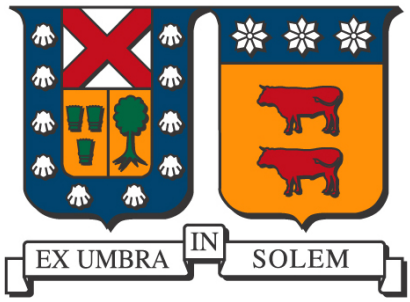
2017

CLUSTERING Y DIVERSIDAD EN SISTEMAS DE RECOMENDACIÓN TOP-N

TORRES RUDLOFF, NICOLÁS IGNACIO

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
SANTIAGO – CHILE



"CLUSTERING Y DIVERSIDAD EN SISTEMAS DE RECOMENDACIÓN TOP-N"

NICOLÁS IGNACIO TORRES RUDLOFF

TESIS PRESENTADA COMO REQUERIMIENTO PARCIAL PARA OPTAR AL GRADO ACADÉMICO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA INFORMÁTICA

PROFESOR GUÍA: DR. MARCELO MENDOZA
PROFESOR CORREFERENTE: DR. RICARDO ÑANCULEF
PROFESOR CORREFERENTE EXTERNO: DR. DENIS PARRA

NOVIEMBRE – 2017

Resumen

Los Sistemas de Recomendación ayudan a la gente a tomar decisiones frente a grandes volúmenes de información. El Filtrado Colaborativo (CF) es una de las técnicas más exitosas en sistemas de recomendación. Está basada en la idea de que la gente obtiene las mejores recomendaciones de alguien con gustos similares. En general, existen dos técnicas principales en CF: model-based y memory-based. El primero, se basa en un modelo aprendido para estimar recomendaciones. El segundo, se basa en un vecindario (Neighborhood-based CF) calculado a través de funciones de similaridad entre usuarios (User-based CF) o productos (Item-based CF). UBCF, es efectivo, pero sufre problemas de escalabilidad en datasets de gran escala. Para enfrentar esos problemas de escalabilidad, clustering-based CF restringen la búsqueda de usuarios a clusters, en lugar de trabajar sobre toda la base de datos. Sin embargo, hay un trade-off entre eficiencia y calidad predictiva.

En esta Tesis de MII, se presenta un nuevo enfoque que combina las ventajas de UBCF y cluster-based CF introduciendo una función de distancia basada en prototipos para calcular vecindarios. La búsqueda de usuarios y productos similares se expande explotando la estructura global de los clusters para inferir las distancias entre usuarios. Estudios experimentales en data de benchmark demostraron que la propuesta es factible y se muestra competitiva en relación a métodos del Estado del Arte. Sin embargo, los sistemas de recomendación son frecuentemente evaluados usando indices basados en variantes y extensiones de métricas relacionadas con precisión. Como estas métricas están sesgadas hacia productos populares, una lista de recomendaciones solo necesitan incluir un par de productos populares para funcionar bien. Para entregar una evaluación más robusta y realista del método de recomendación propuesto, en la segunda parte de la Tesis, se proponen nuevos enfoques para evaluar novedad y diversidad en sistemas de recomendación. Resultados experimentales sugieren que el método propuesto, basado en modelos de clustering, puede favorecer la recuperación de diversidad.

Palabras Clave: Sistemas de Recomendación, Clustering, Evaluación, Novedad, Diversidad.

Abstract

Recommender Systems aim to help people dealing with information overload. Collaborative filtering (CF) is one of the most successful techniques used in recommender systems. It is based on the idea that people often get the best recommendations from someone with similar tastes to themselves. Broadly, there are model-based and memory-based CF techniques. The former, learn a model to make predictions. The latter, uses similarity measures to compute the proximity between users (User-based) or items (Item-based) and build a neighborhood (Neighborhood-based CF). UBCF, while effective, suffers from scalability problems as the database grows. To address the scalability issue, clustering-based CF algorithms constraint the seek of users within small user clusters instead of the entire database. However, there is a trade-off between efficiency and prediction accuracy.

In this Msc. Thesis, we present a novel approach that combines the advantages of UBCF and cluster-based CF methods by introducing a cluster-based distance function used for neighborhood computation. To expand the search of relevant users/items we use a novel measure that is able to exploit the global cluster structure to infer user's distances. Empirical studies on widely known benchmark datasets suggest that our proposal is feasible. Nevertheless, recommender systems are frequently evaluated using indexes based on variants and extensions of precision-like measures. As these measures are biased toward popular items, a list of recommendations just need to include a few popular items to perform well. To provide a more robust and realistic evaluation of our proposed method, in the second part of this Thesis, new approaches for novelty and diversity evaluation have been proposed. Experimental results show that our proposed method, based on cluster models, can promote diversity retrieval.

Keywords: Recommender Systems, Clustering, Evaluation, Novelty, Diversity.

Índice general

Índice de figuras

ÍNDICE DE FIGURAS

Índice de tablas

Capítulo 1

Introducción

Esta Tesis se enfoca en el diseño de un sistema de recomendación basado en técnicas de clustering capaz de competir con algoritmos del Estado del Arte en términos de calidad predictiva, manteniendo la eficiencia del modelo original. Como las medidas actuales de calidad no permiten evaluar correctamente la diversidad de los sistemas de recomendación, se propone un nuevo esquema de evaluación enfocado en las necesidades de satisfacción de calidad desde el punto de vista del usuario. Esta Tesis está dividida explícitamente en dos partes. En la Parte I se estudian los algoritmos de clustering, se propone un nuevo modelo basado en distancias entre los prototipos y se evalúa su calidad en términos de precisión en listas de recomendación. En la Parte II, se propone un nuevo enfoque de evaluación destinado a medir diversidad en listas de recomendación.

1.1. Objetivos

1.1.1. Objetivos Generales

1. Diseñar un sistema de recomendación basado en técnicas de clustering sobre usuarios o productos.

2. Proponer una nueva función de proximidad que sea capaz de explotar la estructura global de los clusters.

3. Comparar el desempeño del modelo propuesto con respecto a métodos competitivos del Estado del Arte.

1.1.2. Objetivos Específicos

1. Medir la calidad de las técnicas de clustering aplicadas al contexto de recomendación Top-N.

2. Explorar el impacto de las funciones de proximidad en sistemas de recomendación.

3. Documentar el método propuesto y los experimentos realizados en una Tesis MII.

4. Presentar los resultados de la Tesis a la comunidad científica en un artículo de revista de corriente principal.

1.2. Hipótesis del Trabajo

Es posible reducir la brecha en términos de precisión que existe entre métodos de recomendación basados en modelos de clustering y algoritmos basados en vecinos más cercanos (k-NN). Esto se consigue a través de dos factores principales:

- Modificando la función de proximidad utilizada para seleccionar a los vecinos más cercanos.

- Generando una solución de clustering con clusters bien formados y razonablemente balanceados.

Restricción

- Mantener la eficiencia de los métodos de recomendación basados en clustering.

1.3. Metodología y Plan de Trabajo

La metodología se basa en el modelo de procesos utilizado en minería de datos conocido como CRISP-DM (Cross Industry Standard Process for Data Mining).

CRISP-DM resume el proceso en seis fases principales:

1.3.1. Comprensión del tema

Es la fase inicial y se centra en entender el problema, definiendo los objetivos (Sección 1.1) y requisitos del proyecto. Ese entendimiento del tema se convierte en una definición formal del problema (Sección 3.3) con un plan preliminar diseñado para alcanzar los objetivos. La etapa incluye el Estudio de las investigaciones más relevantes (Sección 3.2).

1.3.2. Comprensión de los datos

Esta fase busca familiarizarse con los datos y descubrir los primeros patrones ocultos en los mismos. Incluye una descripción inicial de los datos y un análisis exploratorio de los datos (Sección 3.4.1).

1.3.3. Preparación de los datos

Esta fase cubre todas las actividades cuyo fin es construir el conjunto de datos. Las tareas, que son ejecutadas en múltiples oportunidades y sin orden, incluyen la manipulación de tablas, registros, atributos y limpieza de datos.

1.3.4. Modelado

En esta fase se seleccionan y aplican técnicas de modelado de datos, y se calibran los parámetros para obtener resultados óptimos. Hay varias técnicas que tienen requerimientos específicos para la forma de los datos, por lo que frecuentemente es necesario volver a la fase anterior.

- Construcción y descripción el modelo (Sección 3.3.2).
- Definir las técnicas de evaluación (Sección 3.4.2).

1.3.5. Evaluación

En esta etapa se evalúa la calidad del modelo propuesto desde una perspectiva de análisis de datos.

- Evaluar los resultados (Sección 3.5).

- Revisar el proceso (Sección 3.6).
- Establecer los pasos a seguir (Sección 3.6.1).

1.3.6. Despliegue

Esta fase corresponde a la generación de un reporte documentando los resultados obtenidos durante la investigación.

1.4. Aportes y Resultados Esperados

1.4.1. Clustering en Recomendación Top-N

En la Parte I, junto con parametrizar la función de proximidad empleada en sistemas de recomendación, cosa que no ha sido estudiada hasta el momento, tres grandes contribuciones surgen de esta Tesis:

- En sistemas reales, incluso usuarios ávidos poseen ratings sobre menos del 1 % de los productos. De acuerdo a esto, un método de recomendación basado en vecinos más cercanos puede ser incapaz de producir recomendaciones para un usuario que acaba de ingresar al sistema, debido a la falta de información utilizada para asociar alguna medida de proximidad (Cold-start). Esto trae como resultado una precisión del sistema de muy mala calidad. Se propone un sistema de recomendación capaz de manejar problemas de dispersión de datos, explotando los beneficios que traen las técnicas de clustering al enriquecer los vectores de los usuarios con información procedente del clúster al cual fue asignado.
- Tanto los métodos basados en vecinos más cercanos como los basados en clustering sobre explotan la localidad de los datos enfocándose en parte de los datos limitados por vecindarios acotados. Por el contrario, se propone un método flexible capaz de capturar la estructura global de los datos. Como los clusters agrupan a usuarios similares, los cuales suelen gustar de productos similares, las técnicas de clustering tradicionales no promueven la diversidad. Por lo tanto, la inclusión de productos recuperados desde diferentes clusters puede ayudar en el contexto de la diversificación.
- Al agrupar datos en diferentes clusters, cada grupo puede ser manipulado en una máquina diferente de manera independiente (en paralelo). Esta estrategia es útil en muchos escenarios, especialmente al trabajar con conjuntos de datos a gran escala.

La solución propuesta pretende superar el problema de escalabilidad que sufren algoritmos de recomendación tradicionales, almacenando en memoria vectores de usuarios basados en los prototipos obtenidos de la técnica de clustering. Este resultado, es aplicable a dominios de recuperación de información distribuida.

1.4.2. Diversidad en Sistemas de Recomendación

En la Parte II, la contribución de esta Tesis es un nuevo esquema de evaluación capaz de medir diversidad en listas de recomendación. Se introduce un nuevo enfoque de evaluación para sistemas de recomendación basado en cuán novedosos son los elementos que componen cada producto. Entonces, se proponen nuevas medidas para evaluar novedad en contenido.

1.5. Estructura de la Tesis

La Tesis se encuentra dividida en dos partes. La Parte I incluye los Capítulos 2 y 3, y se enfoca en estudiar distintos modelos de clustering en recomendación Top-N. Se propone y evalúa un sistema de recomendación basado en una nueva función de distancia para calcular vecindarios. En la Parte I, primero en el Capítulo 2, se entrega una introducción general a los algoritmos de clustering. En adición, se revisen notaciones, definiciones y funciones de proximidad. En el Capítulo 3, se presenta una nueva función de distancia para calcular vecindarios, evaluando su desempeño en diversos datasets. Una extensa serie de experimentos para evaluar el impacto de esta función, en conjunto al método de recomendación userCL que funciona sobre distintos modelos de clustering, son presentados. Los resultados demuestran que userCL entrega resultados promisorios, manteniendo la eficiencia del modelo original. El capítulo finaliza entregando conclusiones acerca de la etapa experimental, y varias direcciones para futura investigación en la misma línea argumental de la propuesta.

La Parte II, se enfoca en diversidad en sistemas de recomendación. Esta parte incluye los Capítulos 4 y 5. Similar a la primera parte, una revisión de las métricas dedicadas a evaluar diversidad se detalla en el Capítulo 4, finalizando con la propuesta de una serie de medidas destinadas a evaluar novedad en el contenido de los productos en listas de recomendación Top-N. En el Capítulo 5, se evalúa el método propuesto en la primera parte, utilizando las métricas propuestas en la segunda parte. Interesantes conclusiones son presentadas, y discusión para futura investigación.

1.5 ESTRUCTURA DE LA TESIS

Parte I

CLUSTERING EN RECOMENDACIÓN TOP-N

Capítulo 2

Clustering

2.1. Introducción

Clustering es una división de datos en grupos de objetos semejantes. Cada grupo, llamado *cluster*, consiste en objetos que son similares entre sí y diferentes a objetos de otros grupos. Cuando se representan los datos con pocos clusters, es posible perder ciertos detalles (parecido a los métodos de compresión de datos con pérdida), pero se logra simplificación. El modelado de datos ubica al clustering en una perspectiva emparentada con matemáticas, estadísticas y análisis numérico. Desde el punto de vista de machine learning los clusters corresponden a patrones ocultos, y la búsqueda de clusters corresponde a aprendizaje no supervisado. Entre las aplicaciones de clustering en minería de datos destacan: exploración de datos científicos, recuperación de información, sistemas de recomendación, minería de textos, bases de datos espaciales, análisis Web, entre otras.

El procedimiento de clustering estándar puede dividirse en las siguientes etapas:

1. Extraer y seleccionar las características más representativas del conjunto de datos original.
2. Diseñar el algoritmo de agrupación de acuerdo a las características del problema.
3. Evaluar el resultado de la agrupación y validar el algoritmo.
4. Entregar una explicación práctica acerca del resultado obtenido.

En el resto del capítulo, las medidas de proximidad más comunes serán introducidas en la sección 2.3, y los diferentes tipos de algoritmos de clustering se analizarán a partir de la sección 2.5. Finalizando el capítulo con un resumen tabulado de los algoritmos estudiados en la sección 2.13.

2.2. Notaciones

Con el objetivo de fijar un contexto y clarificar terminología utilizada durante el resto del documento se define lo siguiente. Considerar un conjunto de datos (dataset) X, compuesto de puntos (objetos, instancias) $x_i = (x_{i1}, \ldots, x_{id}), i = 1 : n$, en un espacio donde cada componente es un atributo (dimensión, característica) numérico o nominal.

La tarea principal de las técnicas de clustering es asignar puntos a un sistema finito de k subconjuntos (clusters). Usualmente (no siempre), los clusters no se intersectan, y su unión es igual al dataset completo con la posible excepción de los *outliers*.

$$X = C_1 \cup \cdots \cup C_k \cup C_{outliers}, \quad C_i \cap C_j = 0, \quad i \neq j. \tag{2.1}$$

2.3. Funciones de proximidad

Distancia (disimilaridad) y similaridad son la base para construir algoritmos de clustering. Se utilizan funciones de distancia o medidas de similaridad para cuantificar que tan "similares" son dos puntos de datos y poder agruparlos.

Para medir la distancia entre los datos se utiliza, por lo general, una función de la familia de distancias Minkowski.

La distancia Minkowski de orden p entre dos puntos:

$$X = (x_1, x_2, \ldots, x_d), Y = (y_1, y_2, \ldots, y_d) \in \mathbb{R}^d, \tag{2.2}$$

está definida como:

$$d(X,Y) = \left(\sum_{i=1}^{d} |x_i - y_i|^p \right)^{1/p}, \tag{2.3}$$

con $1 \leq p < \infty$. Cuando $p = 1$, d es la distancia Manhattan, que corresponde a la estimación más robusta (no se ve tan afectada por *outliers*). Para $p = 2$, d es la distancia Euclidiana, que representa estadísticamente la varianza total inter-cluster.

La familia de distancias Minkowski asumen independencia entre atributos. En cam-

bio, la distancia de Mahalanobis

$$d(X,Y) = \sqrt{(X-Y)^T \Sigma (X-Y)}, \tag{2.4}$$

con matriz de covarianza Σ, toma en cuenta la correlación entre las variables aleatorias.

Similitud Coseno. Es una medida de similaridad entre dos vectores en un espacio que posee un producto interior que mide el coseno del ángulo comprendido entre los dos vectores. La similitud de coseno se utiliza particularmente en el espacio positivo, donde el resultado está claramente delimitado en $[0,1]$. Lo que se mide es la orientación y no magnitud.

$$\text{sim}_{Cos}(X,Y) = \frac{X \cdot Y}{\|X\|\|Y\|} = \frac{\sum_{i=1}^{d} x_i y_i}{\sqrt{\sum_{i=1}^{d} x_i^2}\sqrt{\sum_{i=1}^{d} y_i^2}} \tag{2.5}$$

En tanto, se habla de distancia coseno para referirse a $d_{Cos}(X,Y) = 1 - \text{sim}_{Cos}(X,Y)$. Aunque, es importante destacar que esta función no es una métrica de distancia propiamente tal porque no cumple con la propiedad de la desigualdad triangular. Una de las razones que explican la popularidad de la similaridad coseno es su eficiencia para evaluar, especialmente, vectores dispersos, ya que solo se necesita considerar dimensiones no nulas.

Coeficiente de correlación de Pearson. Es una medida de la relación lineal entre dos variables aleatorias cuantitativas. El valor del índice de correlación varía en el intervalo $[-1,1]$. Si $r = 0$, no existe relación lineal. Pero, esto no necesariamente implica que las variables son independientes: pueden existir todavía relaciones no lineales entre las dos variables.

$$\rho(X,Y) = \frac{\sum_{i=1}^{d} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{d} (x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{d} (y_i - \bar{y})^2}} \tag{2.6}$$

Medida	Función	Detalles y Resultados
Manhattan	$d(X,Y) = \left(\sum_{i=1}^{d} \lvert x_i - y_i \rvert \right)$	Caso especial de Minkowski para $p = 1$. Tiende a formar clusters hiperrectangulares.
Euclidiana	$d(X,Y) = \left(\sum_{i=1}^{d} \lvert x_i - y_i \rvert^2 \right)^{1/2}$	Caso especial de Minkowski para $p = 2$. Tiende a formar clusters hiperesféricos.
Mahalanobis	$d(X,Y) = \sqrt{(X-Y)^T \Sigma (X-Y)}$, Σ : matriz de covarianza dentro del cluster.	Invariante a cualquier transformación lineal no-singular. Tiende a formar clusters hiperelipsoidales.
Pearson	$\rho(X,Y) = \frac{\sum_{i=1}^{d} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{d} (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^{d} (y_i - \bar{y})^2}}$ $-1 \leq \rho(X,Y) \leq 1$.	No es una métrica. Se deriva de un coeficiente de correlación. No mide magnitud.
Coseno	$\text{sim}_{Cos}(X,Y) = \frac{\sum_{i=1}^{d} x_i y_i}{\sqrt{\sum_{i=1}^{d} x_i^2} \sqrt{\sum_{i=1}^{d} y_i^2}}$	Es eficiente, especialmente en vectores dispersos (independiente de la cantidad de componentes en el vector).
Jaccard	$J(A,B) = \frac{\lvert A \cap B \rvert}{\lvert A \cup B \rvert} = \frac{\lvert A \cap B \rvert}{\lvert A \rvert + \lvert B \rvert - \lvert A \cap B \rvert}$ $0 \leq J(A,B) \leq 1$.	Mide cardinalidades: la intersección dividido por la unión entre dos conjuntos.

Tabla 2.1: Funciones de distancia y similaridad.

2.4. Criterio de clustering

Los algoritmos de clustering deben definir un criterio que pueda ser expresado a través de una función objetivo o algún otro tipo de reglas. Se debe tener en cuenta el tipo de clusters que se esperan obtener en el dataset para que el criterio de clustering definido se ajuste bien al particionamiento de datos.

Como no existe una definición clara acerca de que es un “buen cluster”, cada algoritmo tiene su propio criterio a la hora de optimizar. Formalmente, el propósito de un algoritmo de clustering es encontrar una solución que maximice la función objetivo.

El criterio más común a optimizar es el error cuadrático medio (MSE):

$$\text{MSE} = \sum_{j=1}^{k} \sum_{X_i \in C_j} \frac{\|X_i - C_j\|^2}{n} \tag{2.7}$$

2.5. Partitional Clustering

La idea básica en algoritmos de clustering particionales es considerar el centro de los datos como prototipo del correspondiente cluster.

K-means [65] es uno de los más simples y famosos algoritmos de clustering. Se basa en prototipos porque define un centroide para cada cluster, el cual es usualmente la media de un grupo de objetos. El algoritmo comienza eligiendo k centroides iniciales, donde k representa el número de clusters deseados por el usuario, y entonces iterativamente refina y actualiza los centroides hasta que no existan más cambios (o se cumpla algún criterio de parada).

El algoritmo k-means es rápido en la práctica, $O(nkdi)$, donde n es el número de vectores d-dimensionales, k es el número de clusters, con i el número de iteraciones necesarias hasta converger. Sin embargo, en peor caso, puede ser lento cuando el número de iteraciones es grande. Una cota superior para el número de iteraciones puede ser $O(n^{kd})$ [7].

K-medoids es una mejora de K-means para lidiar con datos discretos, tomando el objeto más cercano del centro de los datos como el representante de cada cluster. El resto de los típicos algoritmos de clustering basados en particiones incluye a PAM [58], CLARA [59], CLARANS [68].

Las ventajas de este tipo de algoritmos es que presentan tiempos de complejidad relativamente bajos y alta eficiencia computacional (en general).

Las desventajas: no son adecuados para datos no convexos, son relativamente sensibles a outliers, caen fácilmente en óptimos locales, necesitan de antemano conocer el número de clusters y el resultado puede ser muy sensible a este último parámetro.

2.6. Density-based Clustering

La idea básica de este tipo de algoritmos es que la región de los datos con alta densidad de objetos es considerada dentro del mismo cluster.

Clustering basado en densidad normalmente se encarga de buscar grupos con gran concentración de objetos, separados por regiones dispersas que representan el ruido. DBSCAN [34], OPTICS [6] y Mean-shift [26] son algoritmos de clustering basados en densidad. DBSCAN visita cada punto de la base de datos, posiblemente muchas veces. OPTICS es una mejora de DBSCAN que no presenta tanta sensibilidad a los dos parámetros iniciales de DBSCAN, que son el radio del vecindario y el número mínimo de puntos en el vecindario.

En el peor caso, el tiempo de ejecución del algoritmo tiene complejidad $O(n^2)$, que podría bajarse a $O(n \log n)$ si se utiliza alguna estructura de indexación adicional.

La ventaja de los algoritmos de clustering basados en densidad, específicamente DBSCAN, es que no requieren establecer a priori el número de clusters, caso contrario a k-means. Al incluir la noción de ruido se comportan de manera más robusta frente a outliers. Sin embargo, la elección de los parámetros puede ser inapropiada y la agrupación no será precisa en data sets con grandes diferencias de densidades.

Otra ventaja es que son relativamente eficientes y adecuados para trabajar sobre datos con formas arbitrarias.

2.7. Hierarchical Clustering

La idea básica en este tipo de algoritmos de clustering es construir relaciones jerárquicas entre los datos con el fin de poder agruparlos.

En clustering jerárquico se busca construir una jerarquía de clusters. La estrategia a seguir para lograrlo puede ser aglomerativa o divisiva. La primera, es un acercamiento ascendente o "*bottom up*": cada objeto comienza en su propio cluster, y los clusters son mezclados en pares de tal forma que uno sube en la jerarquía. La segunda, es un acercamiento descendente o "*top down*": todos los objetos comienzan en un solo cluster, y se realizan divisiones recursivamente tal que uno baja en la jerarquía.

La complejidad en tiempo de ejecución de los algoritmos de clustering jerárquico es a lo menos $O(n^2)$.

Algoritmos jerárquicos típicos incluyen a BIRCH [88], CURE [39], ROCK [40], Chamaleon [55]. BIRCH realiza el clustering construyendo un árbol de características (CF tree), sobre el cual cada nodo representa un subcluster. El CF tree crece dinámicamente cuando nuevos objetos de datos aparecen. CURE, se adapta a datos a gran escala, realiza un muestreo aleatorio para agrupar muestras de datos separadamente y al final integra los resultados. ROCK es una mejora de CURE para lidiar con datos enumerados, que toma en consideración el efecto de la similaridad alrededor de los datos con

respecto al cluster. Chamaleon, al comienzo divide los objetos originales en clusters con pocos datos basándose en el grafo de vecinos más cercanos, y entonces los grupos más pequeños se mezclan en un cluster más grande basándose en algoritmos aglomerativos hasta satisfacer la condición de término.

Las ventajas de este tipo de algoritmos es que son adecuados para datos con formas arbitrarias y atributos de cualquier tipo, las relaciones jerárquicas son fáciles de detectar y son relativamente escalables en general.

Las desventajas es que son de alta complejidad en general, y el número de clusters necesita conocerse de antemano.

2.8. Fuzzy Clustering

La idea básica detrás de este tipo de algoritmos es cambiar los valores discretos de las etiquetas $\{0, 1\}$ a valores en intervalos continuos $[0, 1]$, con el fin de poder describir de manera más razonable las relaciones de pertenencia entre objetos.

La lógica difusa (*fuzzy logic*, en inglés) fue formulada en 1965 por el ingeniero y matemático Lotfi A. Zadeh para tratar problemas de la vida real. Este tipo de lógica utiliza reglas lógicas simples para resolver problemas complejos y no lineales. Ruspini fue el primero en implementar un algoritmo de clustering difuso en 1969.

En algoritmos de clustering tradicionales se agrupan los objetos de tal forma que pertenecen a exactamente un cluster. En métodos fuzzy (también llamado *soft* clustering), los objetos pueden pertenecer a más de un cluster, es decir, cada elemento posee un grado de pertenencia difuso a los distintos grupos.

Fuzzy C-Means (FCM) [32, 14] es un algoritmo típico de clustering difuso, extensión del clásico C-Means. FCM intenta encontrar el punto más característico en cada cluster, el cual puede ser considerado como el "centroide" del cluster, y entonces calcula el grado de membresía o pertenencia de cada objeto hacia el cluster.

Otros algoritmos típicos de esta clase son FCS [29] y MM [87]. FCS difiere de FCM en que considera una hiperesfera multidimensional como el prototipo de cada cluster, por ende para agrupar objetos necesita una función de distancia basada en hiperesferas.

Las ventajas de este tipo de algoritmos es que son más realistas al entregar una probabilidad de pertenencia y son relativamente precisos en la agrupación.

Las desventajas son la baja escalabilidad que presentan en general, caen fácilmente en óptimos locales, el clustering es altamente sensible a los parámetros iniciales, y el número de clusters se necesitan de antemano.

2.9. Clustering High Dimensional Data

2.9.1. Hypergraph Partitioning

Hypergraph-based clustering [43] es un enfoque de clustering en espacios altamente dimensionales basado en hipergrafos. Los hipergrafos son una extensión de los grafos regulares, en donde una arista puede conectar más de dos vértices. Hypergraph-based clustering consiste en las siguientes etapas:

1. Definir la condición para conectar varios objetos (cada objeto es un vértice del hipergrafo) por una hiperarista.
2. Definir una medida para la fuerza o peso de una hiperarista.
3. Usar un algoritmo de graph-partitioning [57] para particionar el hipergrafo en dos partes de tal forma de minimizar el peso en el corte mínimo de las hiperaristas.
4. Continuar el particionamiento hasta lograr un número fijo de partes, o hasta que una nueva partición produzca un cluster de peor calidad, según algún criterio de medición.

2.9.2. Grid Based Approaches

En su forma más básica, grid based clustering es relativamente simple:

1. Dividir el espacio de datos en celdas (hiper) rectangulares, particionando los datos de cada dimensión en celdas de igual tamaño. La Figura 2.1 muestra un ejemplo de grid para dos dimensiones.

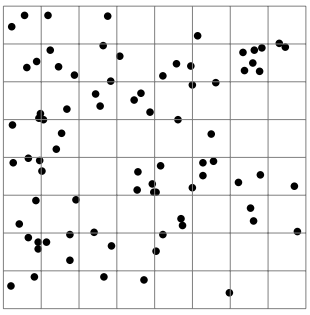


Figura 2.1: Grid de dos dimensiones para detección de clusters.

2. Descartar celdas de baja densidad. Las regiones de alta densidad representan un cluster, mientras que las regiones de baja densidad representan ruido.

3. Combinar celdas adyacentes de alta densidad para formar clusters. Si las regiones de alta densidad son adyacentes, entonces se unen para formar un solo grupo.

El enfoque Grid Based Clustering asume una definición basada en densidad de clusters, es decir, sufre las mismas desventajas que cualquier método de clustering density-based; no puede agrupar correctamente datasets con grandes diferencias de densidades.

Los algoritmos de clustering en alta dimensionalidad basados en celdas o grillas (Grid Based) más conocidos son CLIQUE [4], MAFIA [38], DENCLUE [44], OptiGrid [45].

CLIQUE [4] se basa en la siguiente observación: una región densa en un espacio particular debe crear regiones densas cuando se proyecta a subespacios de menor dimensionalidad. No obstante, CLIQUE necesita heurísticas para reducir los subconjuntos de dimensiones, es decir la complejidad de CLIQUE, aunque lineal con el número de objetos, no lo es con respecto al número de dimensiones.

MAFIA [38] es un refinamiento de CLIQUE, encuentra mejores clusters y consigue alta eficiencia usando celdas no uniformes. Específicamente, en lugar de dividir arbitrariamente los datos en un predeterminado número de intervalos de igual tamaño, MAFIA divide cada dimensión usando un número variable de “intervalos adaptativos”, los cuales reflejan mejor la distribución de los datos en esa dimensión.

DENCLUE [44] aborda el problema de una manera un poco diferente, puede ser visto como una generalización de enfoques de clustering basados en densidad como DBSCAN. DENsity CLUstEring modela la densidad de todos los objetos como la suma de funciones asociadas a cada objeto. La función de densidad global resultante tendrá peaks locales, i.e., máximos de densidad local, y esos peaks locales de densidad pueden ser usados para definir clusters de manera sencilla. Específicamente, para cada objeto, usando la heurística hill climbing se puede hallar el peak más cercano asociado a cada objeto, y el conjunto de todos los objetos asociados a un peak en particular (llamado atractor de densidad local) forman un cluster. Si la densidad del peak local es muy baja, los objetos en el cluster asociado son clasificados como ruido y se descartan. DENCLUE presenta buen desempeño en baja dimensionalidad, aunque cuando aumentan las dimensiones del espacio, también lo hace el ruido. Para lidiar con ese inconveniente, se propuso OptiGrid [45], que es muy similar a MAFIA en el sentido que crea celdas usando particionamiento dependiente a los datos (“adaptativo”). Aunque, a diferencia de MAFIA, no procura encontrar el mejor subespacio para realizar particionamiento. Desde el punto de vista de la eficiencia, es mucho mejor, aunque sus vagos detalles de implementación y el gran número de parámetros no favorecen su popularidad.

2.9.3. A Shared Nearest Neighbor Approach (SNN)

El algoritmo de clustering shared nearest neighbor, propuesto por Jarvis-Patrick [50], se sustenta en la idea de utilizar los vecinos más cercanos en el grafo.

1. En primer lugar, se calculan los n vecinos más cercanos de todos los puntos. En términos de grafos esto se refiere a mantener solo los n enlaces más fuertes entre un punto y el resto de los puntos en el grafo de proximidad. Los enlaces más débiles entre puntos del grafo son suprimidos. Esto forma el denominado "nearest neighbor graph" que viene a ser una versión dispersa del grafo de similaridad original.

2. Entonces, se determina el número de vecinos más cercanos compartidos entre todo par de puntos. En terminología de grafos se conoce como "shared nearest neighbor graph". Esto se realiza reemplazando cada enlace (en el grafo "nearest neighbor graph") entre dos puntos por el número de vecinos que comparten esos dos puntos. En otras palabras [13], esto es el número de caminos de longitud 2 entre dos puntos cualquiera en el nearest neighbor graph. En la Figura 2.2 los enlaces entre nodos indican que tan similares son.

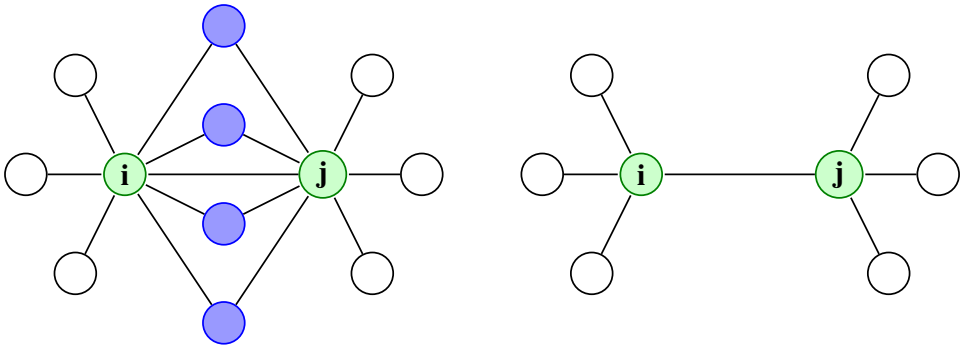


Figura 2.2: Se reemplazan los enlaces entre puntos compartidos por ambos nodos.

3. Todos los pares de puntos son comparados y si cualquiera de los dos puntos comparte más de T vecinos, i.e., tienen un enlace en shared nearest neighbor graph con un peso mayor que el valor del umbral $T(T \leq n)$, entonces los dos puntos y cualquier cluster al que pertenezcan se fusionan. En otras palabras, los clusters corresponden a las componentes conexas en el shared nearest neighbor graph, después de dispersarlo utilizando un umbral.

Este enfoque muestra propiedades positivas: es capaz de manejar clusters de diferentes densidades, ya que, SNN es self-scaling. Además, esta estrategia es transitiva, i.e., si el punto p, comparte muchos de sus vecinos más cercanos con el punto q, quién a su vez

comparte muchos de sus vecinos más cercanos con el punto r , entonces, los puntos p, q y r pertenecen al mismo cluster. Esta propiedad transitiva permite manejar clusters de diferentes tamaños y formas. Sin embargo, presenta limitantes de memoria en conjuntos de datos de gran tamaño (Big Data). Solo para calcular la matriz de similitud entre usuarios más cercanos requiere un tiempo $\Omega(N^2)$.

2.10. Clustering con restricciones de balance

Diversas aplicaciones reales que trabajan en el manejo de datos, a menudo necesitan segmentos de los datos de un tamaño comparable; ya sea para ofrecer promociones a grupos de usuarios proporcionales o poder agrupar grandes conjuntos de información evitando la aparición de jerarquías altamente sesgadas.

Además de los requerimientos de balance que puedan tener ciertas aplicaciones, balanced clustering puede ser muy útil porque tiende a disminuir la sensibilidad en parámetros iniciales y evitar la formación de clusters outliers (representantes altamente subutilizados).

Desafortunadamente, los métodos de clustering tradicionales, por naturaleza construyen de manera secuencial los clusters. Esto impide tener una visión global de la agrupación emergente que se necesita para obtener clusters bien formados y razonablemente balanceados.

Incluso el venerable K-Means no tiene ninguna manera explícita de garantizar que haya al menos un cierto número mínimo de puntos por cluster. Este método junto a sus variantes son cada vez más propensos a dar soluciones desequilibradas a medida que aumenta la dimensionalidad de la entrada. Este problema se exacerba cuando se necesita un gran número de clusters (diez o más), y es bien sabido que tanto *Hard* como *Soft* K-Means, suelen generar algunos clusters vacíos en tales escenarios.

Un método de clustering aglomerativo puede ser adaptado fácilmente de modo tal que cuando un cluster alcance un cierto tamaño en el proceso de aglomeración bottom-up, sea descartado de operaciones aditivas. Sin embargo, esto puede afectar significativamente la calidad del cluster.

Un enfoque reciente para obtener clusters balanceados es convertir el problema de clustering en un problema de particionamiento de grafos [56, 78]. Se construye un grafo con pesos, donde los vértices son los puntos. Una arista conecta dos vértices con peso igual a la proximidad (medida con alguna función de similaridad) correspondiente entre ambos datos. El grafo resultante puede ser dividido por algoritmos eficientes de corte mínimo como METIS [56], que también incorporan una restricción de soft-balancing. Aunque este enfoque da muy buenos resultados, al igual que el algoritmo SNN, no es

escalable.

En [17] se presenta el algoritmo constrained k-means, que es similar a k-means, pero el número de puntos de cada cluster es un parámetro del algoritmo. La etapa de asignación de cada cluster se maneja como un problema de optimización de coste mínimo satisfaciendo las restricciones de tamaño en los clusters. No obstante, la complejidad $O(n^3)$ presenta incluso peores opciones de escalar en big data.

Malinen *et al*. [66] presentan el algoritmo balanced k-means, donde el número de puntos de cada cluster es el mismo. Funciona de manera similar a k-means, pero en lugar de asignar el dato al centroide más cercano es destinado a uno de los n espacios (*slots*) disponibles (n/k slots por cluster). Esto obliga a los clusters a quedar con el mismo tamaño, asumiendo que $\lfloor n/k \rfloor = \lceil n/k \rceil = n/k$.

Una nueva alternativa para obtener clusters balanceados es a través de métodos de aprendizaje competitivos [5], donde los clusters de gran tamaño son penalizados para disminuir la probabilidad de asignación. Aunque los métodos de aprendizaje competitivo pueden entregar clusters bastante balanceados en la práctica, no hay ninguna manera obvia de garantizar que cada clúster tendrá al menos un número predefinido de puntos.

Un estudio pionero sobre clustering restringido en grandes bases de datos fue presentado por Tung *et al*. [81]. Ellos describen una variedad de restricciones que se pueden imponer a una solución de agrupación, pero posteriormente se centran únicamente en una restricción de balance sobre determinados objetos clave denominados objetos pivote. Comienzan con cualquier agrupación (que involucra a *todos* los objetos) que satisface las restricciones dadas. Esta solución se refina entonces para reducir el coste de la agrupación, medido como dispersión neta de los representantes más cercanos, satisfaciendo al mismo tiempo la restricción.

A partir de esta última idea, en [9] se aborda el problema de desarrollar algoritmos de clustering escalables que satisfagan las restricciones de balance en el tamaño de los clusters, es decir, el número de objetos en cada cluster. Se presenta un método para agrupar n puntos en k clusters de tal forma que cada cluster tenga a lo menos m puntos para un $m \leq \frac{n}{k}$. La complejidad total del método es $O(kn \log n)$ y se divide en tres etapas: (I) hace un muestreo de los datos, (II) realiza clustering sobre los datos muestreados y (III) completa los clusters a partir de los datos restantes con el fin de satisfacer la restricción de balance.

En [89] presentan un framework general que permite adaptar cualquier algoritmo de clustering generativo (model-based) para proveer soluciones balanceadas. Los algoritmos de clustering model-based pueden ser vistos como un proceso de optimización iterativo de dos etapas: (re)estimación del modelo y (re)asignación. Zhong *et al*. [90] resuelven el problema de asignación balanceada usando una heurística greedy en cada iteración del proceso. La heurística obtiene una solución aproximada en tiempo $O(kn(k+\log n))$.

Los métodos de clustering tradicionales buscan disminuir la distancia intra-cluster y aumentar la inter-cluster tanto como sea posible. El criterio más común de optimización es el error cuadrático medio (MSE). Por lo tanto, naturalmente, clustering con restricciones de balance emerge como un problema de optimización 2-objective, donde dos objetivos se cruzan: minimizar MSE y balancear el tamaño de los clusters. Para optimizar ambos objetivos, hay dos alternativas: BALANCE-CONSTRAINED y BALANCE-DRIVEN clustering. En el primero, lograr clusters balanceados es obligatorio, y minimizar el MSE es secundario. En el segundo, conseguir balance es un objetivo, pero no es imprescindible. Los algoritmos antes mencionados y su clasificación se pueden encontrar en la Tabla 2.2.

BALANCE-CONSTRAINED		
MÉTODO	TIPO	COMPLEJIDAD
Balanced k-means [66]	k-means	$O(n^3)$
Constrained k-means [17]	k-means	$O(k^{3,5}n^{3,5})$
Size constrained [91]	ILP	–
mk-means [89]	k-means	$O(kn(k+\log n))$
BALANCE-DRIVEN		
MÉTODO	TIPO	COMPLEJIDAD
Cluster sampled data [9]	k-means	$O(kn\log n)$
Competitive learning [5]	asignación	–
Min-max Cut [31]	divisivo	–

Tabla 2.2: Clasificación de algunos algoritmos de clustering balanceados.

2.11. Ventajas y Aplicaciones

El análisis de clusters es una herramienta importante en gran número de aplicaciones, en diferentes campos de la ciencia y los negocios.

- **Reducción de datos.** Clustering puede contribuir en la compresión de información incluida en los datos. En muchos casos, la cantidad de datos disponibles es muy grande y procesarla se vuelve demandante. Clustering puede ser usado para particionar el dataset en un número determinado de grupos. Entonces, en lugar de procesar el dataset como entidad, se trabaja con datos representativos de cada grupo.
- **Web mining.** En este caso, clustering es usado para descubrir grupos de documentos relevantes en la inmensa colección web de documentos semi-estructurados.

Esta clasificación de documentos web ayuda en *Information Discovery*.

- **Sistemas de recomendación.** Análogo al problema de recuperar información relevante en *Information Retrieval* (IR), clustering puede ser utilizado para agrupar usuarios con gustos similares y recomendar productos de su interés que no hayan sido consumidos en el pasado.

2.12. La "maldición de la dimensionalidad"

Fue aparentemente Richard Bellman quién acuñó la frase "la maldición de la dimensionalidad", en un libro de teoría del control [12]. El problema al que Bellman se refería es a la imposibilidad de optimizar una función de varias variables por medio de una búsqueda a fuerza bruta en una grilla espacial discreta multidimensional (el número de puntos en la grilla aumenta exponencialmente con la dimensionalidad, i.e., con el número de variables). Con el paso del tiempo, la maldición de la dimensionalidad se ha utilizado para referirse a cualquier problema en análisis de datos que resulta de un gran número de variables (atributos).

En términos generales, problemas con alta dimensionalidad son consecuencia del hecho que un número fijo de puntos de datos se vuelven cada vez más "escasos" o "dispersos" (sparse) a medida que aumenta la dimensionalidad. Para visualizar esto, considerar 100 puntos distribuidos con una distribución uniforme aleatoria en el intervalo [0,1]. Si este intervalo es dividido en 10 celdas, entonces es altamente probable que todas las celdas contengan al menos 1 punto. Sin embargo, considerar que sucede si se mantiene el mismo número de puntos, pero ahora se distribuyen sobre un cubo unitario (situación donde cada punto tiene dos dimensiones). Si se mantiene la unidad de discretización en 0.1 para cada dimensión, entonces existirán 100 celdas de dos dimensiones, y es bastante probable que más de alguna celda esté vacía. Para 100 puntos y en tres dimensiones, la mayoría de las 1000 celdas estarán vacías, ya que hay por lejos más celdas que puntos. Conceptualmente, los datos se "pierden en el vacío del espacio" mientras aumentan las dimensiones.

Para propósitos de clustering, el aspecto más importante de la maldición de la dimensionalidad es el impacto sobre la función de distancia o similaridad. En particular, muchas técnicas de clustering dependen de manera crítica en una métrica de distancia o similaridad, y requieren que los objetos dentro de un mismo cluster estén, en general, más cerca entre ellos que con respecto a objetos de otros clusters (de lo contrario, los algoritmos de clustering producirían clusters sin sentido).

Existen trabajos que han analizado el comportamiento de funciones de distancia en data de alta dimensionalidad. En [13], se muestra, para ciertas distribuciones de datos, que la diferencia relativa entre el punto más cercano y más lejano de un punto indepen-

dientemente seleccionado tiende a 0 cuando la dimensionalidad aumenta, i.e.,

$$\lim_{d\to\infty} \frac{\max d - \min d}{\min d} = 0$$

Este fenómeno ocurre si todos los atributos son i.i.d. (idénticamente e independientemente distribuidos). Así, se dice usualmente que "en espacios altamente dimensionales, las distancias entre puntos se vuelven relativamente uniformes". En tales casos, la noción de vecino más cercano pierde el sentido.

2.13. Resumen algoritmos de clustering

		Complejidad (tiempo)	Complejidad (espacio)	Forma deseada de los datos	Escalabilidad	High Dim Data	Sensibilidad a los parámetros	Sensibilidad a Ruido/outliers
Partitional	K-Means	$O(nkd)$	$O(n+k)$	Convexa	★☆	✗	★★	★★
	K-medoids	$O(k(n-k)^2)$	$O(n+k)$	Convexa	☆☆	✗	★☆	☆☆
	CLARA	$O(k(40+k)^2+k(n-k))$		Convexa	★★	✗	★☆	☆☆
	CLARANS	$O(kn^2)$	–	Convexa	★☆	✗	★★	☆☆
Density	DBSCAN	$O(n\log n)$	–	Arbitraria	★☆	✗	★☆	☆☆
	OPTICS	$O(n\log n)$	–	Arbitraria	★☆	✗	☆☆	☆☆
	Mean-shift	(kernel)	–	Arbitraria	☆☆	✗	☆☆	☆☆
Hierarchical	HC	$O(n^2)$	$O(n^2)$	Arbitraria	★☆	✗	★☆	☆☆
	BIRCH	$O(n)$	–	Convexa	★★	✗	★☆	☆☆
	CURE	$O(n_s^2\log n_s)$	$O(n_s)$	Arbitraria	★★	✓	★☆	☆☆
	ROCK	$O(n^2\log n)$	–	Arbitraria	★☆	✓	★☆	☆☆
	FCM	$O(n)$	–	Convexa	★☆	✗	★☆	★★
	CLIQUE	$O(n+k^2)$	–	Convexa	★★	✓	☆☆	★☆

n : Número de objetos (puntos) en el dataset. k : Número definido de clusters.
d : Número de dimensiones en el dataset. n_s : Número de objetos en la muestra s.

Tabla 2.3: Características de los principales algoritmos de clustering.

2.13 RESUMEN ALGORITMOS DE CLUSTERING

	Parámetros de Entrada	Resultados	Criterio de clustering
K-Means	Número de clusters	Centroides de los clusters	$\text{mín}_{v_1,v_2,\ldots,v_k}(E_k)$, con $E_k = \sum_i^k \sum_k^n d^2(x_k, v_i)$
CLARA	Número de clusters	Medoides de los clusters	$\text{mín} \sum_j C_{jih}$
CLARANS	Número de clusters, máximo de vecinos examinados	Medoides de los clusters	$\text{mín} \sum_j C_{jih}$
DBSCAN	Radio del cluster, número mínimo de objetos	Asignación de objetos a los clusters	Mezclar objetos con densidad accesible en un mismo cluster.
DENCLUE	Radio del cluster σ, número mínimo de objetos ξ	Asignación de objetos a los clusters	$f^D_{Gauss}(x^*) = \sum_{x_1 \in near(x^*)} e^{\frac{d(x^*,x_1)^2}{2\sigma^2}}$
BIRCH	Radio de los clusters (threshold) T, factor de ramificación (branching factor) B,	Clustering Feature $CF = (n, \vec{LS}, SS)$, n: puntos, $\vec{LS}$: linear sum, SS : squared sum	Asignar objeto al nodo más cercano hasta satisfacer un threshold.
CURE	Número de clusters, número de representantes	Asignación de objetos a los clusters	Meclar los clusters con el par de representantes más cercano.
ROCK	Número de clusters	Asignación de objetos a los clusters	máx $\sum_i^k n_i \times \sum_{p_q,p_r \in V_i} \frac{link(p_q,p_r)}{n_i^{1+2f(\theta)}}$
FCM	Número de clusters	Centroides, membresía	$\text{mín}_{U,v_1,v_2,\ldots,v_k}(J_m(U,V))$

CLARA, CLARANS: C_{jih} = el costo de reemplazar el centroide i con h en lo que se refiere a O_j.
DENCLUE: x^* atractor de densidad para un punto x si $F_{Gauss} > \xi$, entonces x se une al cluster de x^*.
ROCK: v_i centro del cluster i. $link(p_q, p_r)$: número de vecinos en común entre p_q y p_r.

Tabla 2.4: Parámetros de los principales algoritmos de clustering.

Capítulo 3

Clustering para recomendaciones *Top-N*

3.1. Introducción

El Filtrado Colaborativo (CF) es una de las técnicas de recomendación más exitosa. Está basada en la idea de que personas con gustos similares pueden entregar mejores y más precisas recomendaciones sobre productos. Como principal representante de CF está *User-based*, que a través de una métrica específica calcula la similaridad y retorna los k usuarios más cercanos. Aunque, *user* k-NN es efectivo, sufre de problemas de escalabilidad cuando la cantidad de usuarios es grande (aumenta considerablemente el número de operaciones involucradas en el cómputo de las distancias).

Por otra parte, se han utilizado algoritmos de clustering para reducir el tiempo de cómputo. Clustering es útil para mejorar la eficiencia porque el número de operaciones se reduce. Sin embargo, al agrupar datos sobre algún criterio se obtienen recomendaciones menos personalizadas y en la mayoría de los casos de peor calidad que métodos de vecinos más cercanos. Es decir, se genera un *trade-off* entre eficiencia y precisión. Hasta el día de hoy, no se ha logrado reducir esa brecha con respecto a *user* CF y es poco probable que puedan ser un aporte para mejorar la precisión. Por lo tanto, clustering debe ser integrado al sistema recomendador con meticulosidad, sopesando el compromiso entre mejorar la eficiencia y un posible severo declive en precisión.

Clustering, visto en machine learning como aprendizaje no supervisado, consiste en asignar objetos a grupos de tal forma que datos asignados al mismo grupo sean siempre más similares a los ubicados en diferentes grupos. El objetivo es descubrir grupos naturales (o significativos) que existan en los datos. La similaridad es determinada utilizando una función de proximidad. La finalidad de un algoritmo de clustering es minimizar las

distancias intra-cluster y maximizar las distancias inter-cluster.

Hay dos importantes categorías en los algoritmos de clustering: por particiones y jerárquico. En algoritmos de clustering particional se particiona los objetos de datos en clusters no-solapados de tal forma que cada objeto esté ubicado en exactamente un cluster. Agrupamiento jerárquico o Hierarchical Clustering (HC), produce un conjunto de clusters anidados organizados como un árbol jerárquico (dendrograma). HC no necesita asumir por adelantado un número particular de clusters. Se puede obtener el número deseado de clusters seleccionando el nivel apropiado en el árbol.

Muchos algoritmos de clustering intentan minimizar una función que mide la calidad de la agrupación. Esta función de calidad se conoce, generalmente, como función objetivo. Así, clustering puede ser visto como un problema de optimización. El algoritmo de clustering ideal debe considerar todas las posibles particiones de los datos y entregar como salida la partición que minimice la función de calidad. Pero, el correspondiente problema de optimización es NP-hard, entonces muchos algoritmos recurren a heurísticas (e.g., *k*-means clustering utiliza solo procedimientos de optimización local, cayendo potencialmente en mínimos locales). El punto central es que clustering es un problema difícil porque encontrar la solución óptima global no siempre es posible. Por esta misma razón, la elección de un algoritmo de clustering en particular y sus parámetros (e.g., la función de similaridad) depende de muchos factores, incluyendo las características de los datos. En lo que sigue, se describen experimentos de recomendación realizados utilizando diferentes alternativas de clustering y funciones de proximidad.

3.2. Estado del Arte

Los datos pueden ser particionados usando algoritmos de clustering para distribuir grupos de usuarios alrededor de diferentes máquinas, logrando escalar a datasets de gran escala. Este tipo de algoritmos utilizan un método de clustering para realizar la agrupación, restringiendo la recuperación de vecinos más cercanos al cluster de cada usuario. Enfoques basados en memoria distribuida como Hadoop obtienen beneficios de las particiones de datos, evitando la asignación del dataset completo en memoria. El primer algoritmo de CF en adoptar esta idea, aplicó un algoritmo de clustering sobre la base de datos de ratings usuario-producto para dividir el dataset en clusters de usuarios [75]. El algoritmo de clustering podía generar particiones de tamaño fijo, o basándose en un umbral de similaridad, generar un número solicitado de particiones de tamaño variable. Entonces, se aplicaba un algoritmo UBCF para producir recomendaciones, restringiendo el cálculo de cada vecino (usuario) a la frontera del cluster. Resultados experimentales demostraron que esta propuesta es razonable, pero a costa de deteriorar la precisión de las recomendaciones. De aquí en más, para referirse a esta clase de métodos de recomendación se empleará el término `PureCL`.

Una idea ligeramente diferente fue explorada por Xue *et al*. [86] donde se utilizó clustering para mejorar la tasa de estimación. En este trabajo, la distancia entre el usuario objetivo y sus vecinos más cercanos fue modificada reemplazando las entradas nulas del vector por la tasa promedio en cada vecindario del cluster. Recientemente, una nueva medida de similaridad para calcular el vecindario basada en clustering fue propuesta [79]. La medida de similaridad, llamada LiRa, utiliza clustering para calcular la relación de probabilidad entre la diferencia de tasas de los usuarios que pertenecen al mismo cluster y las diferencias producidas por casualidad. Resultados experimentales demostraron que la función de similaridad se comporta bien en datasets dispersos.

Kelleher *et al*. [60] presentaron un método CF basado en clustering jerárquico aglomerativo (HAC) para obtener particiones de datos. Este método consideraba una recopilación de información con respecto a los ratings de los usuarios derivada a partir del HAC. Ellos mostraron que la propuesta era eficiente en términos del tiempo computacional involucrado y aceptable en precisión.

George *et al*. [37] usaron co-clustering para realizar particiones de datos en un enfoque simultáneo sobre usuarios y productos. Resultados experimentales demostraron que su propuesta era razonable al mejorar la precisión en algunos casos.

El clustering iterativo entre productos y usuarios fue explorado por Jiang *et al*. [54] reforzando el proceso de clustering. Resultados experimentales demostraron que la utilización de dos estrategias de clustering en simultáneo (item y user-based) pueden aliviar el problema de dispersión de datos en sistemas recomendadores. La localización de cada usuario fue explotada por Das *et al*. [27], realizando clustering espacial para crear particiones de usuarios. Para lograr este objetivo, un enfoque basado en diagramas de Voronoi ponderado fue usado, restringiendo el cálculo del vecindario a las fronteras de cada cluster espacial. Clustering de usuarios basado en géneros sobre bases de datos de películas fue explorado utilizando DBSCAN [28]. Un máximo de cuatro géneros por usuario fue permitido para realizar el clustering, creando particiones de usuarios de acuerdo a sus gustos. Entonces, el cálculo del vecindario fue restringido a cada cluster, demostrando que los clusters basados en géneros son útiles para mejorar la calidad de las recomendaciones.

En resumen, clustering ha sido considerado como una etapa de pre-procesamiento en los sistemas de recomendación, usado para crear particiones y explotar cada partición con el fin de evitar el cálculo de vecindarios sobre todo el conjunto de datos. Por lo tanto, este tipo de sistema de recomendación utiliza métodos tradicionales para producir listas de productos, siendo los métodos basados en CF las técnicas más comunes empleadas para este propósito. El estado del arte carece de propuestas basadas en la combinación de datos intra/inter cluster a través de funciones de distancia para expandir vecindarios, el principal objetivo de esta investigación.

3.3. Propuesta

Los algoritmos de recomendación basados en modelos de clustering han demostrado ser eficientes en data sets reales porque pueden ser utilizados como una etapa previa en forma de preprocesamiento de los datos. Así, agrupando usuarios/productos en clusters, es posible trabajar con cada cluster de manera independiente (en paralelo) y aumentar significativamente la eficiencia del modelo. El problema es que todos estos métodos sobreexplotan la localidad de los datos, descartando el uso de la estructura global subyacente en los mismos. Esto perjudica en términos predictivos, afectando en la precisión y diversidad del modelo. Para manejar esta limitación, se propone explorar el uso de funciones de distancias que permitan inferir la estructura global de los grupos generados por medio de las técnicas de clustering. Para expandir la búsqueda de usuarios y productos relevantes se propone modificar la función de proximidad empleada en sistemas de recomendación. El objetivo es que usuarios y productos relevantes ubicados en diferentes clusters (que son inaccesibles para un algoritmo de recomendación tradicional), puedan ser recuperados y aprovechados para generar recomendaciones. A partir de esta propuesta surgen dos contribuciones importantes:

1. Se reduce la brecha en términos de precisión entre modelos de clustering y algoritmos de memoria de completa (K-NN).

2. Aumenta la eficiencia en el cómputo de usuarios/productos relevantes utilizando los prototipos, en lugar de vectores altamente dimensionales.

Se propone crear un algoritmo de recomendación basado en modelos de clustering, que no se encuentre restringido por la agrupación de los datos. La idea es poder reducir el uso de memoria beneficiándose de los prototipos de los clusters, y ser capaz al mismo tiempo de aumentar la precisión del modelo rescatando información relevante fuera de cada cluster.

Tomando como inspiración el índice de Bray-Curtis, usado en ecología para medir la diversidad de una población [18], se pretende adaptar la función de proximidad entre usuarios (o productos) para lograr un equilibrio entre dos aspectos:

- **Explotación (Intensificación):** el algoritmo se beneficia de la técnica de clustering agrupando usuarios (o productos) similares, evitándose de esta manera trabajar sobre todo el conjunto de datos. Por lo tanto, el método puede enfocarse en una región del espacio en particular y buscar la mejor solución posible en ella.

- **Exploración (Diversificación):** el algoritmo es capaz de acceder a información fuera del cluster, visitando diferentes regiones del espacio con el fin de identificar

zonas de búsqueda que sean relevantes, y a la vez novedosas. Esto permite aumentar la diversidad de las recomendaciones, y no verse limitado por la localidad del algoritmo K-NN.

Comparado con estrategias de recomendación tradicionales basadas en vecindarios (k-NN), este enfoque supone mayor eficiencia, ya que, al trabajar con los prototipos el costo en el cálculo de las distancias y el almacenamiento de los datos es significativamente menor. Además, entrega mayor flexibilidad porque permite capturar relaciones entre los datos, considerando la estructura global de los mismos, logrando de esta manera mayor diversidad en las recomendaciones.

3.3.1. Definiciones y Notaciones

Para construir la propuesta dentro de un contexto común, se requiere introducir ciertas definiciones. Sea $U = \{u_1, \ldots, u_N\}$ un conjunto de usuarios y $V = \{v_1, \ldots, v_M\}$ un conjunto de productos. Una matriz de ratings R, creada a partir de $U \times V$ con N filas y M columnas, engloba una colección de pares de ratings usuario-producto. Se denota $R_{i,j}$ el rating correspondiente al usuario u_i sobre el producto v_j. Si un par usuario-producto fue inferido de manera implícita (por ejemplo, productos vistos o *views*) cada $R_{i,j}$ toma valores binarios: 1 para productos vistos, 0 en otro caso. Si los ratings han sido obtenidos de manera explícita (juicios humanos), $R_{i,j}$ puede tomar valores en escala ordinal (normalmente, 1-5). Notar que la matriz R contiene solo un subconjunto de sus entradas observadas, en general gran porción de la matriz contendrá entradas nulas (`NA`, 0). El desafío de los sistemas de recomendación es predecir ratings para cada entrada nula de R, con el fin de producir recomendaciones para cada usuario sobre productos no vistos/consumidos.

Sea $U(v)$ un grupo de usuarios quienes han mostrado preferencia sobre el producto v. Sea u un usuario activo (objetivo, *target*) y v un producto no visto por u. Un método tradicional basado en usuarios (UBCF) estima $R(u,v)$ a partir de R, según:

$$\hat{R}(u,v) = \sum_{u' \in U(v)} \text{Sim}(u,u') \cdot R(u',v), \tag{3.1}$$

donde $\text{Sim}(u,u')$ es una función de similaridad aplicada sobre el par u,u'.

En general, los métodos UBCF registren $U(v)$ a los usuarios más cercanos (similares) a u, evitando la inclusión de usuarios con gustos muy distintos que puedan generar ruido en la estimación. En adición, existen extensiones a la ecuación 3.1, donde se incluye un factor para evitar el sesgo, tal como mean-centering o Z-score. Una detallada revisión a esas y más variantes se puede encontrar en [76].

3.3.2. El modelo propuesto

Se propone un modelo basado en algoritmos de clustering para recomendación *top-N* (denominado `userCL`), que utiliza una nueva función de distancia capaz de recuperar usuarios similares fuera de las fronteras del cluster. La función de distancia propuesta funciona sobre los prototipos del cluster, asumiendo que para cada usuario el sistema tiene un prototipo que representa el cluster (e.g., un centroide si se utiliza el algoritmo K-Means Clustering). Como la función de distancia funciona sobre los prototipos, puede manejar datasets de alta dimensionalidad, reduciendo el impacto de la 'maldición de la dimensionalidad'. Además, el costo de almacenamiento es significativamente menor.

Suponer que se ha aplicado un algoritmo de hard clustering entre los usuarios sobre el espacio de vectores de productos, obteniendo k clusters disjuntos. Sea C_x el prototipo del cluster al cual pertenece el objeto x. La idea central es modificar la función de distancia tradicional empleada en Filtrado Colaborativo entre pares de objetos (usuarios), con el fin de considerar que tan lejos se ubican los objetos con respecto a sus prototipos. De esta forma, es posible encontrar vecinos similares que han quedado ubicados fuera de las fronteras del cluster, si los objetos están cerca del prototipo.

Si dos objetos pertenecen al mismo cluster, la función propuesta se define como la distancia entre los dos objetos (sin considerar el prototipo). Sin embargo, si los dos objetos pertenecen a diferentes clusters la distancia es calculada según la proximidad de sus dos prototipos. La idea es forzar una separación entre objetos en un espacio de características de alta dimensionalidad, evitando tener que calcular directamente la distancia entre los objetos. Si el algoritmo de clustering produce los clusters de manera correcta, la distancia entre los prototipos de los clusters será grande. Entonces, midiendo las distancias entre prototipos, los puntos que pertenecen a diferentes clusters estarán más separados, incluso si las fronteras de ambos clusters están muy juntas. De acuerdo a esto, se define la función de distancia $D(x,y)$ para un par de objetos x,y según:

$$D(x,y) = \begin{cases} d(x,y) & \text{if } C_x = C_y, \\ d(C_x,C_y)^{exp} & \text{if } C_x \neq C_y, \end{cases} \tag{3.2}$$

donde $d(x,y)$ es una función de distancia directa entre el par de objetos x,y (e.g., la distancia Euclidiana).

Notar que la distancia entre prototipos $d(C_x,C_y)$ considera un exponente exp. Se proponen tres variantes para exp:

[1] Multiplicative variant:

$$exp = 1 - d(x,C_x) \cdot d(y,C_y) \tag{3.3}$$

[2] Additive variant:

$$exp = 1 - \frac{1}{2}\left(d(x, C_x) + d(y, C_y)\right) \tag{3.4}$$

[3] Additive-sigmoid variant:

$$exp = 1 - \left(1 + \frac{1}{\alpha} e^{-\beta\left(\frac{1}{2}(d(x,C_x)+d(y,C_y))-1\right)}\right) \tag{3.5}$$

donde α y β son parámetros reales estrictamente positivos.

El razonamiento detrás de cada variante propuesta se explica a continuación.

Variante 1: Multiplicativa ($C_x \neq C_y$)

Si x está cerca de C_x `OR` y está cerca de C_y

$$d(x, C_x) \cdot d(y, C_y) \to 0 \Rightarrow D(x, y) \approx d(C_x, C_y)$$

Si x está lejos de C_x `AND` y está lejos de C_y

$$d(x, C_x) \cdot d(y, C_y) \to 1 \Rightarrow D(x, y) \to 1$$

La idea detrás de la variante multiplicativa es penalizar la distancia entre objetos en distintos clusters, si ambos objetos están lejos de sus respectivos prototipos. Si uno, o ambos objetos, están cerca de sus respectivos prototipos, la distancia entre los objetos es aproximadamente la distancia entre sus prototipos. Por otro lado, si ambos objetos están lejos de sus respectivos prototipos, la distancia tiende a 1.

Variante 2: Aditiva ($C_x \neq C_y$)

Si x está cerca de C_x `AND` y está cerca de C_y

$$\frac{1}{2}(d(x, C_x) + d(y, C_y)) \to 0 \Rightarrow D(x, y) \approx d(C_x, C_y)$$

Si x está lejos de C_x `AND` y está lejos de C_y

$$\frac{1}{2}(d(x, C_x) + d(y, C_y)) \to 1 \Rightarrow D(x, y) \to 1$$

La idea detrás de la variante aditiva es que si ambos objetos están cerca de sus respectivos prototipos, la distancia entre ambos es aproximadamente la distancia entre sus prototipos. Por otro lado, si ambos objetos están lejos de sus respectivos prototipos, la distancia tiende a 1. Por lo tanto, la variante aditiva es una versión más exigente que

la variante multiplicativa, porque aproxima la distancia entre objetos como la distancia de sus prototipos, si y solo si, ambos objetos están cerca de sus respectivos prototipos.

Variante 3: Aditiva-sigmoide ($C_x \neq C_y$)

Si x está cerca de C_x `AND` y está cerca de C_y

$$\texttt{Sigmoid}(\frac{1}{2}(d(x,C_x)+d(y,C_y))) \to 0 \Rightarrow D(x,y) \approx d(C_x,C_y)$$

Si x está lejos de C_x `OR` y está lejos de C_y

$$\texttt{Sigmoid}(\frac{1}{2}(d(x,C_x)+d(y,C_y))) \to 1 \Rightarrow D(x,y) \to 1$$

La última variante del *exp* es una extensión de la variante aditiva que incluye una función sigmoide, ayudando a incrementar la penalización en los bordes de la distancia entres pares. El argumento de la función sigmoide es un factor aditivo, comprimiendo los extremos de la variante aditiva.

3.3.3. El método UserCL

El algoritmo `userCL` comienza agrupando a cada usuario en un determinado cluster, por medio de alguna técnica de clustering basada en prototipos. Entonces, basándose en la ecuación 3.2, calcula las distancias entre el usuario objetivo dado y el resto de los usuarios de entrenamiento, seleccionando los K-vecinos más cercanos (KNN), según alguna de las tres variantes propuestas.

UserCL funciona en un tándem junto al algoritmo de recomendación UBCF, que utiliza los ratings de los usuarios en $U(v)$ para restringir el vecindario de u y estimar $R(u,v)$. No obstante, la función de distancia propuesta puede ser combinada con IBCF, agrupando productos similares en lugar de usuarios (como se realiza por defecto en UBCF).

3.4. Configuración Experimental

3.4.1. Datasets

Para evaluar el método propuesto (`userCL`), se utilizan cuatro diferentes datasets que son conocidos en la comunidad de sistemas de recomendación. Los primeros dos conjuntos de datos forman parte del proyecto de investigación de GroupLens, conocido

como MovieLens[1]. ML100K y ML1M corresponden a ratings de películas extraídos de datos reales y cuentan con 100 mil y 1 millón de ratings respectivamente. El tercer dataset es BX, un subconjunto del proyecto Book-Crossing[2], en el cual cada usuario ha calificado al menos 50 productos y cada producto ha sido calificado por al menos 30 usuarios; con un máximo de 300 usuarios por producto. Finalmente, LastFM dataset corresponde a una muestra de datos extraídos desde el sistema de música online lastfm[3]. Este último conjunto de datos incluye una larga lista de preferencias sobre una gran colección de artistas (más de 100 mil artistas fueron incluidos en el full-dataset). En esta oportunidad se seleccionaron usuarios restringiendo el proceso de muestreo con el fin de obtener un conjunto de todo aquel que haya escuchado al menos 50 artistas diferentes. Entonces, se consiguió un dataset con aproximadamente 500 usuarios sobre una colección de casi 40 mil artistas. Este dataset usa el número de reproducciones por artista como ratings implícitos, por ende, cuando un usuario dado reproduce más canciones de un artista dado, el rating toma un valor muy alto para ese par usuario,artista. La importancia de LastFM dataset es el efecto *long-tail* (larga cola), en la distribución de ratings. Este tipo de distribución en los ratings, presenta el fenómeno conocido como Ley de Zipf sobre las preferencias de los usuarios, donde unos pocos artistas (items) concentran la mayoría de las reproducciones. Un análisis detallado del efecto *long-tail* en las recomendaciones fue discutido por Celma en [23].

Un resumen de las estadísticas de los datasets usados en el capítulo se muestra en la Tabla 3.1.

	ML100K	ML1M	BX	LastFM
Usuarios (N)	943	6040	2963	495
Items (M)	1664	3706	7064	39,814
Ratings (nnzs of $\mathscr{R}$)	99,392	1,000,208	244,403	95,711
min $\lvert\mathscr{R}_u\rvert$[a]	20	20	20	50
Densidad[b]	6.33 %	4.47 %	1.17 %	0.48 %

[a] Mínimo número de ratings por usuario.
[b] Ratings (nnzs of $\mathscr{R}$) sobre el producto $M \times N$.

Tabla 3.1: Estadísticas de los datasets.

3.4.2. Metodología de evaluación

Como método de validación para los resultados se utiliza 5-fold cross validation. En cada ejecución (5 runs), el conjunto de datos es dividido usando un 80 % de los usuarios

[1] http://grouplens.org/datasets/movielens/
[2] http://www2.informatik.uni-freiburg.de/∼cziegler/BX/
[3] https://www.last.fm/

como conjunto de entrenamiento y el restante 20 % como conjunto de pruebas. Para cada usuario en el conjunto de prueba, 15 productos calificados son seleccionados al azar como historial y entregado al recomendador, el resto de los ratings se utilizan para evaluar la calidad de la estimación. Entonces, para cada usuario en el conjunto de prueba una lista *top-N* es generada por cada método de recomendación y se mide la calidad de las listas para $N = 1$ y $N = 10$ con las métricas Precision, Recall y nDCG. El proceso de evaluación fue implementado a través del framework en R recommenderlab [41].

3.5. Resultados Experimentales

En esta sección, se presentan los resultados experimentales del método propuesto `userCL`, en comparación a user k-Nearest Neighbors Collaborative Filtering (`userkNN`) y Clustering-Based Collaborative Filtering (`PureCL`), ambos métodos detallados en las secciones 3.1 y 3.2, respectivamente.

Los resultados fueron divididos en tres conjuntos de experimentos. En primer lugar, se estudia el comportamiento de la función de proximidad propuesta (Ecuación 3.2), en relación a funciones tradicionales como distancia Euclidiana Normalizada y similaridad Coseno. Se analiza su distribución entre pares de usuarios, en dos datasets de diferentes características: ML100K y LastFM. El primero, tiene baja dimensionalidad. El segundo, cuenta con casi 40 mil dimensiones y tiene muy baja densidad, lo que permite analizar el desempeño de las funciones de proximidad bajo condiciones de alta dimensionalidad y dispersión de datos.

En segundo lugar, se entregan resultados exhaustivos acerca de diferentes modelos de clustering utilizados para generar recomendaciones *top-N*. Esta sección permite conocer el alcance y real impacto de los algoritmos de clustering aplicados sobre sistemas de recomendación, un tópico poco explorado y sin demasiado éxito en investigaciones del área.

Finalmente, se comparan los métodos en términos de eficiencia, encontrando diferencias significativas en tiempo de recomendación entre `userCL` y `userkNN`.

Los parámetros generales de cada algoritmo son: Para `userkNN` y `userCL` el número de vecinos (tamaño vecindario) $nn = 50$; Para `PureCL` y `userCL` el número de clusters $k = 20$. El mejor desempeño de cada métrica (con diferencia estadísticamente significativa), se destaca en **negrita**.

El tamaño del cluster es un elemento clave del método propuesto. La idea es restringir el tamaño de los clusters (en promedio) para forzar la recuperación de algunos vecinos ubicados fuera de las fronteras del cluster. Para lograrlo, se requiere ajustar el número de clusters (k) y el tamaño del vecindario (nn) para forzar que en promedio

$nn \geq \frac{N}{k}$, donde N es el número de usuarios en el dataset. La capacidad de producir clusters balanceados por tamaño dependerá del tipo de algoritmo de clustering usado para este propósito.

3.5.1. El efecto de la función de proximidad en consultas KNN

En las Figuras 3.1 y 3.2 se explora el efecto de la función de proximidad en el cálculo de los KNN sobre dos datasets diferentes en términos de dimensionalidad. En la Figura 3.1 se muestra la distribución de las funciones de distancia y similaridad en ML100K y LastFM para `userkNN`. La distancia Euclidiana Normalizada, obtiene una mala distribución entre pares de usuarios para ambos datasets, con solo una pequeña fracción de pares a distancia 1 y una gran mayoría a distancia cero. Por otra parte, cuando la similaridad Coseno es utilizada, la distribución mejora, aunque en LastFM el resultado es marginal. Esto permite demostrar el efecto de la alta dimensionalidad sobre funciones de distancia en KNN. Más aún, en situaciones de alta dispersión, la diferencia en distancia entre objetos se vuelve despreciable (la consulta del usuario más cercano se vuelve inestable).

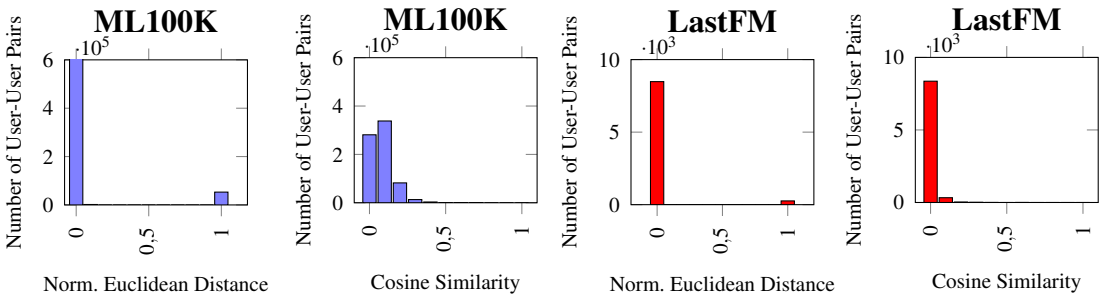


Figura 3.1: Histogramas de proximidad entre pares Usuario-Usuario en `userkNN`.

En `userCL` la función de proximidad no se aplica directamente sobre los datos porque los objetos se encuentran distribuidos en clusters alrededor de uno o más usuarios de entrenamiento, y la consulta KNN realiza la búsqueda en uno o más clusters. La Figura 3.2 muestra los histogramas de proximidad entre pares de usuarios conseguidos a partir de la función de distancia propuesta. De acuerdo a los histogramas, el uso de la nueva función de distancia produce mejores distribuciones para las tres variantes propuestas. La función propuesta, aplicada sobre similaridad Coseno, entrega los mejores resultados al generar una gran cantidad de bins (intervalos) en el rango [0,1].

A diferencia de una típica consulta KNN, donde la función de proximidad es aplicada directamente sobre los datos, la función propuesta se beneficia de modelos de clustering para manejar la "maldición de la dimensionalidad" a través de vectores de usuarios suavizados. En ese sentido, el modelo propuesto puede potencialmente codifi-

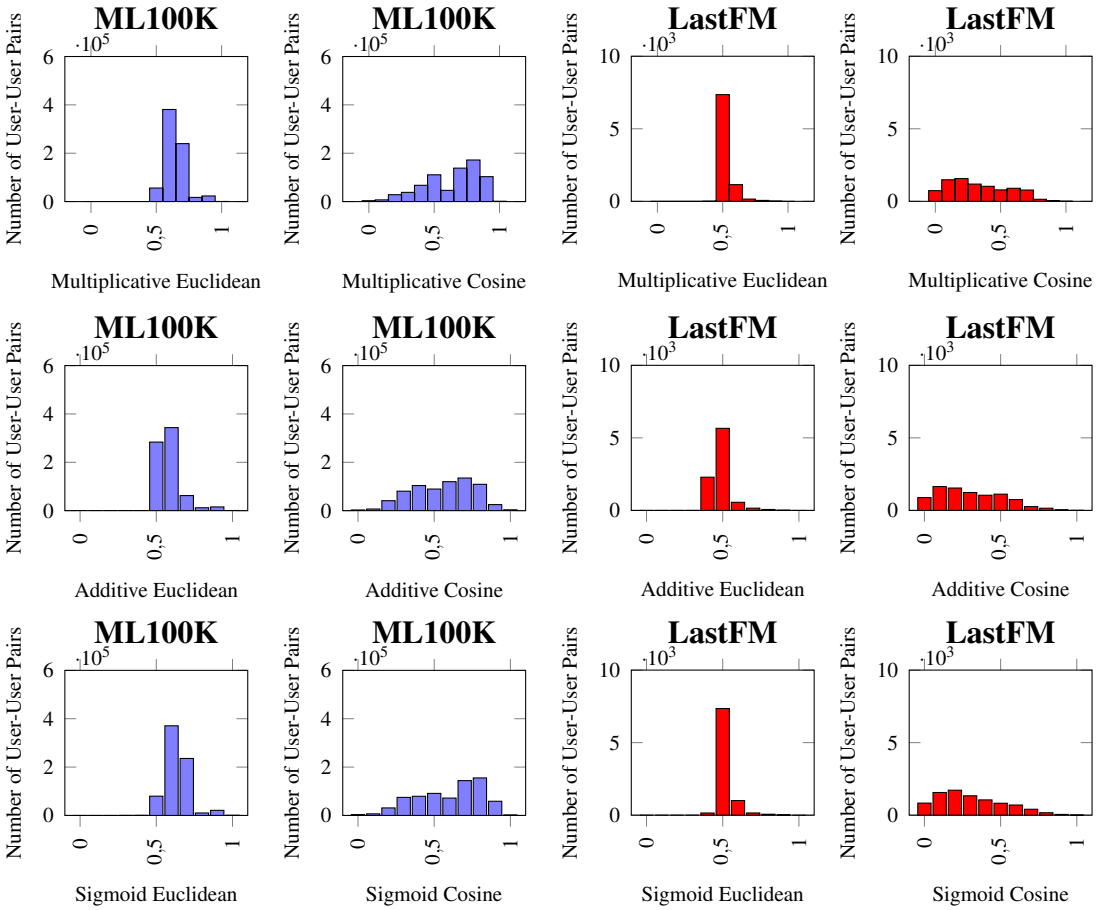


Figura 3.2: Histogramas de proximidad entre pares Usuario-Usuario en `userCL`.

car relaciones ricas y sutiles entre usuarios/productos que no son fácilmente detectadas a través de funciones de distancias convencionales entre usuario-usuario/item-item.

3.5.2. Recomendación *Top-N* sobre distintos modelos de clustering

En esta sección, se compara el desempeño de diversos métodos de clustering destinados a producir recomendaciones *top-N* a través del algoritmo de recomendación propuesto `userCL`.

Los modelos de clustering estudiados a lo largo de la sección incluyen a K-Means, K-Medians, Spectral Clustering, algoritmos de clustering con restricciones de balance (Balanced Clustering), K-Means y K-Medians en sus versiones Soft (Fuzzy) y Clustering Jerárquico Aglomerativo (HAC).

Todos los métodos de clustering se evaluaron sobre ML100K y LastFM para tener un contraste del desempeño de cada algoritmo entre un dataset de menor escala (ML100K) y un dataset de alta dimensionalidad y dispersión de datos (LastFM). En tanto, los métodos basados en `userCL` que demostraron un desempeño competitivo en relación a `userkNN`, fueron evaluados en datasets de mayor escala (ML1M y BX) para lograr una validación más exhaustiva.

K-Means Clustering

La Tabla 3.2 muestra los resultados en ML100K y LastFM utilizando K-Means para `userCL` y `PureCL`. Para calcular el vecindario se analiza el desempeño de la distancia Euclidiana Normalizada y la similaridad Coseno en recomendaciones *top*-1 y *top*-10.

método	parámetros		**ML100K**						**LastFM**					
			Precision		Recall		nDCG		Precision		Recall		nDCG	
	clust	*d*	@1	@10	@1	@10	@1	@10	@1	@10	@1	@10	@1	@10
`userkNN`	–	`Euc`	0.38	0.26	0.01	0.07	0.82	0.82	0.18	0.14	4e-4	3e-3	0.04	0.12
`PureCL`	`Kmean`	–	0.34	0.26	0.01	0.08	0.80	0.81	0.46	0.33	1e-3	0.01	0.09	0.15
`userCL[1]`	`Kmean`	`Euc`	0.50	0.31	0.02	0.12	0.83	0.81	0.37	0.32	9e-4	0.01	0.10	0.16
`userCL[2]`	`Kmean`	`Euc`	0.50	0.32	0.02	0.13	0.84	0.81	0.58	**0.40**	2e-3	0.01	0.13	0.16
`userCL[3]`	`Kmean`	`Euc`	0.43	0.30	0.02	0.13	0.82	0.81	0.58	0.38	2e-3	0.01	0.12	0.16
`userkNN`	–	`Cos`	**0.61**	**0.45**	**0.03**	**0.19**	0.85	0.85	0.41	0.36	9e-4	0.01	0.10	0.18
`userCL[1]`	`Kmean`	`Cos`	0.44	0.32	0.02	0.10	0.84	0.85	**0.59**	0.38	2e-3	0.01	**0.16**	0.16
`userCL[2]`	`Kmean`	`Cos`	0.45	0.34	0.02	0.12	0.84	0.84	0.54	0.38	1e-3	0.01	**0.16**	0.17
`userCL[3]`	`Kmean`	`Cos`	0.50	0.32	0.02	0.12	0.85	0.85	0.55	0.38	2e-3	0.01	**0.16**	0.17

Tabla 3.2: K-Means Clustering en Recomendación *Top-N*.

La Tabla 3.2 muestra que el desempeño de `userCL` es sustancialmente superior a `PureCL`. Sin embargo, en ML100K todavía existe una brecha con respecto a `userkNN`, especialmente en términos de Prec y Recall. En LastFM, `userCL` logra superar a `userkNN` con resultados significativamente mejores en Prec@1,10 y en nDCG@1. En Rec@1,10 no hay diferencias estadísticamente significativas. Con respecto a las tres variantes propuestas de `userCL`, en general muestran resultados similares.

Spectral Clustering

Los datasets en sistemas de recomendación suelen estar compuestos por gran cantidad de productos, que en este caso corresponden a las dimensiones de la matriz de

ratings[4]. Esto es común en sistemas con alta granularidad como plataformas online de recomendación de vídeos (YouTube) o servicios de música (Spotify). Entonces, como principal desafío surge el hecho de poder escalar en Big Data, y al mismo tiempo lograr buena precisión en High Dimensional Data. Una alternativa es aplicar técnicas de "Agrupamiento Espectral" (Spectral Clustering). Esta clase de algoritmos utiliza los valores y vectores propios de la matriz de similaridad para realizar una reducción de la dimensionalidad antes de aplicar una técnica de clustering tradicional como *k*-means.

método	parámetros		**ML100K**						**ML1M**					
			Precision		Recall		nDCG		Precision		Recall		nDCG	
	clust	*d*	@1	@10	@1	@10	@1	@10	@1	@10	@1	@10	@1	@10
`userkNN`	–	`Euc`	0.38	0.26	0.01	0.07	0.82	0.82	0.40	0.31	8e-3	0.06	0.86	0.85
`PureCL`	spec	–	0.33	0.25	0.01	0.07	0.80	0.81	0.36	0.29	8e-3	0.05	0.84	0.84
`userCL[1]`	spec	`Euc`	0.51	0.30	0.03	0.14	0.81	0.80	0.54	0.39	0.02	0.10	0.86	0.84
`userCL[2]`	spec	`Euc`	0.50	0.32	0.03	0.14	0.82	0.81	0.56	0.41	0.02	0.10	0.86	0.85
`userCL[3]`	spec	`Euc`	0.54	0.31	0.03	0.14	0.82	0.80	0.54	0.40	0.02	0.10	0.86	0.85
`userkNN`	–	`Cos`	**0.61**	**0.45**	0.03	**0.19**	0.85	0.85	**0.63**	**0.50**	0.02	**0.13**	0.88	0.87
`userCL[1]`	spec	`Cos`	0.46	0.33	0.02	0.13	0.85	0.85	0.36	0.33	8e-3	0.06	0.87	0.87
`userCL[2]`	spec	`Cos`	0.52	0.35	0.02	0.14	0.85	0.85	0.45	0.35	0.01	0.08	0.87	0.87
`userCL[3]`	spec	`Cos`	0.51	0.34	0.02	0.13	0.83	0.84	0.40	0.35	0.01	0.07	0.87	0.87

método	parámetros		**BX**						**LastFM**					
			Precision		Recall		nDCG		Precision		Recall		nDCG	
	clust	*d*	@1	@10	@1	@10	@1	@10	@1	@10	@1	@10	@1	@10
`userkNN`	–	`Euc`	0.05	0.03	3e-3	0.02	0.39	0.42	0.18	0.14	4e-4	3e-3	0.04	0.12
`PureCL`	spec	–	0.03	0.03	2e-3	0.02	0.36	0.41	0.39	0.30	9e-4	0.01	0.09	0.14
`userCL[1]`	spec	`Euc`	0.07	0.04	5e-3	0.03	0.39	0.43	0.53	0.38	2e-3	0.01	**0.15**	0.17
`userCL[2]`	spec	`Euc`	0.07	0.04	5e-3	0.03	0.40	0.43	0.53	0.39	2e-3	0.01	**0.15**	0.16
`userCL[3]`	spec	`Euc`	0.07	0.04	5e-3	0.03	0.39	0.43	0.60	0.40	2e-3	0.01	0.13	0.16
`userkNN`	–	`Cos`	**0.13**	**0.07**	**0.01**	**0.06**	**0.44**	0.45	0.41	0.36	9e-4	0.01	0.10	0.18
`userCL[1]`	spec	`Cos`	0.05	0.03	4e-3	0.02	0.37	0.42	**0.62**	0.42	2e-3	0.01	**0.15**	0.17
`userCL[2]`	spec	`Cos`	0.09	0.05	6e-3	0.03	0.40	0.44	**0.62**	**0.44**	2e-3	0.01	**0.15**	0.17
`userCL[3]`	spec	`Cos`	0.07	0.04	5e-3	0.03	0.40	0.43	**0.62**	**0.44**	2e-3	0.01	**0.15**	0.17

Tabla 3.3: Spectral Clustering en Recomendación *Top-N*.

Una de las grandes ventajas de aplicar técnicas de reducción de dimensionalidad sobre la matriz de datos R es que se trabaja con una nueva matriz de menor rango (R_k), que corresponde a una representación densa (suavizada) de R, permitiendo, entre otras cosas, lidiar con la escasez de ratings (Cold Start) y aumentar la eficiencia del modelo (menos dimensiones que abarcar). Como $R \approx R_k$, en el sentido de los mínimos cuadrados, la perdida de información es mínima.

Los resultados de la Tabla 3.3 muestran que al utilizar algoritmos de clustering con reducción de dimensionalidad (Spectral Clustering) se producen recomendaciones que

[4]En IBCF los usuarios son los atributos (dimensiones) de la matriz.

son consistentemente mejores que al aplicar clustering directamente sobre los datos. Esto sugiere, no solo una ventaja en términos de calidad predictiva, sino que un enorme beneficio de tiempo al dirigir el cómputo de los métodos sobre un espacio de características acotadas (se conserva lo relevante).

Comparando `userCL` con respecto a `userkNN`, se observa que este último muestra mejor desempeño (especialmente en términos de Precision y Recall) sobre tres de los cuatro datasets. Sin embargo, en LastFM, `userCL` logra superar al resto de los métodos. Esto se sustenta en el hecho de que este dataset presenta un espacio de alta dimensionalidad, mucho mayor que los otros datasets, y cualquier función de proximidad tradicional pierde el sentido (Figura 3.2). En otras palabras, `userkNN` puede funcionar bien y superar al resto de los algoritmos en datasets con un número limitado de dimensiones, luego su función de proximidad se vuelve inestable. Por otro lado, `userCL` es capaz de mantenerse competitivo en los cuatro datasets, trabajando sobre menos dimensiones; lo que implica un aumento significativo en la eficiencia del modelo (más detalles en la Sección 3.5.4).

K-Medians Clustering

La presencia de ciertos objetos (usuarios) lejanos del resto, considerados *outliers* por tener opiniones divergentes (preferencias atípicas, sesgadas), pueden afectar negativamente el desempeño de cualquier aplicación estadística, incluyendo las técnicas de clustering.

Particularmente, en sistemas de recomendación, la robustez se entiende como la estabilidad de la recomendación en presencia de información falsa o tendenciosa, que pueda influir (negativa o positivamente) sobre los datos.

En algoritmos de clustering, *k*-medians es una variación de *k*-means en donde en lugar de calcular la media de cada cluster para determinar su centroide, se calcula la mediana. Esto tiene el efecto de minimizar el error con respecto a la norma-1, lo cual entrega una estimación más robusta (se encuentra menos afectada por outliers).

K-Medians suele tener ventaja por sobre *k*-means debido a la robustez de la mediana como medida estadística. La media es una medida altamente vulnerable a outliers. Apenas un simple valor muy atípico del resto puede arruinar el valor de la media de todo el conjunto de datos, lo que puede llegar a ser preocupante sobretodo en datasets de gran escala. Por otro lado, la mediana es una medida estadística increíblemente resistente a outliers, por lo tanto debería agrupar de mejor manera los datos y entregar mayores beneficios a la hora de formar el vecindario.

Comparando las ganancias que pueden obtenerse al utilizar una medida estadística resistente a outliers a través de los diferentes datasets, se puede ver que los resultados

método	parámetros		ML100K						ML1M					
			Precision		Recall		nDCG		Precision		Recall		nDCG	
	clust	*d*	@1	@10	@1	@10	@1	@10	@1	@10	@1	@10	@1	@10
`userkNN`	–	`Euc`	0.38	0.26	0.01	0.07	0.82	0.82	0.40	0.31	8e-3	0.06	0.86	0.85
`PureCL`	`Kmed`	–	0.36	0.27	0.02	0.09	0.80	0.81	0.36	0.28	7e-3	0.05	0.84	0.84
`userCL[1]`	`Kmed`	`Euc`	0.50	0.32	0.02	0.13	0.82	0.82	0.44	0.34	0.01	0.08	0.85	0.84
`userCL[2]`	`Kmed`	`Euc`	0.51	0.35	0.02	0.15	0.83	0.83	0.47	0.34	0.01	0.08	0.86	0.84
`userCL[3]`	`Kmed`	`Euc`	0.55	0.31	0.03	0.14	0.83	0.82	0.42	0.31	0.01	0.08	0.84	0.84
`userkNN`	–	`Cos`	**0.61**	**0.45**	0.03	**0.19**	0.85	0.85	**0.63**	**0.50**	0.02	**0.13**	0.88	0.87
`userCL[1]`	`Kmed`	`Cos`	0.49	0.35	0.02	0.14	0.82	0.84	0.43	0.33	0.01	0.07	0.87	0.86
`userCL[2]`	`Kmed`	`Cos`	0.54	0.37	0.03	0.15	0.83	0.84	0.43	0.35	0.01	0.07	0.87	0.86
`userCL[3]`	`Kmed`	`Cos`	0.52	0.36	0.02	0.15	0.84	0.84	0.40	0.35	0.01	0.07	0.86	0.86

método	parámetros		BX						LastFM					
			Precision		Recall		nDCG		Precision		Recall		nDCG	
	clust	*d*	@1	@10	@1	@10	@1	@10	@1	@10	@1	@10	@1	@10
`userkNN`	–	`Euc`	0.05	0.03	3e-3	0.02	0.39	0.42	0.18	0.14	4e-4	3e-3	0.04	0.12
`PureCL`	`Kmed`	–	0.03	0.02	2e-3	0.01	0.35	0.41	0.30	0.29	9e-4	0.01	0.08	0.14
`userCL[1]`	`Kmed`	`Euc`	0.06	0.03	5e-3	0.02	0.38	0.42	0.53	0.38	2e-3	0.01	0.13	0.16
`userCL[2]`	`Kmed`	`Euc`	0.06	0.03	4e-3	0.02	0.38	0.42	0.50	0.36	1e-3	0.01	0.12	0.16
`userCL[3]`	`Kmed`	`Euc`	0.06	0.03	5e-3	0.02	0.39	0.42	0.56	0.37	2e-3	0.01	0.13	0.16
`userkNN`	–	`Cos`	**0.13**	**0.07**	**0.01**	**0.06**	**0.44**	**0.45**	0.41	0.36	9e-4	0.01	0.10	0.18
`userCL[1]`	`Kmed`	`Cos`	0.06	0.03	4e-3	0.02	0.39	0.43	**0.61**	0.44	2e-3	0.01	**0.15**	0.17
`userCL[2]`	`Kmed`	`Cos`	0.06	0.03	4e-3	0.02	0.39	0.43	0.59	**0.45**	2e-3	0.01	**0.15**	0.17
`userCL[3]`	`Kmed`	`Cos`	0.07	0.04	5e-3	0.02	0.40	0.43	0.57	0.44	2e-3	0.01	**0.15**	0.17

Tabla 3.4: K-Medians Clustering en Recomendación *Top-N*.

obtenidos no son uniformes. Solo en ML100K, *k*-medians clustering provee beneficios, aunque mínimos, mientras que las ganancias mostradas sobre los otros datasets son nulas. Esto sugiere que la técnica de clustering no está jugando un papel preponderante en la estimación de las recomendaciones (Tabla 3.4).

La Figura 3.3 muestra el tamaño del cluster en donde cada usuario de prueba (usuario objetivo) fue ubicado. Además, se muestra el número de usuarios de entrenamiento ubicados en el mismo cluster que el usuario de prueba (`inside users`), que fueron utilizados como parte de su vecindario (seleccionados entre los *k* más similares).

En general, los algoritmos de clustering basados en prototipos entregan clusters desbalanceados. El tamaño óptimo debiese estar en torno a los 50 usuarios (n/k). En la práctica, K-Means y K-Medians generan clusters muy grandes y algunos grupos de muy pocos elementos. Spectral Clustering obtiene en promedio mejor desempeño en las recomendaciones, esto coincide con el hecho de que este algoritmo de clustering entrega clusters más pequeños (balanceados) en relación a *k*-means y *k*-medians, utilizando mayor cantidad de `inside users` en el vecindario (segunda fila de la Figura 3.3). Sin embargo, el método de clustering está lejos de brindar resultados ideales y

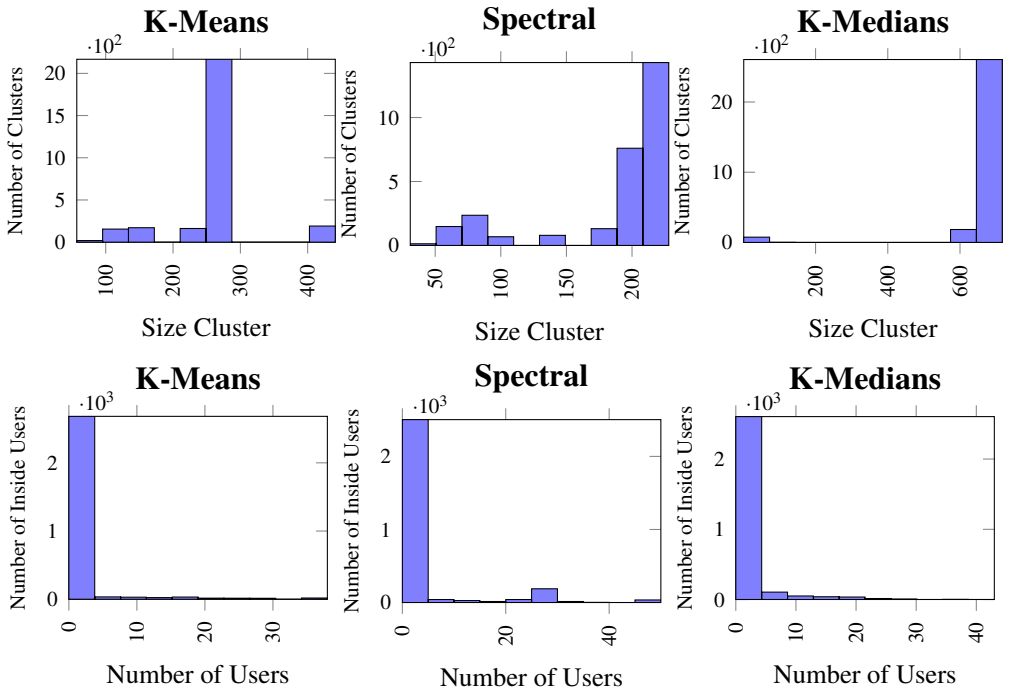


Figura 3.3: Histogramas del tamaño del cluster y número de `inside users` en ML100K.

agrupar los datos de manera precisa.

Una técnica de clustering que entrega una correcta agrupación de datos debiese generar clusters de tamaño balanceado, para que el algoritmo de recomendación pueda buscar usuarios similares solo dentro del cluster del usuario objetivo. De hecho, una respuesta apropiada a una consulta-NN sería retornar todos los usuarios dentro del cluster (del usuario objetivo) y solo algunos usuarios entre los clusters más cercanos. Desafortunadamente, *k*-means y *k*-medians no funcionan bien con clusters (en los datos originales) de diferentes formas y densidades. Esta es la principal razón por la cual el algoritmo de clustering no está siendo significativo y no se han obtenido grupos de usuarios bien formados.

Además, se debe tener en consideración que si no es posible garantizar que el punto de consulta (usuario objetivo) se ubica dentro de un cluster adecuado, entonces el cluster desde el cual se seleccionan los vecinos más cercanos está sujeto a los mismos problemas que sufre el algoritmo k-NN en su versión original (UBCF). En otras palabras, si no se obtienen clusters bien formados se está calculando distancias entre objetos extremadamente juntos en un espacio de alta dimensionalidad y la estimación de los vecinos más cercanos pierde el sentido (como le ocurre a `userkNN` en LastFM).

En la Figura 3.4 se muestra el número de `inside users` en cada dataset. Solo en

LastFM los `inside` superan a los `outside users`, y no es casualidad porque en este dataset el algoritmo de recomendación entrega sus mejores resultados. La cantidad de `inside users` está ligada con el correcto desempeño de la técnica de clustering, y por ende, con el éxito o fracaso del algoritmo de recomendación.

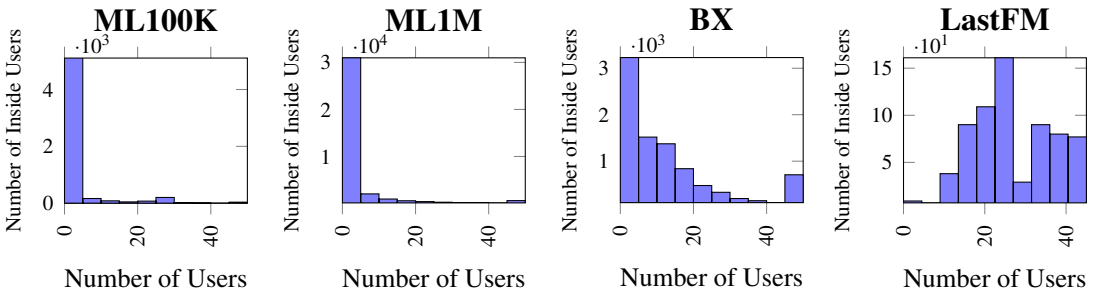


Figura 3.4: Histogramas de `inside users` en todos los datasets.

Balanced Clustering

Una de las ventajas de tener clusters balanceados por tamaño es poder parametrizar la función de proximidad y forzar una cierta proporción entre `inside/outside users` en el vecindario. Si la agrupación es realizada correctamente, los usuarios más cercanos (similares) deben quedar ubicados, en su mayoría, dentro del cluster. Por lo tanto, si se pretende mejorar la precisión, el algoritmo debe basarse en ese grupo de usuarios para generar las recomendaciones. Sin embargo, en algunos escenarios puede ser conveniente trabajar con menor cantidad de `inside users`, para favorecer la diversidad. La desventaja es que el proceso de balancear explícitamente los clusters puede resultar muy costoso computacionalmente.

método	parámetros		**ML100K**						**LastFM**					
			Precision		Recall		nDCG		Precision		Recall		nDCG	
	clust	*d*	@1	@10	@1	@10	@1	@10	@1	@10	@1	@10	@1	@10
`userCL[1]`	`Kmean`	`Cos`	0.44	0.32	0.02	0.10	0.84	0.85	**0.59**	**0.38**	2e-3	0.01	**0.16**	0.16
`userCL[2]`	`Kmean`	`Cos`	0.45	0.34	0.02	0.12	0.84	0.84	0.54	**0.38**	1e-3	0.01	**0.16**	0.17
`userCL[3]`	`Kmean`	`Cos`	0.50	0.32	0.02	0.12	0.85	0.85	0.55	**0.38**	2e-3	0.01	**0.16**	0.17
`userkNN`	–	`Cos`	**0.61**	**0.45**	**0.03**	**0.19**	0.85	0.85	0.41	0.36	9e-4	0.01	0.10	0.18
`userCL[1]`	`BKmean`	`Cos`	0.45	0.30	0.02	0.11	0.81	0.82	0.23	0.24	4e-4	6e-3	0.09	0.14
`userCL[2]`	`BKmean`	`Cos`	0.45	0.30	0.02	0.11	0.82	0.82	0.24	0.24	4e-4	6e-3	0.10	0.13
`userCL[3]`	`BKmean`	`Cos`	0.44	0.30	0.02	0.11	0.82	0.81	0.27	0.26	4e-4	5e-3	0.09	0.14

Tabla 3.5: Balanced Clustering en Recomendación *Top-N*.

La Tabla 3.5 muestra que el balanceo explícito de clusters en *k*-means produce peores recomendaciones. Esta clase de algoritmos de clustering basados en prototipos tie-

nen ciertas limitaciones en relación a la formación de clusters en condiciones de alta dimensionalidad. Como se demostró en la Figura 3.1, todos los objetos tienden a estar extremadamente juntos en un espacio de alta dimensionalidad. K-Means se basa implícitamente en distancias Euclidianas entre pares de objetos, por ende sufre los mismos problemas que `userkNN`.

La Figura 3.5 confirma que el balanceo de clusters es realizado correctamente. Sin embargo, el número de `inside users` utilizados para formar el vecindario tiende a cero en ambos datasets. Esto, anteriormente no ocurría en LastFM, y en este caso produce una merma en la calidad de las recomendaciones (Tabla 3.5). Por lo tanto, la técnica de clustering no es capaz de agrupar los datos correctamente, y el balanceo de clusters no entrega un beneficio en términos de precisión.

Para lograr una mejora significativa en las recomendaciones *top-N* (en términos de precisión), es necesario generar clusters bien formados y razonablemente balanceados. No obstante, algoritmos de clustering basados en prototipos no entregan garantías al respecto.

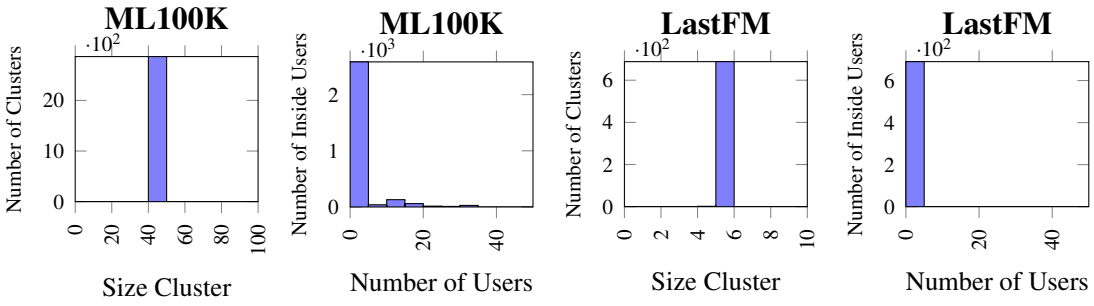


Figura 3.5: Histogramas del tamaño del cluster y número de `inside users` utilizando BALANCED K-means.

Soft Clustering

En modelos de clustering difuso (*soft* o *fuzzy clustering*) los objetos tienen un grado de pertenencia o membresía sobre los distintos clusters. A priori, este escenario puede calzar perfectamente con los sistemas de recomendación porque usuarios y productos suelen estar relacionados con más de un grupo. Por ejemplo, si los objetos son películas, naturalmente tienen distintos géneros que pueden servir para ser agrupados (de manera difusa) en distintos clusters. Lo mismo ocurre con los usuarios, que pueden mostrar interés en más de un género.

Como el objetivo es evaluar la función de distancia propuesta (discreta) en soft clustering, se asume un escenario donde el objeto queda ubicado en el cluster con mayor gra-

do de pertenencia. No obstante, una interesante línea de investigación puede ser adaptar la función para trabajar con proximidad continua.

Aunque soft clustering podría ser la forma más natural de agrupar usuarios/productos en sistemas de recomendación, tener objetos asociados a más de un tópico vuelve la clasificación compleja y no permite obtener los resultados esperados. De hecho, la calidad de las recomendaciones utilizando *hard* clustering es considerablemente mejor que al utilizar sus mismas variantes de manera *soft*, para *k*-means y *k*-medians (Tabla 3.6).

método	parámetros		**ML100K**						**LastFM**					
			Precision		Recall		nDCG		Precision		Recall		nDCG	
	clust	*d*	@1	@10	@1	@10	@1	@10	@1	@10	@1	@10	@1	@10
`userkNN`	–	`Euc`	0.38	0.26	0.01	0.07	0.82	0.82	0.18	0.14	4e-4	3e-3	0.04	0.12
`PureCL`	`FKmean`	–	0.37	0.28	0.02	0.10	0.80	0.81	0.27	0.24	6e-4	6e-3	0.09	0.16
`userCL[1]`	`FKmean`	`Euc`	0.42	0.31	0.02	0.12	0.79	0.81	0.25	0.35	5e-4	0.01	0.09	0.16
`userCL[2]`	`FKmean`	`Euc`	0.39	0.31	0.02	0.11	0.81	0.82	0.25	0.35	5e-4	0.01	0.09	0.16
`userCL[3]`	`FKmean`	`Euc`	0.44	0.32	0.02	0.11	0.81	0.82	0.25	0.35	5e-4	0.01	0.09	0.16
`PureCL`	`FKmed`	–	0.31	0.24	0.01	0.08	0.80	0.81	0.19	0.23	5e-4	6e-3	0.10	0.14
`userCL[1]`	`FKmed`	`Euc`	0.40	0.32	0.02	0.16	0.80	0.82	0.32	0.26	6e-4	6e-3	0.07	0.16
`userCL[2]`	`FKmed`	`Euc`	0.40	0.30	0.02	0.11	0.82	0.82	0.32	0.26	6e-4	6e-3	0.07	0.16
`userCL[3]`	`FKmed`	`Euc`	0.35	0.29	0.02	0.11	0.82	0.83	0.32	0.26	6e-4	6e-3	0.07	0.16
`userkNN`	–	`Cos`	**0.61**	**0.45**	0.03	**0.19**	**0.85**	**0.85**	**0.41**	0.36	9e-4	0.01	0.10	**0.18**
`userCL[1]`	`FKmean`	`Cos`	0.39	0.29	0.02	0.10	0.80	0.82	0.18	0.25	5e-4	6e-3	0.08	0.13
`userCL[2]`	`FKmean`	`Cos`	0.37	0.28	0.02	0.10	0.80	0.82	0.18	0.25	5e-4	6e-3	0.08	0.13
`userCL[3]`	`FKmean`	`Cos`	0.37	0.29	0.02	0.11	0.81	0.82	0.18	0.25	5e-4	6e-3	0.08	0.13
`userCL[1]`	`FKmed`	`Cos`	0.37	0.29	0.02	0.11	0.81	0.82	0.32	0.24	6e-4	5e-3	0.07	0.15
`userCL[2]`	`FKmed`	`Cos`	0.34	0.29	0.01	0.11	0.80	0.82	0.32	0.24	6e-4	5e-3	0.07	0.15
`userCL[3]`	`FKmed`	`Cos`	0.36	0.27	0.02	0.10	0.81	0.83	0.32	0.24	6e-4	5e-3	0.07	0.15

Tabla 3.6: Soft (*Fuzzy*) Clustering en Recomendación *Top-N*.

La simplicidad y eficiencia de los algoritmos de particionamiento, específicamente *k*-means, han ensombrecido otros tipos de clustering. Alternativas a *k*-means no han sido exploradas y se desconoce su alcance. No está claro si los enfoques de clustering basados en densidad o la agrupación jerárquica tienen algo que ofrecer en el ámbito de las recomendaciones.

Hierarchical Clustering

Los métodos de agrupamiento jerárquico tienen por objetivo construir una jerarquía de grupos. Para incluir esta clase de algoritmos de clustering en `userCL` (la función propuesta se basa en prototipos), se utilizó una estrategia aglomerativa en donde cada usuario comienza en su propio cluster y se mezclan clusters cercanos mientras uno sube en la jerarquía. A partir de eso se obtiene un dendrograma, que es utilizado para obtener

un número deseado de clusters a través de un corte en el árbol a una altura dada. Posteriormente, se estiman los prototipos utilizando el promedio del grado de membresía entre cada usuario de cada cluster.

La Tabla 3.7 muestra los resultados de clustering jerárquico aglomerativo (HAC) en recomendación *top-N*.

método	parámetros		ML100K						LastFM					
			Precision		Recall		nDCG		Precision		Recall		nDCG	
	clust	*d*	@1	@10	@1	@10	@1	@10	@1	@10	@1	@10	@1	@10
`userkNN`	–	`Euc`	0.38	0.26	0.01	0.07	0.82	0.82	0.18	0.14	4e-4	3e-3	0.04	0.12
`PureCL`	HAC	–	0.34	0.25	0.01	0.08	0.81	0.81	0.43	0.31	1e-3	8e-3	0.09	0.14
`userCL[1]`	HAC	`Euc`	0.44	0.32	0.02	0.12	0.85	0.84	0.34	0.29	7e-4	7e-3	0.08	0.16
`userCL[2]`	HAC	`Euc`	0.50	0.34	0.02	0.14	0.84	0.80	0.32	0.28	6e-4	7e-3	0.08	0.16
`userCL[3]`	HAC	`Euc`	0.49	0.33	0.02	0.14	0.82	0.79	0.32	0.27	6e-4	6e-3	0.08	0.16
`userkNN`	–	`Cos`	**0.61**	**0.45**	0.03	**0.19**	0.85	0.85	0.41	0.36	9e-4	0.01	0.10	0.18
`userCL[1]`	HAC	`Cos`	0.48	0.33	0.02	0.12	0.84	0.83	**0.61**	0.42	2e-3	0.01	0.16	0.16
`userCL[2]`	HAC	`Cos`	0.51	0.34	0.02	0.14	0.84	0.83	**0.61**	**0.43**	2e-3	0.01	0.16	0.16
`userCL[3]`	HAC	`Cos`	0.52	0.35	0.02	0.14	0.84	0.83	0.54	0.37	2e-3	0.01	**0.17**	0.18

Tabla 3.7: Clustering Jerárquico (HAC) en Recomendación *Top-N*.

La Tabla 3.7 indica que HAC puede ser una alternativa muy razonable para generar clusters de usuarios en recomendación *top-N*, entregando resultados equivalentes a Spectral Clustering y K-Medians.

3.5.3. Desempeño general de cada método de clustering

La Tabla 3.8 muestra el desempeño general (ganancia o perdida) de cada método de clustering en relación a `userkNN`, considerando el promedio de las diferentes funciones de proximidad propuestas. Los resultados demuestran que al utilizar distancia Euclidiana, todos los algoritmos de clustering, en combinación con `userCL`, entregan recomendaciones consistentemente mejores que `userkNN`, en ambos datasets.

Particularmente, en ML100K se destaca K-Medians como el método más preciso (14 % y 7 %, mejor que `userkNN` en P@1 y P@10, respectivamente), considerando solo distancia Euclidiana. En LastFM, Spectral Clustering supera en un 37 % y un 25 % a `userkNN` en P@1 y P@10, respectivamente.

Por otra parte, al considerar similaridad Coseno en el dataset de menores dimensiones (ML100K), todos los métodos muestran resultados negativos (para toda métrica). Centrándose en Precision, el método que menor diferencia evidencia es K-Medians (solo un 9 % por debajo de `userkNN` en P@1,10).

Sin embargo, al considerar el dataset de mayor dimensionalidad y dispersión (LastFM), varios algoritmos de clustering son capaces de superar a `userkNN`, siendo Spectral Clustering el que entrega los mejores resultados en Precision (probablemente gracias a la reducción de dimensionalidad). En P@1 está un 21 % por encima de `userkNN`. En P@10, lidera K-Medians con un 8 % y Spectral lo sigue muy de cerca con un 7 %.

En LastFM, no parece existir diferencia estadísticamente significativa entre los métodos en términos de Recall, porque los valores son extremadamente bajos. Esto demuestra que el problema de recomendación en LastFM es difícil por la baja densidad del dataset (dispersión del 99.52 %) y el gran desbalance entre usuarios y productos (efecto *Long-Tail*).

En general, entre todas las métricas y ambos datasets, los métodos de clustering de peor desempeño (mayor diferencia negativa en relación a `userkNN`) son los modelos de Soft Clustering y Balanced K-Means. Ninguno es capaz de superar a `userkNN` en LastFM, donde el resto se impone y marca una gran diferencia positiva.

Un hecho interesante es el buen desempeño mostrado por Clustering Jerárquico Aglomerativo (HAC), logrando resultados similares a Spectral Clustering y K-Medians. Este tipo de algoritmos de clustering, escasamente explorado en sistemas de recomendación, puede resultar una alternativa interesante en escenarios de diversidad.

		ML100K						**LastFM**					
parámetros		Precision		Recall		nDCG		Precision		Recall		nDCG	
clust	*d*	@1	@10	@1	@10	@1	@10	@1	@10	@1	@10	@1	@10
Kmean	Euc	0.10	0.05	0.01	0.06	0.01	-0.01	0.33	0.23	0.00	0.01	0.08	0.04
spectral	Euc	0.14	0.05	0.02	0.07	0.00	-0.02	0.37	0.25	0.00	0.01	0.10	0.04
Kmed	Euc	0.14	0.07	0.01	0.07	0.01	0.00	0.35	0.23	0.00	0.01	0.09	0.04
FKmean	Euc	0.04	0.05	0.01	0.04	-0.02	0.00	0.07	0.21	0.00	0.01	0.05	0.04
FKmed	Euc	0.00	0.04	0.01	0.06	-0.01	0.00	0.14	0.12	0.00	0.00	0.03	0.04
HAC	Euc	0.10	0.07	0.01	0.07	0.02	-0.01	0.15	0.14	0.00	0.00	0.04	0.04
Kmean	Cos	-0.15	-0.12	-0.01	-0.08	-0.01	0.00	0.15	0.02	0.00	0.00	0.06	-0.01
spectral	Cos	-0.11	-0.11	-0.01	-0.06	-0.01	0.00	0.21	0.07	0.00	0.00	0.05	-0.01
Kmed	Cos	-0.09	-0.09	-0.01	-0.04	-0.02	-0.01	0.18	0.08	0.00	0.00	0.05	-0.01
BKmean	Cos	-0.16	-0.15	-0.01	-0.08	-0.03	-0.03	-0.16	-0.11	0.00	0.00	-0.01	-0.04
FKmean	Cos	-0.23	-0.16	-0.01	-0.09	-0.05	-0.03	-0.23	-0.11	0.00	0.00	-0.02	-0.05
FKmed	Cos	-0.25	-0.17	-0.01	-0.08	-0.04	-0.03	-0.09	-0.12	0.00	-0.01	-0.03	-0.03
HAC	Cos	-0.11	-0.11	-0.01	-0.06	-0.01	-0.02	0.18	0.05	0.00	0.00	0.06	-0.01

Tabla 3.8: Diferencia positiva/negativa de cada método de clustering y su función de distancia con respecto a `userkNN`.

3.5.4. Eficiencia en Recomendación *Top-N*

Todos los algoritmos comparados en la Sección 3.5.4 han sido implementados en R, y su evaluación ha sido realizada a través del framework recommenderlab. Todas las implementaciones en recommenderlab se basan en la misma estructura para asegurarse que cualquier diferencia de tiempo sea debido al algoritmo mismo, y no debido a la implementación. Los experimentos han sido realizados en un Intel(R) Core i5 CPU 3.20 GHz con 16 GB RAM.

La Tabla 3.9 muestra el tiempo de cómputo experimental aproximado de cada algoritmo en recomendación *top-N*. Las columnas correspondientes a mt (model time) y tt (testing time), representan el tiempo usado por el método en el aprendizaje del modelo (e.g., el clustering entre usuarios en `userCL`) y el tiempo de recomendación (e.g., el cálculo de los kNN), respectivamente.

	ML100K						**LastFM**					
	d	mt	tt	*d*	mt	tt	*d*	mt	tt	*d*	mt	tt
`userkNN`	`Euc`	–	20.99(m)	`Cos`	–	2(m)	`Euc`	–	6.18(m)	`Cos`	–	31.25(s)
K-Means	`Euc`	0.92(s)	50.36(m)	`Cos`	0.54(s)	17.34(m)	`Euc`	2.9(s)	22.83(m)	`Cos`	2.61(s)	26.33(m)
spectral	`Euc`	4.26(s)	40.23(s)	`Cos`	4.24(s)	14.36(s)	`Euc`	1.46(s)	0.22(s)	`Cos`	1.2(s)	0.05(s)
K-Median	`Euc`	4.74(s)	39.32(s)	`Cos`	4.58(s)	11.54(s)	`Euc`	1.22(s)	0.32(s)	`Cos`	1.11(s)	0.08(s)
BKMeans	`Euc`	11.85(m)	33.59(s)	`Cos`	11.63(m)	6.74(s)	`Euc`	6.74(s)	0.43(s)	`Cos`	6.94(s)	0.09(s)
FKMeans	`Euc`	3.42(m)	6.12(s)	`Cos`	3.19(m)	1.87(s)	`Euc`	9.37(m)	0.07(s)	`Cos`	9.1(m)	0.03(s)
FKMedian	`Euc`	44.94(m)	5.86(s)	`Cos`	41.62(m)	1.86(s)	`Euc`	13.16(m)	0.07(s)	`Cos`	13.63(m)	0.03(s)
HAC	`Euc`	10.6(s)	44.96(m)	`Cos`	10.65(s)	12.99(m)	`Euc`	4.8(s)	1.02(h)	`Cos`	4.26(s)	49.72(m)

Los números en mt/tt con (s), (m) y (h) representan el tiempo en segundos, minutos y horas, respectivamente.

Tabla 3.9: Tiempo de cómputo experimental de cada método en ML100K y LastFM.

La Tabla 3.9 demuestra que la similaridad Coseno es muy eficiente, especialmente en este caso donde los vectores son muy dispersos (solo necesita considerar dimensiones no nulas para el cálculo de la similaridad).

Al comparar `userkNN` con respecto a `userCL` se aprecia que la eficiencia del modelo varía según el algoritmo de clustering utilizado en la fase de entrenamiento. Los modelos de soft clustering son muy eficientes en términos de recomendación (tt), pero son costosos de entrenar (mt). En contraste, K-Means y HAC realizan el aprendizaje de forma altamente eficiente, pero el tiempo de recomendación es elevado (incluso peor que `userkNN`). Este gasto extra de tiempo se debe al aumento en la densidad de la matriz de ratings original y al ser incapaz de manejar outliers correctamente. Los modelos con mejor rendimiento son Spectral Clustering y K-Medians. En ML100K, la aceleración con respecto a `userkNN` es de 32x considerando distancia Euclidiana y 10x con similaridad Coseno. En LastFM, la aceleración es aún mayor, 1200x considerando distancia Euclidiana y 400x con similaridad Coseno.

La Tabla 3.10 permite explorar los tiempos de cómputo para los otros dos datasets restantes, considerados de mayor escala (ML1M y BX). En este caso, se comparan los tiempos de los algoritmos más eficientes (Spectral Clustering y K-Medians), con respecto a `userkNN`.

	ML1M						BX					
	d	mt	tt	*d*	mt	tt	*d*	mt	tt	*d*	mt	tt
`userkNN`	`Euc`	–	28.52(h)	`Cos`	–	2.48(h)	`Euc`	–	12.48(h)	`Cos`	–	1.13(h)
spectral	`Euc`	3.68(m)	21.26(m)	`Cos`	3.59(m)	5.56(m)	`Euc`	2.88(m)	4.28(m)	`Cos`	2.88(m)	1.04(m)
K-Median	`Euc`	3.73(m)	23.30(m)	`Cos`	3.78(m)	4.55(m)	`Euc`	2.97(m)	5.14(m)	`Cos`	2.89(m)	0.97(m)

Los números en mt/tt con (s), (m) y (h) representan el tiempo en segundos, minutos y horas, respectivamente.

Tabla 3.10: Tiempo de cómputo experimental de cada método en ML1M y BX.

La Tabla 3.10 demuestra que la diferencia de tiempo entre Spectral clustering o K-Medians, con respecto a `userkNN`, aumenta significativamente en datasets large-scale. El tiempo de recomendación (tt) en `userkNN` está en el orden de las horas, mientras que los métodos basados en clustering (`userCL`), solo requieren minutos para entregar una lista *top-N* en ML1M y BX. Estos resultados ilustran el potencial de la propuesta en términos de eficiencia.

3.6. Discusión y Conclusiones

En este capítulo, se propuso un método para recomendación *top-N* basado en clustering, `userCL`, el cual utiliza una nueva función de distancia para calcular el vecindario de un usuario dado. Se compararon distintos modelos de clustering con la finalidad de evaluar el desempeño de `userCL` en relación a `userkNN` y `PureCL`. Este último, es computacionalmente eficiente, pero sufre de baja calidad predictiva al restringir el espacio de búsqueda. En tanto, `userkNN` entrega alta precisión, aunque es costoso en datasets a gran escala. Finalmente, `userCL` genera las recomendaciones agregando ratings de usuarios que son similares a aquellos localizados en el mismo cluster. Aunque, a diferencia de `PureCL`, `userCL` considera los ratings de usuarios en diferentes clusters, permitiendo mejorar considerablemente la calidad de las recomendaciones. Los experimentos sugieren que `userCL` es capaz de reducir la brecha en términos de precisión con respecto a `userkNN` (uno de los principales objetivos de esta Tesis), y entrega una ventaja significativa en términos de eficiencia en el cómputo de los KNN (una aceleración de 10x utilizando Spectral Clustering o K-Medians en ML100K). El mérito de este resultado es que los modelos de clustering tienen como propósito principal reducir el tiempo computacional involucrado en sistemas a gran escala, donde no es posible almacenar la

matriz de ratings en memoria principal, y en este caso, son también capaces de producir recomendaciones de calidad similar a un método de memoria completa.

Resulta adecuado agregar que la ventaja obtenida en términos de eficiencia puede ser aún más significativa si `userCL` es implementado en un sistema distribuido, donde cada cluster sea procesado de forma paralela en una máquina diferente. La gran ventaja en términos de eficiencia de este modelo de recomendación, basado en clustering, es que al utilizar los prototipos para estimar las distancias KNN, el costo en tiempo y espacio se reduce drásticamente en comparación a cuando se trabaja con vectores en un espacio de alta dimensionalidad.

A pesar que Spectral K-Means y K-Medians lograron resultados promisorios en los datasets evaluados, ambos tienen evidentes limitaciones. En primer lugar, estos algoritmos no pueden prevenir la formación de clusters altamente sesgados y/o desbalanceados. Los algoritmos de clustering basados en prototipos usualmente generan algunos clusters que son extremadamente pequeños. Es necesario obtener clusters bien formados y razonablemente balanceados para lograr una mejora significativa en la calidad de las recomendaciones. No obstante, los resultados obtenidos son esperanzadores considerando la ventaja que demostró `userCL` con respecto a `userkNN` en LastFM. Este dataset tiene alta dimensionalidad y dispersión de datos, un escenario común en sistemas reales de información. Por lo tanto, bajo condiciones relativamente ideales, `userkNN` saca diferencia en términos de precisión. Pero, cuando los datos comienzan a tener mayor cantidad de atributos (dimensiones) y menor densidad (más entradas nulas) se originan serios problemas para realizar correctamente el cómputo de los KNN.

Los experimentos también demostraron las propiedades positivas de `userCL`, con respecto a otros métodos de recomendación. En términos de las consultas de usuarios más cercanos (NN), la ventaja es grande porque las técnicas de clustering proveen una forma de regularización (suavizado) que parece disminuir el efecto que produce la dispersión de datos en el cálculo de las distancias sobre un punto consulta dado (usuario objetivo). Además, al realizar clustering sobre un espacio de dimensionalidad reducida, aumenta la eficiencia del modelo y se obtienen clusters bien separados. Sin embargo, si el modelo se desvía un poco (por ejemplo, si el usuario objetivo no se ubica dentro de un cluster bien formado), la capacidad del método puede perder estabilidad.

Finalmente, es importante destacar el hecho de que las métricas utilizadas en este capítulo para evaluar los algoritmos (Precision, Recall, NDCG) están enfocadas en medir la exactitud del modelo en términos de cuantos productos relevantes el algoritmo es capaz de recuperar. Sin embargo, esta forma de evaluación tiene limitaciones y no permite medir de una forma realista la calidad de un sistema de recomendación (desde el punto de vista de la satisfacción real del usuario hacia las recomendaciones). Para lograr conclusiones más acabadas sobre el real aporte del método propuesto es necesario explorar otras formas de evaluación no tradicionales.

3.6.1. Trabajo Futuro

- Explorar el real alcance de la función de proximidad propuesta en términos de las relaciones latentes entre usuarios y productos que la función es capaz de formar, incluso en vectores con alta dispersión.
- Adaptar la función de proximidad a espacios continuos, aprovechando la ventaja de los modelos de soft clustering. En este mismo escenario, la función de proximidad puede incluir un factor de decaimiento.
- Realizar una evaluación más extensiva de la propuesta, considerando un sistema real de Big Data. Una implementación de userCL en Hadoop es aconsejable para cumplir esta exigencia.
- Extender el método propuesto para funcionar sobre algoritmos basados en productos más cercanos (Item-based) surge como una promisoria línea de investigación, ya que en este tipo de métodos los productos son representados por vectores sobre el espacio de características de los usuarios, incrementando la dimensionalidad de los vectores usados para el cálculo de los KNN, un escenario que userCL es capaz de manejar satisfactoriamente.
- Implementar el método propuesto en un sistema distribuido, asignando cada cluster a una máquina diferente. De esta manera, es posible calcular las distancias intra cluster de una forma muy eficiente. Dado que para el cómputo de las distancias inter cluster solo es necesario mantener en memoria los prototipos, el intercambio de información entre servidores resulta sencillo.
- Desarrollar un sistema de recomendación incremental basado en el modelo propuesto capaz de asignar un nuevo usuario al cluster más cercano en tiempo constante. Esto supone una ventaja considerable en comparación a UBCF dado que este algoritmo requiere calcular nuevamente la matriz de similaridad cuando ingresan nuevos datos al sistema.

Parte II

DIVERSIDAD EN RECOMENDACIÓN TOP-N

Capítulo 4

Evaluación de novedad y diversidad

4.1. Motivación

Una de las más importantes conclusiones originadas a partir de los experimentos de la Parte I es que al evaluar sistemas de recomendación a través de métricas enfocadas en la precisión, algoritmos voraces siempre prevalecerán debido a que por su naturaleza avara eligen la opción óptima en cada paso, llevando a cabo una búsqueda completa por sobre todo el espacio de posibles soluciones. Por esta razón, *k-Nearest Neighbor* es capaz de recomendar (recuperar) mayor cantidad de instancias relevantes en una lista *top-N*.

Entonces, surge la necesidad de encontrar nuevas medidas de evaluación, más realistas y capaces de medir correctamente la satisfacción del usuario frente a recomendaciones. La relevancia binaria de precision/recall, aunque utilizada en la actualidad, es insuficiente para medir la calidad de una lista de recomendación. En tanto, NDCG, se encuentra más en la línea de evaluación que se pretende seguir. Desafortunadamente, la evaluación no tradicional en sistemas de recomendación (ampliable a recuperación de información), no ha sido explorada lo suficiente para disponer de una métrica adecuada que agrupe todas las condiciones necesarias para evaluar listas *top-N*. Así, el objetivo de la Parte II de la Tesis se centra en lograr formular una propuesta de métrica aceptada por la comunidad que permita valorar las necesidades del usuario. Básicamente, evaluar la "utilidad" de un sistema recomendador, o de un motor de búsqueda texto-documento en IR.

Los sistemas de recomendación han sido frecuentemente evaluados usando indices basados en variantes y extensiones de métricas relacionadas con precisión. Como esas medidas están sesgadas hacia productos populares, una lista de recomendaciones solo necesita incluir un par de productos populares para funcionar bien. Con el objetivo de encontrar una solución a este problema, durante el Capítulo 4 se exploran propuestas

de evaluación disponibles en el Estado del Arte y se proponen nuevos enfoques para evaluar novedad y diversidad.

Los enfoques basados en novedad modelan el estado "nuevo" en relación a algo que ya se conoce. Entonces, los enfoques de novedad están comúnmente basados en productos vistos o calificados para calcular estas medidas. Por otra parte, enfoques basados en diversidad modelan la cualidad de un producto de estar compuesto por diferentes elementos o características. Medidas de diversidad se basan en medir que tan diverso es el contenido de un producto en términos de la presencia/ausencia de un número predefinido de *nuggets* (tópicos, características) de información. Sin embargo, como los contenidos de los productos están también sesgados a contenidos populares (e.g., el género drama en las películas, o pop en música), medidas basadas en diversidad sufren también el efecto de ese sesgo hacia lo "popular". De acuerdo a esto, en este capítulo se presenta un nuevo enfoque de evaluación para sistemas de recomendación basado en cuán novedosos son los elementos que componen cada producto. Entonces, se proponen nuevas medidas para evaluar *novedad* en *contenido* (*content-novelty*). Los resultados experimentales muestran que la propuesta resulta exitosa al medir novedad y diversidad en sistemas de recomendación. La evaluación es efectiva, atenuando el efecto del sesgo en el contenido y proporcionando una alternativa válida y consistente con respecto a las medidas actuales en el Estado del Arte.

4.2. Introducción

La diversificación se ha convertido en un tópico fundamental en sistemas de recomendación, principalmente porque es un forma de aumentar la calidad de la experiencia del usuario frente al uso del sistema. La diversidad, aunque interesante, es relativamente nueva (descrita por primera vez en 2001 por Bradley y Smith [16]), y por ende, ofrece una gran posibilidad para nuevas investigaciones. Es también interesante porque a diferencia de otros problemas en RS, la diversidad y novedad requieren una participación mucho más activa por parte del usuario, ya que pueden ser percibidas de forma muy distinta entre dos personas.

Novedad es la cualidad de ser nuevo o diferente a lo que ya se conoce. Por lo tanto, la novedad de un producto se percibe desde el punto de vista del usuario. De acuerdo a esto, es común usar *views* (visualizaciones de un usuario sobre un producto) o *rates* (calificaciones/preferencias de un usuario sobre un producto) para calcular novedad. Naturalmente, como los *rates* son una declaración de preferencia explícita permite lograr estimaciones más precisas.

Las recomendaciones pueden ser consideradas novedosas cuando se entregan productos que el usuario no conoce. Una manera sencilla y fácil de lograr novedad es filtrar todos los productos que el usuario ya conoce. Sin embargo, muchas veces no se co-

nocen todos los productos que el usuario ha consumido en el pasado, por ende no es trivial lograr novedad en listas de recomendación. Es importante que en el intento por aumentar la diversidad en las recomendaciones, introduciendo productos novedosos, no se pierda de vista la relevancia de los mismos. Mientras productos irrelevantes pueden ser nuevos para el usuario, su aparición en la lista de recomendación puede carecer de valor intrínseco para el mismo.

Diversidad es la cualidad de estar compuesto por diferentes elementos. Por ende, la diversidad se percibe desde el punto de vista del producto. Es común usar características basadas en contenido para caracterizar el contenido de un producto en términos de la presencia/ausencia de un número predefinido de tópicos de información. Es también posible usar conocimiento experto para etiquetar productos según su contenido. La diversidad ha sido definida como el opuesto a similaridad. En algunos casos, sugerir un conjunto de productos similares puede no ser útil para un usuario, porque le puede tomar mucho más tiempo explorar la gama de productos disponibles y encontrar el que busca.

4.3. Estado del Arte

El Estado del Arte posee medidas de novedad y diversidad para evaluar sistemas de recomendación. La investigación se encuentra enfocada en la formulación de métricas de evaluación capaces de proveer información acerca de la diversidad en las listas de recomendaciones. El concepto de diversificar comienza con Bradley y Smyth [16], quienes definen diversidad como el opuesto de similaridad. Esto fue seguido por Fleder y Hosanagar [35] en el 2007, quienes realizan un experimento para mostrar que la mayoría de los sistemas de recomendación reducen la diversidad de los productos recomendados enfocándose en la precisión. Clarke *et al.* [25] van un paso más allá y tratan de combinar diversidad con novedad dentro de una nueva medida de relevancia para recuperación de documentos, basándose en la conocida métrica Normalized Discounted Cumulative Gain (NDCG). En 2011, Vargas [84] define una métrica de evaluación para diversidad de un producto basada en una función de descuento decreciente. Castells *et al.* [22] desarrollan esa idea con una métrica que también considera la posición del producto y la relevancia cuando se está determinando la diversidad de la lista de recomendaciones. Por otro lado, Hu y Pu [48] se enfocan en determinar como la diversidad de la recomendación es percibida por los usuarios llevando a cabo un estudio real subjetivo, donde muestran que la organización y la categorización juegan un rol importante en como los usuarios perciben la diversidad. Vargas sigue esta sugerencia y formaliza métodos de diversificación y técnicas de evaluación, considerando el ranking y la relevancia como aspectos fundamentales en el proceso de recomendación. En 2013 [20] Castagos *et al.* también realizan un interesante caso de estudio con usuarios reales donde comparan la aceptación y satisfacción de los usuarios en las listas de recomendaciones presentadas.

Sus resultados experimentales concluyen que si bien la diversificación puede reducir la tasa de aceptación de los usuarios, al mismo tiempo aumenta la satisfacción con el sistema. Hullier *et al.* [64] realizan un experimento similar presentando un recomendador de música que mantiene las canciones de preferencia de los usuarios, el contexto y la diversidad de todos los productos musicales escuchados por el usuario. Jiang *et al.* [53] manejan el problema de la evaluación de diversidad desde un ángulo diferente y miden la calidad de las recomendaciones (diversidad) tomando como base la probabilidad de elección. Su objetivo era combinar la evaluación de relevancia y diversidad en una misma medida. Vargas *et al.* [85] se enfocan en el género como uno de los atributos claves en la evaluación de diversidad y proponen un framework Binomial para medir diversidad de géneros en cada lista de recomendación.

Detalles acerca de la definición (y ecuación si está disponible) de diversidad, técnica de evaluación implementada y otros puntos relevantes que mencionar, sobre las propuestas para medir diversidad y novedad, se pueden encontrar en la Tabla 4.1.

Ref	Ecuación	Evaluación	Comentario
[16]	$D = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}(1-\text{sim}(c_i,c_j))}{n/2(n-1)}$	Dataset de anuncios.	Diversidad es la disimilaridad promedio entre pares de productos.
[35]	$G = 1 - 2\int_0^1 L(u)du$	Data sintética en ambiente simulado.	Diversidad es representada como el coeficiente de Gini.
[25]	$G[k] = \sum_{i=1}^{m} J(d_k,i)(1-\alpha)^{r_i,k-1}$	TREC2006 dataset.	Diversidad forma parte de la métrica NDCG.
[84]	$ILD(i_k\|u,R) = C'_k\sum_l \text{disc}(l\|k)p(rel\|i_l,u)\text{dist}(i_k,i_l)$	MovieLens 1M	Diversidad como relevancia, similaridad y posición en la lista.
[48]	Diversidad percibida por usuarios. N/A	Cuestionario con 20 participantes.	No ofrece una nueva medida de diversidad.
[Vargas]	$IA-nDCG = \sum p(a\|u)\text{NDCG}(u\|a)$ $G[k] = \sum_a r(i_k,u)(1-\alpha)^{\sum_{l=0}^{k-1} r(i_l,u;a)}$	Movielens y Last.fm	Variación de NDCG, reemplaza los *nuggets*.
[20]	$D = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}(1-\text{sim}(c_i,c_j))}{n/2(n-1)}$	250 voluntarios.	Estudio que confirma la viabilidad de [16].
[64]	$D = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}(1-\text{sim}(c_i,c_j))}{n/2(n-1)}$	Last.fm dataset.	No ofrece una nueva medida de diversidad.
[85]	$\text{BinomDiv}(R) = \text{Coverage}(R) * \text{NonRed}(R)$	Movielens y Netflix.	Diversidad es visto como covertura de género y no-redundancia.

Tabla 4.1: Definición y evaluación de diversidad en papers más relevantes.

4.3.1. Medidas de Novedad

Castells *et al.* [21] proponen un framework probabilista para evaluar novedad y diversidad en Sistemas de Recomendación. El framework propuesto desarrolla un modelo probabilista del problema siguiendo un enfoque producto-novedad. Básicamente, hay dos enfoques para medir novedad, el primero se basa en cuán difícil es descubrir un producto en un dataset dado. El segundo, se basa en distancias entre el producto y el resto del dataset. Cada medida es definida para un contexto dado, el cual incluye la perspectiva desde donde se estima la medida. La novedad *long-tail* puede ser medida usando una matriz de ratings R, donde la "*sorpresa*" (novedad) se mide de la lista de

productos consumidos o vistos por el usuario objetivo I_u, la diversidad intra-lista es medida desde la lista de recomendaciones que el sistema estima para el usuario objetivo R_u. Finalmente, el contexto general corresponde a todo el conjunto de recomendaciones provistas por el sistema a todos los usuarios $\{R_v^s\}_{v \in U}$.

La dificultad de descubrir un producto depende del contexto donde se considera el descubrimiento. Sea θ un contexto del producto i. La probabilidad de descubrir un producto dado en un dataset es denotado por $p(seen|i,\theta)$. Entonces, la novedad de i puede ser modelada por el complemento del descubrimiento $\text{nov}^C = 1 - p(seen|i,\theta)$, esto es recíproco a $\text{nov}^R(i|\theta) = \frac{1}{p(seen|i,\theta)}$. Con respecto a las medidas basadas en distancias, la novedad puede ser definida por $\text{nov}^{AD}(i|\theta) = \sum_{j \in \theta} p(j|choose,u,\theta) \cdot \text{dist}(i,j)$, donde $p(j|choose,u,\theta)$ es la preferencia del usuario u sobre el producto j en un contexto θ. Entonces, reemplazando θ por un contexto dado, se define una medida de novedad específica en el framework. Por ejemplo, $\theta = R$ define el complemento de popularidad de $\text{nov}^C(i|R) = 1 - p(seen|i,R)$ o el libre descubrimiento $\text{nov}^S(i|R) = -\log(p(seen|i,R))$. En tanto, $\theta = I_u$ (conjunto de productos evaluados por el usuario), define un perfil para el cálculo de las distancias en $\text{nov}^{AD}(i|I_u) = \sum_{j \in I_u} p(j|choose,u,R) \cdot \text{dist}(i,j)$. Finalmente, $\theta = R_u \setminus \{i\}$ define la distancia intra-lista $\text{nov}^{ILD}(i|R \setminus \{i\}) = \sum_{j \in R_u \setminus \{i\}} p(j|choose,u,R_u \setminus \{i\}) \cdot \text{dist}(i,j)$.

Dentro del framework se han incorporado funciones de ganancia acumulada. Un factor de descuento es agregado dentro de la función de ganancia modelando el costo de alcanzar un producto relevante luego de $r-1$ recomendaciones. La función de ganancia acumulada sobre R_u está definida por $m(R_u) = C_u \sum_{i \in R_u} \text{disc}(k_i) \cdot p(rel|i,u) \cdot \text{nov}(i)$, donde $p(rel|i,u)$ es el rating relevante para el producto i provisto por el usuario u, $\text{disc}(k_i)$ es una función de descuento y C_u es un factor de normalización. Notar que $m(R_u)$ permite definir nDCG [52] reemplazando $\text{disc}(k_i)$ con $\frac{1}{\log(r)}$, descartando la medida de novedad. Si se reemplaza $\text{nov}(i)$ con $\alpha^{r_j,k-1}$ se obtiene α-nDCG [25].

Estimadores convencionales para $p(seen|i,R)$ son $\frac{|U_i|}{|U|}$ o $\frac{|U_i|}{|R|}$. Para una elección relativa el framework utiliza $p(j|choose,u,I_u) = \frac{p(rel|j,u)}{\sum_{j' \in I_u} p(rel|j',u)}$ en la estimación del perfil o $p(j|choose,u,R_u \setminus \{i\}) = \frac{\text{disc}(k_j|k_i) \cdot p(rel|j,u)}{\sum_{j' \in R_u \setminus \{i\}} \text{disc}(k'_j|k_i) p(rel|j',u)}$ para estimación intra-lista. Finalmente, $p(rel|i,u) = 1$ si $(u,i) \in R_{test} \wedge r_{u,i} \geq \rho$, (en caso contrario 0), donde ρ corresponde al umbral de relevancia (todos los productos con rating mayor o igual a ρ son considerados relevantes).

Entonces, Vargas [83] define una familia de medidas de novedad para sistemas recomendadores. Las más destacadas son "Expected Popularity Complement (EPC)", "Expected Free Discovery (EFD), "Expected Profile Distance (EPD)", "Expected Intra-List Distance (EILD)" y "Expected Inter-User Discovery Complement (EIUDC)", detalladas a continuación:

$$\mathrm{EPC}(R_u) = C_u \sum_{i \in R_u} \mathrm{disc}(k_i) \cdot p(rel|i,u) \cdot \left(1 - \frac{|U_i|}{|U|}\right)$$

$$\mathrm{EFD}(R_u) = -C_u \sum_{i \in R_u} \mathrm{disc}(k_i) \cdot p(rel|i,u) \cdot \log\left(\frac{|U_i|}{|R|}\right)$$

$$\mathrm{EPD}(R_u) = C_u \sum_{i \in R_u} \sum_{j \in I_u} \mathrm{disc}(k_i) \cdot p(rel|i,u) \cdot p(rel|j,u) \cdot \mathrm{dist}(i,j)$$

$$\mathrm{EILD}(R_u) = \sum_{i,j \in R_u} C_i \cdot \mathrm{disc}(k_i) \cdot \mathrm{disc}(k_j|k_i) \cdot p(rel|i,u) \cdot p(rel|j,u) \cdot \mathrm{dist}(i,j)$$

$$\mathrm{EIUDC}(\{R_v^S\}_{v \in U}) = C_S \sum_{u \in U} C_u \sum_{i \in R_u} \mathrm{disc}(k_i^{R_u}) \cdot p(rel|i,u) \cdot \left(1 - \frac{\sum_{v \in U} \mathrm{disc}(k_i^{R_v})}{|U|}\right)$$

4.3.2. Medidas de Diversidad

La comparación de una lista de productos recomendados R_u con una lista ideal es una idea exitosa que viene de la comunidad de recuperación de información (information retrieval). J*ä*rvelin y Kek*älä*inen [52] propusieron la medida *normalized discount cumulative gain, nDCG*, que incorpora el modelo tradicional de evaluación del enfoque precision/recall. La idea fue considerar la posición relativa de los documentos en el ranking para descontar desde la función de ganancia un factor proporcional a la posición en la lista. Documentos relevantes ubicados en las primeras posiciones de la lista producen mayores ganancias que si estuvieran ubicados al final de la lista. Siguiendo esta idea, Clarke *et al.*[25] proponen una variante de nDCG para evaluación de diversidad, llamada α-nDCG. En este enfoque, cada producto contribuye a R_u con tópicos o *nuggets*, que pueden ser géneros de películas/música/libros o intervenciones del usuario en el caso de las consultas web. Sea $\{n_1, \ldots, n_m\}$ un conjunto de nuggets en un dataset. Se define la función $N(i, n_j) = 1$ si el producto i contiene al nugget n_j, 0 en caso contrario. Entonces, el número de nuggets en i corresponde a $\sum_{j=1}^{m} N(i, n_j)$. Para penalizar la redundancia de nuggets en R_u, se define $r_{j,k-1} = \sum_{i=1}^{k-1} N(i, n_j)$, que corresponde al número de productos en R_u que contiene n_j hasta la posición $k-1$ de la lista. Por conveniencia, $r_{j,0} = 0$. Entonces, la función de ganancia en la posición k esta definida por:

$$G[k] = \sum_{j=1}^{m} N(d_k, n_j) \cdot \alpha^{r_{j,k-1}}, \quad 0 \leq \alpha \leq 1. \tag{4.1}$$

Si se desea incrementar el peso relativo de la redundancia de nuggets, se deben tomar valores bajos para α. Aplicando una función de descuento, se puede obtener la curva α-DCG de R_u. Finalmente, la versión normalizada α-nDCG se obtiene comparando α-DCG con la lista ideal en términos de diversidad. La lista ideal es voraz en términos del

número de nuggets, maximizando la ganancia acumulada en cada nivel de R_u.

4.4. Propuesta de evaluación

Se propone un conjunto de medidas de novedad basadas en contenido para sistemas de recomendación. La propuesta pretende cerrar la brecha existente entre medidas de diversidad (α-NDCG) y medidas de novedad. La idea es usar el enfoque de novedad en el contexto de análisis de diversidad, un enfoque que se conocerá como novedad basada en contenido. Los principales esfuerzos en métricas de novedad están basados en similaridad en el contenido del producto. Aquellas métricas están definidas en términos de diversificación de contenido, logrando el óptimo cuando el conjunto de productos recomendados maximizan su disimilaridad interna. Este enfoque tiene inconvenientes: en datos de alta dimensionalidad, la estimación de disimilitud sufre los efectos de la "maldición de la dimensionalidad". Con el aumento del número de dimensiones, los productos tienden a juntarse unos a otros, afectando la maximización de la disimilaridad interna. Este escenario es común cuando el contenido del producto es calculado sobre un espacio de características basado en el usuario (los productos son vectores de los ratings del usuario). En este caso, la dimensionalidad del espacio de características es mucho mayor que el número de productos, cayendo en la maldición de la dimensionalidad. Para manejar esta limitación se propone un enfoque agnóstico basado en dimensionalidad para análisis de diversidad. En lugar de usar funciones de disimilaridad para estimar la diversidad en el contenido, se aprovechan los nuggets para estimar la diversidad. Usando nuggets de contenido para análisis de diversidad se podrá reducir el efecto del sesgo inherente en el contenido de los productos (géneros desbalanceados) o las calificaciones de los usuarios (productos populares sobrevalorados).

4.4.1. Sesgo en los géneros

Se usará MovieLens 100K (ML100K) para ilustrar el efecto del sesgo presente en los géneros de los productos. ML100K registra 100000 ratings de 943 usuarios sobre una colección de 1664 películas. Las cuales han sido etiquetadas sobre géneros por expertos y muchas películas registran más de una etiqueta, mostrando que el contenido se correlaciona con más de un género. MovieLens considera 19 géneros (`Action`, `Adventure`, `Animation`, `Children's`, `Comedy`, `Crime`, `Documentary`, `Drama`, `Fantasy`, `Film-Noir`, `Horror`, `Musical`, `Mystery`, `Romance`, `Sci-Fi`, `Thriller`, `War`, `Western`). En la Figura 4.1 se muestra el número de películas por género en ML100K.

La Figura 4.1 muestra que ML100K es un dataset multi-etiqueta. Muchas películas están etiquetadas dentro de tres géneros principales (`Drama`, `Comedy` o `Action`), y solo

unos pocos están etiquetados en géneros como `Fantasy`, `Film Noir` o `Western`. Usando los géneros como nuggets de contenido, se espera que muchas de las listas de recomendadas incluyan géneros comunes como `Drama` y solo unas pocas incluyan `Western` o `Fantasy`. Como las etiquetas están sesgadas hacia algunos géneros específicos las recomendaciones basadas en estos datos también lo estarán. La naturaleza multi-etiqueta del proceso se ilustra en la Tabla 4.2, donde se muestra la distribución del número de géneros en ML100K.

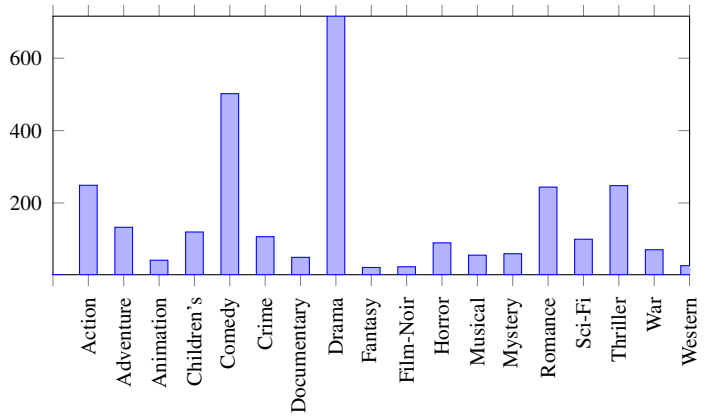


Figura 4.1: Número de películas por género en ML100K.

# géneros	1	2	3	4	5	6
# películas	824	563	212	51	11	3

Tabla 4.2: Distribución de géneros en ML100K. La mayoría de las películas han sido etiquetadas en dos o más géneros por expertos.

Como las películas registran dos o más géneros, se espera que las recomendaciones incluyan diferentes géneros. Sin embargo, usando user-based Collaborative Filtering (user-KNN) sobre ML100K, la mayoría de las listas *top*-5 están concentradas en dos géneros, como se muestra en la Tabla 4.3.

Géneros	`Drama`	`Action`	`Documentary`	`Fantasy`	`Western`
# listas	932 (97.6%)	849 (88.9%)	5 (0.5%)	4 (0.4%)	2 (0.2%)

Tabla 4.3: Distribución de géneros en listas recomendadas por `userkNN` en ML100K. La mayoría de las listas se concentran en dos géneros principales.

La Tabla 4.3 muestra que la mayoría de las listas de productos recomendados incluyen películas de drama o acción. Por otra parte, solo cinco listas incluyen documentales y solo cuatro incluyen películas de fantasía, evidenciando la presencia de un sesgo en

los géneros al usar user-KNN para generar recomendaciones. Sin embargo, si se evalúa diversidad a través de α-NDCG en ML100K, para medir user-KNN en listas *top*-5, se logra una valor de 0.95 en términos de diversidad de géneros, una medición casi perfecta para esta métrica. Como α-NDCG esta definida en términos del número de nuggets en la lista de recomendaciones, se consiguen altos valores aún cuando algunos géneros sean despreciados en todas las listas.

4.4.2. Sesgo en los ratings

El sesgo presente en los ratings, debido a productos populares que suelen ser más valorados por los usuarios, es un fenómeno bien conocido en sistemas de recomendación que limita en términos de novedad la calidad de las recomendaciones. El sesgo por popularidad emerge cuando unos pocos productos populares se llevan la mayoría de los ratings. Como los ratings de los usuarios son usados para determinar la relevancia de los productos, las recomendaciones generadas a través de esta fuente de retroalimentación explícita, se encuentran sesgadas hacia los productos más populares. Se puede ilustrar este fenómeno en ML100K explorando como los ratings de los usuarios tienden a sesgar las recomendaciones en user-KNN. La Tabla 4.4 muestra algunas de las películas que encabezan con gran frecuencia todas las listas de recomendaciones generadas por user-KNN en ML100K. La lista completa de películas y el número de veces en que cada película fue incluida en alguna de las listas *top*-5 generadas por user-KNN se muestra en la Figura 4.2.

Película	# listas	covertura en test set
`Star Wars (1977)`	725	76 %
`The Godfather (1972)`	373	39 %
`Titanic (1997)`	372	39 %
`Fargo (1996)`	357	37 %

Tabla 4.4: Presencia de películas populares en listas recomendadas por `userkNN`.

Como se indica en la Tabla 4.4 y en la Figura 4.2, las películas populares son incluidas en muchas de las listas que se recomiendan a los usuarios. Sin embargo, esta clase de recomendaciones no son diversas en términos de contenido usando R como contexto para medir novedad. Las medidas actuales de diversidad son insensibles frente a esta clase de efectos (notar que α-NDCG puntúa 0.95 en términos de diversidad de género). Este tipo de aspectos se manejan a través de medidas de novedad, la cual se espera entregue su óptimo cuando nuevos productos sean recomendados a los usuarios. No obstante, como las medidas de novedad están basadas en ratings de usuarios, son incapaces de medir el efecto del sesgo por popularidad en términos de géneros.

El sesgo por popularidad está relacionado al sesgo por género. En la Tabla 4.4 se

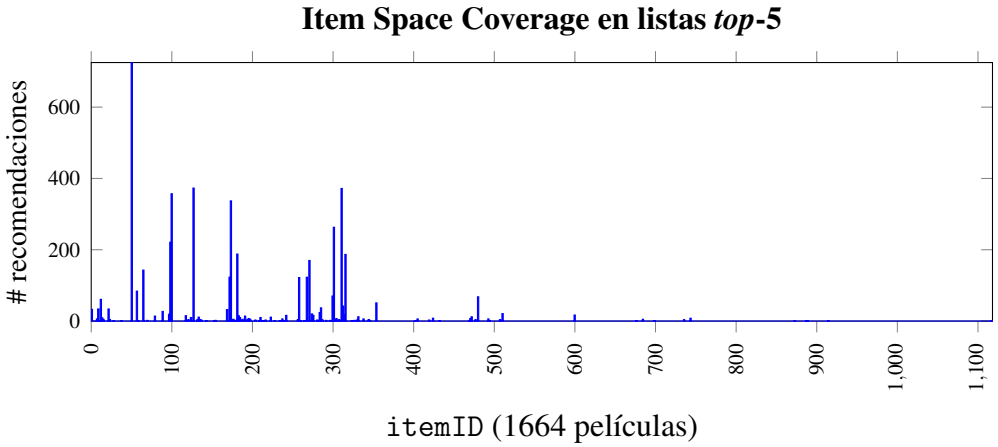


Figura 4.2: Número de veces que cada película fue recomendada por `userkNN`.

muestra que las películas más populares se concentran en algunos géneros (`Drama`, `Action`, o ambos), por lo que los ratings de los usuarios refuerzan el efecto del sesgo por género en las recomendaciones. Como las preferencias de los usuarios están concentradas en pocas películas, y esas películas pertenecen a unos pocos géneros, los sistemas de recomendación basados en filtrado colaborativo se ven afectados en términos de diversidad por el efecto combinado de ambos factores. Se espera que las medidas de diversidad puedan detectar este tipo de efectos presentes en las recomendaciones. Por lo tanto, se proponen métricas capaces de medir este tipo de fenómenos.

4.4.3. Unicidad en los productos

La primera métrica de diversidad que se introducirá tiene el objetivo de medir la unicidad de productos a través de las listas de recomendaciones. Se modelará esto como un factor de la medida α-NDCG, premiando listas con productos que hayan sido recomendados solo en esa lista (únicos entre todas las listas del conjunto de prueba). Sea $V(u_i)$ una función igual a 1 si al menos un producto recomendado a u_i fue recomendado **únicamente** a u_i, en caso contrario 0. Entonces, se define TOT DIV como una medida global de diversidad en el conjunto de prueba, calculada como el producto entre α-NDCG y la fracción de listas que poseen al menos un producto único:

$$\text{TOT DIV} = \alpha\text{-NDCG} \cdot \frac{\sum_{i=1}^{n} V(u_i)}{n}, \tag{4.2}$$

donde n es el número de listas incluidas en el conjunto de prueba. Notar que TOTAL DIVERSITY penaliza α-NDCG cuando las listas tienden a incluir solo productos populares porque la redundancia a lo largo de las listas es penalizada por la función de unicidad. Así, TOTAL DIVERSITY será igual a α-NDCG si y solo si todas las listas

en el conjunto de prueba incluyen al menos un producto único. En otro caso, TOTAL DIVERSITY será la fracción de α-NDCG que corresponde a la proporción de listas con productos únicos sobre el total del número de listas en el conjunto de pruebas.

4.4.4. Nuggets frecuentes en los productos

La segunda métrica de diversidad que se propondrá incluye descuento por nuggets frecuentes. La idea es incluir un factor de descuento en α-NDCG para penalizar la inclusión de nuggets frecuentes, un tipo de enfoque basado en novedad de géneros aplicado a medidas de diversidad. Como contexto para novedad se utilizará I, el conjunto de productos en el sistema. Se usará la función de conteo de nuggets $N(d_i, n_j)$, usada en α-NDCG para definir este factor. El recíproco de un conteo de nuggets para un nugget dado n_j es $\frac{N}{\sum_{i=1}^{N} N(d_i,n_j)}$, donde N es el número de productos en I. Un nugget infrecuente, por ejemplo un nugget que contiene un solo producto, consigue un máximo valor en este factor (N). Por otro lado, el valor mínimo para este factor se consigue cuando el nugget es incluido en todos los productos de I (4.1). Se usará el logaritmo de este factor, de manera similar al IDF usado en recuperación de información. Entonces, se define un factor de descuento exponencial $\exp_1^j$ para un nugget n_j dado:

$$\exp_1^j = \log_N \left[\frac{N}{\sum_{i=1}^{N} N(d_i, n_j)} \right] - 1 \tag{4.3}$$

Un nugget novedoso n_j (e.g. un nugget incluido en solo un producto de I), tendrá $\exp_1^j = 0$. Nuggets frecuentes n_j tendrán valores negativos, con un mínimo alcanzado para un nugget n_j incluido en todos los productos de I, que corresponda a $\exp_1^j = -1$. Como $\exp_1^j$ está en el rango $[-1, 0]$, se usará como un factor de descuento exponencial para $\beta \geq 1$, definiendo la medida de diversidad $\alpha\beta$-NDCG:

$$\alpha\beta\text{-NDCG} = \sum_{j=1}^{m} N(d_k, n_j) \cdot \alpha^{r_j, k-1} \cdot \beta^{\exp_1^j}, \quad 0 \leq \alpha \leq 1, 1 \leq \beta. \tag{4.4}$$

Como para nuggets novedosos ocurre que $\exp_1^j = 0$, $\alpha\beta$-NDCG $= \alpha$-NDCG. Sin embargo, para nuggets frecuentes $\exp_1^j < 0$, $\alpha\beta$-NDCG $= \alpha$-NDCG $\cdot \frac{1}{\beta^{|e_1^j|}}$, una fracción de α-NDCG definida por la frecuencia inversa del nugget en I.

4.4.5. Nuggets frecuentes en los productos de cada usuario

Una tercera métrica de diversidad que se introducirá incluye un descuento por nuggets frecuentes en I_u, los productos vistos/calificados por el usuario u. La idea detrás de esta medida es cambiar el contexto para medir la novedad del nugget en relación al perfil del usuario I_u (productos consumidos por u), midiendo la novedad a nivel del usuario. En este sentido, nuggets frecuentes en I_u serán penalizados por un factor de descuento, y nuggets infrecuentes en I_u producirán recompensas en términos de α-NDCG. Nuggets poco frencuentes en I_u representan novedad en contenido para el usuario (e.g. un género nuevo para u). Un producto altamente recomendado con nuggets infrecuentes representa un producto recomendado que es inesperado en términos del perfil del usuario. Para medir este efecto, se define un factor de descuento exponencial $\exp_2^j$ para un nugget n_j dado:

$$\exp_2^j = \log_N \left[\frac{N_u}{\sum_{i=1}^{N_u} N(d_i, n_j)} \right] - 1. \tag{4.5}$$

donde N_u es el número de productos vistos/calificados por el usuario u (productos en I_u). Notar que los productos considerados en $\exp_2^j$ están incluidos en I_u, una diferencia significativa con relación a $\exp_1^j$, donde los productos estaban en I (todo el universo de productos disponibles en el dataset). Entonces, se define $\alpha\gamma$-NDCG:

$$\alpha\gamma\text{-NDCG} = \sum_{j=1}^{m} N(d_k, n_j) \cdot \alpha^{r_{j,k-1}} \cdot \gamma^{\exp_2^j}, \quad 0 \leq \alpha \leq 1, 1 \leq \gamma. \tag{4.6}$$

Nuggets novedosos en I_u producirán altos valores en $\alpha\gamma$-NDCG, alcanzando un valor máximo cuanto la lista de productos recomendados incluya solo nuggets novedosos para u. En ese caso, $\alpha\gamma$-NDCG $= \alpha$-NDCG. Por otro lado, nuggets frecuentes introducirán descuentos en esta medida, y $\alpha\gamma$-NDCG logrará solo una fracción de α-NDCG definida por la frecuencia inversa de los nuggets frecuentes en I_u.

Finalmente, una cuarta métrica que puede ser usada en análisis de diversidad contiene la combinación de las medidas previas. Se propone $\alpha\beta\gamma$-NDCG, la cual combina diversidad a nivel de I y I_u en una sola medida:

$$\alpha\beta\gamma\text{-NDCG} = \sum_{j=1}^{m} N(d_k, n_j) \cdot \alpha^{r_{j,k-1}} \cdot \beta^{\exp_1^j} \cdot \gamma^{\exp_2^j}, \quad 0 \leq \alpha \leq 1, 1 \leq \beta, 1 \leq \gamma \tag{4.7}$$

En tanto, $\alpha\beta\gamma$-TOT DIV, combina unicidad de productos y diversidad en I y I_u:

$$\alpha\beta\gamma\text{-TOT DIV} = \sum_{j=1}^{m} N(d_k, n_j) \cdot \alpha^{r_j,k-1} \cdot \beta^{\exp_1^j} \cdot \gamma^{\exp_2^j} \cdot \frac{\sum_{i=1}^{n} V(u_i)}{n}, \qquad (4.8)$$

con $0 \leq \alpha \leq 1, 1 \leq \beta, 1 \leq \gamma$.

4.5. Experimentos

4.5.1. Ejemplos Ilustrativos

Considerar el siguiente conjunto de películas con sus respectivos géneros.

v_1:`The Silence of the Lambs(1991)::Crime|Horror|Thriller`
v_2:`Titanic(1997)::Drama|Romance`
v_3:`It's a Wonderful Life(1946)::Drama|Fantasy|Romance`
v_4:`Unforgiven(1992)::Drama|Western`
v_5:`Pulp Fiction(1994)::Comedy|Crime|Drama`
v_6:`The Godfather(1972)::Crime|Drama`
v_7:`Forrest Gump(1994)::Comedy|Drama|Romance|War`
v_8:`Goodfellas(1990)::Crime|Drama`
v_9:`The Shawshank Redemption(1994)::Drama`
v_{10}:`Schindler's List(1993)::Drama|War`

Alta Novedad, Baja Diversidad Inter

Asumir que se tiene tres listas *top*-1 que muestran alta novedad en términos de géneros, incluyendo películas de `Crime`, `Horror` y `Thriller`. Baja diversidad inter (las tres listas recomiendan los mismos géneros) y se enfrenta un escenario de Cold Start (los tres usuarios han calificado solo unas pocas películas).

Notar que los tres géneros etiquetados en v_1 son nuevos para los usuarios u_1, u_2 y u_3 (recomendaciones altamente novedosas en términos de géneros). Las métricas de *content novelty*, para este caso, se muestran abajo (el promedio).

Como la recomendación es novedosa, variantes de α-NDCG consiguen altos valores a pesar del hecho de que las listas tienen baja diversidad inter (las tres listas recomien-

dan la misma película). Esta situación es penalizada por TOT DIV y $\alpha\beta\gamma$–TOT DIV entregando valores bajos para este escenario.

User	Recommendations	I_u
u_1	L_1: $\{v_1\}$, $\langle$CRIME, HORROR, THRILLER$\rangle$	$\{v_2, v_3\}$
u_2	L_2: $\{v_1\}$, $\langle$CRIME, HORROR, THRILLER$\rangle$	$\{v_4, v_7\}$
u_3	L_3: $\{v_1\}$, $\langle$CRIME, HORROR, THRILLER$\rangle$	$\{v_9, v_{10}\}$

α-NDCG	$\alpha\beta$-NDCG	$\alpha\gamma$-NDCG	$\alpha\beta\gamma$-NDCG	TOT DIV	$\alpha\beta\gamma-$ TOT DIV
1	0.886	1	0.886	0.333	0.295

Baja Novedad, Alta Diversidad Inter

Asumir que se tienen tres listas *top*-1 que muestran baja novedad (`Drama` es recomendado en tres listas), alta diversidad inter (las tres listas incluyen una película diferente) y Cold Start (I_u tiene solo dos películas para los tres usuarios).

User	Recommendations	I_u
u_1	L_1: $\{v_9\}$, $\langle$DRAMA$\rangle$	$\{v_1, v_2\}$
u_2	L_2: $\{v_6\}$, $\langle$CRIME, DRAMA$\rangle$	$\{v_1, v_2\}$
u_3	L_3: $\{v_8\}$, $\langle$CRIME, DRAMA$\rangle$	$\{v_1, v_2\}$

Notar que los géneros recomendados no son nuevos para los usuarios (están incluídos en I_u). Las métricas de *content novelty*, para este caso, se muestran abajo (en promedio y a nivel de usuario).

	α-NDCG	$\alpha\beta$-NDCG	$\alpha\gamma$-NDCG	$\alpha\beta\gamma$-NDCG	TOT DIV	$\alpha\beta\gamma$-TOT DIV
u_1	1	0.516	1	0.516	-	-
u_2	1	0.587	1	0.587	-	-
u_3	1	0.587	1	0.587	-	-
avg	1	0.564	1	0.564	1	0.564

Como la recomendación es diversa en términos de géneros inter-listas, TOT DIV alcanza su máximo valor. Sin embargo, las métricas de *content novelty* $\alpha\beta$-NDCG y $\alpha\beta\gamma$-NDCG penalizan esta situación entregando valores bajos para este escenario.

Recomendación experta

Ahora, suponer que se tienen tres usuarios con muchas películas vistas/calificadas (usuarios expertos). Las recomendaciones en esta situación alcanzarán baja novedad. Así, asumir que las tres listas tienen alta diversidad inter (las tres listas incluyen una película diferente).

Notar que las tres listas son diferentes y cada usuario ha visto/calificado todas las películas en I, excepto la película recomendada. Las métricas de *content novelty*, para este caso, se muestran abajo (en promedio y a nivel de usuario).

User	Recommendations	I_u
u_1	L_1: $\{v_9\}$, $\langle$DRAMA$\rangle$	$\{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_{10}\}$
u_2	L_2: $\{v_6\}$, $\langle$CRIME, DRAMA$\rangle$	$\{v_1, v_2, v_3, v_4, v_5, v_7, v_8, v_9, v_{10}\}$
u_3	L_3: $\{v_8\}$, $\langle$CRIME, DRAMA$\rangle$	$\{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_9.v_{10}\}$

	α-NDCG	$\alpha\beta$-NDCG	$\alpha\gamma$-NDCG	$\alpha\beta\gamma$-NDCG	TOT DIV	$\alpha\beta\gamma$-TOT DIV
u_1	1	0.516	0.519	0.268	-	-
u_2	1	0.587	0.613	0.367	-	-
u_3	1	0.587	0.613	0.367	-	-
avg	1	0.564	0.582	0.334	1	0.334

Como las tres listas son diferentes TOT DIV alcanza su valor máximo. Sin embargo, como la película recomendada no es novedosa para los usuarios, las métricas de *content novelty* penalizan esta situación entregando valores bajos. El valor más bajo es alcanzado por $\alpha\beta\gamma$-NDCG, mostrando que es la medida más estricta en términos de novedad.

En resumen, recomendaciones con baja novedad serán penalizadas por $\alpha\beta$-NDCG y $\alpha\gamma$-NDCG, recomendaciones con baja diversidad inter serán penalizadas por TOT DIV. Las medidas combinadas $\alpha\beta\gamma$-NDCG y $\alpha\beta\gamma$-TOT DIV serán útiles en recomendaciones para usuarios expertos, donde la novedad y la diversidad son difíciles de lograr.

4.5.2. Resultados Experimentales

En esta sección, se presentan los resultados de las métricas de novedad y diversidad propuestas comparando seis algoritmos de recomendación diferentes. Se evaluaron las recomendaciones provistas por listas Random (Rnd), listas ordenadas por Popularidad (Pop), Non-Negative Matrix Factorization (MF) [61], probabilistic Latent Semantic Analysis (pLSA) [47], User-based Collaborative Filtering (UB) [71] y Item-based Co-

llaborative Filtering (IB) [74]. Se utilizó RecommenderLab, un framework de sistemas de recomendación implementado en R [42], para llevar a cabo el proceso de validación.

Los resultados para los datasets ML100K, ML1M y MovieTweetings se muestran en las Tablas 4.5, 4.6 y 4.7, respectivamente.

	α-NDCG	$\alpha\beta$-NDCG	$\alpha\gamma$-NDCG	$\alpha\beta\gamma$-NDCG	Tot Div	$\alpha\beta\gamma$-Tot Div
Rnd	0.8221	0.6030	0.7240	0.4408	0.2461	0.1084
Pop	0.7393	0.6012	0.7441	0.4576	0.0038	0.0017
MF	0.8483	0.6098	0.7157	0.4406	0.5962	0.2625
pLSA	0.8604	0.6107	0.7077	0.4357	0.1892	0.0824
UB	0.8399	0.6106	0.7153	0.4407	0.5849	0.2577
IB	0.8314	0.6037	0.7221	0.4408	0.5607	0.2472

Tabla 4.5: Content Novelty Measures en ML100K.

	α-NDCG	$\alpha\beta$-NDCG	$\alpha\gamma$-NDCG	$\alpha\beta\gamma$-NDCG	Tot Div	$\alpha\beta\gamma$-Tot Div
Rnd	0.8008	0.5951	0.7181	0.4316	0.0749	0.03234
Pop	0.7894	0.5779	0.7031	0.4138	0.0012	0.00051
MF	0.8343	0.5999	0.6845	0.4143	0.3002	0.12440
pLSA	0.8577	0.6038	0.6863	0.4178	0.0462	0.01934
UB	0.8365	0.6001	0.6841	0.4141	0.2996	0.12410
IB	0.8379	0.5969	0.7052	0.4254	0.2513	0.10691

Tabla 4.6: Content Novelty Measures en en ML1M.

	α-NDCG	$\alpha\beta$-NDCG	$\alpha\gamma$-NDCG	$\alpha\beta\gamma$-NDCG	Tot Div	$\alpha\beta\gamma$-Tot Div
Rnd	0.8718	0.5840	0.8110	0.4788	0.2714	0.1299
Pop	0.8012	0.5742	0.7680	0.4448	0.0006	0.0002
MF	0.9047	0.5791	0.7368	0.4299	0.6125	0.2633
pLSA	0.9169	0.5697	0.7288	0.4250	0.1982	0.0843
UB	0.9048	0.5791	0.7368	0.4299	0.6128	0.2635
IB	0.8742	0.5812	0.7783	0.4573	0.3723	0.1702

Tabla 4.7: Content Novelty Measures en MovieTweetings.

Las Tablas 4.5, 4.6 y 4.7 muestran que α-NDCG es la métrica más permisiva, entregando los mejores resultados para cada método en los tres datasets. En efecto, recomendaciones Random y por Popularidad logran valores que están muy cerca a los resultados conseguidos por los métodos más complejos y competitivos del Estado del Arte. En particular, listas Random logran casi el mismo desempeño en términos de α-NDCG que UB o IB, con solo 1 punto de diferencia en ML100K, y casi el mismo desempeño que IB en MovieTweetings. En tanto, Popularidad, aunque se acerca al resto, entrega los peores resultados para α-NDCG en todos los datasets.

Las medidas basadas en Novedad de Contenido (*Content Novelty Measures*) son más estrictas, como lo muestran sus resultados. En específico, la medida de desempeño más estricta es $\alpha\beta\gamma$-NDCG y la más permisiva es $\alpha\gamma$-NDCG. TOT DIV es la métrica que exhibe mayor varianza entre todos los métodos evaluados. De acuerdo a TOT DIV, el peor método es Popularidad, como se podría esperar. Nótese que Popularidad recomienda siempre los productos más consumidos por los usuarios. Por lo tanto, este tipo de recomendaciones tienen un sesgo hacia los productos más populares y no considera nunca los productos de la cola de la distribución (*long-tail*). Dos métodos consiguen el mejor desempeño en esta evaluación, MF y UB, con resultados bastante similares en TOT DIV. El mejor resultado se consigue en MovieTweetings, porque ese dataset tiene mayor cantidad de productos para recomendar. Por lo tanto, diversificar, encontrando productos nuevos, es más sencillo debido a la mayor cantidad de opciones disponibles en comparación a MovieLens.

Cuando se combinan diversidad y novedad en contenido en la métrica $\alpha\beta\gamma$-TOT DIV, MF y UB son penalizados, demostrando que un buen desempeño en términos de diversidad (TOT DIV) no implica necesariamente un buen desempeño en Novedad basada en contenido. De hecho, para esta última medida, la diferencia entre UB, MF y IB disminuye a solo 1 punto en ML100K y ML1M. La diferencia en favor de UB y MF aumenta en MovieTweetings. En general, estos resultados muestran que $\alpha\beta\gamma$-TOT DIV consigue mejores resultados en *content novelty*, permitiendo evaluar en una misma medida ambos aspectos de un sistema de recomendación, e.g., cuán diferentes son los productos en una lista dada y cuán diferentes son las listas recomendadas entre ellas.

En la Tabla 4.8 se muestran los resultados conseguidos usando RankSys, el framework propuesto por Vargas para evaluar novedad y diversidad [83]. Se utilizaron en estos experimentos cuatro medidas discutidas en la sección 4.3.

La Tabla 4.8 muestra que las cuatro medidas logran diferentes resultados. EPC muestra alta novedad para recomendaciones producidas en MovieTweetings, con casi todos los métodos alcanzando casi el máximo valor en esta medida. Sorpresivamente, MF logra peores resultados en términos de EPC que Random o Popularidad en MovieTweetings. En tanto, en MovieLens EPC muestra menor novedad en las recomendaciones, donde los mejores resultados son conseguidos por Random y los peores por Popularidad. EFD muestran resultados similares a EPC, introduciendo un factor de escala que ayuda a aumentar la separación entre los resultados obtenidos por los diferentes métodos. cuando se utiliza EPD, los mejores resultados se consiguen con Popularidad, hecho que indica exactamente lo opuesto a EPC y EFD en ML100K y ML1M. Una evaluación más estricta se logra con EILD, alcanzando resultados similares que EPC y EFD, pero con valores de menor desempeño. Como EILD y EPD se basan en distancias de contenido, es esperable que dichas medidas tengan un comportamiento similar a las métricas propuestas de novedad en contenido. Por ende, métodos como MF y UB deberían mostrar mejor desempeño que el resto en diversidad. Sin embargo, los resultados entregados por estas métricas indican algo completamente diferente. Para las medidas

EILD y EPD, los mejores métodos son Random y Popularidad. En particular, en MovieTweetings el mejor resultado en términos de EPD es alcanzado por Popularidad, con un desempeño casi perfecto en términos de novedad para este dataset. Así, en estos datasets, el uso de métricas basadas en funciones de distancias para medir novedad (EPD) o diversidad (EILD), no permite detectar diferencias con métricas basadas en novedad pura como EPC y EFD.

		EPC	EFD	EPD	EILD
ML100K	Rnd	0.9556	6.8384	0.8325	0.7400
	Pop	0.7660	2.5209	0.8436	0.6968
	MF	0.7765	2.2391	0.7634	0.6545
	pLSA	0.8633	3.1743	0.7584	0.6486
	UB	0.7999	2.4903	0.7743	0.6734
	IB	0.8382	2.9080	0.7519	0.6295
ML1M	Rnd	0.9797	7.3859	0.8305	0.7379
	Pop	0.7074	1.7839	0.8415	0.6948
	MF	0.7961	2.4184	0.7615	0.6527
	pLSA	0.8420	2.9390	0.7603	0.6486
	UB	0.7802	2.3057	0.7762	0.6753
	IB	0.9076	4.1094	0.7361	0.6061
MovieTweetings	Rnd	0.9999	13.0805	0.9448	0.8995
	Pop	0.9977	8.7958	0.9883	0.9000
	MF	0.9869	10.8246	0.8852	0.8653
	pLSA	0.9997	12.5692	0.8964	0.8953
	UB	0.9999	13.2147	0.9871	0.8960
	IB	0.9998	13.0070	0.9625	0.8966

Tabla 4.8: User Oriented Metrics del framework RankSys en ML100K, ML1M y MovieTweetings.

Análisis de resultados

Los resultados obtenidos a partir de las métricas de la sección anterior entregan información sobre diferentes aspectos en las recomendaciones de cada método en los datasets estudiados. Para verificar el nivel de dependencia entre cada métrica de evaluación propuesta se efectúa una prueba de los rangos con signo de Wilcoxon (Wilcoxon Signed Rank Test) sobre cada par de medidas en cada dataset. Las Tablas 4.9, 4.10 y 4.11 muestran los resultados (p-values) de cada test de Wilcoxon. Valores pequeños del p-value indican que la hipótesis nula es rechazada.

Las Tablas 4.9, 4.10 y 4.11 muestran que casi todas las comparaciones rechazan la hipótesis nula, resaltando en **negrita** los casos donde la evidencia no permite rechazar la

	$\alpha\beta$-NDCG	$\alpha\gamma$-NDCG	$\alpha\beta\gamma$-NDCG	Tot Div	$\alpha\beta\gamma$-Tot Div	EPC	EFD	EPD	EILD
α-NDCG	0.03	**0.06**	0.03	0.03	0.03	**0.84**	0.03	**0.56**	0.03
$\alpha\beta$-NDCG	-	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03
$\alpha\gamma$-NDCG	-	-	0.03	0.03	0.03	0.03	0.03	0.03	**0.06**
$\alpha\beta\gamma$-NDCG	-	-	-	**0.43**	0.03	0.03	0.03	0.03	0.03
Tot Div	-	-	-	-	0.03	0.03	0.03	0.03	0.03
$\alpha\beta\gamma$-Tot Div	-	-	-	-	-	0.03	0.03	0.03	0.03
EPC	-	-	-	-	-	-	0.03	0.03	0.03
EFD	-	-	-	-	-	-	-	0.03	0.03
EPD	-	-	-	-	-	-	-	-	0.03

Tabla 4.9: Test de Wilcoxon Signed-Rank en ML100K.

	$\alpha\beta$-NDCG	$\alpha\gamma$-NDCG	$\alpha\beta\gamma$-NDCG	Tot Div	$\alpha\beta\gamma$-Tot Div	EPC	EFD	EPD	EILD
α-NDCG	0.03	0.03	0.03	0.03	0.03	**1**	0.03	**0.15**	0.03
$\alpha\beta$-NDCG	-	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03
$\alpha\gamma$-NDCG	-	-	0.03	0.03	0.03	0.03	0.03	0.03	**0.15**
$\alpha\beta\gamma$-NDCG	-	-	-	0.03	0.03	0.03	0.03	0.03	0.03
Tot Div	-	-	-	-	0.03	0.03	0.03	0.03	0.03
$\alpha\beta\gamma$-Tot Div	-	-	-	-	-	0.03	0.03	0.03	0.03
EPC	-	-	-	-	-	-	0.03	**0.21**	0.03
EFD	-	-	-	-	-	-	-	0.03	0.03
EPD	-	-	-	-	-	-	-	-	0.03

Tabla 4.10: Test de Wilcoxon Signed-Rank en ML1M.

	$\alpha\beta$-NDCG	$\alpha\gamma$-NDCG	$\alpha\beta\gamma$-NDCG	Tot Div	$\alpha\beta\gamma$-Tot Div	EPC	EFD	EPD	EILD
α-NDCG	0.03	0.03	0.03	0.03	0.03	0.03	0.03	**0.15**	**0.68**
$\alpha\beta$-NDCG	-	0.03	0.03	**0.15**	0.03	0.03	0.03	0.03	0.03
$\alpha\gamma$-NDCG	-	-	0.03	0.03	0.03	0.03	0.03	0.03	0.03
$\alpha\beta\gamma$-NDCG	-	-	-	0.03	0.03	0.03	0.03	0.03	0.03
Tot Div	-	-	-	-	0.03	0.03	0.03	0.03	0.03
$\alpha\beta\gamma$-Tot Div	-	-	-	-	-	0.03	0.03	0.03	0.03
EPC	-	-	-	-	-	-	0.03	0.03	0.03
EFD	-	-	-	-	-	-	-	0.03	0.03
EPD	-	-	-	-	-	-	-	-	0.03

Tabla 4.11: Test de Wilcoxon Signed-Rank en MovieTweetings.

hipótesis nula, y de acuerdo al test, ambos resultados provienen de la misma población.

En la Tabla 4.9, donde se muestran los resultados del Test de Wilcoxon para ML100K, el test muestra una fuerte dependencia entre los resultados de α-NDCG y $\alpha\gamma$-NDCG, EPC, EPD. Una fuerte dependencia también se observa entre $\alpha\gamma$-NDCG y EILD, y entre $\alpha\beta\gamma$-NDCG y Tot Div.

$\alpha\beta\gamma$-Tot Div y EPC son las únicas medidas que rechazan la hipótesis nula para cualquier comparación en todos los datasets.

En la Tabla 4.10, donde se muestran los resultados del Test de Wilcoxon para ML1M, se aprecia una fuerte dependencia entre α-NDCG y EPC, EFD. También se observa una fuerte dependencia entre $\alpha\gamma$-NDCG y EILD, y entre EPC y EPD. En esta evaluación, las propuestas $\alpha\beta\gamma$-NDCG, Tot Div y $\alpha\beta\gamma$-Tot Div, rechazan todas las hipótesis nulas. De RankSys, EFD rechaza la hipótesis nula en todas las comparaciones.

En MovieTweetings, casi todas las medidas comparadas producen resultados diferentes. Como la Tabla 4.11 lo indica, solo fueron encontradas dependencia de pares entre α-NDCG y EPD, EILD, y entre $\alpha\beta$-NDCG y Tot Div. $\alpha\gamma$-NDCG, $\alpha\beta\gamma$-NDCG y $\alpha\beta\gamma$-Tot Div rechazaron la hipótesis nula en todas las comparaciones. De RankSys, EPC y EFD rechazaron la hipótesis nula en todas las comparaciones.

Los resultados en las Tablas 4.9, 4.10 y 4.11 muestran que el número de dependencias entre medidas de desempeño decrece cuando la complejidad del dataset aumenta. En ML100K, el dataset más pequeño considerado en los experimentos, que presenta la mayor densidad en términos de ratings, solo dos medidas rechazan el test en todas las

comparaciones, indicando que la variabilidad entre las distintas métricas de desempeño es baja. En cambio, en MovieTweetings, el dataset más grande considerado en los experimentos, que presenta la menor densidad de datos, cinco medidas rechazan el test en todas las comparaciones, indicando alta variabilidad entre las métricas de novedad y diversidad. Cuando se contrastan los resultados entre todos los datasets, $\alpha\beta\gamma$-TOT DIV y EFD, son las únicas medidas que rechazan la hipótesis nula en todas las comparaciones, demostrando consistentemente que esas medidas producen resultados que son diferentes a los obtenidos por cualquier otra medida.

4.6. Conclusiones

En este capítulo se propusieron un conjunto de nuevas métricas para evaluar el desempeño de los sistemas de recomendación, orientadas a novedad y diversidad de listas *top-N*. Las medidas propuestas están basadas en un nuevo enfoque para evaluación denominado *content novelty* o novedad en el contenido de las recomendaciones, que intenta medir cuán novedoso es el contenido de los productos presentados en una lista. Para medir la novedad en el contenido se utilizan nuggets (descriptores de contenido como géneros en películas o estilos musicales en canciones), un enfoque previamente estudiado en recuperación de información [25].

Se propusieron tres variantes del conocido α-NDCG. En adición, se presentó una medida de diversidad general entre listas (inter-list) llamada Total Diversity, que considera el conjunto de listas recomendadas por un sistema. Se compararon las medidas propuestas con α-NDCG y con respecto a cuatro funciones propuestas en RankSys [83], un framework para evaluación de diversidad y novedad en sistemas de recomendación. Los experimentos efectuados en tres datasets diferentes, usando seis métodos diferentes de recomendación, demostraron que las medidas propuestas producen resultados consistentes y fácilmente interpretables.

Cuando se evaluó la dependencia entre las medidas propuestas, la métrica híbrida $\alpha\beta\gamma$-TOT DIV, la cual combina en una sola función todas las variantes propuestas para α-NDCG e incluye la diversidad general inter listas, demostró entregar resultados consistentemente diferentes al resto para cualquier escenario de evaluación. En adición, los resultados demostraron que EFD, una medida propuesta por RankSys, es también independiente a cualquier otra. Los resultados derivados de utilizar EFD y $\alpha\beta\gamma$-TOT DIV, son bastante diferentes. EFD indica que el mejor método en términos de novedad es Random, una conclusión que señala que la aleatorización ayuda a maximizar la novedad, algo esperable. Cuando se utiliza $\alpha\beta\gamma$-TOT DIV, el mejor método es MF, demostrando que este método ayuda en la diversificación de géneros y en la reducción del efecto producido por el sesgo de popularidad presente en los ratings (distribución *long-tail*). En consecuencia, resultados derivados de $\alpha\beta\gamma$-TOT DIV no son solo diferentes a EFD sino que complementarios entre sí. El enfoque propuesto es capaz de resaltar

la novedad en términos del contenido, y no solo en términos de las ocurrencias de los productos en la matriz de ratings.

En este capítulo se demostró que la novedad en términos de ocurrencias de los productos es solo una visión parcial del concepto de novedad. Se argumenta que incluir diferentes aspectos de novedad, como la propuesta de *Content Novelty*, ayuda a mejorar la evaluación de un sistema de recomendación, de acuerdo a métodos capaces de realizar dicha labor.

4.6 CONCLUSIONES

Capítulo 5

Diversificación en sistemas de recomendación

5.1. Introducción

La diversificación se ha transformado en los últimos años en un tópico fundamental de discusión en sistemas de recomendación como resultado de un gran número de publicaciones involucradas en este tema. La importancia de la diversificación en sistemas de recomendación recae en dos factores primordiales: aumentar la satisfacción de los usuarios y mitigar el efecto del sobre-ajuste (over-fitting).

Este capítulo tiene el objetivo de concluir la integración de las métricas de diversidad propuestas en el Capítulo 4, estudiando el impacto de la diversificación en las recomendaciones *top-N*.

5.1.1. Satisfacción del usuario

En algunos casos, recomendar los productos con el mayor rating estimado (maximizando la precisión del sistema) puede no ser tan útil para los usuarios. Generar una lista de recomendaciones enfocándose únicamente en la precisión puede entregar productos muy similares entre sí. Como los sistemas de recomendación surgen y son utilizados en la industria deben considerar la satisfacción del usuario como medida importante de calidad, es por ello que la inclusión de novedad y diversidad en las soluciones resulta primordial en escenarios reales donde los RS son utilizados (comercio electrónico, streaming de series y películas, etcétera).

5.1.2. Sobre-ajuste (Over-fitting)

Una vez que el sistema está capacitado para generar recomendaciones de manera consistente para cada usuario, un nuevo problema surge cuando el sistema comienza a recomendar productos relacionados a un determinado espectro de interés, que obviamente satisface a un selecto grupo de usuarios. Este problema puede ocurrir cuando un usuario entrega feedback de manera explícita únicamente sobre el contenido que a él más le agrada o consume. Esto lleva a la creación de un modelo demasiado específico que conoce perfectamente los intereses de cada usuario, y por lo tanto es incapaz de detectar cualquier otro tipo de producto interesante. Esto es debido a que el usuario no mostró en el pasado ningún tipo de interés en productos de otro estilo. Este fenómeno se conoce como sobre-ajuste (over-fitting) o sobre-especialización.

5.2. Impacto de la diversificación en la recomendación

Varios grupos de investigación se han enfocado en estudiar el efecto de la diversificación en la calidad de las recomendaciones. Para llevar a cabo la evaluación ellos han utilizado una combinación entre medidas propias de los RS (como F-measure, MAE, NMAE [73]) y algunas propuestas no tradicionales que tienden a incluir la diversidad en la evaluación. Todo ello para determinar cuánto es el real impacto de la diversificación en el desempeño general de un sistema recomendador.

Adomavicius y Kwon [3] evaluaron algoritmos de recomendación basados en productos (IBCF) y determinaron que son capaces de ofrecer recomendaciones diversas, manteniendo ciertos niveles de precisión comparables. Hurley y Zhang [49] modelaron el trade-off entre diversidad y precisión como un problema de optimización binario. Ge *et al*. [36] consideraron un ángulo diferente y realizaron una serie de experimentos para determinar el impacto de ubicar productos muy diversos en una lista de recomendaciones. Tintarev *et al*. [80] realizaron un interesante estudio en el cual no se enfocaron en el impacto de la diversificación en la calidad de la recomendación, sino que en el impacto de la diversificación en usuarios con diferentes personalidades y gustos. Ellos encontraron que a los usuarios que les gustaban las nuevas experiencias preferían mayor diversidad en sus recomendaciones. Aytekin y Karakaya [8] fueron un paso más allá y ofrecieron directo control al usuario sobre la diversidad en la recomendación, para medir la satisfacción. Esto fue explorado aún más, cuando Ekstrand *et al*. [33] efectuaron una evaluación real de usuarios, en la cual los usuarios entregaban feedback no solo sobre la diversidad en la lista de recomendación, sino también sobre la novedad, precisión, satisfacción y el nivel de personalización. Javari y Jalili [51] diseñaron un experimento en el cual ellos usaban un sistema recomendador híbrido y podían controlar el trade-off entre diversidad y precisión. Ribeiro *et al*. [72] continuó esta tendencia modelando el recomendador como un problema multi-objetivo que permitía combinar varios algoritmos

de recomendación, tratando de maximizar la precisión y la diversidad.

La Tabla 5.1 resume los resultados de los papers más importantes presentados en esta sección, con respecto al impacto que produce la diversificación en la calidad del proceso de recomendación.

Ref	Algoritmo de diversificación	Impacto y forma (métricas) de evaluación utilizadas	Consecuencias directas observadas
[3]	Recomendación Item-Based: item popularity, item average rating, item likeability,	Aumento de los productos *long-tail* en las listas de recomendación.	$-5\,\%$ de precisión. $+58\,\%$ productos *long-tail* recomendados.
[49]	Función objetivo parametrizando el trade-off entre diversidad y precisión usando Greedy.	Aumento de la diversidad midiendo el contraste entre prec/rec y novedad en las recomendaciones.	No se indican.
[36]	Guía para organizar los productos recomendados en listas *top-N*.	Satisfacción del usuario y diversidad percibida (feedback).	Usuarios notan la diversidad cuando se entrega en bloques.
[80]	No se propone un algoritmo. El estudio muestra como la experiencia puede incidir en la diversidad.	Diversidad medida por autor y género en forma binaria (0/1) y por temas en 3 niveles (0/0.3/1).	La personalidad de los usuarios tiene relación con la diversidad.
[8]	ClusDiv: clustering sobre productos. Las listas se generan seleccionado productos de diferentes clusters.	Aumento de la diversidad (z-diversity) con pequeña disminución en el recall.	Ganancia considerable en complejidad computacional.
[33]	No se propone un algoritmo. Experimento mide la diversidad y novedad de distintos algoritmos de recomendación.	RMSE para precisión, mean popularity rank para novedad, ILD para diversidad.	La diversidad impacta positivamente en la satisfacción del usuario.
[51]	SM - probabilistic specification maximizer model	Aumento de la diversidad midiendo diversidad inter-listas.	Supera a modelos clásicos de Markov en términos de diversidad.
[72]	Pareto-efficient multi-objective ranking	Aumento de la diversidad midiendo con el modelo basado en distancias de Vargas [84].	La propuesta permite enfocarse en un objetivo o maximizar los tres.

Tabla 5.1: El impacto de la diversificación en la calidad de la recomendación.

5.3. Algoritmos de diversificación

Varios autores han realizado investigaciones enfocadas en desarrollar nuevos algoritmos de diversificación y evaluarlos utilizando alguna de las medidas descritas en la Tabla 5.1. Ziegler *et al*. [92] fueron los primeros en usar diversificación de tópicos para aumentar la diversidad de las recomendaciones a costa de disminuir la precisión del sistema. Slaney y White [77], enfocados en mejorar la diversidad del recomendador, generaron listas de música proyectando cada canción a un espacio de características multidimensional creado a partir de las características musicales de cada canción en el dataset, extraídas utilizando una descomposición en valores singulares (SVD). Adomavicius y Kwon [3] presentaron una serie de técnicas para recomendar productos, aumentando sustancialmente la diversidad, sin perder demasiada precisión. Ellos también propusieron un enfoque basado en la teoría de grafos [2] para aumentar la diversidad basándose en algoritmos de flujo máximo. Usando una filosofía similar, Premchaiswadi *et al*. [69] propuso un nuevo método híbrido llamado Total Diversity Effect Ranking (TDE) que mejora la diversidad general de la recomendación considerando el efecto de la diversidad de cada producto en la lista de recomendación final.

Choi y Han [24] se enfocaron en consultas web, implementando un algoritmo que calculaba correlaciones de categorías con el objetivo de encontrar el usuario con resultados de búsqueda más diversos. De manera similar, Abbassi *et al*. [1] incrementaron la diversidad de documentos recuperados en sitios web, evitando mostrar muchos documentos de la misma categoría. Ellos lograron esto usando un algoritmo de restricción matroide. Bridge y Kelly [19] aumentaron la diversidad de la recomendación usando solo datos colaborativos, que obtuvieron usando Distancia Hamming para comparar productos. Lathia *et al*. [62] estudiaron el aspecto temporal de la diversidad en CF. Ellos analizaron como un sistema de recomendación colaborativo cambia durante el tiempo y el impacto que esos cambios producen sobre la diversidad de las recomendaciones resultantes. Mourao *et al*. [67] trataron con un problema similar al introducir el Oblivion problem, que intenta explotar productos que fueron relevantes para el usuario en el pasado, pero perdieron esa relevancia en el tiempo por distintos factores. Boim *et al*. [15] agregaron diversidad en los productos comparando todos los ratings dados a los productos por distintos usuarios. Usando este enfoque ellos crearon clusters de productos y generaron listas de recomendación con alta diversidad. Vaishnavi *et al*. [82] manejaron el problema de la diversidad en marketing electrónico proponiendo un enfoque basado en LCM versión 2 y I-Tree.

Basille *et al*. [10] propusieron un enfoque que combinaba varios algoritmos existentes con el fin de lograr alta diversidad usando una gran variedad de factores semánticos disponibles. Ho *et al*. [46] usaron la diversificación para manejar el problema *long-tail* en RS (productos que tienen pocos ratings y, por ende, rara vez se recomiendan a los usuarios). Ellos descubrieron que eran capaces de recomendar productos relevantes y mejorar la experiencia de los usuarios. Lee y Lee [63] presentaron un algoritmo de re-

comendación basado en la teoría de grafos que usaba solo los productos con ratings positivos de los usuarios para crear un grafo no-dirigido y usaban la entropía para encontrar recomendaciones novedosas y relevantes.

Ren *et al*. [70] usaron un enfoque interdisciplinario y propusieron un algoritmo de diversificación basado en economía (índice de Giny) y física (proceso de conducción del calor) para mejorar la novedad y diversidad de las recomendaciones. Bedi *et al*. [11] desarrollaron un enfoque basado en clustering y utilizaron la métrica Semi-Partial R-Squared (SPRS) para introducir diversidad en recomendaciones de noticias. Además, ofrecieron explicaciones para recomendaciones inesperadas con el objetivo de aumentar la calidad en la experiencia de los usuarios al usar el sistema. Di Noia *et al*. [30] dieron vuelta el problema de la diversidad modelando la tendencia de los usuarios a seleccionar productos diversos y luego utilizaron ese modelo para volver a ranquear los productos en listas de recomendación *top-N*.

El pseudocódigo de los algoritmos más relevantes descritos en esta sección aparecen en la Tabla 5.2. Se incluye el algoritmo propuesto y presentado en la Parte I, Capítulo 3, Sección 3.3.2, `userCL`.

Se puede apreciar que todos los algoritmos que crean una lista de recomendaciones diversa funcionan re-ordenando las recomendaciones después de haber sido generadas usando alguno de los enfoques existentes. Esto significa que tales algoritmos asumen que las recomendaciones ya son diversas y solamente necesitan ser ordenadas correctamente para lograr la máxima diversidad posible. La mayoría de esos algoritmos son colaborativos [19, 15, 46, 63] o basados en contenido [92, 69]. Notables excepciones [24, 1] intentan diversificar productos durante el proceso de recomendación. El resto de los algoritmos ([77, 62]) no diversifican las recomendaciones sino más bien se enfocan en medir la diversidad general de cada lista de productos entregada sin cambiar ningún producto. La Tabla 5.3 entrega una comparación de las ventajas y desventajas de dichos algoritmos, incluyendo la propuesta de esta Tesis, el método de recomendación `userCL`.

Ref	Pseudocódigo de los algoritmos
[92]	1. Generar predicciones (al menos listas de recomendación *top-N* = 5). 2. Para cada producto en la posición $N+1$ calcular ILS (diversidad) si el producto fue parte de la lista *top-N*. 3. Ordenar los productos restantes en reversa (de acuerdo al ranking ILS) para obtener su ranking de disimilaridad. 4. Calcular un nuevo ranking para cada producto según $r = a*P+b*P_d$, donde P es el ranking original, P_d es el ranking de disimilaridad y a,b constantes en el rango $[0,1]$. 5. Seleccionar los *top-N* productos de acuerdo al nuevo ranking calculado.
[77]	1. Analizar y extraer características relevantes de los archivos de audio. 2. Realizar una SVD seguido de multi-class LDA para transformar todas las características en un espacio SVD de 2 dimensiones. 3. Ajustar un modelo de probabilidad Gaussiana en cada una de las listas de audios (una canción sería un punto en el espacio SVD). 4. Calcular la diversidad para cada lista de audio estimando el volumen del modelo ajustado (elipsoide).
[69]	1. Calcular ratings predichos para el usuario actual usando la suma promedio ponderada. 2. Generar una lista de recomendaciones *top-N* + *S* (*N* entre 3 y 10; *S* entre 1 y 10). 3. Calcular el TDE de cada producto como la suma de las distancias a todos los otros productos ($N+S-1$) en la lista. 4. Remover los *S* productos con el TDE más bajo y así generar las recomendaciones *top-N* para el usuario actual.
[24]	1. Recolectar información del género de cada película en el dataset. 2. Calcular la correlación de género para todas las películas contando el número de ocurrencias de cada posible par de géneros, y normalizar los valores. 3. Generar recomendaciones estimando ratings según $\sum R_M * r_{ij}$, con R_M los ratings promedio de las películas y r_{ij} las correlaciones de género para todos los géneros que le gustan al usuario i, y que son atribuibles al producto j. 4. Seleccionar los *top-N* productos con el rating estimado más alto y presentárselos al usuario.

[1]
1. Representar el espacio de documentos (D) y tópicos (T) como un grafo bipartito, donde el peso de las aristas equivale a la relevancia del documento D sobre el tópico T.
2. Ordenar los documentos relevantes (aquellos considerados para recomendación) de acuerdo a su cobertura ponderada (suma de los pesos para todos los tópicos relevantes).
3. Para cada documento: (i) Si el tamaño de la lista de recomendación es menor que el deseado, agregar el documento a la lista. (ii) Si reemplazar cualquiera de los documentos de la lista con el actual aumenta la diversidad general, efectuar el reemplazo.

[19]
1. Generar recomendaciones usando una lista de productos ranqueados de acuerdo a los ratings promedio de los vecinos más cercanos.
2. Ofrecer esos productos al usuario y chequear si quiere más recomendaciones.
3. Si es así, pedirle al usuario que seleccione el producto que considera más relevante.
4. Crear una lista de posibles recomendaciones usando un algoritmo Greedy, seleccionar los mejores y agregarlos a la lista de recomendación final. Los mejores se determinan calculando la calidad de cada producto como combinación del rating estimado y el ILD.
5. Presentar la lista de recomendaciones al usuario.

[62]
1. Generar recomendaciones usando MF, kNN o popularidad.
2. Ver si el usuario califica/selecciona alguno de los productos recomendados.
3. Generar una nueva recomendación usando la información actualizada (todos los nuevos ratings entregados por el usuario).
4. Calcular la diversidad de la nueva lista de recomendaciones, comparándola con la anterior. Nota: Este enfoque mide como la diversidad de las recomendaciones a los usuarios cambia con el tiempo, no mide la diversidad de los productos en la lista de recomendación.

[15]
1. Crear una lista de recomendaciones para un usuario dado usando alguno de los recomendadores basados en CF existentes.
2. Agrupar esas recomendaciones en medoids basados en prioridad, comparando los ratings dados por los usuarios a esos productos.
3. Determinar como representante del cluster al producto con mayor rating estimado, no el con menor distancia a los otros miembros del cluster.
4. Construir un árbol de cobertura usando los representantes de cada cluster.
5. Seleccionar un nivel del árbol que contenga k productos.
6. Presentar esos productos al usuario como lista de recomendación.

[46]	1. Crear recomendaciones para todos los usuarios usando algún enfoque CF. 2. Para cada producto que fue recomendado al menos una vez, calcular el puntaje 5*D*: Precisión, Balance, Cobertura, Calidad y Cantidad de productos *long-tail*. 3. Para cada usuario crear una lista de posibles recomendaciones y: (i) ordenar la lista según rating estimado para asignar un rank r a cada producto. (ii) Crear un ranking combinado $r_n = r * r_{5D}$ y ordenar los productos de acuerdo a este puntaje. 4. Presentar los top N productos al usuario.
[63]	1. Crear una matrix de adyacencia $M \times M$ (producto/producto), con $x_{mn} =$ número de veces que los productos m y n fueron ambos calificados con un rating superior al promedio. 2. Para cada usuario: remover todas las filas (productos) que no han sido aún calificadas por el usuario (productos no vistos/consumidos). 3. Para cada nodo (producto) restante: calcular la entropía de Shannon. 4. Ordenar los nodos de acuerdo a sus pesos (número de ocurrencias en la matriz). 5. Remover los productos con valor de entropía menor al umbral.
`userCL` [3.3.2]	1. Agrupar cada usuario en un cluster de acuerdo a algún algoritmo de clustering basado en prototipos. 2. Calcular las distancias para un usuario dado con respecto al resto según una función de proximidad propuesta (Ecuación 3.2). 3. Formar un vecindario de usuarios, privilegiando optimizar la precisión o la diversidad en las recomendaciones: 3.1 Precisión: Incluir a los usuarios más cercanos en distancia, ubicados en cualquier cluster. 3.2 Diversidad: Incluir a los usuarios más cercanos en distancia, pero que estén ubicados fuera de las fronteras del cluster. 4. Generar una lista de productos *top-N*, usando los ratings de los usuarios en el vecindario.

Tabla 5.2: Pseudocódigo de los algoritmos de diversificación más relevantes.

Ref	Ventajas	Desventajas
[92]	Flexible. Puede ser utilizado en cualquier sistema capaz de trabajar con funciones de distancia (similaridad) entre objetos.	La diversificación es aplicada después de generar una recomendación, significa que si los productos en la lista no son diversos, no existirá un aumento notable en la diversidad final.
[77]	La definición de la función de diversidad está muy bien detallada y explicada. Cada producto (canción) debe ser procesado solo una vez.	No es generalizable: específico para el dominio musical y dependiente de los géneros definidos.
[69]	Flexible. Puede ser utilizado en cualquier sistema capaz de trabajar con funciones de distancia (similaridad) entre objetos.	Como la mayoría de los sistemas orientados a diversidad depende de la definición de similaridad/distancia entre dos productos.
[24]	No requiere mucho cómputo, solo necesita la descripción del género de cada producto.	Necesita recalcular valores para cada producto agregado, solo funciona si los productos incluyen descripción en meta-data, para funcionar requiere información explícita del usuario.
[1]	Implementación simple, representación gráfica.	Requiere tópicos, e.g., meta-data del contenido.
[19]	Flexible. Puede ser utilizado en cualquier sistema capaz de trabajar con funciones de distancia (similaridad) entre objetos.	Depende de la definición de similaridad/distancia entre dos productos, solo considera diversidad después de completar el proceso de recomendación.
[62]	No requiere conocer las recomendaciones para aumentar la diversidad. Utiliza múltiples RS para introducir diversidad.	Necesita paralelización – requiere varios RS trabajando al mismo tiempo.
[15]	Puede conectar cualquier algoritmo CF existente, diversifica la lista construida.	No usa información semántica de los productos.

[46]	Algoritmo genérico que puede ser aplicado a cualquier sistema. Funciona como un modulo adicional que se introduce entre el cálculo de los ratings recomendados y la generación de listas *top-N*.	Funciona cuando es posible generar una gran cantidad de recomendaciones, requiere muchos ratings para lograr un ranking 5*D* adecuado.
[63]	Representación gráfica, relativa simpleza en el algoritmo, innovación en el cálculo de la diversidad.	Vulnerable al Cold-Start – productos sin ratings no serán nunca recomendados.
`userCL` [3.3.2]	**Flexible. Diversifica antes de generar las recomendaciones y permite controlar el *trade-off* entre diversidad y precisión. Innovación en la función de distancia. Menor tiempo online de recomendación.**	**La calidad de las recomendaciones depende del algoritmo de clustering y de los clusters generados (suele haber desbalance en alta dimensionalidad).**

Tabla 5.3: Ventajas y desventajas de los algoritmos de diversificación más relevantes.

5.4. Evaluando diversidad en recomendaciones *Top-N*

En esta sección se evaluará diversidad en recomendaciones *top-N* utilizando las métricas de *content novelty* definidas en el Capítulo 4. En contraste con el proceso de evaluación realizado en el Capítulo 3, donde se midió precisión en las recomendaciones, al medir diversidad es necesario contar con datasets donde los productos tengan asociados tópicos o *nuggets* (géneros en el caso de las películas). Al ya conocido MovieLens[1], se agrega en esta oportunidad MovieTweetings[2]. Este último, sigue la línea de MovieLens, recolectando ratings reales a partir de Tweets bien estructurados (no incluye ruido porque la información explícita fue extraída de la forma: ``I rated The Matrix 9/10 #IMDb''). Además, tiene la ventaja de ser considerablemente más grande (en términos de usuarios, productos y ratings) que MovieLens. Al igual que este último, MovieTweetings cuenta con un número no despreciable de publicaciones en la literatura de sistemas recomendadores que han utilizado sus datos para investigación, aunque en menor medida debido a su corta existencia (fue presentado en la conferencia ACM RecSys 2013[3]).

[1] http://grouplens.org/datasets/movielens/

[2] http://www.recsyswiki.com/wiki/Movietweetings

[3] http://crowdrec2013.noahlab.com.hk/

La Tabla 5.4 muestra un resumen de las estadísticas de los datasets usados para evaluación de diversidad.

	MovieLens	MovieTweetings
Usuarios (N)	943	49,572
Items (M)	1664	28,377
Ratings (*nnz*s of $\mathscr{R}$)	99,392	618,051
min $\lvert\mathscr{R}_u\rvert$[a]	20	15
Densidad[b]	6.33 %	0.04 %

[a] Mínimo número de ratings por usuario.
[b] Ratings (*nnz*s of $\mathscr{R}$) sobre el producto $M \times N$.

Tabla 5.4: Estadísticas de los datasets usados para medir diversidad.

La Tabla 5.5 entrega el desempeño en términos de *content novelty measures* del método de recomendación propuesto `userCL`, y lo compara a `userkNN`, en los datasets MovieLens y MovieTweetings.

		α-NDCG	$\alpha\beta$-NDCG	$\alpha\gamma$-NDCG	$\alpha\beta\gamma$-NDCG	Tot Div	$\alpha\beta\gamma$-Tot Div
MovieLens	`userkNN`	0.8406	0.6105	0.7130	0.4392	0.5651	0.2482
	`userCL[1]`	0.9221	0.6157	0.7225	0.4473	0.0116	0.0052
	`userCL[2]`	0.9190	0.6156	0.7220	0.4470	0.0115	0.0052
	`userCL[3]`	0.9143	0.6171	0.7264	0.4509	0.0105	0.0048
Tweetings	`userkNN`	0.9047	0.5791	0.7368	0.4299	0.6128	0.2635
	`userCL[1]`	0.9364	0.5802	0.7509	0.4390	0.0023	0.0010
	`userCL[2]`	0.9529	0.5803	0.7415	0.4326	0.0024	0.0010
	`userCL[3]`	0.9601	0.5807	0.7485	0.4372	0.0046	0.0020

Tabla 5.5: Comparación entre `userCL` y `userkNN` usando Content Novelty Measures.

La Tabla 5.5 demuestra que `userCL` permite diversificar las recomendaciones, superando a `userkNN` en términos de α-NDCG, $\alpha\beta$-NDCG, $\alpha\gamma$-NDCG y $\alpha\beta\gamma$-NDCG. Sin embargo, a pesar de que `userCL` alcanza alta diversidad entre los elementos de cada lista (intra-list), expresado por α-NDCG, muestra un pobre desempeño en términos de Tot Div (diversidad inter-listas). Por lo tanto, las recomendaciones ofrecen productos diversos, pero todas las listas *top-N* recomiendan casi siempre los mismos productos.

La razón es que este método fue diseñado y propuesto con el objetivo de optimizar las métricas de precisión, como fue ampliamente discutido en el Capítulo 3. El método no tiene por objetivo diversificar globalmente, sino local. Al estar basado en UBCF, va a tender a formar vecindarios siempre con usuarios similares al usuario activo (favoreciendo la precisión). A pesar de esto, y como se detalla en la Tabla 5.3, una de las ventajas de `userCL` es su gran flexibilidad en el diseño, que permite parametrizar el número de usuarios intra-cluster usados en el proceso de predicción. En el Capítulo 3, esto no fue necesario debido a que se requería que el algoritmo encontrara y utilizara a los usuarios más similares para lograr la mayor precisión posible. Por el contrario, en escenarios donde se requiere diversificar las soluciones, es útil incluir este factor como parámetro del algoritmo, y ajustarlo según las necesidades del usuario.

5.4.1. Controlando la diversidad

Uno de los desafíos, y al mismo tiempo objetivo de esta Tesis, era poder afrontar el *trade-off* entre precisión y diversidad en sistemas de recomendación. Entendiendo por precisión, todas las métricas destinadas a medir exactitud en los métodos como Precision, Recall, NDCG. En tanto, diversidad se asocia con novedad y serendipity[4].

Un aspecto interesante y diferenciador del método propuesto `userCL` tiene relación con la capacidad de poder controlar el *trade-off* entre precisión y diversidad. Esto se consigue parametrizando el número de vecinos (usuarios en el vecindario) que pertenecen al mismo cluster del usuario objetivo. El proceso de agrupación, de haber sido realizado correctamente, produce grupos (clusters) de usuarios con preferencias hacia productos similares. Por ende, al utilizar una gran cantidad de usuarios del mismo cluster (`Inside Users`) para formar el vecindario, los productos recomendados compartirán características similares. Por el contrario, limitar la cantidad de `inside users`, utilizando gran cantidad de `outside users` (usuarios de otros clusters), favorecerá el proceso de diversificación del algoritmo.

La Tabla 5.6 muestra el aumento en TOT DIV a medida que disminuye el porcentaje de `inside users`. En tanto, el resto de los indicadores de diversidad (α, β, γ, en todas sus variantes), presentan solo un leve decaimiento en su desempeño al incluir mayor número de `outside users`.

Los resultados obtenidos en la Tabla 5.6 son interesantes en el sentido de que los algoritmos de recomendación tradicionales no fueron diseñados para garantizar diversidad, menos aún, controlar el trade-off que se produce en relación a la precisión. Por otro lado, el método `userCL` demostró ser capaz de garantizar alta novedad (según $\alpha\beta-$ y $\alpha\gamma-$ NDCG) y al mismo tiempo poder incorporar diversidad según las necesidades del

[4]Descubrimiento o hallazgo afortunado e inesperado, pero relevante, que se produce cuando se estaba buscando otra cosa distinta.

usuario. Esto último, es posible gracias al parámetro `inside users`.

inside users	α-NDCG	$\alpha\beta$-NDCG	$\alpha\gamma$-NDCG	$\alpha\beta\gamma$-NDCG	Tot Div	$\alpha\beta\gamma$-Tot Div
–	0.9321	0.6161	0.7237	0.4483	0.0117	0.0052
1	0.9254	0.6154	0.7215	0.4466	0.0145	0.0065
0.5	0.9099	0.6163	0.7233	0.4483	0.0153	0.0069
0.1	0.8820	0.6140	0.7166	0.4432	0.0657	0.0291
0.02	0.8588	0.6106	0.7124	0.4387	0.1907	0.0837
0.01	0.8493	0.6096	0.7142	0.4393	0.2322	0.1020

Tabla 5.6: Controlando la diversidad a través de `inside users` en `userCL`.

5.5. Conclusiones

Casi todo grupo de investigación en RS está de acuerdo en que la diversidad es importante y debe ser evaluada. Sin embargo, aún no existe consenso de qué manera y con qué métrica hacerlo. Mientras algunas métricas aparecen más a menudo, la comunidad todavía no ha adoptado una (o varias) de ellas como medida de diversidad predilecta. Una vez que esto se logre, será más sencillo cuantificar y comparar resultados de diferentes grupos de investigación (como sucede al evaluar precisión).

Un notable resultado obtenido es que un aumento de la diversidad, no implica necesariamente una disminución en la precisión. Un enfoque bien aplicado podría traer beneficios en ambos factores. Es más, si una medida que evalúa precisión tiende a disminuir cuando aumenta la diversidad es porque no se está midiendo correctamente la precisión del sistema. Tradicionalmente, la precisión se ha evaluado por medio de métricas del área de recuperación de información (IR). Este tipo de métricas se basan en entender y medir la relevancia de los productos en las recomendaciones. Sin embargo, una medida de precisión definida correctamente debería incluir en su evaluación la diversidad como un factor complementario y no opuesto. Las métricas basadas en content novelty, propuestas en el Capítulo 4, están enfocadas en medir la diversidad del sistema, basándose en una medida de calidad de ranking como NDCG. Por lo tanto, permiten evaluar la precisión en las recomendaciones, incluyendo la diversidad como factor de relevancia.

Por otro lado, aunque todavía es necesario seguir explorando el impacto de los modelos de clustering en diversificación de listas, la estructura de grupos y la función de proximidad intra/inter cluster, pueden aportar propiedades valiosas en términos de con-

trolar la diversidad en una lista *top-N*. La creación de clusters con características similares permite manejar, hasta cierto punto, el *trade-off* entre precisión y diversidad. Esto, es tan importante como el hecho de poder ofrecer flexibilidad en el diseño del algoritmo de recomendación.

En el futuro, la evaluación de los sistemas de recomendación muestra una tendencia a emplear juicios de usuarios en tiempo real, dejando atrás los esquemas de evaluación offline test/train. Aunque costoso, es ideal para medir calidad, ya que la percepción de diversidad es en parte propia de cada individuo. No obstante algunos aspectos de la diversidad son altamente subjetivos, la evaluación de diversidad se beneficiaría de incluir conocimiento experto del campo de la psicología en el desarrollo de nuevas métricas de diversidad.

Otro punto importante es que la diversidad debe ser considerada durante el proceso de recomendación, en lugar de ser aplicada como una etapa de re-ordenamiento posterior a generar la lista de productos. Este enfoque, presente en la mayoría de los algoritmos del Estado del Arte, asume que las recomendaciones son diversas y solo necesitan un orden correcto. Por otro lado, si un sistema sufre de sobre-ajuste se pueden generar casos en donde casi todos los productos de la recomendación sean más o menos lo mismo, lo cual causará que el proceso de diversificación falle. Por lo tanto, la diversificación debe estar presente desde el inicio del procedimiento de recomendación y debe incluirse en el proceso de ranking/estimación de ratings.

5.5.1. Trabajo Futuro

- La debilidad de la mayoría de las métricas destinadas a evaluar diversidad (incluidas las propuestas), es que comparan productos de acuerdo a su descripción extraída a partir de meta-data. El problema surge cuando se requiere manejar sistemas que funcionan con diferentes tipos de productos o cuando no se tiene disponibilidad sobre los descriptores. Una alternativa sería inferir automáticamente los nuggets a través de topic models.
- Realizar un análisis exhaustivo del impacto de los modelos de clustering en el proceso de diversificación. Una alternativa sería intentar homologar los experimentos realizados en la Parte I, utilizando las métricas de content-novelty enunciadas en la Parte II. De esta forma, sería posible explorar el alcance de los algoritmos de clustering en términos de diversidad.
- Incluso aunque algunos trabajos se han encargado de realizar estudios de casos reales, no son aún concluyentes. Mientras ellos han preguntado a los usuarios acerca de su percepción de la diversidad en las recomendaciones, aún no se han cuestionado como los usuarios realmente definen y entienden este concepto. Ese valioso conocimiento puede ser incluido en el desarrollo de una nueva versión de las métricas propuestas.

Bibliografía

[1] Zeinab Abbassi, Vahab S Mirrokni, and Mayur Thakur. Diversity maximization under matroid constraints. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 32–40. ACM, 2013.

[2] Gediminas Adomavicius and YoungOk Kwon. Maximizing aggregate recommendation diversity: A graph-theoretic approach. In *Proc. of the 1st International Workshop on Novelty and Diversity in Recommender Systems (DiveRS 2011)*, pages 3–10, 2011.

[3] 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.

[4] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. *Automatic subspace clustering of high dimensional data for data mining applications*, volume 27. ACM, 1998.

[5] Stanley C Ahalt, Ashok K Krishnamurthy, Prakoon Chen, and Douglas E Melton. Competitive learning algorithms for vector quantization. *Neural networks*, 3(3):277–290, 1990.

[6] Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: ordering points to identify the clustering structure. In *ACM Sigmod record*, volume 28, pages 49–60. ACM, 1999.

[7] David Arthur and Sergei Vassilvitskii. How slow is the k-means method? In *Proceedings of the twenty-second annual symposium on Computational geometry*, pages 144–153. ACM, 2006.

[8] Tevfik Aytekin and Mahmut Özge Karakaya. Clustering-based diversity improvement in top-n recommendation. *Journal of Intelligent Information Systems*, 42(1):1–18, 2014.

[9] Arindam Banerjee and Joydeep Ghosh. On scaling up balanced clustering algorithms. In *SDM*, volume 2, pages 333–349. SIAM, 2002.

[10] Pierpaolo Basile, Cataldo Musto, Marco de Gemmis, Pasquale Lops, Fedelucio Narducci, and Giovanni Semeraro. Aggregation strategies for linked open data-enabled recommender systems. In *European Semantic Web Conference*, 2014.

[11] Punam Bedi, Shikha Agarwa, Archana Singhal, Ena Jain, and Gunjan Gupta. A novel semantic clustering approach for reasonable diversity in news recommendations. In *Computational Intelligence in Data Mining-Volume 1*, pages 437–445. Springer, 2015.

[12] Richard E. Bellman. *Adaptive control processes - A guided tour*. Princeton University Press, Princeton, New Jersey, U.S.A., 1961.

[13] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is "nearest neighbor" meaningful? In *International conference on database theory*, pages 217–235. Springer, 1999.

[14] James C Bezdek, Robert Ehrlich, and William Full. Fcm: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2-3):191–203, 1984.

[15] Rubi Boim, Tova Milo, and Slava Novgorodov. Diversification and refinement in collaborative filtering recommender. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 739–744. ACM, 2011.

[16] Keith Bradley and Barry Smyth. Improving recommendation diversity. In *Proceedings of the Twelfth Irish Conference on Artificial Intelligence and Cognitive Science, Maynooth, Ireland*, pages 85–94. Citeseer, 2001.

[17] PS Bradley, KP Bennett, and Ayhan Demiriz. Constrained k-means clustering. *Microsoft Research, Redmond*, pages 1–8, 2000.

[18] J Roger Bray and John T Curtis. An ordination of the upland forest communities of southern wisconsin. *Ecological monographs*, 27(4):325–349, 1957.

[19] Derek Bridge and John Paul Kelly. Ways of computing diverse collaborative recommendations. In *AH*, pages 41–50. Springer, 2006.

[20] Sylvain Castagnos, Armelle Brun, and Anne Boyer. When diversity is needed... but not expected! In *International Conference on Advances in Information Mining and Management*, pages 44–50. IARIA XPS Press, 2013.

[21] P. Castells, N. Hurley, and S. Vargas. *Novelty and diversity in recommender systems*. In Ricci, F., Rokach, L., and Shapira, B., editors, Recommender Systems Handbook, 2nd Edition, Springer US., 2015.

[22] Pablo Castells, Saúl Vargas, and Jun Wang. Novelty and diversity metrics for recommender systems: choice, discovery and relevance. 2011.

[23] O. Celma. *Music Recommendation and Discovery in the Long Tail*. Springer, 2010.

[24] Sang-Min Choi and Yo-Sub Han. A content recommendation system based on category correlations. In *Computing in the Global Information Technology (ICCGI), 2010 Fifth International Multi-Conference on*, pages 66–70. IEEE, 2010.

[25] C. Clarke, M. Kolla, G. Cormack, O. Vechtomova, A. Ashkan, S. B*ü*ttcher, and I. MacKinnon. Novelty and diversity in information retrieval evaluation. In *Proceedings of the 31st International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 659–666, 2008.

[26] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):603–619, 2002.

[27] Joydeep Das, Subhashis Majumder, Debarshi Dutta, and Prosenjit Gupta. *Iterative Use of Weighted Voronoi Diagrams to Improve Scalability in Recommender Systems*, pages 605–617. Springer, 2015.

[28] Joydeep Das, Partha Mukherjee, Subhashis Majumder, and Prosenjit Gupta. Clustering-based recommender system using principles of voting theory. In *International Conference on Contemporary Computing and Informatics (IC3I)*, pages 230–235. IEEE, 2014.

[29] Rajesh N Dave and Kurra Bhaswan. Adaptive fuzzy c-shells clustering and detection of ellipses. *IEEE Transactions on Neural Networks*, 3(5):643–662, 1992.

[30] Tommaso Di Noia, Vito Claudio Ostuni, Jessica Rosati, Paolo Tomeo, and Eugenio Di Sciascio. An analysis of users' propensity toward diversity in recommendations. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 285–288. ACM, 2014.

[31] Chris HQ Ding, Xiaofeng He, Hongyuan Zha, Ming Gu, and Horst D Simon. A min-max cut algorithm for graph partitioning and data clustering. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 107–114. IEEE, 2001.

[32] Joseph C Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. 1973.

[33] Michael D Ekstrand, F Maxwell Harper, Martijn C Willemsen, and Joseph A Konstan. User perception of differences in recommender algorithms. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 161–168. ACM, 2014.

[34] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[35] Daniel M Fleder and Kartik Hosanagar. Recommender systems and their impact on sales diversity. In *Proceedings of the 8th ACM conference on Electronic commerce*, pages 192–199. ACM, 2007.

[36] Mouzhi Ge, Fatih Gedikli, and Dietmar Jannach. Placing high-diversity items in top-n recommendation lists. In *Workshop chairs*, page 65, 2011.

[37] Thomas George and Srujana Merugu. A scalable collaborative filtering framework based on co-clustering. In *Data Mining, Fifth IEEE International Conference on*, pages 4–pp. IEEE, 2005.

[38] Sanjay Goil, Harsha Nagesh, and Alok Choudhary. Mafia: Efficient and scalable subspace clustering for very large data sets. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 443–452. ACM, 1999.

[39] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Cure: an efficient clustering algorithm for large databases. In *ACM Sigmod Record*, volume 27, pages 73–84. ACM, 1998.

[40] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Rock: A robust clustering algorithm for categorical attributes. *Information systems*, 25(5):345–366, 2000.

[41] M. Hahsler. Recommenderlab: A framework for developing and testing recommendation algorithms, r package, 2016.

[42] M. Hahsler. *Recommenderlab: A Framework for Developing and Testing Recommendation Algorithms, R package, In: https://CRAN.R-project.org/package=recommenderlab*. 2016.

[43] Eui-Hong Han, George Karypis, Vipin Kumar, and Bamshad Mobasher. Clustering in a high-dimensional space using hypergraph models. *Proceedings of data mining and knowledge discovery*, 1997.

[44] Alexander Hinneburg and Daniel A Keim. An efficient approach to clustering in large multimedia databases with noise. In *KDD*, volume 98, pages 58–65, 1998.

[45] Alexander Hinneburg and Daniel A Keim. Optimal grid-clustering: Towards breaking the curse of dimensionality in high-dimensional clustering. 1999.

[46] Yu-Chieh Ho, Yi-Ting Chiang, and Jane Yung-Jen Hsu. Who likes it more?: mining worth-recommending items from long tails by modeling relative preference. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 253–262. ACM, 2014.

[47] T. Hofmann. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems*, 22(1):89–115, 2004.

[48] Rong Hu and Pearl Pu. Helping users perceive recommendation diversity. In *DiveRS@ RecSys*, pages 43–50, 2011.

[49] Neil Hurley and Mi Zhang. Novelty and diversity in top-n recommendation–analysis and evaluation. *ACM Transactions on Internet Technology (TOIT)*, 10(4):14, 2011.

[50] Raymond A Jarvis and Edward A Patrick. Clustering using a similarity measure based on shared near neighbors. *IEEE Transactions on Computers*, 100(11):1025–1034, 1973.

[51] Amin Javari and Mahdi Jalili. A probabilistic model to resolve diversity–accuracy challenge of recommendation systems. *Knowledge and Information Systems*, 44(3):609–627, 2015.

[52] K. J*ä*rvelin and J. Kek*ä*l*ä*inen. Ir evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 41–48, 2000.

[53] Hai Jiang, Xin Qi, and He Sun. Choice-based recommender systems: a unified approach to achieving relevancy and diversity. *Operations Research*, 62(5):973–993, 2014.

[54] Xue-Mei Jiang, Wen-Guan Song, and Wei-Guo Feng. Optimizing collaborative filtering by interpolating the individual and group behaviors. In Xiaofang Zhou, Jianzhong Li, Heng Tao Shen, Masaru Kitsuregawa, and Yanchun Zhang, editors, *Frontiers of WWW Research and Development - APWeb 2006: 8th Asia-Pacific Web Conference*, pages 568–578. Springer, 2006.

[55] George Karypis, Eui-Hong Han, and Vipin Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, 1999.

[56] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1):359–392, 1998.

[57] George Karypis and Vipin Kumar. hmetis 1.5: A hypergraph partitioning package. Technical report, Technical report, Department of Computer Science, University of Minnesota, 1998. Available on the WWW at URL http://www. cs. umn. edu/metis, 1998.

[58] Leonard Kaufman and Peter J Rousseeuw. Partitioning around medoids (program pam). *Finding groups in data: an introduction to cluster analysis*, pages 68–125, 1990.

[59] Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009.

[60] Jerome Kelleher and Derek Bridge. Rectree centroid: An accurate, scalable collaborative recommender. *AICS 2003*, page 7, 2003.

[61] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, 2009.

[62] Neal Lathia, Stephen Hailes, Licia Capra, and Xavier Amatriain. Temporal diversity in recommender systems. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 210–217. ACM, 2010.

[63] Kibeom Lee and Kyogu Lee. Escaping your comfort zone: A graph-based recommender system for finding novel recommendations among relevant items. *Expert Systems with Applications*, 42(10):4851–4858, 2015.

[64] Amaury L'Huillier, Sylvain Castagnos, and Anne Boyer. Understanding usages by modeling diversity over time. In *22nd Conference on User Modeling, Adaptation, and Personalization*, volume 1181, 2014.

[65] J. B. MacQueen. "Some methods for classification and analysis of multivariate observations". *in Proceedings of the 5th Symposium on Math, Statistics, and Probability*, pages 281–297, 1967.

[66] Mikko I Malinen and Pasi Fränti. Balanced k-means for clustering. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 32–41. Springer, 2014.

[67] Fernando Mourão, Claudiane Fonseca, Camila Souza Araujo, and Wagner Meira Jr. The oblivion problem: Exploiting forgotten items to improve recommendation diversity. In *DiveRS@ RecSys*, pages 27–34, 2011.

[68] Raymond T. Ng and Jiawei Han. Clarans: A method for clustering objects for spatial data mining. *IEEE transactions on knowledge and data engineering*, 14(5):1003–1016, 2002.

[69] Wichian Premchaiswadi, Pitaya Poompuang, Nipat Jongswat, and Nucharee Premchaiswadi. Enhancing diversity-accuracy technique on user-based top-n recommendation algorithms. In *Computer Software and Applications Conference Workshops (COMPSACW), 2013 IEEE 37th Annual*, pages 403–408. IEEE, 2013.

[70] Xiaolong Ren, Linyuan Lü, Runran Liu, and Jianlin Zhang. Avoiding congestion in recommender systems. *New Journal of Physics*, 16(6):063057, 2014.

[71] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the Conference on Computer Supported Cooperative Work, CSCW*, pages 175–186, 1994.

[72] Marco Tulio Ribeiro, Nivio Ziviani, Edleno Silva De Moura, Itamar Hata, Anisio Lacerda, and Adriano Veloso. Multiobjective pareto-efficient approaches for recommender systems. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(4):53, 2015.

[73] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer, 2011.

[74] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the Tenth International World Wide Web Conference, WWW*, pages 285–295, 2001.

[75] Badrul M Sarwar, George Karypis, Joseph Konstan, and John Riedl. Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. In *Proceedings of the fifth international conference on computer and information technology*, volume 1. Citeseer, 2002.

[76] JHJB Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. Collaborative filtering recommender systems. *The adaptive web*, pages 291–324, 2007.

[77] Malcolm Slaney and William White. Measuring playlist diversity for recommendation systems. In *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, pages 77–82. ACM, 2006.

[78] Alexander Strehl and Joydeep Ghosh. Relationship-based clustering and visualization for high-dimensional data mining. *INFORMS Journal on Computing*, 15(2):208–230, 2003.

[79] Veronika Strnadová-Neeley, Aydin Buluç, John R. Gilbert, Leonid Oliker, and Weimin Ouyang. Lira: A new likelihood-based similarity score for collaborative filtering. In *Workshop on Large Scale Recommender Systems co-located with ACM RecSys*, 2016.

[80] Nava Tintarev, Matt Dennis, and Judith Masthoff. Adapting recommendation diversity to openness to experience: A study of human behaviour. In *International Conference on User Modeling, Adaptation, and Personalization*, pages 190–202. Springer, 2013.

[81] Anthony KH Tung, Jiawei Han, Laks VS Lakshmanan, and Raymond T Ng. Constraint-based clustering in large databases. In *International Conference on Database Theory*, pages 405–419. Springer, 2001.

[82] S Vaishnavi, A Jayanthi, and S Karthik. Ranking technique to improve diversity in recommender systems. *International Journal of Computer Applications*, 68(2), 2013.

[83] S. Vargas. *Novelty and diversity evaluation and enhancement in recommender systems*. PhD thesis, Universidad Autónoma de Madrid, Spain, 2015.

[84] Saúl Vargas. New approaches to diversity and novelty in recommender systems. In *Fourth BCS-IRSG symposium on future directions in information access (FDIA 2011), Koblenz*, volume 31, 2011.

[85] Saúl Vargas, Linas Baltrunas, Alexandros Karatzoglou, and Pablo Castells. Coverage, redundancy and size-awareness in genre diversity for recommender systems. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 209–216, New York, NY, USA, 2014. ACM.

[86] Gui-Rong Xue, Chenxi Lin, Qiang Yang, WenSi Xi, Hua-Jun Zeng, Yong Yu, and Zheng Chen. Scalable collaborative filtering using cluster-based smoothing. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 114–121. ACM, 2005.

[87] Ronald R Yager and Dimitar P Filev. Approximate clustering via the mountain method. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(8):1279–1284, 1994.

[88] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: an efficient data clustering method for very large databases. In *ACM Sigmod Record*, volume 25, pages 103–114. ACM, 1996.

[89] Shi Zhong and Joydeep Ghosh. Scalable, balanced model-based clustering. In *SDM*, pages 71–82. SIAM, 2003.

[90] Shi Zhong and Joydeep Ghosh. A unified framework for model-based clustering. *Journal of machine learning research*, 4(Nov):1001–1037, 2003.

[91] Shunzhi Zhu, Dingding Wang, and Tao Li. Data clustering with size constraints. *Knowledge-Based Systems*, 23(8):883–889, 2010.

[92] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*, pages 22–32. ACM, 2005.