izquierdo representa el valor de cada una de las métricas en los dos *datasets* de MovieLens y el eje de ordendadas secundario el valor de dichas métricas para el *dataset* de Jester. En todos los conjuntos de datos se observa el mismo comportamiento: un empeoramiento de la novedad y de la diversidad, excepto en el caso de Gini y de ISC en Jester que siempre consigue un valor de 1 pues está formado por poco más de 100 ítems.

El descenso de la novedad es razonable pues la restricción impuesta es que al menos $n$ vecinos hayan votado el ítem para considerarlo de confianza, por tanto, cuanto mayor es la restricción más ítems populares se tiende a recomendar, ya que implícitamente se exige un número mayor

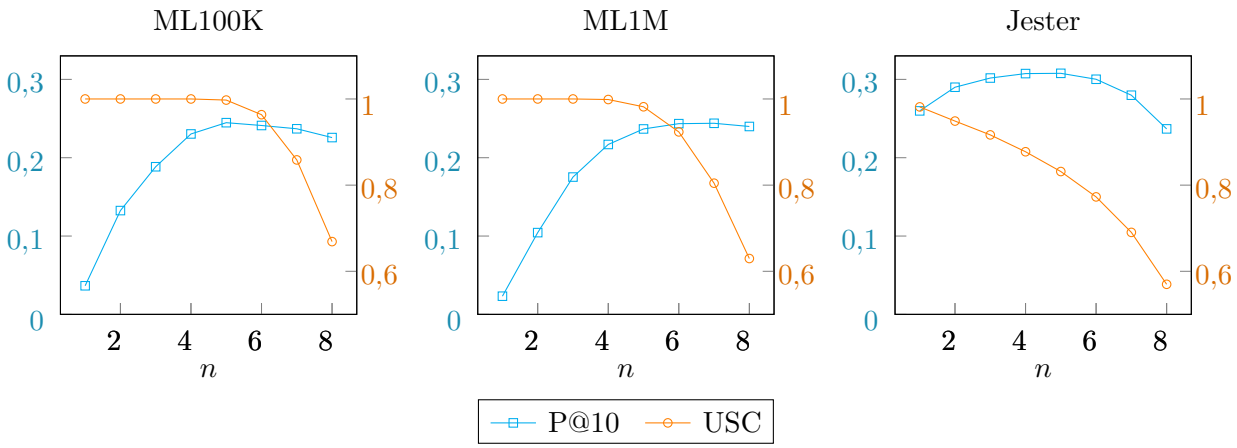


Figura 5.6: Precisión ($P@10$) vs. cobertura de usuario ($USC$) usando como estrategia de decisión el **soporte de la predicción** del algoritmo de KNN con $k = 10$ y similitud coseno en tres *datasets* distintos: ML100K, ML1M y Jester.

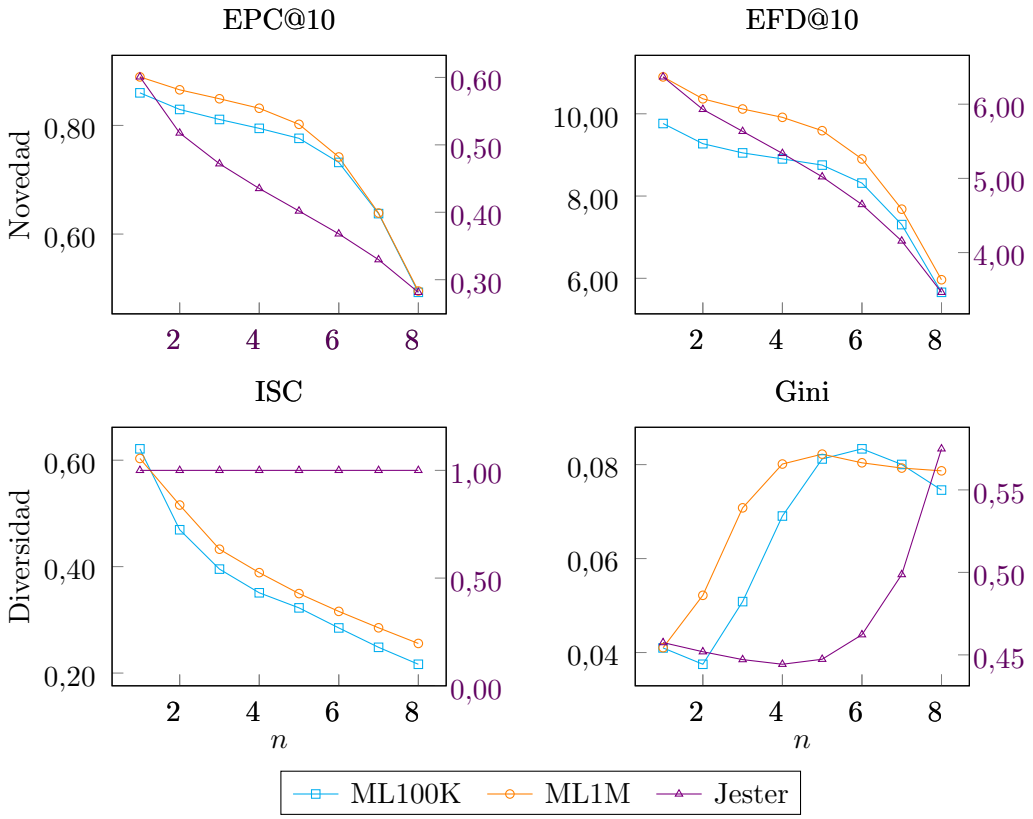


Figura 5.7: Métricas de novedad y diversidad usando como estrategia de decisión el **soporte de la predicción** del algoritmo de KNN con $k = 10$ y similitud coseno en tres *datasets* distintos: ML100K, ML1M y Jester.

de interacciones para cada ítem. Como cada vez se van recomendando ítems más populares, el número de ítems diferentes que se recomiendan baja (*ISC* decrece). Sin embargo, como estos ítems se reparten de manera relativamente uniforme, Gini obtiene cada vez valores más altos.

### 5.2.3. Según la incertidumbre de la predicción

En muchos algoritmos de recomendación la estimación final del rating que un usuario daría a un ítem viene dada por una media y, en algunos casos como el de *Variational Bayesian*, se proporciona además una fórmula para calcular explícitamente la desviación típica. Esta desviación típica es lo que en el Capítulo 3 se denomina incertidumbre de la predicción. En esta sección se van a poner en práctica las ideas propuestas en ese capítulo, resumidas en la Ecuación 3.1.

En primer lugar, se explicarán los resultados que se obtienen al aplicar estas modificaciones al algoritmo de *Variational Bayesian* y, en segundo lugar, al aplicarlas a un KNN.

#### 5.2.3.1. Variational Bayesian

La Tabla 5.5 recoge los resultados obtenidos tras realizar un experimento en el que se proporcionaba al algoritmo *Variational Bayesian* la capacidad de decidir cuándo no recomendar un ítem según su incertidumbre, $\sigma_\tau$. Se varió dicho parámetro sobre el *dataset* de ML100K y se evaluaron los resultados. Como se ve en la tabla mencionada, a medida que la restricción

| $\sigma_\tau$ | P@10 | *USC* | *ISC* | $F_1$ | $F_2$ | $F_{0,5}$ | $G_{1,1}$ | $G_{1,2}$ | $G_{2,1}$ | *UC* | *RUC* | *IC* | *RIC* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0,093 | **100,0** | 22,7 | 0,170 | 0,338 | 0,113 | 0,304 | 0,453 | 0,205 | 0,093 | 0,093 | 0,001 | 0,009 |
| 0,82 | **0,326** | 28,2 | 9,1 | 0,303 | 0,290 | **0,316** | 0,303 | 0,296 | 0,311 | 0,100 | 0,094 | 0,001 | 0,006 |
| 0,84 | 0,283 | 59,0 | 15,1 | **0,382** | 0,484 | 0,316 | 0,408 | 0,462 | **0,361** | 0,174 | 0,170 | 0,002 | 0,011 |
| 0,86 | 0,214 | 80,9 | 19,6 | 0,338 | **0,520** | 0,251 | **0,416** | 0,519 | 0,333 | **0,177** | **0,176** | **0,002** | 0,012 |
| 0,88 | 0,181 | 95,6 | 22,2 | 0,304 | 0,514 | 0,216 | 0,415 | **0,548** | 0,315 | 0,176 | 0,176 | 0,002 | **0,013** |
| 0,90 | 0,165 | 99,5 | 24,8 | 0,283 | 0,495 | 0,198 | 0,405 | 0,546 | 0,300 | 0,165 | 0,165 | 0,002 | 0,013 |
| 0,92 | 0,156 | 100,0 | 26,0 | 0,269 | 0,480 | 0,187 | 0,395 | 0,538 | 0,289 | 0,156 | 0,156 | 0,002 | 0,012 |
| 0,94 | 0,145 | 100,0 | 27,3 | 0,254 | 0,459 | 0,175 | 0,381 | 0,526 | 0,276 | 0,145 | 0,145 | 0,002 | 0,011 |
| 0,96 | 0,139 | 100,0 | 28,2 | 0,245 | 0,447 | 0,168 | 0,373 | 0,518 | 0,269 | 0,139 | 0,139 | 0,002 | 0,011 |
| 0,98 | 0,133 | 100,0 | **28,6** | 0,235 | 0,435 | 0,161 | 0,365 | 0,511 | 0,261 | 0,133 | 0,133 | 0,002 | 0,011 |

Tabla 5.5: Comparación de métricas de evaluación cuando se usa la estrategia de toma de decisión basada en la **incertidumbre de la predicción** para el algoritmo de Variational Bayesian sobre el *dataset* ML100K.

$\sigma_\tau$ es más estricta, con valores más bajos, aumenta la precisión, $P$@10, y disminuyen tanto la cobertura de usuario como la de ítem.

Con ello, y al igual que en el experimento anterior, se cumple nuestra hipótesis inicial: al eliminar recomendaciones de ítems calculados con datos muy dispersos y, por lo tanto, poco consistentes y fiables se consigue aumentar la precisión del sistema. Además, cuando la condición sobre $\sigma_\tau$ es muy exigente se eliminan cada vez más recomendaciones haciendo más probable que un usuario pueda no recibir ninguna recomendación o que un ítem no sea recomendado a ningún usuario.

Con la modificación propuesta, se consigue mejorar, con $\sigma_\tau = 0{,}82$, hasta en un $250,5\,\%$ la precisión, pasando de un 0,093 del *baseline* del algoritmo a un 0,326, a costa de un empeoramiento tanto en la cobertura de usuario como en la de ítem. En concreto, se pasa de una cobertura de usuario del 100 % a una de tan sólo 28,2 % y de un 22,7 % a un 9,1 % en el caso de la cobertura de ítem. En este punto surge una de las preguntas clave en este proyecto: ¿Compensa el aumento de la precisión con el drástico descenso de la cobertura? Existen otras opciones en las que la precisión sigue siendo mejor que la de partida y la cobertura no ha disminuído tan drásticamente, como en el caso de $\sigma_\tau = 0{,}86$ donde se mejora un 130,1 % la precisión con una cobertura de usuario del 80,9 %. Es aquí donde cobran sentido las métricas propuestas en el Capítulo 4 y recogidas en esta misma tabla.

Observando las métricas que combinan precisión y cobertura de usuario ($F_\beta$ y $G_{\alpha_1,\alpha_2}$) se puede ver, en primer lugar, que la métrica $F_1$-score es demasiado sensible a la precisión y elige la opción $\sigma_\tau = 0{,}84$ como mejor opción a pesar de obtener una cobertura del 59,0 %. Si se pondera en esta métrica más alto a la cobertura, para compensar esta sensibilidad hacia la precisión, se consigue con $F_2$-Score que el máximo se alcance en $\sigma_\tau = 0{,}86$, en el que la precisión es de 0,214 (una mejora del 130,1 % respecto a la precisión base de 0,093) y con $USC = 80{,}9\,\%$. Si por el contrario se desea dar aún más importancia a la precisión, basta con ponderar a $F_\beta$ con $\beta < 1$ como en el caso de $F_{0,5}$, donde el máximo se alcanza con $\sigma_\tau = 0{,}82$ y muy seguido por $\sigma_\tau = 0{,}84$, donde la precisión alcanza sus valores más altos. La métrica G-score ($G_{1,1}$) se comporta de manera similar a $F_2$-score y alcanza su máximo en $\sigma_\tau = 0{,}86$. Si se pondera en esta métrica más a $USC$ que a $P$@10, $G_{1,2}$-score, el máximo se alcanza con $\sigma_\tau = 0{,}88$ y se ven valores más altos en los casos en los que la cobertura es entorno al 95 %.

En cuanto a las métricas propuestas $UC$ y $RUC$, ambas proponen como mejor solución $\sigma_\tau = 0{,}86$, al igual que $F_2$ o $G_{1,1}$. En el caso de $RUC$ además presenta un empate en $\sigma_\tau = 0{,}88$ donde $P$@10 = 0,181 (mejora del 94,6 %) y $USC = 95{,}6\,\%$. La ventaja de estas métricas respecto a las que simplemente combinan las métricas es que, por un lado, no están parametrizadas y, por tanto, no necesitan del criterio subjetivo de qué parámetro elegir a la hora de utilizarlas y, por otro lado, ambas penalizan más los fallos que las no recomendaciones.

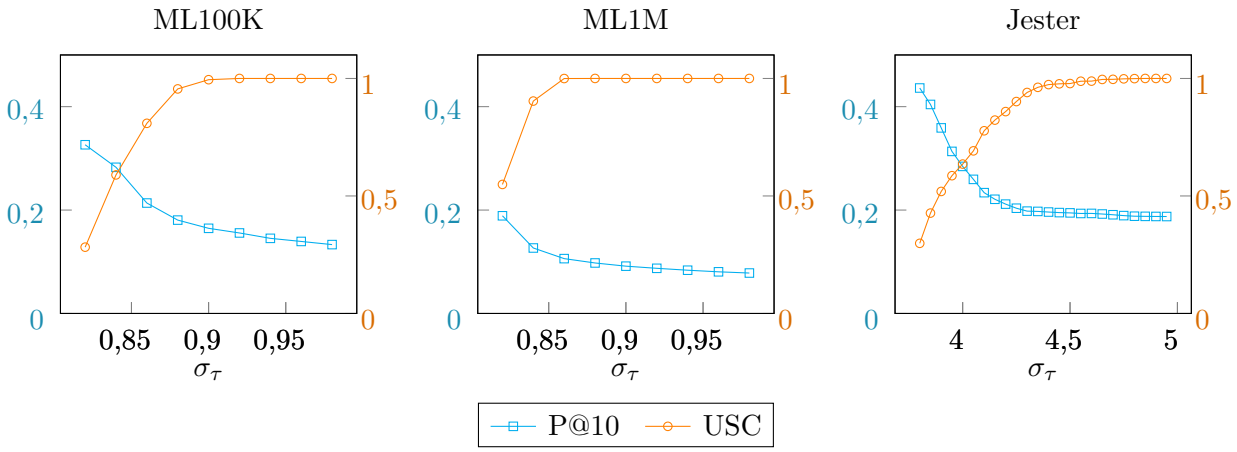


Figura 5.8: Precisión ($P@10$) vs. cobertura de usuario ($USC$) usando como estrategia de decisión la **incertidumbre de la predicción** del algoritmo Variational Bayesian en tres *datasets* distintos: ML100K, ML1M y Jester.

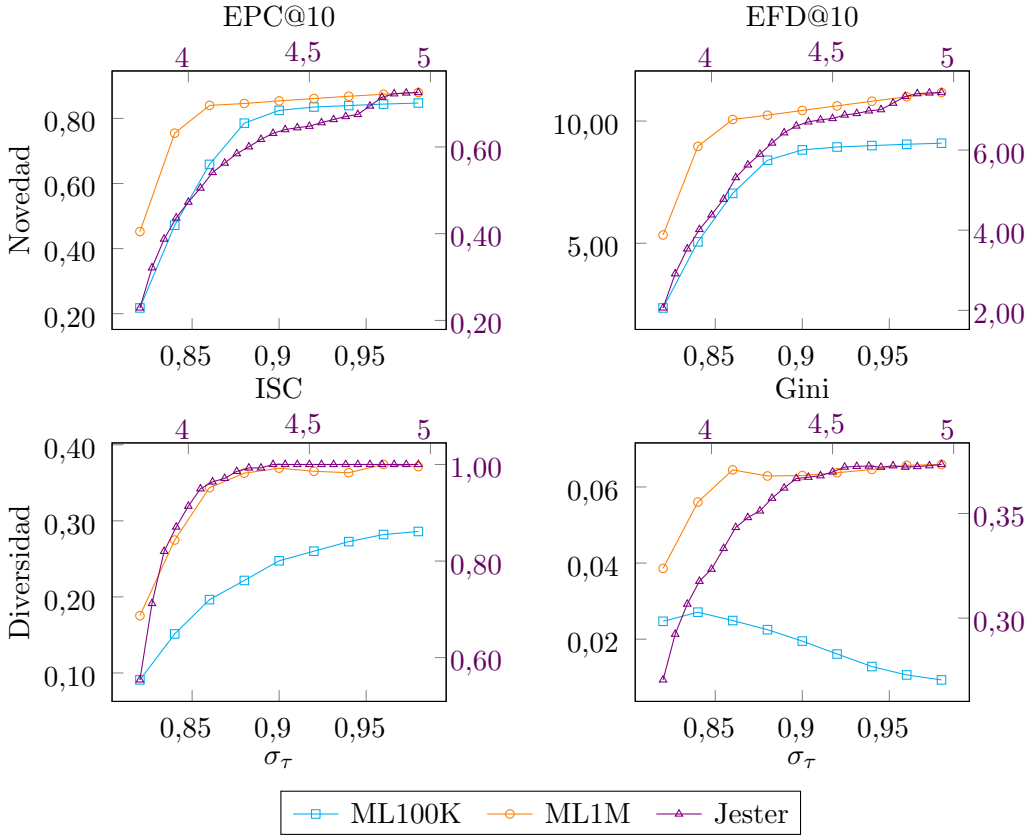


Figura 5.9: Métricas de novedad y diversidad usando como estrategia de decisión la **incertidumbre de la predicción** del algoritmo Variational Bayesian en tres *datasets* distintos: ML100K, ML1M y Jester.

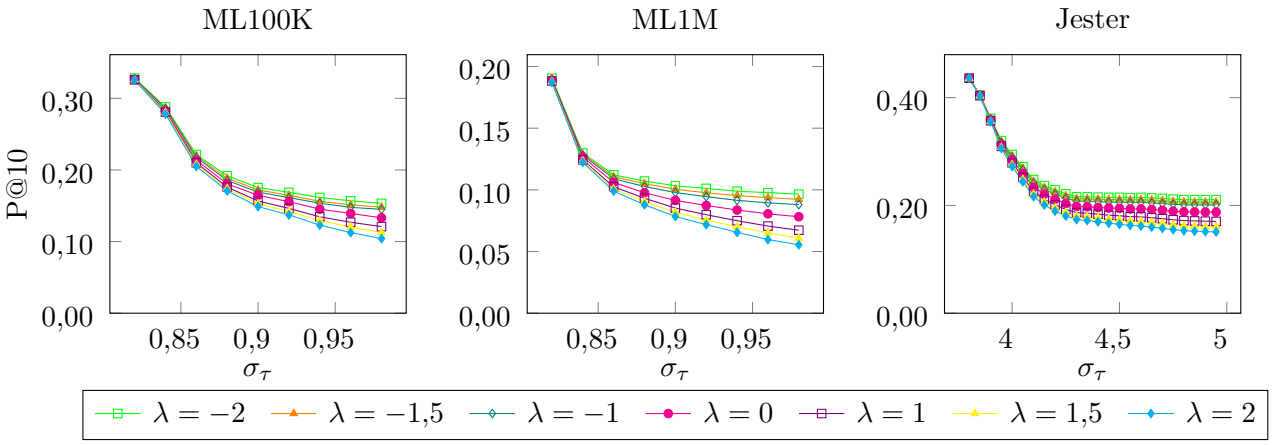


Figura 5.10: Precisión ($P$@10) usando como estrategia de decisión la **incertidumbre de la predicción** del algoritmo Variational Bayesian y aplicando la Ecuación 3.1 con distintos valores de $\lambda$ en tres *datasets* distintos: ML100K, ML1M y Jester.

Las métricas de ItemCorrectness $IC$ y $RIC$ tienen en cuenta también la cobertura de ítem ($ISC$), además de precisión y cobertura de usuario indirectamente, tal y como se explicó en el Capítulo 4. En el caso de $IC$ los valores obtenidos son muy bajos por la naturaleza de la métrica y tiene una mayor inclinación hacia la cobertura de usuario que $RIC$. Por ello, $IC$ alcanza su máximo en $\sigma_\tau = 0{,}86$ donde $ISC = 19{,}6\,\%$ mientras que $RIC$ lo alcanza en $\sigma_\tau = 0{,}88$ donde $ISC = 22{,}2\,\%$.

Este experimento se aplicó, como en casos anteriores, a dos conjuntos de datos más: ML1M y Jester. En la Figura 5.8 se muestra cómo se comportan los tres *datasets* respecto a la precisión ($P$@10) y a la cobertura de usuario ($USC$). En todos ellos, la precisión aumenta cuando $\sigma_\tau$ disminuye (situación más restrictiva) y la cobertura desciende.

Como en experimentos anteriores, en la Figura 5.9 se muestra cómo influye la toma de decisiones, en este caso la variación de $\sigma_\tau$, en métricas de novedad y diversidad. En los tres *datasets* a medida que la restricción $\sigma_\tau$ se hace menos restrictiva (crece su valor) aumentan tanto las métricas de novedad como las de diversidad.

Si además de aplicar la toma de decisiones según la incertidumbre de la predicción aplicamos la Ecuación completa 3.1 con distintos factores $\lambda$, obtenemos los resultados que se pueden ver en la Figura 5.10. Los resultados para los tres *datasets* presentan el mismo comportamiento: a medida que la restricción sobre $\sigma_\tau$ es más estricta, la precisión aumenta. Además, en todos los experimentos la subestimación presenta mejores resultados que la sobrestimación. Una razón que podría explicar esto es que subestimando se penaliza más a aquellos ratings con mayor incertidumbre y, por tanto, calculados a partir de datos menos consistentes.

#### 5.2.3.2. $K$ Vecinos Próximos

Una vez analizado cómo influye la estrategia de decisión basada en la incertidumbre de la predicción en el algoritmo *Variational Bayesian*, surge plantearse qué ocurriría si se aplicara al algoritmo de $K$ Vecinos Próximos (KNN). Tal y como se explicó en la Sección 3.3.2, la idea consiste en calcular la desviación estandar ponderada de cada estimación y, basándose en ésta, decidir cuándo una estimación es de confianza, entendiendo como confianza aquellas con una incertidumbre pequeña, es decir, menor que un *threshold* $\sigma_\tau$. Cabe destacar que este algoritmo, al contrario que otros como *Variational Bayesian*, no proporciona una fórmula explícita con la

| $\sigma_\tau$ | P@10 | *USC* | *ISC* | $F_1$ | $F_2$ | $F_{0,5}$ | $G_{1,1}$ | $G_{1,2}$ | $G_{2,1}$ | *UC* | *RUC* | *IC* | *RIC* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **0,037** | **100,0** | 62,1 | **0,070** | **0,159** | **0,045** | **0,191** | **0,332** | **0,110** | **0,037** | **0,037** | **0,000** | **0,015** |
| 0,40 | 0,025 | 100,0 | **64,2** | 0,048 | 0,112 | 0,031 | 0,157 | 0,291 | 0,085 | 0,025 | 0,025 | 0,000 | 0,014 |
| 0,45 | 0,026 | 100,0 | 64,0 | 0,050 | 0,117 | 0,032 | 0,161 | 0,296 | 0,087 | 0,026 | 0,026 | 0,000 | 0,014 |
| 0,50 | 0,027 | 100,0 | 63,8 | 0,053 | 0,123 | 0,034 | 0,165 | 0,301 | 0,091 | 0,027 | 0,027 | 0,000 | 0,015 |
| 0,55 | 0,030 | 100,0 | 63,5 | 0,058 | 0,133 | 0,037 | 0,173 | 0,310 | 0,096 | 0,030 | 0,030 | 0,000 | 0,014 |
| 0,60 | 0,031 | 100,0 | 63,1 | 0,061 | 0,139 | 0,039 | 0,177 | 0,315 | 0,099 | 0,031 | 0,031 | 0,000 | 0,014 |
| 0,65 | 0,032 | 100,0 | 63,1 | 0,061 | 0,140 | 0,039 | 0,178 | 0,316 | 0,100 | 0,032 | 0,032 | 0,000 | 0,014 |
| 0,70 | 0,032 | 100,0 | 63,1 | 0,062 | 0,141 | 0,039 | 0,178 | 0,317 | 0,100 | 0,032 | 0,032 | 0,000 | 0,014 |
| 0,75 | 0,032 | 100,0 | 62,7 | 0,062 | 0,141 | 0,040 | 0,178 | 0,317 | 0,100 | 0,032 | 0,032 | 0,000 | 0,014 |
| 0,80 | 0,032 | 100,0 | 62,6 | 0,063 | 0,143 | 0,040 | 0,180 | 0,319 | 0,102 | 0,032 | 0,032 | 0,000 | 0,014 |
| 0,85 | 0,034 | 100,0 | 62,5 | 0,066 | 0,149 | 0,042 | 0,184 | 0,324 | 0,105 | 0,034 | 0,034 | 0,000 | 0,015 |
| 0,90 | 0,034 | 100,0 | 62,4 | 0,066 | 0,150 | 0,042 | 0,185 | 0,324 | 0,105 | 0,034 | 0,034 | 0,000 | 0,015 |
| 0,95 | 0,035 | 100,0 | 62,4 | 0,067 | 0,152 | 0,043 | 0,186 | 0,326 | 0,106 | 0,035 | 0,035 | 0,000 | 0,015 |
| 1,00 | 0,035 | 100,0 | 62,4 | 0,068 | 0,154 | 0,043 | 0,187 | 0,327 | 0,107 | 0,035 | 0,035 | 0,000 | 0,015 |
| 1,05 | 0,036 | 100,0 | 62,3 | 0,069 | 0,156 | 0,044 | 0,189 | 0,329 | 0,108 | 0,036 | 0,036 | 0,000 | 0,015 |
| 1,10 | 0,036 | 100,0 | 62,3 | 0,069 | 0,156 | 0,044 | 0,189 | 0,329 | 0,108 | 0,036 | 0,036 | 0,000 | 0,015 |
| 1,15 | 0,036 | 100,0 | 62,3 | 0,069 | 0,156 | 0,044 | 0,189 | 0,330 | 0,109 | 0,036 | 0,036 | 0,000 | 0,015 |
| 1,20 | 0,036 | 100,0 | 62,3 | 0,069 | 0,157 | 0,045 | 0,190 | 0,330 | 0,109 | 0,036 | 0,036 | 0,000 | 0,015 |
| 1,25 | 0,036 | 100,0 | 62,3 | 0,070 | 0,157 | 0,045 | 0,190 | 0,330 | 0,109 | 0,036 | 0,036 | 0,000 | 0,015 |
| 1,30 | 0,036 | 100,0 | 62,3 | 0,070 | 0,158 | 0,045 | 0,190 | 0,331 | 0,109 | 0,036 | 0,036 | 0,000 | 0,015 |
| 1,35 | 0,036 | 100,0 | 62,3 | 0,070 | 0,158 | 0,045 | 0,190 | 0,331 | 0,109 | 0,036 | 0,036 | 0,000 | 0,015 |
| 1,40 | 0,036 | 100,0 | 62,3 | 0,070 | 0,158 | 0,045 | 0,190 | 0,331 | 0,109 | 0,036 | 0,036 | 0,000 | 0,015 |
| 1,45 | 0,036 | 100,0 | 62,2 | 0,070 | 0,159 | 0,045 | 0,191 | 0,332 | 0,110 | 0,036 | 0,036 | 0,000 | 0,015 |

Tabla 5.6: Comparación de métricas de evaluación cuando se usa la estrategia de toma de decisión basada en la **incertidumbre de la predicción**, es decir, cuando se varía el *threshold* $\sigma_\tau$ usando $\lambda = 0$, para el algoritmo de KNN con $k = 10$ y similitud coseno sobre el *dataset* ML100K.

que calcular esta incertidumbre. Como consecuencia, y como se analizará en esta sección, no se obtienen resultados tan buenos como en la sección anterior con el algoritmo de factorización de matrices probabilístico *Variational Bayesian*.

Se realizó el experimento con el algoritmo KNN basado en usuarios con $k = 10$, similitud coseno y sobre el conjunto de datos de ML100K variando el *threshold* $\sigma_\tau$ de tal manera que si la desviación estandar de un rating es mayor que dicho *threshold* es eliminado de la lista de recomendación. Los resultados se recogen en la Tabla 5.6. En este caso, y al contrario que en el algoritmo de factorización de matrices probabilístico *Variational Bayesian*, la variación de este umbral no influye o lo hace negativamente en la precisión ($P$@10). La cobertura de usuario ($USC$) se mantiene en su valor máximo 1, esto se debe a que existen muchos ratings calculados a partir de pocos valores y en esos casos es más probable que la desviación sea pequeña e inferior

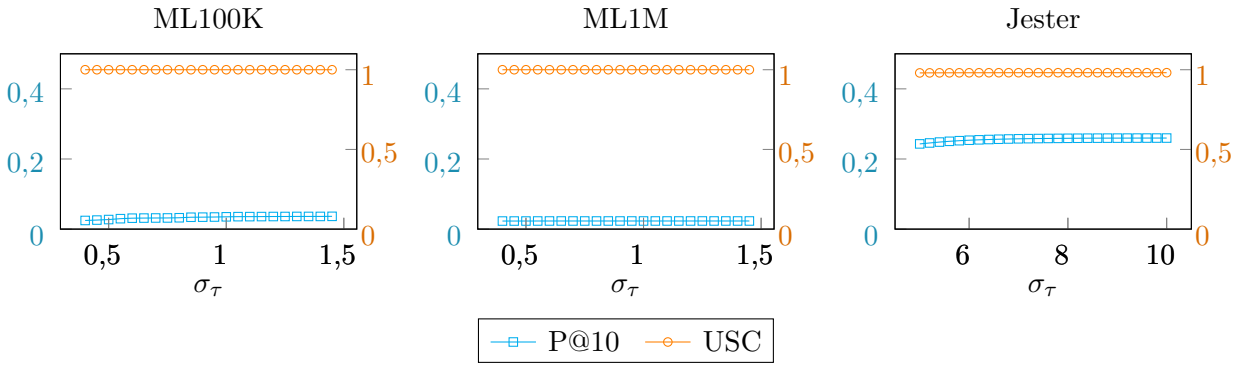


Figura 5.11: Precisión ($P$@10) vs. cobertura de usuario ($USC$) usando como estrategia de decisión la **incertidumbre de la predicción** del algoritmo KNN en tres *datasets* distintos: ML100K, ML1M y Jester.

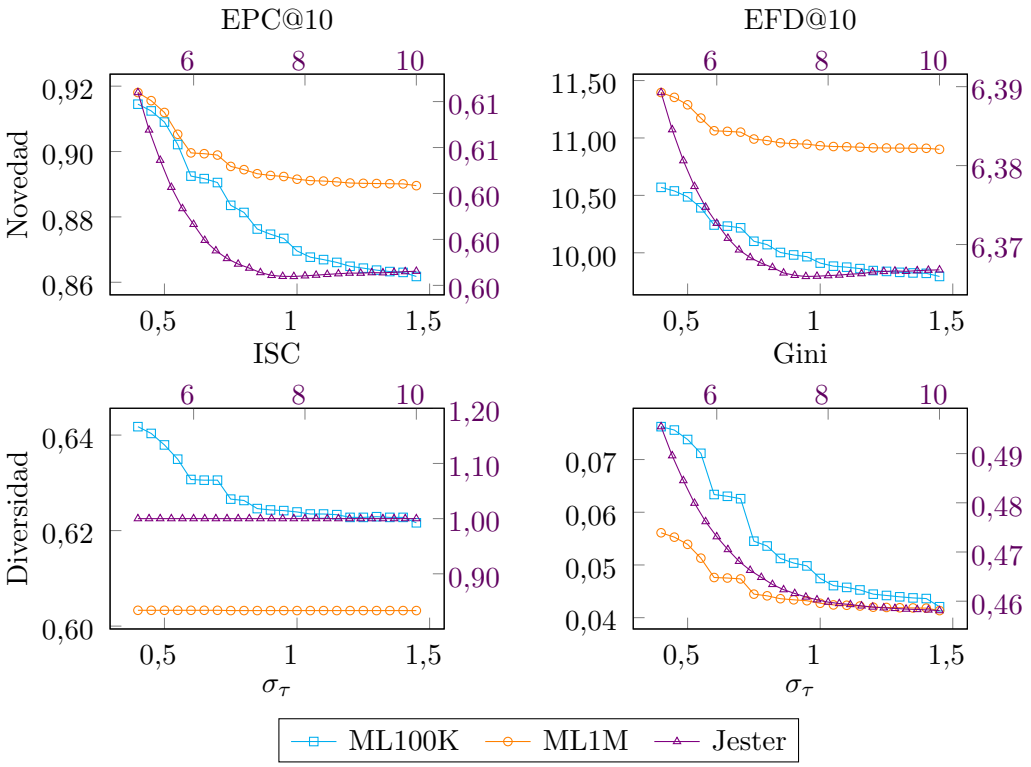


Figura 5.12: Métricas de novedad y diversidad usando como estrategia de decisión la **incertidumbre de la predicción** del algoritmo KNN en tres *datasets* distintos: ML100K, ML1M y Jester.

al *threshold* establecido. En cuanto a la cobertura de ítem ($ISC$), mejora levemente para los valores más bajos de $\sigma_\tau$ pero, en general, se mantiene estable.

Con los *datasets* de ML1M y Jester se obtuvieron resultados similares tal y como se ve en la Figura 5.11. En los tres casos la variación del umbral no influye ni en la precisión ni en la cobertura de usuario. Por tanto, este parámetro, que en el algoritmo *Variational Bayesian* influía bastante en su eficacia, en un algoritmo KNN parece no tener ningún efecto, independientemente del conjunto de datos o de la métrica de eficacia.

Si se mira, como en otras ocasiones, la influencia de este parámetro en otras dimensiones de evaluación (novedad y diversidad), obtenemos resultados un poco diferentes. Como se puede observar en la Figura 5.12, la novedad se ve afectada por los valores de $\sigma_\tau$ en los tres *datasets*, en particular, la novedad disminuye a la vez que crece el umbral. Este resultado parece contradecir lo observado anteriormente: cuanto más restrictivo es el parámetro (valores menores) se devuelven ítems más novedosos, al contrario de lo que se mostraba en el experimento usando *Variational Bayesian* (Figura 5.9). Un motivo para esta inconsistencia es la menor base teórica que se tiene al calcular el paŕametro $\sigma_\tau$ en este algoritmo; además, esta desviación típica ponderada combina tanto el soporte como la incertidumbre en la predicción: un ítem puntuado por muy pocos vecinos puede tener una incertidumbre muy baja si esos vecinos han coincidido en sus valores, no obstante, la confianza que se tiene en dicha incertidumbre es menor que cuando muchos vecinos han puntuado un mismo ítem, lo cual puede generar incertidumbres mayores o menores, pero en cualquier caso, con una mayor confianza en su resultado. Según el modelo propuesto, esta confianza en la incertidumbre no se tiene en cuenta, y esperamos poder considerarlo e incorporarlo en la toma de decisiones en el futuro.

Por otro lado, del mismo modo que en el experimento con *Variational Bayesian*, se aplicó la

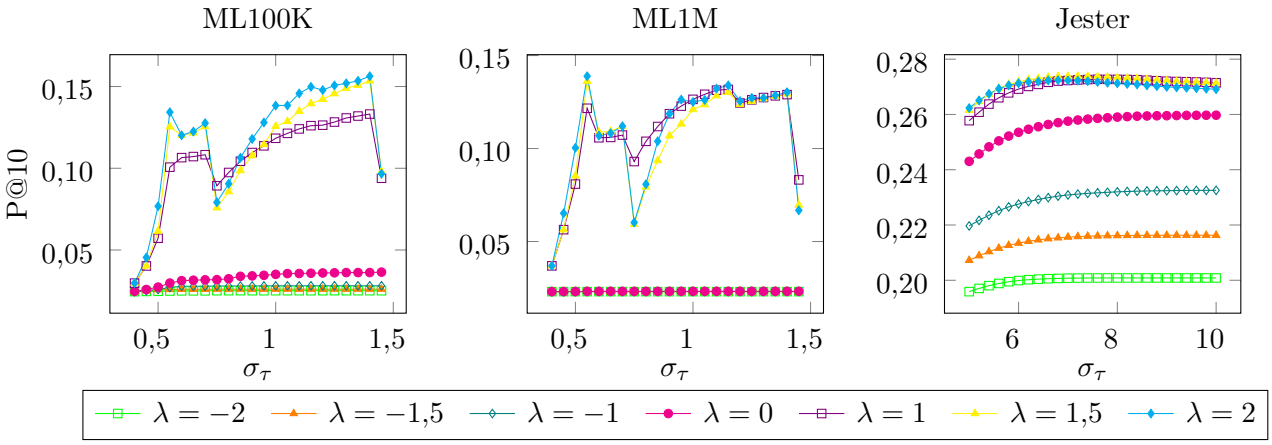


Figura 5.13: Precisión ($P$@10) usando como estrategia de decisión la **incertidumbre de la predicción** del algoritmo KNN y aplicando la Ecuación 3.1 con distintos valores de $\lambda$ en tres *datasets* distintos: ML100K, ML1M y Jester.

Ecuación 3.1, además de la toma de decisión según la incertidumbre de la predicción. Se puede ver en la Figura 5.13 que tanto en el caso de ML100K como en el de ML1M se obtienen importantes mejoras en la precisión cuando se sobreestima, es decir, cuando $\lambda > 0$. El conjunto Jester presenta también dicha mejora pero más levemente.

En comparación con el experimento de *Variational Bayesian*, no se observa un comportamiento similar de las diferentes curvas entre sí ni entre los datasets como ocurría en *Variational Bayesian*, donde las curvas tenían la misma forma para todo $\lambda$ y en los tres *datasets*. Además, en *Variational Bayesian* la precisión mejoraba a medida que los $\sigma_\tau$ disminuían, se hacían más restrictivos, mientras que en ML100K y ML1M es para valores más altos donde se alcanza el máximo. Por otro lado, en el caso de *Variational Bayesian* las mejoras en la precisión se obtenían subestimando en lugar de sobreestimando. El hecho de que en KNN se consiga mejorar cuando se sobreestima puede estar relacionado con que cuantos más vecinos forman parte del cálculo de la estimación es más probable que la desviación sea más alta favoreciendo a estos ratings cuando los $\lambda$s son positivos.

Como se observó en el experimento de toma de decisión según el soporte de la predicción de la Sección 3.2, cuanto mayor es el número de vecinos que ha puntuado al ítem más fiable es la estimación y ello influye favorablemente en la precisión. La ventaja que presenta utilizar la Ecuación 3.1 con $\lambda$s positivos y la toma de decisión según la incertidumbre frente a la toma de decisión según el soporte de la predicción es que, como se verá a continuación, las precisiones mejoran manteniendo una cobertura de usuario del 100 %.

La Tabla 5.7 recoge los resultados de ML100K con $\lambda = 2$ para poder analizar cómo varía la precisión a medida que varía el *threshold* $\sigma_\tau$, puesto que en la Tabla 5.6 con $\lambda = 0$ no se podía ver este efecto. El valor máximo se alcanza en $\sigma_\tau = 1{,}4$ donde $P$@10 es igual a 0,156, un 116,7 % mejor que el baseline de $\lambda = 2$ y un 321,6 % respecto al baseline general ($\lambda = 0$) del KNN utilizado. Para valores menores y mayores la precisión tiene peores resultados aunque todos ellos, excepto en el caso más restrictivo, mejoran el baseline del algoritmo. La cobertura de usuario, como ya se ha adelantado, se mantiene en el 100 % mientras que la de ítem empeora en la mayoría de los casos. Debido a que la mejora en precisión es tan alta, todas las métricas de combinación de cobertura y precisión coinciden en destacar el algoritmo con $\sigma_\tau = 1{,}4$ como el mejor, incluso $IC$ y $RIC$, que son las únicas que tienen la cobertura de ítem en cuenta.

| $\sigma_\tau$ | P@10 | $USC$ | $ISC$ | $F_1$ | $F_2$ | $F_{0,5}$ | $G_{1,1}$ | $G_{1,2}$ | $G_{2,1}$ | $UC$ | $RUC$ | $IC$ | $RIC$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0,072 | **100,0** | 64,7 | 0,135 | 0,281 | 0,089 | 0,269 | 0,417 | 0,174 | 0,072 | 0,072 | 0,001 | 0,017 |
| 0,40 | 0,030 | 100,0 | 64,1 | 0,058 | 0,133 | 0,037 | 0,172 | 0,310 | 0,096 | 0,030 | 0,030 | 0,000 | 0,015 |
| 0,45 | 0,045 | 100,0 | **64,8** | 0,087 | 0,192 | 0,056 | 0,213 | 0,357 | 0,127 | 0,045 | 0,045 | 0,001 | 0,016 |
| 0,50 | 0,077 | 100,0 | 64,1 | 0,143 | 0,294 | 0,094 | 0,277 | 0,425 | 0,181 | 0,077 | 0,077 | 0,001 | 0,019 |
| 0,55 | 0,134 | 100,0 | 58,8 | 0,237 | 0,437 | 0,162 | 0,366 | 0,512 | 0,262 | 0,134 | 0,134 | 0,002 | 0,022 |
| 0,60 | 0,120 | 100,0 | 53,5 | 0,215 | 0,406 | 0,146 | 0,347 | 0,493 | 0,243 | 0,120 | 0,120 | 0,001 | 0,021 |
| 0,65 | 0,122 | 100,0 | 53,0 | 0,218 | 0,411 | 0,149 | 0,350 | 0,497 | 0,247 | 0,122 | 0,122 | 0,001 | 0,021 |
| 0,70 | 0,128 | 100,0 | 52,7 | 0,226 | 0,422 | 0,155 | 0,357 | 0,503 | 0,254 | 0,128 | 0,128 | 0,001 | 0,022 |
| 0,75 | 0,079 | 100,0 | 58,7 | 0,147 | 0,301 | 0,097 | 0,281 | 0,429 | 0,184 | 0,079 | 0,079 | 0,001 | 0,018 |
| 0,80 | 0,091 | 100,0 | 58,4 | 0,166 | 0,332 | 0,111 | 0,301 | 0,449 | 0,202 | 0,091 | 0,091 | 0,001 | 0,019 |
| 0,85 | 0,106 | 100,0 | 57,1 | 0,192 | 0,373 | 0,130 | 0,326 | 0,474 | 0,225 | 0,106 | 0,106 | 0,001 | 0,020 |
| 0,90 | 0,118 | 100,0 | 56,1 | 0,211 | 0,401 | 0,143 | 0,343 | 0,490 | 0,240 | 0,118 | 0,118 | 0,001 | 0,021 |
| 0,95 | 0,128 | 100,0 | 55,0 | 0,227 | 0,423 | 0,155 | 0,358 | 0,504 | 0,254 | 0,128 | 0,128 | 0,001 | 0,022 |
| 1,00 | 0,138 | 100,0 | 52,2 | 0,243 | 0,445 | 0,167 | 0,372 | 0,517 | 0,267 | 0,138 | 0,138 | 0,002 | 0,023 |
| 1,05 | 0,138 | 100,0 | 51,1 | 0,243 | 0,445 | 0,167 | 0,372 | 0,517 | 0,268 | 0,138 | 0,138 | 0,002 | 0,023 |
| 1,10 | 0,146 | 100,0 | 50,2 | 0,254 | 0,460 | 0,176 | 0,382 | 0,526 | 0,277 | 0,146 | 0,146 | 0,002 | 0,023 |
| 1,15 | 0,150 | 100,0 | 49,8 | 0,261 | 0,468 | 0,181 | 0,387 | 0,531 | 0,282 | 0,150 | 0,150 | 0,002 | 0,024 |
| 1,20 | 0,148 | 100,0 | 49,4 | 0,258 | 0,464 | 0,178 | 0,384 | 0,529 | 0,280 | 0,148 | 0,148 | 0,002 | 0,024 |
| 1,25 | 0,151 | 100,0 | 49,0 | 0,262 | 0,470 | 0,182 | 0,388 | 0,532 | 0,283 | 0,151 | 0,151 | 0,002 | 0,024 |
| 1,30 | 0,152 | 100,0 | 49,0 | 0,264 | 0,473 | 0,183 | 0,390 | 0,534 | 0,285 | 0,152 | 0,152 | 0,002 | 0,024 |
| 1,35 | 0,153 | 100,0 | 48,6 | 0,266 | 0,476 | 0,185 | 0,392 | 0,535 | 0,287 | 0,153 | 0,153 | 0,002 | 0,024 |
| 1,40 | **0,156** | 100,0 | 48,5 | **0,270** | **0,481** | **0,188** | **0,395** | **0,539** | **0,290** | **0,156** | **0,156** | **0,002** | **0,025** |
| 1,45 | 0,097 | 100,0 | 60,3 | 0,176 | 0,348 | 0,118 | 0,311 | 0,459 | 0,211 | 0,097 | 0,097 | 0,001 | 0,019 |

Tabla 5.7: Comparación de métricas de evaluación cuando se usa la estrategia de toma de decisión basada en la **incertidumbre de la predicción**, es decir, cuando se varía el *threshold* $\sigma_\tau$ y se aplica la Ecuación 3.1 con $\lambda = 2$ para el algoritmo de KNN con $k = 10$ y similitud coseno sobre el *dataset* ML100K.

# 6

# Conclusiones y Trabajo Futuro

Para acabar este documento, este último capítulo recoge las conclusiones obtenidas del proyecto desarrollado para este Trabajo Fin de Máster y las tareas con las que continuar en un trabajo futuro.

## 6.1. Conclusiones

En este trabajo hemos investigado cómo aumentar la confianza del usuario en las recomendaciones del sistema incorporando una cierta toma de decisión en el mismo. Para ello, se han propuesto tres estrategias para que el sistema pueda incorporar un parámetro que le indique qué estimaciones son de confianza, o dicho de otro modo, cuáles han sido calculadas a partir de datos consistentes y fiables. Con ello, el sistema puede decidir no recomendar aquellas recomendaciones con una confianza inferior a un determinado umbral. Dichas estrategias son: (1) Independiente del modelo, (2) Según el soporte y (3) Según la incertidumbre.

Tras el exahustivo análisis realizado a lo largo del Capítulo 5 se han extraído algunas conclusiones importantes. Una de las principales conclusiones es que calcular de manera exacta la incertidumbre de la predicción en el método de factorización de matrices propuesto, *Variational Bayesian*, y utilizarla como método de toma de decisión proporciona unos resultados muy buenos, consiguiendo mejorar la precisión *baseline* del algoritmo en un 250 % con el *dataset* de ML100K y $\lambda = 0{,}82$. Con ello se verifica la teoría de que a menor incertidumbre, mayor confianza tiene el sistema en la estimación, aumentando de esta forma la probabilidad de acierto. Sin embargo, si se aplica esta idea al algoritmo de $K$ Vecinos Próximos no se obtienen resultados tan buenos debido a que, en este algoritmo y al contrario que en el caso de *Variational Bayesian*, no se proporciona una fórmula teórica exacta de la incertidumbre de cada predicción. A pesar de ello, sí se observan mejoras en términos de precisión.

Por otro lado, el método según el soporte de la predicción en KNN proporciona mejoras de hasta un 562 %, en el caso de ML100K, cumpliendo la hipótesis de que cuanto mayor es el soporte, en este caso el número de vecinos, sobre el que se basa la estimación, más fiable es el resultado y mayor la probabilidad de acierto.

En el caso de la toma de decisión independiente del modelo, la variación del parámetro de confianza $\gamma$ no muestra importantes variaciones en las métricas evaluadas.

Estas tres estrategias se han evaluado en términos de precisión, cobertura, novedad y diversidad. Debido a que el sistema es capaz de no recomendar un ítem a un usuario si no lo considera de confianza, la cobertura de usuario y de ítem pueden decrecer a pesar del aumento de la precisión. Por ello, se han propuesto diversas métricas para poder evaluar estos nuevos algoritmos que combinen resultados de varias métricas en un solo valor. Por un lado, se propone combinar la precisión y la cobertura a través de una media armónica (F-score) o a través de una media geométrica (G-score), ambas parametrizadas. Con esta parametrización se consigue dar más o menos importancia a una de las métricas según el objetivo de la medición. Por otro lado, se proponen la métricas *correctness* con las que poder evaluar cuándo un sistema decide no recomendar. Estas métricas premian no contestar frente a fallar en su sugerencia.

En particular, se propusieron cuatro métricas *correctness*: $UC$, $RUC$, $IC$ y $RIC$. En el caso de $UC$ y $RUC$ se toma la perspectiva del usuario, es decir, por cada usuario se evalúa la lista de recomendaciones recibida calculando la proporción de ítems relevantes teniendo por base los $N$ ítems que debería haber recibido. Si la lista es incompleta, a esta proporción de acierto se suma una cierta cantidad de la proporción de ítems no retornados y es en este punto donde ambas métricas se diferencian. $IC$ y $RIC$ parten de la perspectiva del ítem y evalúan por cada ítem la lista de usuarios del sistema midiendo la proporción de usuarios a los que se ha recomendado el ítem y era relevante, premiando no ser recomendado frente a fallar. Al igual que antes, la diferencia entre ambas métricas es cómo premian no recomendar.

Como se pudo ver en el experimento de *Variational Bayesian* variando el umbral de la incertidumbre $\sigma_\tau$, en la Sección 5.2.3.1, tanto $UC$ como $RUC$ eligen como mejor opción el $\sigma_\tau$ para el que la ganancia de precisión no conlleva un gran dencenso de la cobertura de usuario (esto ocurre también en el resto de experimentos, pero este caso es donde mejor se observa este comportamiento). La ventaja de estas métricas frente a $F_\beta$ y $G_{\alpha_1,\alpha_2}$ es que, estas últimas deben ser parametrizadas para su uso añadiendo, por tanto, la subjetividad de la persona que realiza el experimento, mientras que las métricas *correctness* no. $IC$ y $RIC$ tienen más en cuenta el equilibrio entre precisión y cobertura de ítem, aunque indirectamente también la de usuario.

## 6.2. Trabajo futuro

En el futuro, nos gustaría ampliar los experimentos a otros *datasets*, como por ejemplo, LastFM[1], *dataset* sobre música, o MovieTweeting[2], *dataset* sobre películas, para verificar los resultados ya obtenidos y hacerlos más sólidos. De igual forma, pretendemos continuar estudiando nuevas estrategias para incorporar la toma de decisiones en diversos algoritmos de recomendación y evaluar su efecto en diferentes dimensiones, como se ha mostrado en esta memoria. Además, ahora que hemos sido capaces de definir unas métricas que combinan el concepto de precisión y cobertura, consideramos interesante extender este estudio a otras métricas no basadas en la precisión, como NDCG, o incluso a métricas de diversidad como el índice de Gini. Para ello, prevemos que será necesario estudiar en detalle las métricas que se quieran combinar y plantear los mismos pasos que se han mostrado en el Capítulo 4. De igual forma, nos gustaría analizar si se podrían definir métricas equivalentes a la familia *correctness* cuando se usan metodologías de evaluación diferentes, más allá de TestItems [6].

Por otro lado, un problema que sigue estando abierto es el ser capaz de decidir cuándo el incremento de una métrica (en nuestro caso, precisión) compensa la pérdida de otra métrica (cobertura). En este trabajo hemos definido y experimentado con varias métricas pero sigue sin estar claro que el ranking de sistemas generado por ellas sea mejor que el generado por precisión,

[1] https://grouplens.org/datasets/hetrec-2011/
[2] https://github.com/sidooms/MovieTweetings

por ejemplo. Para ello, seguramente sea necesario hacer estudios con usuarios y determinar si las métricas aquí definidas correlacionan más o menos con la satisfacción final del usuario.

De manera adicional, el aspecto psicológico de las recomendaciones también debería ser considerado, ya que si un usuario espera recibir $N$ recomendaciones, su confianza en el sistema puede ser menor cuando menos de esas $N$ recomendaciones son devueltas, a pesar de que sean todas relevantes. De hecho, este efecto podría llegar a ser contraproducente, ya que por la llamada *paradoja de la elección* los usuarios se pueden llegar a frustar si todas las elecciones que se le muestran son igual de relevantes [30]. Por lo tanto, y como se ha llegado a estudiar en el contexto de los sistemas de recomendación [7], un algoritmo que sólo devolviera ítems relevantes para el usuario podría tener efectos colaterales negativos, como una mayor sobrecarga en el proceso cognitivo del usuario así como tiempos de adquisición mayores, aunque estas consecuencias también dependen del tamaño de la lista devuelta, por lo que sigue quedando abierto el problema de encontrar el número óptimo de recomendaciones a devolver, donde tanto los algoritmos como las métricas propuestas en este Trabajo de Fin de Máster pueden ayudar a resolver algunas de estas situaciones.

# Glosario

- **EFD:** Expected Free Discovery. Métrica de novedad definida en la Sección 2.3.2.2.
- **EPC:** Expected Popularity Complement. Métrica de novedad definida en la Sección 2.3.2.2.
- **IC:** Ítem Correctness. Métrica propuesta para combinar cobertura de ítem y precisión, definida en la Sección 4.2.2.
- **ISC:** Item Space Coverage (cobertura de ítem). Métrica de cobertura definida en la Sección 2.3.2.2.
- **KNN:** K Nearest Neighbors (K Vecinos Próximos). Más información en la Sección 2.2.1.
- **ML100K:** *dataset* hecho público por MovieLens y que contiene alrededor de 100,000 puntuaciones.
- **ML1M:** *dataset* hecho público por MovieLens con cerca de un millón de puntuaciones.
- **P@N:** Precisión at N, métrica de evaluación definida en la Sección 2.3.2.1.
- **R@N:** Recall at N, métrica de evaluación definida en la Sección 2.3.2.1.
- **RIC:** Recall Ítem Correctness. Métrica propuesta para combinar cobertura de ítem, recall y precisión, definida en la Sección 4.2.2.
- **RUC:** Recall User Correctness. Métrica propuesta para combinar cobertura de usuario, recall y precisión, definida en la Sección 4.2.2.
- **UC:** User Correctness. Métrica propuesta para combinar cobertura de usuario y precisión, definida en la Sección 4.2.2.
- **USC:** User Space Coverage (cobertura de usuario). Métrica de cobertura definida en la Sección 2.3.2.2.

# Bibliografía


[1] Bayesian inference. `https://en.wikipedia.org/wiki/Bayesian_inference`, 2016.

[2] Pondered average. `https://en.wikipedia.org/wiki/Weighted_arithmetic_mean`, 2016.

[3] Variational bayesian methods. `https://en.wikipedia.org/wiki/Variational_Bayesian_methods`, 2016.

[4] Gediminas Adomavicius, Sreeharsha Kamireddy, and Youngok Kwon. *Towards more confident recommendations: Improving recommender systems using filtering approach based on rating variance*, pages 152–157. Social Science Research Network, 2007.

[5] Gediminas Adomavicius and YoungOk Kwon. Improving aggregate recommendation diversity using ranking-based techniques. *IEEE Transactions on Knowledge and Data Engineering*, 24(5):896–911, 2012.

[6] Alejandro Bellogin, Pablo Castells, and Ivan Cantador. Precision-oriented evaluation of recommender systems: an algorithmic comparison. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 333–336. ACM, 2011.

[7] Dirk G. F. M. Bollen, Bart P. Knijnenburg, Martijn C. Willemsen, and Mark P. Graus. Understanding choice overload in recommender systems. In *RecSys*, pages 63–70. ACM, 2010.

[8] Pablo Castells, Neil J Hurley, and Saul Vargas. Novelty and diversity in recommender systems. In *Recommender Systems Handbook*, pages 881–918. Springer, 2015.

[9] Nick Craswell. Mean reciprocal rank. In *Encyclopedia of Database Systems*, pages 1703–1703. Springer, 2009.

[10] Marco de Gemmis, Pasquale Lops, Cataldo Musto, Fedelucio Narducci, and Giovanni Semeraro. Semantics-aware content-based recommender systems. In *Recommender Systems Handbook*, pages 119–159. Springer, 2015.

[11] Michael D Ekstrand, John T Riedl, and Joseph A Konstan. Collaborative filtering recommender systems. *Foundations and Trends in Human-Computer Interaction*, 4(2):81–173, 2011.

[12] Asela Gunawardana and Guy Shani. Evaluating recommender systems. In *Recommender Systems Handbook*, pages 265–308. Springer, 2015.

[13] Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl. An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Inf. Retr.*, 5(4):287–310, 2002.

[14] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM*, pages 263–272. IEEE Computer Society, 2008.

[15] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

[16] George Karypis. Evaluation of item-based top-n recommendation algorithms. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 247–254. ACM, 2001.

[17] Yehuda Koren, Robert Bell, Chris Volinsky, et al. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[18] Yehuda Koren and Robert M. Bell. Advances in collaborative filtering. In *Recommender Systems Handbook*, pages 77–118. Springer, 2015.

[19] Yew Jin Lim and Yee Whye Teh. Variational bayesian approach to movie rating prediction. In *Proceedings of KDD cup and workshop*, volume 7, pages 15–21. Citeseer, 2007.

[20] Greg Linden, Brent Smith, and Jeremy York. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, 7(1):76–80, 2003.

[21] Sean M. McNee, Shyong K. Lam, Catherine Guetzlaff, Joseph A. Konstan, and John Riedl. Confidence displays and training in recommender systems. In *INTERACT*. IOS Press, 2003.

[22] Shinichi Nakajima and Masashi Sugiyama. Theoretical analysis of bayesian matrix factorization. *Journal of Machine Learning Research*, 12(Sep):2583–2648, 2011.

[23] Xia Ning, Christian Desrosiers, and George Karypis. A comprehensive survey of neighborhood-based recommendation methods. In *Recommender Systems Handbook*, pages 37–76. Springer, 2015.

[24] John O'Donovan and Barry Smyth. Trust in recommender systems. In *IUI*, pages 167–174. ACM, 2005.

[25] Anselmo Peñas and Álvaro Rodrigo. A simple measure to assess non-response. In Dekang Lin, Yuji Matsumoto, and Rada Mihalcea, editors, *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pages 1415–1424. The Association for Computer Linguistics, 2011.

[26] Francesco Ricci, Lior Rokach, and Bracha Shapira, editors. *Recommender Systems Handbook*. Springer, 2015.

[27] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th international conference on Machine learning*, pages 880–887. ACM, 2008.

[28] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *NIPS*, volume 20, pages 1–8, 2011.

[29] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

[30] Barry Schwartz. *The Paradox of Choice: Why More Is Less*. Harper Perennial, January 2005.

[31] Nathan Srebro, Tommi Jaakkola, et al. Weighted low-rank approximations. In *Icml*, volume 3, pages 720–727, 2003.

[32] Saúl Vargas and Pablo Castells. Rank and relevance in novelty and diversity metrics for recommender systems. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 109–116. ACM, 2011.

[33] Yi Zhang and James P. Callan. Maximum likelihood estimation for filtering thresholds. In *SIGIR*, pages 294–302. ACM, 2001.