Universidad Autónoma de Sinaloa

facultad de informática culiacán
facultad de ciencias de la tierra y el espacio

maestría en ciencias de la información

PROPUESTA DE ALGORITMO DE FILTRADO COLABORATIVO BASADO EN VECINDARIOS PARA SISTEMAS DE RECOMENDACIÓN

Tesis

COMO REQUISITO PARA OBTENER EL GRADO DE

Maestro en Ciencias de la Información

PRESENTA:

Ing. Emmanuel Felix Valenzuela

Director de tesis: Dr. Inés Fernando Vega López
Codirector de tesis: Dr. Arturo Yee Rendón

Culiacán, Sinaloa, México Febrero del 2024

Dedicatoria

A mi esposa Cecilia Angulo Domínguez, a quien le he externado todas mis inseguridades y me hace seguir adelante en cada ocasión.

Agradecimientos

Estoy muy agradecido de haber llegado a este punto ya que en ciertos momentos no me sentía capaz de llegar tan lejos. Tener un momento como este de reflexión acentúa las cosas realmente importantes, en mi caso son aquellas personas de quienes he recibido instrucción y me han hecho quien soy ahora.

Dentro de la gran lista de personas a quienes agradezco, quisiera mencionar en primer lugar a mi madre, a quien le debo la vida, ha sido una gran impulsora de mi educación tanto académica como de vida. En ella puedo destacar su gran sentido del trabajo duro y de la honestidad que me han hecho verla como un modelo a seguir. Por otra parte, agradezco a mi padre quien me ayuda siempre a tener nuevas metas y a mis hermanos quienes me han acompañado a lo largo de la vida.

Sin duda alguna las palabras no me alcanzan para sentir mi agradecimiento a mi director de tesis Inés Fernando Vega López. Su trabajo me deja lecciones de vida mas allá del contenido en este trabajo de investigación sino de una genuina formación científica, su compromiso a este objetivo y sus atenciones para conmigo no desistieron incluso después de la poca flexibilidad de mi parte en cambiar prácticas arraigadas que evitaban el correcto seguimiento de la investigación. Llegado a este punto me quedo convencido que la formación y la cátedra del doctor Inés Vega han rendido sus frutos.

Otras contribuciones importantes de las debo a mis profesores de posgrado, sin duda todos se han mostrado interesados en ayudar a la formación de los estudiantes. Entre ellos se encuentran, Arturo Yee Rendón quien también participó como asesor de la investigación; Gerardo Beltrán Gutiérrez y Jorge Adalberto Navarro Castillo. Estos últimos nos han compartido elementos de gran importancia para esta investigación.

De la manera más cálida posible, agradezco también a mi esposa Cecilia Angulo Domínguez, quien me ha acompañado en este proceso de inicio a fin, quien a velado por el bien de la terminación de esta investigación, quien en muchos momentos fue la única espectadora de mis presentaciones y quien me ayudaba a no desistir de tan ambiciosa meta.

Por último y no menos importante, quiero agradecer a Conahcyt, a la administración del posgrado y a la UAS por haberme permitido continuar con mis estudios.

Abstract

Recommendation systems are tools highly used in recommendation prediction due to their effectiveness in capturing the interests of users. They are a widely studied subject both in the scientific and in the industry fields. However, the new information collection tools as well as the growth of the sectors that use these systems provoke the need to look for increasingly efficient methods that employ the use of better recommendations at the time that users require it.

In this research, various methods and strategies of recommender systems based on collaborative filtering are analyzed. Collaborative filtering is a technique for estimating recommendations to users using information from them as well as from users who share similar characteristics.

In this document is proposed the use of graphics processing units to develop a new algorithm based on state-of-the-art methods, aiming to reduce the time used for providing recommendations to users. The experimental evaluation of our proposal shows that a response time bet-

ween 3 and 29 times shorter can be achieved compared to the sequential algorithm designed to be used with CPU only.

Resumen

Los sistemas de recomendación son herramientas altamente utilizadas para la generación de recomendaciones gracias a su eficacia en captar los intereses de los usuarios. Son un tema altamente estudiado tanto en el ámbito científico como en el ámbito empresarial. Sin embargo, las nuevas herramientas de recolección de información así como el crecimiento de los sectores que disponen de éstos sistemas provocan la necesidad de buscar métodos cada vez más eficientes que permitan el uso de mejores recomendaciones en el momento que los usuarios lo requieran.

En esta investigación se analizan diversos métodos y estrategias de sistemas de recomendación basados en filtrado colaborativo, una técnica de estimación de recomendaciones a usuarios utilizando la información de estos así como de usuarios que comparten características similares.

En el presente documento se propone el uso de unidades de procesamiento gráfico para desarrollar un nuevo algoritmo con base en métodos basados en el estado del arte para reducir el tiempo de obtención de re-

comendaciones para los usuarios. La evaluación experimental de nuestra propuesta muestra que se puede lograr un tiempo de respuesta de entre 3 y 29 veces menor que el tiempo de respuesta del algoritmo secuencial diseñado para usar solo CPU.

Índice general

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

Un sistema de recomendación es una herramienta que permite filtrar información de amplios espacios de búsqueda, mostrando así solo contenido de posible interés para el usuario de forma personalizada [12]. En sistemas digitales basados en servicios de internet donde la cantidad de información existente es muy amplia, los sistemas de recomendación son una solución bastante práctica. Estos reducen el espacio de búsqueda para el usuario de forma que pueda buscar solo entre contenido potencialmente útil. Algunas de las ventajas que tienen los clientes cuando se utiliza un sistema de recomendación son la facilidad en la búsqueda de objetos dentro de sus intereses, encontrar contenido relevante, tener alternativas útiles y mejorar la toma de decisiones [41].

A día de hoy los sistemas de recomendación han tenido mucho auge

principalmente por la gran cantidad de aplicaciones en las que se pueden implementar. Ejemplos de esto son la recomendación de contenido multimedia (Spotify [65], Youtube [75], Netflix [54]), recomendación de otros usuarios (Linkedin [46], Facebook [24], Twitter [71]), recomendación de artículos a comprar (Amazon [5], MercadoLibre [50], Coppel [18]), recomendación de páginas de internet (Google [29], Duckduckgo [23]), recomendación de hospedajes (AirB&B [4], Trivago [70]) y recomendación de artículos científicos (Springer [66], Google Scholar [63]), solo por mencionar algunas aplicaciones.

Esta versatilidad por parte de los sistemas de recomendación da lugar a que tanto el sector industrial, así como la academia aporten conocimiento en el desarrollo de esta área. Para entender un poco la producción científica respecto al tema, los trabajos de Adomavicius [2], Abdollahpouri [1], Koren [44], Lü [48] y Lu [47] ofrecen buenos estudios antológicos enfocados a los sistemas de recomendación.

Dada la gran amplitud de esta área de estudio, nos enfocaremos en una rama de los sistemas de recomendación, el filtrado colaborativo. El filtrado colaborativo busca contenido relevante para un usuario con base en la obtención de una función de similitud que permita asociarlo con otros usuarios. Nos referimos a un usuario como una persona cuyas acciones y recomendaciones se han registrado en un sistema de información. De manera similar, se puede utilizar la asociación entre distintos elementos recomendables conocidos como ítems. Para identificar las po-

sibles preferencias del usuario, se busca dentro de las elecciones de los usuarios evitando el uso de la información de las propiedades o atributos que existan en los ítems o los usuarios del sistema. La función de similitud antes mencionada permite tener valores apreciables y cuantificables para la identificación de los usuarios a tomar en cuenta a la hora de hacer las recomendaciones. A pesar de sus buenos resultados, uno de los principales problemas de la implementación de los filtrados colaborativos es el tiempo que toman en hacer las recomendaciones. Este tiempo suele aumentar de acuerdo al número de usuarios n y al número de ítems m [45].

1.1. Justificación

Los sistemas de recomendación generan cada vez mayor interés interés en la investigación científica y la industria por la facilidad con la que el usuario puede encontrar contenido de valor en las plataformas donde son implementados. Al hablar de contenido de valor, se habla no solo de contenido multimedia (vídeo, musica, imágenes, etc.), sino también de posibles productos a comprar [45], localización de páginas web, uso de repositorios de objetos de aprendizaje [58], entre muchas otras aplicaciones. Lu [47] ofrece un estudio extenso basado en aplicaciones de los sistemas de recomendación.

Algunos ejemplos tangibles del éxito de los sistemas de recomen-

dación son el caso de Netflix, empresa de entretenimiento multimedia cuyos orígenes fueron la renta de DVD's con entrega a domicilio. Netflix en 2006 lanza un reto para mejorar un 10 % la exactitud de su sistema de recomendación. Con un premio de un millón de dolares la plataforma encontró un equipo de desarrolladores capaz de lograr una mejora aproximada del 8 %, hecho que implicó un enorme avance en la investigación de los sistemas de recomendación [10]. Por otro lado, Chen, Wu y Yoon muestran en su investigación que las recomendaciones hechas por la plataforma Amazon tienen un impacto directo en el nivel de compras por parte del cliente [14]. Así mismo se tiene que los sistemas de recomendación, al identificar los gustos de los usuarios, disminuyen el esfuerzo por parte de estos (a menudo desalentador) de buscar información relevante de productos que se ajusten al usuario [14]. Esto último tiene un impacto positivo en la venta de los productos menos populares, llamados también como "la cola larga", una serie de productos conocidos por su poca rotación en inventario. Una cualidad de manejar esta serie de productos es que usualmente tienen los mayores márgenes de ganancia.

Además del beneficio económico de estas empresas, existe también un aumento en la fidelidad por parte del cliente al obtener contenido de interés en sus plataformas [41]. Tanto Amazon como Netflix son grandes referentes para el asentamiento de los sistemas de recomendación en la industria, sin embargo, dichos sistemas tienen efectos similares para una gran variedad de situaciones.

Con lo que respecta a los objetivos destacados de esta investigación, se espera encontrar aportes en la reducción de tiempos de respuesta de los sistemas de recomendación sin impactar significativamente la calidad de las recomendaciones. Es decir, se espera que nuestros métodos no empeoren la calidad de medidas de evaluación basadas en utilidad[1] en más de un 10 %. Además, se espera encontrar y describir las limitantes de estos métodos de paralelización con respecto a los sistemas de recomendación. Como se puede notar, se muestra una preferencia por la disminución de tiempos de respuesta en contraste con la calidad de las recomendaciones otorgadas. A continuación se ilustran los motivos al respecto.

Calidad de las recomendaciones. Con las técnicas actuales de los sistemas de recomendación se puede llegar a resultados suficientemente buenos de forma que aumentar la exactitud de un sistema de recomendación no resulta práctico [59]. Rossetti es uno de varios autores que comprueban esta teoría al comparar las medidas de evaluación de exactitud convencionales con resultados en tiempo real con usuarios, sus resultados pueden ser consultados en su investigación [61].

Tiempo de recomendación. A día de hoy las herramientas de recopilación de información han permitido que el 90 % de los datos existentes en el mundo se hayan creado tan solo en los últimos 3 años [8]. Contar con cantidades tan crecientes de información implica que los costos

[1]Las medidas de evaluación basadas en utilidad son descritas en el Capítulo 5.

computacionales de entrenamiento y explotación de un sistema de recomendación sean altos, es por esto que se debe de continuar el trabajo para la optimización de dichas técnicas. Estos esfuerzos implican una experiencia de usuario más dinámica y una mayor actividad en el sitio donde son aplicados. Finalmente, se tiene la creciente cantidad de sistemas computacionales como sitios web de comercio electrónico o redes sociales. Estos, al estar enfocados en la web necesitan realizar las transacciones en tiempo real. Un tiempo de respuesta de un sistema de recomendación mayor a unos segundos podría resultar excesivo para los usuarios.

1.2. Hipótesis

Se ha realizado este trabajo de investigación con el fin de encontrar y desarrollar los métodos actuales más eficientes computacionalmente en términos de tiempo. Al inicio de la investigación se ha dado lugar a la siguiente hipótesis.

El uso de hardware especializado como el uso de unidades de procesamiento gráfico (GPU) puede reducir en al menos 10 % los tiempos de ejecución de sistemas de filtrado colaborativo.

Con esta hipótesis se busca marcar una pauta para posteriores mejoras en el desarrollo de las recomendaciones haciendo uso de técnicas existentes. El origen de dichas técnicas, no se limitan solamente a técni-

cas utilizadas en los sistemas de recomendación, sino en diversos ámbitos referentes a la computación.

1.3. Definición formal del problema

Como ya se mencionó, los sistemas de recomendación son herramientas para reducir espacios de búsqueda priorizando ítems que le sean de interés al usuario [12]. Formalmente, un sistema de recomendación se puede definir por la Expresión 1.1 [2].

$$\forall\ u \in U, \quad i' = argMax_{i \in I}(f(u, i)) \tag{1.1}$$

Esta expresión nos indica que para cada usuario u perteneciente al conjunto de usuarios U en el sistema de recomendación, se obtiene un ítem i' que es el ítem mejor evaluado para dicho usuario, dado que el ítem i pertenece al conjunto de ítems I. Se somete tanto u como i a la función de evaluación f para obtener una estimación numérica del posible interés del usuario u con respecto al ítem i. En la práctica, es más común tomar un conjunto de los ítems mejor evaluados en vez de solo tomar el mejor ítem. Esto es también conocido como el problema de las mejores N recomendaciones. Los problemas más comunes que atienden los sistemas de recomendación son la estimación del interés de los usuarios sobre los ítems y la selección de los mejores N ítems para

cada usuario.

1.4. Organización de la tesis y aportes de la investigación

El resto de este documento se organiza de la siguiente forma. En el Capítulo 2 tenemos el marco de referencia, en éste se pueden encontrar una serie de conceptos para el entendimiento de la investigación. De manera más precisa, se tienen conceptos relacionados con los sistemas de recomendación, las técnicas de filtrado colaborativo (el eje principal de la investigación) y las unidades de procesamiento gráfico. En el Capítulo 3 se tiene el estado del arte. En el Capítulo 4 se tiene la metodología. Aquí se explican a detalle los métodos utilizados a mejorar, las modificaciones realizadas y la serie de experimentos y parámetros a evaluar. En el Capítulo 5 se presentan nuestros resultados experimentales. Se describe todo el proceso del tratamiento de los conjuntos de datos para una mejor manipulación de estos. Los resultados se han obtenido con base en la comparación de los métodos convencionales con los métodos optimizados mediante el uso de tarjetas gráficas. Finalmente, se presentan las conclusiones de este trabajo de investigación así como posibles extensiones de la investigación a futuro.

Por otro lado, los experimentos realizados están disponibles para su consulta como código abierto [25]. Los aportes obtenidos, derivados

del trabajo de esta investigación reafirman la validez de la hipótesis planteada y consisten en lo siguiente:

- Se ha hecho una comparativa entre algoritmos del estado del arte con el fin de conocer el algoritmo más rápido para los escenarios propuestos.
- Se desarrolló un algoritmo para sistemas de recomendación que permite una rápida recomendación a los usuarios. El método implementado es una variación de los algoritmos del estado del arte por lo que los resultados ofrecidos son los mismos a excepción de los tiempos utilizados para la recomendación.
- Se ha analizado el rendimiento de este algoritmo así como sus ventajas y limitaciones. Los resultados muestran tiempos de respuesta entre 16 y 113 veces menor con nuestros experimentos en contraste con sus versiones del estado del arte a un solo núcleo.

Capítulo 2

Marco de referencia

En este capítulo se describen una serie de conceptos y métodos referentes a este trabajo de investigación para fundamentar la propuesta a implementar.

2.1. Preliminares

A continuación se aborda una serie de conceptos que dan paso al entendimiento de los temas en los cuales se basa la investigación. Estos temas ayudan también a entender la delimitación de nuestra investigación dentro de los sistemas de recomendación.

Tipos de retroalimentación al sistema de recomendación

Las distintas formas de conseguir información de los usuarios impacta directamente al sistema de recomendación [3]. El principal efecto de la implementación de diversos métodos para la obtención de la información es la cantidad de evaluaciones que se pueda disponer. Esto se debe al hecho de que efectuar una evaluación más completa de parte del usuario implica un esfuerzo mental extra por parte de éste. Por ejemplo, alguien puede decir con facilidad que le ha gustado un ítem, pero puede complicarse decidir si le ha parecido bueno, muy bueno, excelente o maravilloso. Esto evita la obtención de dicha evaluación en repetidos casos. Aggarwal [3] categoriza los tipos de recomendación como evaluaciones continuas, discretas, ordinales y retroalimentación unaria. Las evaluaciones continuas son expresadas en un intervalo continuo. El usuario tiene la posibilidad de establecer su preferencia a un ítem con un número real. Estas evaluaciones dificultan al usuario establecer una evaluación al tener la posibilidad de escoger una calificación en un rango muy vasto en posibilidades. Una alternativa es utilizar una barra donde se coloque una marca de acuerdo al nivel de agrado. En este caso la resolución se ve afectada por la granularidad de la interfaz [28]. Por otro lado, con las evaluaciones discretas las posibilidades del usuario varían solamente entre los números enteros. Algunos ejemplos son evaluar desde 1 a 10 puntos o de 1 a 5 puntos, en este caso el usuario tiene la opción de calificar un ítem con $\{1, 2, 3, 4, 5\}$. Otro caso común, consiste en poder elegir entre agrado y desagrado, esta implementación representa una

evaluación discreta de 2 valores. Similarmente se tienen las evaluaciones ordinales, también conocidas como evaluaciones no numéricas, se hacen a partir de un conjunto de palabras o caracteres, de forma que cada elemento representa una categoría única. En algunos casos se puede traducir a equivalentes numéricos, por ejemplo, se puede hablar del gusto por algo con las categorías: {ninguno, poco, normal, bastante, mucho}, sin embargo, no resulta muy claro para el usuario el valor que tiene cada elemento. Finalmente, se tiene la retroalimentación unaria. Como evaluación, la retroalimentación unaria es del tipo más restrictivo, ya que solo permite mostrar gusto o interés por un ítem sin tener la opción de mostrar lo contrario, sin embargo, también se puede obtener en forma de retroalimentación implícita. La retroalimentación implícita es el tipo de retroalimentación más simple, solo expresa que hay relación de usuario a ítem. Por ejemplo, en el caso de cada ítem comprado por el usuario así como en el historial de accesos de un usuario a una serie de ítems. En estas retroalimentaciones se captura la información de acuerdo a las acciones del usuario y no a lo que éste exprese directamente.

Tipos de sistemas de recomendación

Los principales tipos de sistemas de recomendación son de filtrado colaborativo, basados en contenido, o bien, una combinación de distintos tipos conocidos como híbridos. Sin embargo, en el último par de décadas se tiene una gran cantidad de desarrollo en la obtención de la información, que ha dado lugar a nuevos tipos de sistemas de recomendación.

A continuación, vamos a definir brevemente cada uno de estos tipos de sistemas basándonos en la clasificación propuesta por Ricci [59].

Recomendaciones de filtrado colaborativo. El filtrado colaborativo hace recomendaciones para el usuario usando la información histórica de las evaluaciones en el sistema de recomendación sin analizar el contenido de los ítems. Como una de las técnicas de recomendación más exitosas, el filtrado colaborativo ha sido ampliamente estudiado por varias instituciones de investigación así como por la industria y ha sido aplicado en diversas cuestiones prácticas [15].

Al usuario se le recomiendan ítems que le gustaron a personas con gustos y preferencias similares. Dicho de otra forma, la función de estimación de interés de un usuario a un ítem, $f(u, i)$, se estima basada en $f(v, i)|v \in N(u)$; donde $N(u)$ es el conjunto de usuarios similares al usuario u. Así, a partir de las evaluaciones de v y de u a una serie de ítems, se puede obtener el nivel de similitud entre estos usuarios para determinar el grado en que las evaluaciones de v podrían coincidir con el usuario u [2]. En la siguiente sección se profundizará acerca del filtrado colaborativo.

Recomendaciones basadas en contenido. En las recomendaciones basadas en contenido se define para cada usuario un perfil de características $perfil(u)$, construido a partir de las características de cada ítem $contenido(i)$ por los cuales el usuario ha mostrado interés. Esto permite hacer una comparativa directa entre la información del

usuario con las características de un ítem dando lugar a que $f(u, i)$ sea estimada basada en la similitud entre $perfil(u)$ y $contenido(i)$ [2].

Recomendaciones basadas en conocimiento. El sistema basado en conocimiento hace recomendaciones basándose en la necesidad del usuario así como en la medida en que el ítem se ajusta a las necesidades del usuario [59]. Para esto, es común recopilar la información directamente del usuario, ejemplos de estos sistemas son la implementación de una serie de filtros para búsquedas de ítems (estos filtros pueden consistir en rangos de precios, colores, marcas, etc).

Ricci define los dos principales enfoques de los sistemas basados en conocimiento [59], los sistemas de razonamiento basados en casos y los sistemas basados en restricciones. Los primeros determinan las recomendaciones a partir de medidas de similitud que se aplican entre un ítem y un caso, por otro lado, en los sistemas basados en restricciones se prioriza mostrar todos los resultados que cumplan con los criterios descritos por el usuario.

Recomendaciones demográficas. Este tipo de sistema de recomendación hace las recomendaciones basadas en la información demográfica del usuario [59] (ejemplos de información demográfica son edad, sexo, nacionalidad, ocupación y nivel de ingresos). La investigación y desarrollo de las recomendaciones demográficas no es mucho, aún así, algunos ejemplos de esto es el uso de estereotipos con base en los datos del usuario para hacer las recomendaciones [60]; filtrar las re-

comendaciones por rangos de edad; filtrar las páginas web de posible relevancia para los usuarios por el lenguaje que éstos hablan o por el país de origen; o bien, buscar una similitud entre usuarios con el uso del perfil demográfico de éstos [3].

Basados en comunidad. Los sistemas de recomendación sociales o basados en comunidad se basan en el uso de información de los amigos del usuario y personas cercanas a éste para hacer recomendaciones. Se ha demostrado que los usuarios tienden a confiar más en las recomendaciones hechas por sus amigos y seres cercanos que en las recomendaciones hechas por desconocidos, incluso teniendo una mayor similitud al usuario [59].

Como principales herramientas se tienen la recomendación de usuarios a otros usuarios; el uso de etiquetas para asociar tanto al usuario como al objeto con dicha etiqueta; la identificación y el uso de influencia social para el marketing viral; y la estimación del impacto que tienen las evaluaciones de un usuario sobre otro usuario [3].

Recomendaciones híbridas. Habiendo puesto los tipos de sistemas de recomendación anteriores en contexto, vemos que muchas deficiencias de é stos pueden y son tratadas generando un sistema que unifique varios tipos de sistemas de recomendación. Un sistema híbrido combina dos o más técnicas con el fin de aprovechar las ventajas de la primera para resolverlas desventajas de la segunda. Por ejemplo los métodos de filtrado colaborativo, muestran deficiencias para el caso de

nuevos ítems ya que no pueden ser recomendados ítems que no han sido evaluados. Para este caso, implementar de forma adicional una técnica complementaria puede ayudar al sistema de recomendación [59]. A manera de resumen, en la Figura 2.1 se puede apreciar el mapa conceptual de los tipos de sistemas de recomendación.

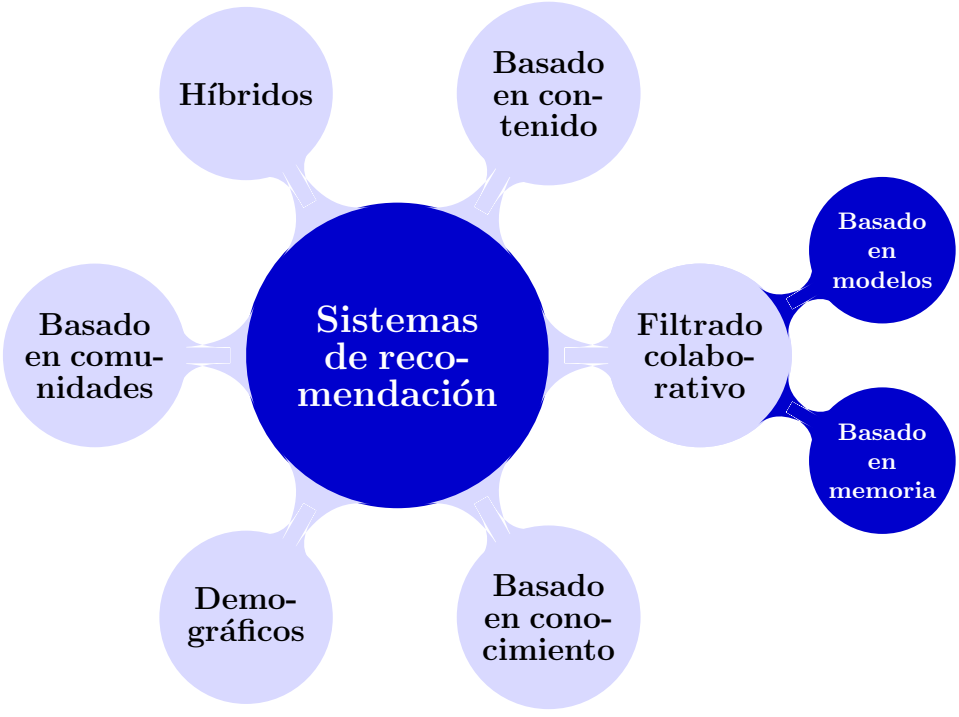

Figura 2.1: Tipos de sistemas de recomendación, basado en las ideas de Ricci [59].

2.2. Técnicas de filtrado colaborativo

El filtrado colaborativo genera buenos resultados en la calidad de las recomendaciones además de que no necesita información contextual del sistema para la elaboración de recomendaciones. Este tema ha sido ya ampliamente estudiado. Sin embargo, aún se encuentran áreas de oportunidad en lo que respecta a los tiempos de recomendación ya que este tipo de sistemas de recomendación es también de los más tardados al momento de ofrecer recomendaciones. Es por esto que solo se profundizará en este tipo de sistemas de recomendación. Dentro de esta sección se muestran las técnicas más utilizadas para el filtrado colaborativo así como los tipos de filtrado colaborativo que existen, es decir, filtros basados en memoria y filtros basados en modelos. Si un sistema utiliza directamente las evaluaciones de los usuarios para hacer recomendaciones se trata de un filtrado colaborativo basado en memoria. En caso de utilizar las evaluaciones para entrenar un modelo predictivo, se trata de un filtrado colaborativo basado en modelos [55]. Los modelos pueden incluir el uso de redes neuronales, técnicas bayesianas, técnicas difusas, máquinas de vectores de soporte y técnicas de aprendizaje profundo, entre otras técnicas. En la siguiente sección se abordará con detalle cada uno de estos conceptos.

2.2.1. Filtrado colaborativo basado en memoria

También es conocido como métodos de vecindades. Este enfoque de filtrado colaborativo consiste en la generación de una matriz de usuarios a usuarios cuyo valor en cada celda es la similitud calculada en forma numérica entre estos elementos. De manera similar, es posible utilizar una matriz de ítems a ítems para determinar la similitud de estos [45]. Siendo n el número de usuarios en el conjunto de usuarios U ($n = |U|$) y m el número de ítems en dicho conjunto I ($m = |I|$), la matriz de similitud puede ser de tamaño $n \times n$ o bien, de tamaño $m \times m$ para el caso de un enfoque basado en usuarios o en ítems, respectivamente.

Los métodos basados en vecindades se justifican con el principio de que usuarios similares tienen gustos similares. La metodología de vecindarios o vecinos más cercanos (KNN) consiste en utilizar los elementos de mayor similitud, específicamente los k vecinos más cercanos para hacer las recomendaciones con base en las evaluaciones de estos.

Esto se puede hacer ya sea que se busque para cada usuario los k usuarios más similares a estos para encontrar los ítems de interés de dichos usuarios, o bien, utilizar similitudes entre los ítems para buscar ítems similares a aquellos a los que se ha mostrado interés. Basar un sistema de recomendación en similitudes con usuarios o en similitudes con ítems representa una elección importante, sus efectos serán abordados en el Capítulo 4 de este documento.

Comúnmente, el cálculo de las similitudes entre usuarios o entre ítems se efectúa en una matriz de similitud en servidores fuera de línea y posteriormente se llevan los datos a un servidor en línea para hacer uso de éstos. Esto permite hacer un análisis único en complejidad $O(n^2m)$ para realizar posteriormente varios conjuntos de recomendaciones en complejidad $O(mk)$. Donde n es el número de usuarios, m el número ítems y k el número de usuarios relevantes a utilizar en las recomendaciones. Para calcular las similitudes entre elementos se hace uso de diversas funciones de semejanza, dentro de las principales, se encuentran el cálculo de la similitud cosenoidal, la correlación de Pearson y la distancia euclidiana. En la siguiente sección se describe con detalle estas funciones de semejanza.

Funciones de similitud

A continuación, vamos a explicar las tres funciones de similitud mayormente utilizadas; la similitud cosenoidal, la correlación de Pearson y la distancia euclidiana. Adicionalmente, se muestra la función de similitud de coseno ajustado, enfocada especialmente para recomendaciones basadas en ítems, una versión de la correlación de Pearson basada en ítems en lugar de usuarios. Para esto, vamos a explicar primero algunos términos relevantes en la Tabla 2.1. Para el caso de la evaluación del usuario u sobre el ítem i (r_{ui}) y los vecinos más similares al usuario u que han calificado al ítem i ($N_i(u)$) se conservan los términos utilizados

Tabla 2.1: Elementos comúnmente utilizados en los sistemas de recomendación.

Elemento	Descripción
I_u	Conjunto de ítems que ha evaluado el usuario u.
U_i	Conjunto de usuarios que han evaluado al ítem i.
I_{uv}	Conjunto de ítems que han evaluados tanto el usuario u como el usuario v ($I_{uv} = I_u \cap I_v$).
U_{ij}	Conjunto de usuarios que han evaluado tanto al ítem i como al ítem j ($U_{ij} = U_i \cap U_j$).
$\vec{u}$	vector representativo del usuario u.
$\|X\|$	Número de elementos en el conjunto X.
r_{ui}	Evaluación del usuario u sobre el ítem i.
$\hat{r}_{ui}$	Estimación que hace el sistema de recomendación para aproximar el interés del usuario u en el ítem i.
μ_u	Calificación media del usuario u ($\mu_u = \frac{1}{\|I_u\|} \sum_{i \in I_u} r_{ui}$).
σ_u	Es la desviación estándar de las evaluaciones del usuario u.
$N_i(u)$	Son los k vecinos más similares del usuario u que han calificado al ítem i.

comúnmente en la literatura derivados de *rating* y *neighbors*, respectivamente.

Similitud Cosenoidal. La similitud cosenoidal mide la relación que existe entre dos vectores. El coseno del ángulo encontrado entre estos vectores sería la similitud cosenoidal. Para un sistema de recomendación se puede interpretar a un usuario como el vector de ítems que ha calificado, dando lugar a que se estime su similitud como en la Ecuación 2.1. Los resultados de dicha función varían en un rango desde 0 hasta 1. El resultado es 1 cuando los vectores tienen la misma dirección, es decir, usuarios con alto grado de similitud. El resultado cero implica que

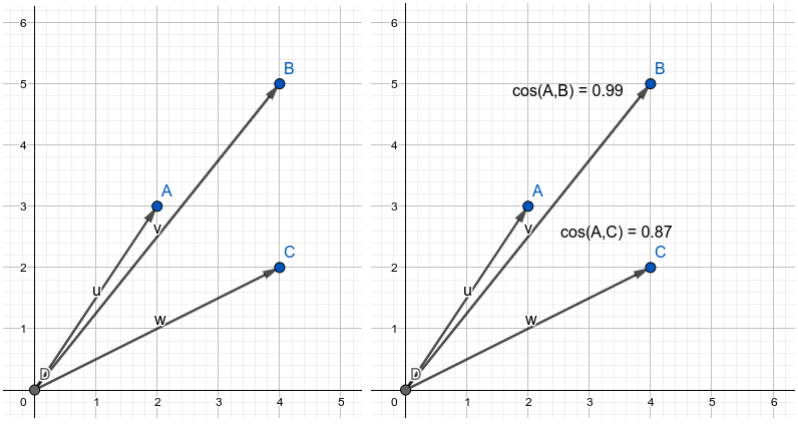

Figura 2.2: Visualización de usuarios en forma de vectores y cálculo coseniodal de sus representaciones vectoriales.

no existen ítems calificados en común para el par de usuarios. Reflejado gráficamente en la Figura 2.2, se puede ver que los posibles usuarios (vectores **A**, **B** y **C**) han evaluado en común 2 ítems (coordenadas **X** y **Y**). Se puede encontrar el usuario más parecido a **A** utilizando la similitud cosenoidal, el cual es **B** por tener un valor de similitud más cercano a 1.

$$Sim(u,v) = Cos(u,v) = \frac{\vec{u} \cdot \vec{v}}{||\vec{u}||_2 \times ||\vec{v}||_2} \tag{2.1}$$

Coeficiente de Correlación de Pearson (PCC). El coeficiente de correlación de Pearson se muestra en la Ecuación 2.2 y se define como la covarianza entre dos usuarios σ_{uv} entre la desviación estándar de estos σ_u y σ_v. El coeficiente de correlación de Pearson delimita la media de

estos conjuntos (μ_u y μ_v) a solamente los elementos utilizados en común. Sin embargo, en sistemas de recomendación es muy común utilizar la Ecuación 2.3, donde se utiliza directamente la media de los usuarios por simplicidad computacional [3]. Para el caso de una similitud entre un par de ítems, se tiene la Ecuación 2.4.

$$Sim(u,v) = PCC(u,v) = \frac{\sigma_{uv}}{\sigma_u \sigma_v} \tag{2.2}$$

$$Sim(u,v) = \frac{\sum_{i \in I_{uv}} (r_{ui} - \mu_u)(r_{vi} - \mu_v)}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - \mu_u)^2 \sum_{i \in I_{uv}} (r_{vi} - \mu_v)^2}} \tag{2.3}$$

$$Sim(i,j) = \frac{\sum_{u \in U_{ij}} (r_{ui} - \mu_i)(r_{uj} - \mu_j)}{\sqrt{\sum_{u \in U_{ij}} (r_{ui} - \mu_i)^2 \sum_{u \in U_{ij}} (r_{uj} - \mu_j)^2}} \tag{2.4}$$

Una característica útil de esta función de similitud, es el hecho de que su rango puede tener valores negativos el cual consta desde -1 a 1. Esto hace fácil de interpretar y de rechazar valores negativos. Para la selección de vecinos cercanos, se ha concluido que usuarios incluso con una fuerte correlación negativa, no aportan información útil en las recomendaciones [3]. En la Figura 2.3 se pueden ver los resultados del coeficiente de correlación de Pearson entre A y B así como entre A y C. Debido a que el ejemplo tiene solamente 2 coordenadas (ítems), las correlaciones entre los vectores están perfectamente definidas y muestra los valores límite para en el rango (1 para **A** y **B**; -1 para **A** y **C**).

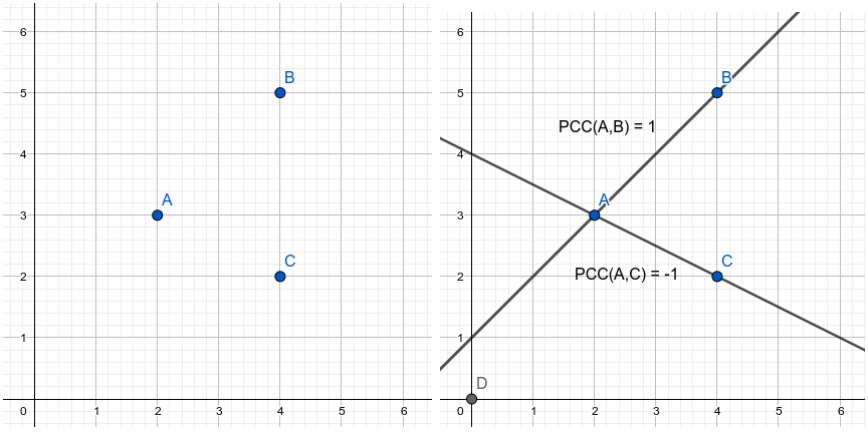

Figura 2.3: Visualización de usuarios en forma gráfica de puntos y Coeficiente de Correlación de Pearson de estos.

Distancia Euclidiana (EUC). EUC calcula la distancia euclidiana entre dos usuarios interpretándolos como puntos en un espacio multidimensional. Esto es gracias a la suposición de que las dimensiones de los usuarios son los ítems tanto calificados por el usuario u como por el usuario v. Gráficamente, cada usuario se puede representar como un punto en el espacio, siendo la distancia euclidiana la distancia más corta entre estos puntos calculado con la Ecuación 2.5. Como se aprecia en la Figura 2.4, la distancia euclidiana es más cercana consiste en el segmento desde **A** hacía **C**. En el caso de las similitudes cosenoidal y PCC los resultados muestran una mayor similitud para el par **A**, **B**.

$$EUC(u,v) = \sqrt{\sum_{i \in I_{uv}} (r_{ui} - r_{vi})^2} \tag{2.5}$$

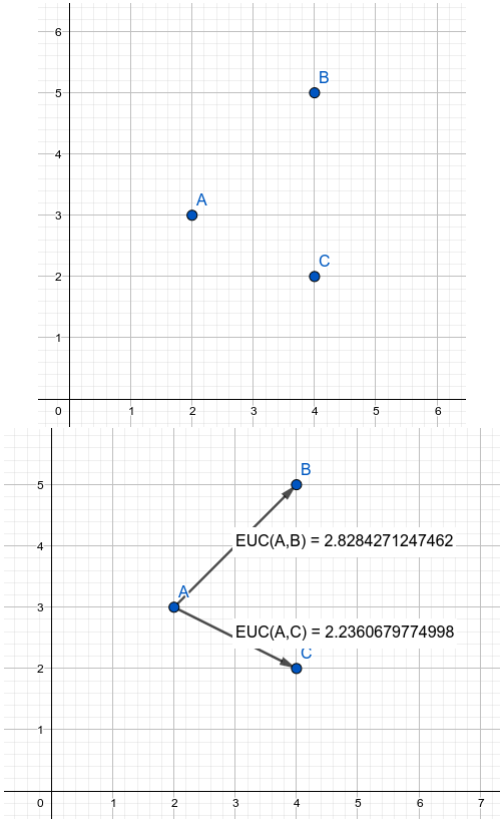

Figura 2.4: Visualización de usuarios en forma gráfica de puntos y distancia euclidiana de estos.

Como medida de similitud, es común utilizar un valor normalizado como es el caso de la Ecuación 2.6 [38]. El valor resultante tiene un rango de 0 a 1, siendo 1 un resultado para un par de elementos colocados en el mismo lugar. Esta medida de similitud ha mostrado muy buenos resultados, especialmente en evaluaciones basadas en clasificación [7] (precisión y exhaustividad, índices descritos posteriormente).

$$Sim(u,v) = \frac{1}{1 + \sqrt{\sum_{i \in I_{uv}} (r_{ui} - r_{vi})^2}} \tag{2.6}$$

Similitud Cosenoidal Ajustada (AC). Esta medida de similitud enfocada a ítems, tiene por objetivo acentuar la diferencia entre la media de los usuarios ya que son estos los que presentan mayor variación al efectuar las evaluaciones [55]. Esto puede verse en la Figura 2.5, en la cual se muestran los elementos de los vectores **A'**, **B'** y **C'** después de descontar la media de los vectores originales **A**, **B** y **C**.

Es posible construir vectores **i'** y **j'** basados en vectores de calificaciones hechas a los ítems i y j que permitan ajustarse al modelo de la Ecuación 2.1 para calcular una similitud cosenoidal. Esto se muestra en la Ecuación 2.8. La construcción de **i'** implica obtener las calificaciones de i con los usuarios en común con j, y restar a cada calificación la media de las calificaciones del usuario que otorgó la calificación. Esta construcción se refleja en la Ecuación 2.7.

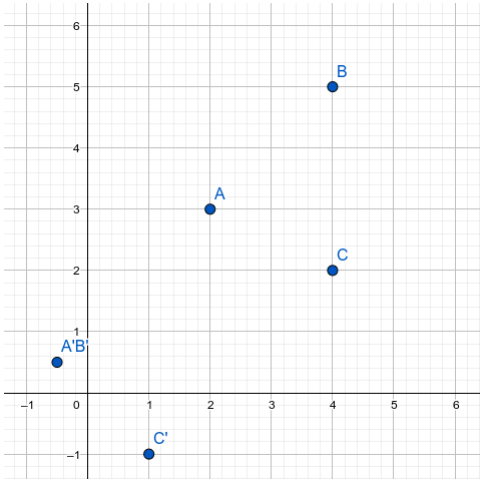

*Figura 2.5: Visualización de vectores en forma gráfica de puntos (**A**, **B** y **C**); visualización de los mismos vectores después de restar la media de los valores de estos (**A'**, **B'** y **C'**). Los elementos **A'** y **B'** se encuentran traslapados.*

$$i' = \{r_{ui} - \mu_u | u \in U_i \cap U_j\} \tag{2.7}$$

Como se puede ver con respecto a PCC en la Ecuación 2.4, en la Ecuación 2.9 (AC) existe un gran parecido ya que el fin es obtener una similitud entre dos ítems, sin embargo, en AC se utiliza la media de las calificaciones de los usuarios, mientras que en PCC se utiliza la media de las calificaciones hechas a los ítems. AC ha mostrado una mejora con respecto a PCC para ítems en varios casos [55].

$$Sim(i,j) = AC(i,j) = COS(i',j') \tag{2.8}$$

$$AC(i,j) = \frac{\sum_{u \in U_{ij}} (r_{ui} - \mu_u)(r_{uj} - \mu_u)}{\sqrt{\sum_{u \in U_{ij}} (r_{ui} - \mu_u)^2 \sum_{u \in U_{ij}} (r_{uj} - \mu_u)^2}} \tag{2.9}$$

Variantes en medidas de similitud. Una variante importante es la consideración de la robustez en el cálculo de la similitud. Si se piensa en dos usuarios con muy pocas calificaciones, es posible inferir en que estos son exactamente iguales por haber calificado uno o dos ítems de la misma manera, sin embargo, esto supone un problema ya que se necesita más información para llegar a estas conclusiones. Es por esto que se busca la manera de penalizar la similitud entre usuarios con pocas calificaciones en común, los métodos más utilizados son la implementación de un umbral [55] el cual afectará proporcionalmente la similitud de usuarios a menos que se sobrepase el número de ítems en común establecidos como puede verse en la Ecuación 2.10. Otro criterio es implementar un ajuste lineal que afecte en todos los casos la similitud entre usuarios, el cual afectará mayormente a los usuarios en medida que tengan un menor número de calificaciones. Esto se ve reflejado en la Ecuación 2.11. En las ecuaciones anteriores γ y β son constantes que afectan la similitud entre dos usuarios a manera de umbral y de forma lineal, respectivamente.

$$Sim'(u,v) = \frac{min(|I_{uv}|, \gamma)}{\gamma} \times Sim(u,v) \tag{2.10}$$

$$Sim'(u,v) = \frac{|I_{uv}|}{|I_{uv}| + \beta} \times Sim(u,v) \tag{2.11}$$

Estimación de calificaciones

Una vez determinada la similitud entre usuarios se tiene que considerar cómo estimar el interés de un usuario sobre un ítem. Como se mencionó en la sección de medidas de similitud, los usuarios aunque califiquen 2 ítems de la misma manera, no significa que opinen igual de éste, esto se debe a las variaciones existentes en las calificaciones de los usuarios. Es por esto la importancia de la normalización de los datos. Las principales normalizaciones para la predicción de calificaciones en sistemas de recomendación son la normalización centrada en la media y la normalización de predictores base [44].

Normalización centrada en la media. Lo que hace esta normalización, mostrada en la Ecuación 2.12, es restar la media μ_v de las evaluaciones de cada usuario v (vecino cercano de u que ha calificado el ítem i) en cada una de las evaluaciones al ítem i. Esto tiene como fin determinar el agrado del usuario v ya sea positivo o negativo con respecto al ítem i. El término $\sum_{v \in N_i(u)} |Sim(u,v)|$ es utilizado para ajustar los resultados a valores cercanos a los evaluados por los usuarios, en caso de utilizar las predicciones para la recolección de una lista de ítems recomendados se puede omitir el término mencionado [13].

$$\hat{r}_{ui} = \mu_u + \frac{\sum_{v \in N_i(u)} Sim(u,v) \times (r_{vi} - \mu_v)}{\sum_{v \in N_i(u)} |Sim(u,v)|} \tag{2.12}$$

Normalización de predictores base. Los predictores base o sesgos consisten en aquellas tendencias sistemáticas de los usuarios para calificar ítems o de ítems al ser calificados que distorsionan la búsqueda real de las preferencias de los usuarios [44]. La normalización de predictores base elimina estos efectos considerando tanto a los sesgos de evaluación de los usuarios b_u, a los sesgos de las evaluaciones de los ítems b_i y la media general de las calificaciones μ, lo cual se puede apreciar en la Ecuación 2.13. La predicción se hace de acuerdo a la Ecuación 2.14.

$$b_{ui} = \mu + b_u + b_i \tag{2.13}$$

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{v \in N_i(u)} Sim(u,v) \times (r_{vi} - b_{vi})}{\sum_{v \in N_i(u)} |Sim(u,v)|} \tag{2.14}$$

Si bien el proceso de búsqueda de estos predictores resulta más complejo que los parámetros de las normalizaciones anteriores, suele tener mejores resultados. Las principales metodologías para estimar estos predictores consisten en el uso de mínimos cuadrados alternados (ALS) [9] y gradiente descendiente estocástico (SGD) [44] para minimizar la función objetivo mostrada en la Ecuación 2.15. El término λ es una constante

utilizada para eliminar el sobre ajuste.

$$min_{b_*} \sum_{u,i \in R} (r_{ui} - \mu - b_u - b_i) + \lambda(\sum_u b_u^2 + \sum_i b_i^2) \tag{2.15}$$

Selección del número de vecinos k. El número de vecinos que se consideren para cada estimación aumentará o reducirá la exactitud del sistema de recomendación, esto se representa en una gráfica de manera cóncava como se puede ver en la Figura 2.6. La mejor manera de escoger la k óptima para nuestro conjunto de datos, es utilizar la validación cruzada [55]. En caso de buscar una mayor taza de elementos no esperados en las recomendaciones, se puede optar por disminuir el valor de k, a pesar de la disminución de la exactitud en el sistema de recomendación [55].

2.2.2. Filtrado colaborativo basado en modelos

Los métodos basados en modelos capturan las características de los ítems y usuarios. Dichas características son aprendidas en el entrenamiento del modelo y se utilizan para predecir recomendaciones [55]. Las propuestas más utilizadas se basan en métodos de factorización de matrices, sin embargo, para la estimación de ítems no es poca la cantidad de métodos existentes basados en modelos. La meta de los sistemas de

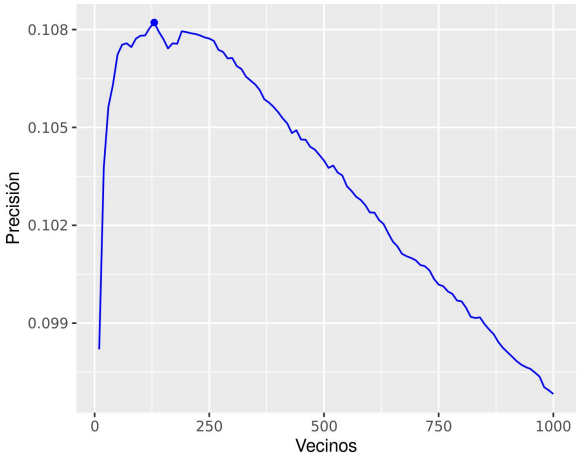

Figura 2.6: Precisión vs número de vecinos en ML1M, óptimo en k=130 basado en [28].

recomendación basados en modelos es generalizar estos conocimientos aprendidos del entrenamiento para la predicción de calificaciones [44]. Algunas ventajas de los métodos basados en modelos son la eficiencia tanto en espacio como en la rapidez de la creación de los modelos. En muchos casos no es necesario trabajar con una matriz de usuarios por ítems para aplicar los modelos lo que reduce significativamente el costo de almacenamiento de los datos. Ejemplo de esto son los métodos de reducciones de matrices [3].

Gran parte del desarrollo de métodos basados en modelos en los sistemas de recomendación se debe a la consideración de que la representación del problema del filtrado colaborativo (realizar recomendaciones de ítems para usuarios) es similar a los problemas de clasificación [3]. El

desarrollo de diversos modelos pensados para la clasificación es un problema bastante estudiado. Identificar las diferencias entre un sistema de recomendación y la clasificación permite hacer los cambios pertinentes en los diversos modelos para clasificación con el fin de obtener modelos enfocados a los sistemas de recomendación.

Modelos de factorización de matrices

Al tener una matriz de evaluaciones de usuarios a ítems M es posible representar a cada usuario como un vector de manera que cada ítem represente una dimensión en éste. En los diversos métodos de factorización de matrices se propone el uso de matrices alternas, normalmente de dimensiones menores, que puedan representar a la matriz M. Los atributos son de caracter abstracto, es decir que no están directamente relacionados a un solo ítem, sino a la relación que hay con todo el conjunto de ítems. Con estos métodos se esperan implementaciones efectivas de los algoritmos con una mayor exactitud, ya que se están eliminando las partes no significativas para el tratamiento de los datos [44]. Las principales ventajas de esta transformación de la matriz consiste en que la información parcial contenida en M permite construir matrices sin valores vacíos y que éstas pueden ser utilizadas para una rápida predicción de evaluaciones.

Segmentación de usuarios

La segmentación de usuarios es comúnmente utilizada en los sistemas de recomendación demográficos. Sin embargo, en los algoritmos de filtrado colaborativo la segmentación representa un gran beneficio en la eficiencia de diversos métodos. La práctica más común es utilizar algún método de segmentación o clasificación no supervisada (clusterización) para el cálculo de los vecinos en métodos basados en vecindarios. Para mostrar la ventaja de la segmentación en sistemas de recomendación suponga segmentar un conjunto de n usuarios en cuatro subgrupos de igual tamaño, las matrices de similitud generadas serán de un tamaño de $n^2/16$ que significa una reducción de uso de memoria y de costo computacional de cuatro veces. A pesar de esto, existe una ligera pérdida de exactitud en la clusterización ya que se está descartando información para el análisis.

Algunos ejemplos de la segmentación se encuentran en el trabajo de Cui [19] donde se utiliza una variación del método de clusterización estándar k-means con el fin de mejorar la selección inicial de vectores con el método "cuckoo search" [74]. Esta variación tiene por nombre csk-means. El método cuckoo search, es un método de optimización basado en partículas, el cual supone que en el espacio de búsqueda los pajaritos (*cuckoos*) colocan huevos en nidos colocados en posiciones aleatorias, para siguientes generaciones prevalecen las soluciones con mejores huevos (mejores soluciones). En caso de que un pajarito encuentre su nido, éste

puede colocar un nuevo nido con otro huevo en él. La aleatoriedad en el desplazamiento de los pajaritos está determinada por la distribución de Lévy y la distribución Normal.

Con su propuesta han conseguido una mejora del 5.2 % en la precisión del modelo en comparación con propuestas base de segmentación de usuarios.

2.3. Unidades de Procesamiento Gráfico GPU

Un factor importante para mejorar el rendimiento de futuros sistemas de recomendación es el uso de hardware especializado para el cómputo en paralelo como las unidades de procesamiento gráfico GPU (acrónimo de *Graphics Processing Unit*). La tarea de paralelización ya es posible con la mayoría de los procesadores actuales, sin embargo, estos solo suelen encontrarse en versiones de hasta 64 núcleos, en el caso de los modelos de mayor costo.

A continuación vamos a explicar las principales propiedades así como los conceptos básicos de programación de una tarjeta gráfica típica de la familia Nvidia. Las GPU son unidades de procesamiento en paralelo que se caracterizan por tener una gran cantidad de núcleos de procesamiento de poca potencia en comparación con el procesador principal. Esto permite a las GPU's realizar tareas de forma distribuida en esa gran cantidad de núcleos.

Para el caso de los sistemas de recomendación es común tratar con información de miles de usuarios e ítems por lo que la posibilidad de dividir la tarea de otorgar recomendaciones entre miles de partes para que cada una sea procesada por algún núcleo de la tarjeta gráfica resulta en ahorros significativos de tiempo incluso contando los tiempos de transferencia de datos desde la memoria RAM de la computadora hasta la memoria de la tarjeta gráfica. Existe una gran cantidad de métodos que pueden paralelizarse en sistemas de recomendación, algunos ejemplos son el método estándar de vecindarios basados en ítems o usuarios [73]; sistemas de recomendación basados en clusterización [6]; y métodos basados en optimización de características para sistemas de recomendación basados en modelos [56], [62].

2.3.1. Construcción a nivel hardware

La GPU de Nvidia consiste en un arreglo de Multiprocesadores SM (*Streaming Multiprocesors*), los cuales son un conjunto de núcleos de procesamiento con memoria compartida. En la Figura 2.7 se pueden ver las diferencias de los núcleos convencionales en CPU en contraste con los Multiprocesadores en una GPU. Principalmente, se tiene que cada fila en la parte de la derecha (GPU) representa un multiprocesador el cual contiene varios núcleos además de su unidad de control y de su memoria interna. En el caso de la CPU, cada núcleo tiene su memoria y su unidad de control.

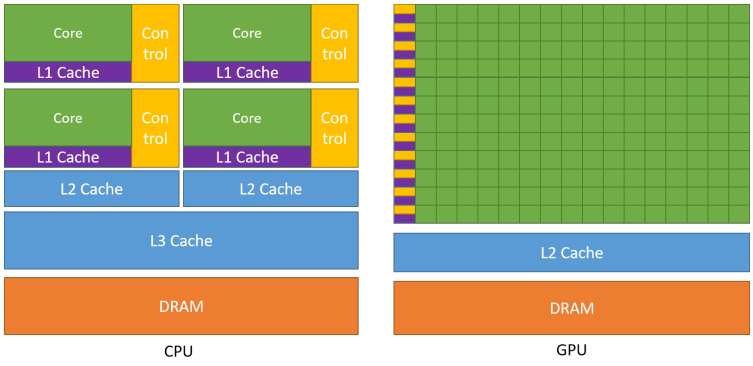

Figura 2.7: Comparación a nivel de hardware CPU vs GPU. Fuente [31].

Mientras que un hilo es una secuencia de operaciones a ejecutar, un bloque de hilos es un arreglo de una, dos o tres dimensiones de hilos de procesamiento que se ejecuta en un multiprocesador. El multiprocesador puede ejecutar varios bloques de hilos simultáneamente. Para esto crea, administra, agenda y ejecuta bloques en paquetes de 32 hilos y son llamados *warp*. El multiprocesador ejecuta cada *warp* con la misma serie de instrucciones. En caso de que exista divergencia en una parte de hilos por alguna condición, el *warp* deshabilita los demás hilos para ejecutar las instrucciones particulares. Al terminar, el *warp* vuelve a habilitar todos los hilos para continuar con la ejecución simultanea [31]. El número de núcleos existente en cada multiprocesador así como el número de *warp* que puede ejecutar concurrentemente lo define la capacidad de cómputo de la tarjeta, la cual es un estándar de diseño en la arquitectura de las GPU [31].

La cantidad de *warps* generados para cada bloque de hilos, se define

con la siguiente función:

$$N = \lceil T/32 \rceil$$

Donde T es la cantidad de hilos por bloque, w es el cantidad de hilos por warp, es decir, 32 y tanto "⌈" como "⌉"son símbolos que indican la función techo, la cual redondea el número con el primer entero igual o mayor a éste. A continuación se proporciona un ejemplo para 50 hilos del uso de la función techo.

$$N = \lceil 50/32 \rceil = 2$$

Para una gran cantidad de problemas es común utilizar el término *grid* o malla para determinar el número y estructura de los bloques de hilos a ejecutar. La malla, al igual que los bloques de hilos se pueden definir en arreglos de una, dos o tres dimensiones.

2.3.2. Ley de Amdahl en optimizaciones con GPU

Dentro del análisis y explicación de las mejoras con GPU vamos a introducir el concepto de la Ley de Amdahl, la cual permite tener un conocimiento más exacto de las mejoras implementadas así como ver niveles teóricos de mejora obtenibles.

En la mayoría de los casos al implementar una mejora para la opti-

mización de tiempo ya sea basada en software o en hardware, en solo en una parte del proceso se aplican las reducciones de tiempo mientras que el resto opera de la misma manera. Con la paralelización mediante GPU nos estamos limitando a mejorar las partes paralelizables de nuestro algoritmo, por lo que resulta útil evaluar el porcentaje de mejora real del sistema. De acuerdo con Hill et al. [36], la Ley de Amdahl, mostrada en la Ecuación 2.16, establece que una mejora en una porción f del proceso de cómputo, al mejorarse en una razón m, genera un rendimiento general M a razón de:

$$M = \frac{1}{(1-f) + \frac{f}{m}} \tag{2.16}$$

Es de gran importancia prestar atención a la parte no mejorable $(1-f)$ la cual define el limite de mejora, es decir, incluso al mejorar la parte de computo f varios ordenes de magnitud, si f solo representa una parte pequeña del proceso no es significativa la mejora. En métodos de sistemas de recomendación de filtrado colaborativo basados en vecindarios las partes no paralelizables se encuentran en la lectura de los datos así como en la muestra de resultados. Adicionalmente, tenemos trabajo adicional al hacer copia de datos desde la memoria de acceso aleatorio RAM del ordenador a la RAM de la tarjeta gráfica y vice versa.

Capítulo 3

Estado del arte

En este capítulo se muestran los avances recientes que han permitido el desarrollo de esta investigación. Las secciones están separadas por categoría para facilitar el desglose de los temas. En este capítulo no se intenta dar explicaciones extensas de los métodos mencionados, sino que se ilustran las principales propiedades de estos métodos.

3.1. Métodos basados en vecindarios

Los métodos basados en vecindarios han sido altamente estudiados por su buen rendimiento, simpleza conceptual y la facilidad de implementarlos dentro de los sistemas de recomendación. En los principales precedentes se encuentra es estudio de Linden en 2003 [45] mostrando

los beneficios del filtrado colaborativo. Similarmente se tiene la investigación de Deshpande y Karypis en 2004 [22] quienes se enfocan en las aplicaciones de recomendación de hasta N ítems a los usuarios con técnicas de filtrado colaborativo basado en ítems. Muestran resultados muy favorables al comparar los tiempos de ejecución de los métodos basados en ítem con los métodos basados en usuarios. Para temas comparativos, vamos a nombrar **Item-KNN** al método descrito por Deshpande y Karypis en futuras menciones del método.

En 2018, Chae, et al. [13] hacen una propuesta basada en vecindarios para ítems con la particularidad de tener una nueva estructura de datos que permite almacenar y calcular las similitudes entre usuarios e ítems de una manera más eficiente en términos de memoria. En esta investigación se comparan extensivamente los métodos basados en vecindarios con los métodos de factorización de matrices. Los autores concluyen con que una muy poca concentración de datos (poca densidad) afecta el rendimiento de los métodos basados en la reducción de matrices. Otros resultados relevantes de los autores es el mejor rendimiento de los métodos basados en ítems con respecto a los métodos basados en usuarios. En posteriores menciones de este método, vamos a nombrarlo como **User-KNN-SP** para el caso de la versión basada en usuarios e **Item-KNN-SP** para la versión basada en ítems, donde la notación **-SP** indica su principal aplicación que es la aplicación en matrices con poca densidad de datos (*sparce*).

En 2019, Jeunen, et al. [43] hacen una mejora en el cálculo de la similitud entre usuarios para métodos de vecinos más cercanos basado en retroalimentación implícita con el método **Dynamic-Index**. Las mejoras que proponen son la reducción del cálculo de relaciones entre ítems y usuarios. Esto se logra al considerar en las recomendaciones el uso temporal o estacionario de algunos ítems (por ejemplo, la vestimenta de temporada la cual solo se debe considerar en ciertos periodos de tiempo). Adicionalmente, en la propuesta es posible agregar dinámicamente ítems o usuarios sin un reentrenamiento total del sistema de recomendación. En el método se obtiene una reducción en la complejidad computacional del cálculo de la matriz de similitud con la ayuda de este nuevo algoritmo de $\mathrm{O}(mp^2q)$ a $\mathrm{O}(Rp)$; R es la cardinalidad del conjunto de evaluaciones validas de los usuarios. Finalmente, se logra una reducción de los tiempos de ejecución con la paralelización de el algoritmo con el paradigma Map-Reduce [21] en procesadores de varios núcleos.

3.2. Aplicaciones con GPU

Un antecedente importante con el uso de GPU en sistemas de recomendación es el estudio de Wang et al. en 2016 [73]. El estudio muestra una sustancial mejora de tiempos de ejecución en la implementación básica de un sistema de recomendación de filtrado colaborativo basado en vecindarios con el uso de múltiples tarjetas gráficas. A pesar de

mostrar una mejora impresionante (mayor a 3600 veces más rápido), el algoritmo solamente se compara con el método estándar **Item-KNN** poco utilizado por su alto costo computacional $O(m^2n)$ que paralelizado se obtiene una complejidad de $O(m^2n/h)$ donde h es el número de hilos utilizados en la paralelización. Aún así, el autor demuestra la gran capacidad de paralelización del algoritmo ya que al aumentar el número de tarjetas gráficas, el rendimiento obtenido aumenta linealmente. Para futuras menciones vamos a nombrar este método como **Item-KNN-GPU**.

Más allá de los sistemas de los métodos basados en vecindarios, se tienen resultados significativos por parte del equipo de desarrollo de Nvidia. En la competencia anual del congreso anual de sistemas de recomendación RecSys Nvidia se ha posicionado con el primer puesto en 2020 [62]. Sus aportes consisten en el uso de ingeniería de características para la evaluación de múltiples modelos de sistemas de recomendación, de los cuales XGBoost [16] ha sido el que ha mostrado mejores resultados. Esto es posible gracias al uso de tecnología basada en GPU para la implementación rápida de los modelos. Su implementación resultó ser 25 veces más rápida que el método optimizado con un CPU Intel Xeon de 20 núcleos. Su configuración consistió en el uso de 4 tarjetas gráficas V100.

Atasu, et al. en 2017 consiguen un tiempo de respuesta 15 veces menor que el estado del arte en el conjunto de datos de Netflix de 100 mi-

llones de evaluaciones [6]. Los autores utilizan un método de balanceado de carga para segmentar en un cluster de GPU's el método de factorización de matrices. El método utilizado esta basado en OCuLaR [35]. Para este proceso se utilizaron 16 tarjetas gráficas Nvidia P100.

3.3. Marcos de trabajo

En 2011, Gantner, et al. proponen Mymedialite [26], un Marco de trabajo basado en el lenguaje C#, sin embargo puede ser llamado desde python o Ruby. Para la predicción de evaluaciones, utiliza variantes de KNN, sencillos métodos base y métodos de factorización de matrices. Sus métodos aceptan retroalimentación explicita e implícita. Ya que los usuarios son difíciles de persuadir para que otorguen evaluaciones explicitas, se cuenta también con otros tipos de retroalimentación, (que incluyen las acciones de los usuarios en una página, vistas o número de clics).

Por otro lado, en el aspecto de aplicaciones para comercio electrónico tenemos el estudio de Geuens, et al. en 2018 [27], quienes utilizan diversas configuraciones en algoritmos de filtrado colaborativo para encontrar un balance adecuado entre exactitud, diversidad de recomendaciones y tiempo de cómputo. Algunos parámetros que utilizan son métodos de compresión de información, método de filtrado colaborativo (basado en ítems o basado en usuarios) y cálculo de similitud (COS, PCC, Jacard).

Sus resultados de acuerdo al método, sin embargo, destacan el uso de métodos basados en ítems con análisis de correspondencia [30] (solo aplicable para retroalimentación binaria [27]).

En 2020, el equipo de desarrollo de Nvidia presenta Merlin [56], un marco de trabajo enfocado específicamente en el uso de tarjetas gráficas para sistemas de recomendación de redes neuronales profundas. Merlin se compone de tres elementos: Una herramienta de preprosesamiento de datos y de ingeniería de características; un repositorio de métodos de sistemas de recomendación basados en aprendizaje profundo; por último, una herramienta de inferencia que selecciona y ordena ítems de acuerdo a la probabilidad de que el usuario interactue con dichos ítems [56]. Los métodos contemplados en este marco de trabajo son Wide & Deep [17], DeepFM [33], DLRM [53], Deep & Cross Networks [72].

3.4. Factorización de matrices

A continuación vamos a describir tres métodos altamente utilizados de factorización de matrices, Descomposición de Valores Singulares SVD, Mínimos Cuadrados Alternados ALS y el filtrado colaborativo de una clase OCCF.

Reducción de valores singulares SVD.

La implementación de este método consiste en utilizar la matriz M

de usuarios e ítems de tamaño $n \times m$ para obtener 3 matrices P, S, Q de tamaños $n \times m$, $m \times m$ y $m \times n$ que cumplan con la propiedad vista en la Ecuación 3.1.

$$M = P \times S \times Q \tag{3.1}$$

Definición vectores singulares y valores singulares. Se dice que para una matriz A, un vector v y un número λ son un vector singular y un valor singular respectivamente de A si cumplen con la Ecuación 3.2 y $v \neq 0$ [67]. Los vectores singulares y valores singulares son conocidos también como vectores característicos y valores característicos.

$$A \times v = \lambda \times v \tag{3.2}$$

En sistemas de recomendación, cualquier vector $p_u \in P$ representaría un conjunto de rasgos distintivos de los usuarios. De manera similar, $q_i \in Q^T$ son vectores singulares de los ítems. La predicción del gusto de un usuario u hacia un ítem i se puede ver en la Ecuación 3.3 [32].

$$\hat{r}_{ui} = q_i^T p_u \tag{3.3}$$

La ventaja de este método es que al disminuir las dimensiones de P, S y Q a tamaños $n \times d$, $d \times d$ y $m \times d$, respectivamente, se tiene la mejor

aproximación para la matriz M utilizando matrices de dimensionalidad d.

Los orígenes de SVD en los sistemas de recomendación son propuestos por Billsus [11]. Algunas propuestas de SVD consisten en utilizar métodos de mínimos cuadrados o descenso de colinas estocástico [32, 44] al tratar SVD como un problema de minimización.

Mínimos cuadrados alternados.

La técnica de mínimos cuadrados alternados, o bien, ALS por sus siglas en inglés, es un método utilizado en sistemas de recomendación tanto para la estimación de recomendaciones con la generación de las matrices P y Q descritas en el método SVD o bien como ayuda en la normalización de predictores base de métodos basados en vecindarios.

El método ALS obtiene las matrices P y Q encontrando una matriz P' óptima que minimice la Ecuación 3.4 dada una matriz Q' generada aleatoriamente para posteriormente iterar y resolver el mismo problema de forma que se encuentre una matriz Q'' óptima dada la matriz P'. El término λ consiste en una constante para evitar el sobreajuste. Lo descrito con anterioridad se describe en el Algoritmo 1. Resolver el problema parcialmente para solo una de las matrices a la vez permite una solución eficiente mediante el uso de mínimos cuadrados [9].

$$\min_{p*,q*} \sum_{u \in U} \sum_{i \in I} (r_{ui} - p_u^T q_i)^2 + \lambda(||p_u||^2 + ||q_i||^2) \tag{3.4}$$

Algoritmo 1: Método mínimos cuadrados alternados ALS [68].

Entrada: R
Salida : P, Q

```
Q ← matriz rellena con pequeños valores aleatorios.
for número determinado de iteraciones do
    Calcula P que minimiza Ecuación 3.4 dada Q.
    Calcula Q que minimiza Ecuación 3.4 dada P.
end
```

Filtrado Colaborativo de Una Clase (OCCF).

En lo que respecta a los sistemas de recomendación con información implícita se tiene el método de Filtrado Colaborativo de Una Clase [57]. Este método utiliza como matriz de entrada M la información implícita registrada con el valor de uno. En caso de no existir información de que un usuario halla mostrado interés en un ítem en la casilla correspondiente en la matriz se infiere una valoración negativa de cero.

La predicción de recomendaciones se consigue con SVD al resolver con ALS una versión modificada la Ecuación 3.4. La nueva función a minimizar considera también el nivel de confianza W_{ui} al determinar M_{ui}. En caso de existir información del par usuario-ítem, se tiene que $W_{ui} = 1$ en caso contrario se infiere que el peso del usuario W_{u*} es proporcional al número de ítems que el usuario u ha calificado. Esto se debe

a que en un enfoque orientado al usuario, en medida que este muestra interés en un mayor número de ítems, se puede suponer desagrado en los ítems restantes con mayor certeza. La función objetivo se muestra en la Ecuación 3.5 en la cual $||x||_F$ es la normalización Frobenius.

$$\min_{p*,q*} \sum_{u \in U} \sum_{i \in I} W_{ui}(r_{ui} - p_u^T q_i)^2 + \lambda(||p_u||_F^2 + ||q_i||_F^2) \tag{3.5}$$

3.5. Discusión del uso del aprendizaje profundo

En 2018 Dacrema, et al. [20] analizaron 18 propuestas para sistemas de recomendación basadas en técnicas de aprendizaje profundo (DL por sus siglas en inglés). La intención de los autores fue analizar la reproducibilidad y la exactitud de dichas técnicas comparándolas con las técnicas convencionales basadas en memoria con un alto nivel de ajuste en los hiper-parámetros[1] de los métodos. Los métodos convencionales lograron superar las técnicas basadas en DL con los conjuntos de datos y medidas mostradas en las mismas propuestas en la mayoría de los casos (seis de siete casos).

Jannach, et al. en 2020 discuten posibles causas por las cuales los métodos basados en DL no han mostrado resultados significativos en las competencias de sistemas de recomendación a pesar de tener un vasto

[1] Los hiper-parámetros son el conjunto de constantes modifican el comportamiento de los algoritmos en los que se encuentran (por ejemplo, el número de iteraciones en algoritmos iterativos).

desarrollo en la investigación científica [42]. Parte de sus análisis incluye los conjuntos de datos, el perfil de los participantes o las particularidades de los procesos de evaluación. Concluyen en la importancia de no olvidar la línea base de los sistemas de recomendación en la investigación científica y destacan la relevancia de los concursos al mostrar problemas reales y necesidades actuales de la industria a la comunidad científica.

En cuanto a la obtención de buenas precisiones se tiene a Chen, et al. en 2018 [15], quien muestra una perspectiva muy clara de qué ha sido de los métodos de filtrado colaborativo desde los métodos básicos hasta los métodos híbridos. El estudio de Chen considera también extensiones en las medidas de similitud, aspectos sociales, avances en reducción de dimensionalidad, clusterización y reducción de matrices. El estudio concluye en el análisis de la confianza (trust) como una medida muy relévate hoy en día así como la necesidad de utilizar más técnicas de DL para resolver problemas como la amplia dispersión de datos y el inicio en frío visto en los sistemas de recomendación con técnicas convencionales.

3.6. Extensión de las capacidades del método

A continuación se mencionan los aportes que se han hecho con la intención de aumentar la exactitud de los métodos a implementar. El objetivo de este apartado es mostrar las capacidades de la paralelización de adaptarse a propuestas que no fueron contempladas en la descripción

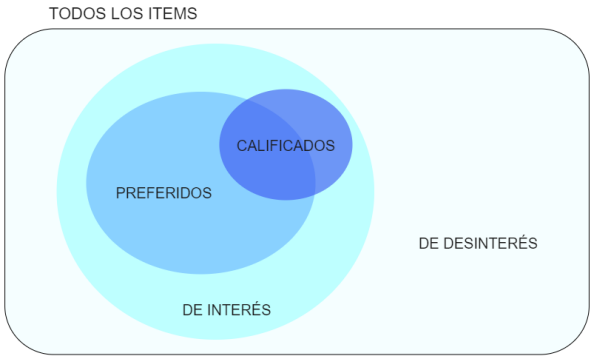

Figura 3.1: Diagrama de Venn, mostrando desinterés, interés y preferencia de ítems. Hecho por [37].

de los métodos. Si bien es de gran utilidad aplicar los métodos de recomendación de forma eficiente, es importante mostrar también que los métodos más eficientes funcionen bajo situaciones cambiantes ante la naturaleza de mejoras en la precisión del método. Una mejora ya descrita es el uso de **correcciones por penalización** ante usuarios con muy pocas calificaciones (Ecuación 2.10 y Ecuación 2.11). En esta sección vamos a describir otra clase de mejoras que puedan aplicarse a los métodos en general como es el caso de la matriz de inserción de ceros (***Zero-Injection Matrix***) [37].

Este método consiste en el análisis de ítems que sean de desinterés para el usuario. Esto se hace basado en la premisa de que los usuarios solo califican una parte pequeña de los ítems que son de interés para ellos, por lo tanto, encontrar y descartar los ítems de desinterés mejora sustancialmente la efectividad de las recomendaciones [37]. El método se

describe en el Algoritmo 2, donde R es el conjunto de recomendaciones de la forma (usuario u, ítem i, calificación r_{ui}) y θ es un nivel mínimo de interés sobre el cual vamos a estimar los elementos de desinterés. La función matrizBinaria(R) genera una matriz que se construye con 1 en caso de existir una calificación en la posición (u, i) de R, nulo en caso contrario. La matriz generada B es de $n \times m$ elementos. Posteriormente, la función OCCF(B) trabaja con la matriz anteriormente generada. OCCF genera la matriz $P \in \mathbb{R}^{n \times m}$, los valores de cada elemento de la matriz están en un rango de entre cero y uno. La lista de desinterés L_{Desint} se calcula con el umbral θ comparado con las estimaciones obtenidas de cada elemento p_{ui} de P, en todos los casos tienen la calificación mínima. La naturaleza de los elementos de Z ya sea en forma de un conjunto o convertidos posteriormente en una matriz permiten que se pueda implementar cualquier sistema de recomendación que acepte una matriz con base en R. Como se puede ver, el pilar del Algoritmo 2 consiste en la generación de un conjunto de calificaciones Z, el cual incluye los elementos contenidos en R así como todos los elementos de desinterés para cada usuario con una calificación de cero L_{Desint}.

Algoritmo 2: *Zero-Injection Matrix* [37].

Entrada: R, θ
Salida : Z

1 $B \leftarrow matrizBinaria(R)$
2 $P \leftarrow OCCF(B)$
3 $L_{Desint} \leftarrow L_{Desint} \cup \{(u, i, 0)\} \ \forall \{(u, i) \mid p_{ui} \in P \text{ AND } p_{ui} < \theta\}$
4 $Z \leftarrow R \cup L_{Desint}$

Capítulo 4

Descripción de la propuesta

La propuesta se ha dividido en 4 secciones. La Figura 4.1 ilustra esta división. En ella, se pueden apreciar las etapas del esquema metodológico así como los elementos relevantes de la investigación. La Sección 4.1 consiste en mostrar los motivos por los cuales se ha llegado a la decisión del uso de métodos basados en memoria dentro de las técnicas de filtrado colaborativo. En la Sección 4.2 se describen detalladamente los métodos con los que se trabajará. Por último en la Sección 4.3 se detallan partes de la implementación de los métodos comparativos y algoritmos basados en GPU así como la metodología para la paralelización.

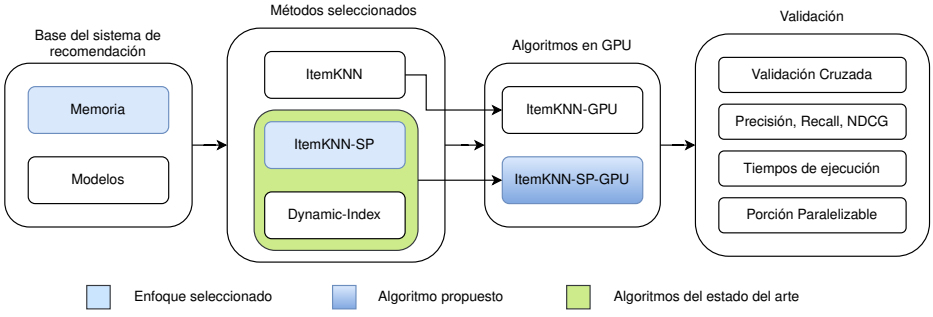

Figura 4.1: Partes principales del esquema metodológico.

4.1. Base del sistema de recomendación

Para la delimitación de métodos solo se consideraron métodos de filtrado colaborativo y se analizaron diversas técnicas basadas tanto en memoria como en modelos. Ya que los métodos basados en memoria son los que ofrecen las recomendaciones más rápidamente, el método a optimizar será un algoritmo de filtrado colaborativo de este tipo. Como ya se ha visto en [20], se pueden obtener resultados muy aceptables en los sistemas de recomendación basados en memoria al compararse con métodos de DL.

Filtrado colaborativo basado en usuarios contra filtrado colaborativo basado en ítems

En general, la cantidad de usuarios suele ser mucho mayor que la cantidad de ítems. Así, la matriz de similitud ítem a ítem se representa

con una menor cantidad de memoria y contiene una mayor concentración de datos.

Tabla 4.1: Costo computacional de implementaciones basadas en usuarios y en ítems para métodos de vecindarios. Fuente: [55].

		Tiempo	
	Espacio	Entrenamiento	En línea
Basado en usuarios	$O(n^2)$	$O(n^2p)$	$O(mk)$
Basado en ítems	$O(m^2)$	$O(m^2q)$	$O(mk)$

Ning [55] considera cinco aspectos a la hora de seleccionar una implementación basada en usuarios o basada en ítems, los cuales son: exactitud, eficiencia, estabilidad, jutificabilidad, serenidad. Debido a la mayor cantidad de usuarios que ítems, buscar similitudes entre usuarios implica que éstos tendrán menos información por usuario en contraste con los ítems, por lo que las similitudes entre los ítems suelen proporcionar una mayor exactitud [55]. Aggarwal le atribuye una mejor exactitud en las similitudes entre ítems a que la información de los intereses de los usuarios se utiliza directamente para la búsqueda de otros ítems. En el caso de similitudes de usuarios se tiene que confiar en la elección de éstos para las recomendaciones [3]. En cuanto a eficiencia, un sistema basado en ítems suele tener un menor espacio en memoria, así como un menor tiempo de entrenamiento tal como se indica en la Tabla 4.1. En dicha tabla n y m son el número de usuarios y de ítems respectivamente, p es el número máximo de ítems que ha evaluado un usuario, q es el número máximo de usuarios que han evaluado un ítem y k es el número de vecinos a considerar en la solución del problema. Dado que hay

una mayor concentración de recomendaciones por ítem, agregar ítems nuevos no es una situación que se presente tan seguido como agregar usuarios nuevos por lo que el cálculo de la matriz de similitud se puede efectuar en intervalos más largos de tiempo [3]. La justificabilidad es otro punto a favor de un sistema basado en ítems ya que a diferencia el sistema basado en usuarios, en éste se pueden describir de manera más clara los orígenes de las recomendaciones (por ejemplo, se puede crear una sección descrita como: "objetos similares a ..."). Por contraparte, los sistemas basados en usuarios tienen una mayor serenidad. La serenidad consiste en la tendencia a mostrar ítems fuera de los intereses mostrados hasta ahora por el usuario. Esto es debido a que muestran una mayor diversidad de ítems y a que las similitudes entre usuarios no son tan rígidas como las similitudes entre ítems [55]. Aggarwal [3] hace un análisis del mismo tipo, llegando a las conclusiones similares.

En algunas aplicaciones, como es el caso de repositorios de archivos, el tamaño del conjunto de ítems resulta ser bastante mayor al tamaño del conjunto de usuarios. Para estos casos se debe considerar que dichas implementaciones tendrán efectos contrarios a los descritos en algunos aspectos. Las propiedades mostradas en esta sección destacan mayor justificabilidad y exactitud por parte de los métodos de vecindarios basados en ítems con respecto a los métodos de vecindarios basados en usuarios. Se trabajará con los métodos basados en ítems para este trabajo de investigación debido a estas razones.

Se plantea el uso de el método estándar **Item-KNN** mostrado por Deshpande en 2004 [22], **Item-KNN-SP** para conjuntos esparcidos [13], **Dynamic-Index** para evaluaciones con información implícita [43] y con sus versiones basadas en el uso de técnicas de GPU, con los conjuntos de datos ya mostrados. En este diseño experimental nos enfocamos en el uso de técnicas basadas en vecindarios debido a su mejor comportamiento con conjuntos de datos altamente dispersos, a diferencia de los modelos de descomposición de matrices [13]. En nuestro análisis vamos a excluir también a los modelos con técnicas avanzadas de inteligencia artificial, ya que no se han encontrado grandes mejoras de exactitud a pesar del aumento en tiempo de ejecución [20].

4.2. Métodos seleccionados

A continuación, se dará una detallada explicación de los métodos a implementar. El método **Item-KNN** consta de dos partes principales, la selección de los k vecinos más cercanos como se puede ver en el Algoritmo 3 y la selección de los principales ítems para recomendar a un usuario como se aprecia en el Algoritmo 4. En el Algoritmo 3 se toma como entrada la matriz de calificaciones de ítems hechas por usuarios R de tamaño $m \times n$ (donde m es el número de ítems y n es el número de usuarios) y el número de vecinos k. El cálculo de la similitud se puede hacer con cualquiera de las funciones descritas en la Sección 2.2.1. El algoritmo calcula el valor de la similitud de todos los ítems contra todos

los ítems en la Línea 5 y coloca los resultados en una matriz de tamaño m x m. En el caso de hacer la comparación del mismo ítem, su valor será la mínima posible como se muestra en la Línea 7. Posteriormente en las líneas 10 a 14, se descartan los valores que no pertenezcan a los k vecinos con mayor similitud. La función $mayoresElementos(A, k)$ mostrada en la Línea 11 retorna el conjunto de los mayores k elementos del vector A para corroborar la existencia de cualquier elemento dentro de este conjunto. La Salida del algoritmo consiste en la matriz generada M.

Algoritmo 3: Item-KNN (Construcción del modelo) [22].

```
   Entrada: R, k
   Salida  : M
 1 M ← ∅
 2 for i = 0; i < m; i = i + 1 do
 3 |   for j = 0; j < m; j = j + 1 do
 4 |   |   if i ≠ j then
 5 |   |   |   M_{i,j} ← sim(R_i, R_j)
 6 |   |   else
 7 |   |   |   M_{i,j} ← 0
 8 |   |   end
 9 |   end
10 |   for j = 0; j < m; j = j + 1 do
11 |   |   if M_{i,j} ∉ mayoresElementos(M_i, k) then
12 |   |   |   M_{i,j} ← 0
13 |   |   end
14 |   end
15 end
```

Para la explotación de método, mostrada en el Algoritmo 4, se tiene

de entrada a la matriz generada M, la lista de ítems que el usuario ya ha calificado U y el número de recomendaciones deseadas n. Este algoritmo en la Línea 1 realiza para cada ítem una estimación del agrado acuerdo con los gustos del usuario. Dicha estimación se calcula de acuerdo a la Ecuación 4.1. En la ecuación se busca obtener para el ítem X_i la suma de todos sus vecinos $M_{i,j}$ multiplicado por calificación otorgada por el usuario para dicho vecino. Así, el vector de estimaciones para un usuario es la multiplicación matricial de la matriz de similitud M por la lista de ítems U ya consumidos por el usuario. Posteriormente, de las Líneas 2 a 6 se descartan los ítems que ya ha consumido el usuario. Se descartan todos los ítems que no pertenezcan a los mejores n ítems en las Líneas 7 a 11. Finalmente, se obtiene la lista de recomendación en la Línea 12 al filtrar la lista de recomendación a solo los ítems distintos de cero y al ordenarlos ítems por orden descendente de estimación. Se retorna una lista de ítems ordenados por posible nivel de agrado.

En los métodos siguientes, no se explicará el método de entrega de recomendaciones ya que no forma parte de la propuesta de los autores ([13, 22]) , sin embargo, tomaremos en cuenta un proceso similar al Algoritmo 4.

$$X_i = \sum (M_{i,j} \times U_j) \tag{4.1}$$

La técnica **Item-KNN-SP** [13] propuesta por Chae muestra dife-

Algoritmo 4: Item-KNN (Aplicación del modelo) [22].

Entrada: $M, R_{*,u}, n$
Salida : Rec

```
1  X ← MR_{*,u}
2  for i = 0; i < m; i = i + 1 do
3  |  if R_{i,u} ≠ 0 then
4  |  |  X_i ← 0
5  |  end
6  end
7  for i = 0; i < m; i = i + 1 do
8  |  if X_i ∉ mayoresElementos(X, n) then
9  |  |  X_i ← 0
10 |  end
11 end
12 Rec = listaRecomendacion(X)
```

rencias significativas en comparación con el método **Item-KNN**. Primeramente, se tiene una mayor eficiencia debido a la consideración de que existe un alto esparcimiento de datos, por lo que evita hacer búsquedas innecesarias entre ítems que no tengan recomendaciones de un mismo usuario. La propuesta de Linden [45] ya también evitaba el cálculo completo de la matriz ítems a ítems, sin embargo, Chae en su método evita también el almacenamiento de una matriz de tamaño $[n \times m]$ de usuarios por ítems. Al almacenar los elementos del conjunto de datos en una lista de n usuarios y n listas de ítems con su respectiva calificación (una lista por usuario) se logra también una significativa reducción de espacio. El método de Chae puede verse en el Algoritmo 5 donde R es la entrada de datos de forma que cada elemento de R contiene la 3-tupla

(usuario, ítem, calificación). En las Líneas 1 y 2 se definen $U[i]$ y $I[u]$, estos son índices de tablas que contienen la información necesaria para hacer recomendaciones con mayor eficiencia de memoria. El uso de tablas de hash es una herramienta común para el diseño de estos índices. Se puede ver en las Líneas 3 a 6 como U e I se utilizan para almacenar los vectores de usuarios y de ítems respectivamente. En las Líneas 8 a 18 se tiene el cálculo de la similitud. Este se hace buscando para cada ítem los usuarios que lo han recomendado, agregando a esto los ítems a los cuales estos usuarios han hecho recomendaciones y buscando solo entonces la similitud. Note que por eficiencia y control de los datos, en las Líneas 11 a 15 se hace la distinción del ítem con mayor id. Esto permite eliminar redundancia de los datos. S es el resultado de nuestro algoritmo y la matriz esparcida que almacena el valor de similitud de distintos pares de ítems.

Dynamic-Index [43] es un método enfocado a obtener una rápida recomendación para conjuntos de conjuntos de datos implícitos, es decir, donde cada registro en el conjunto de evaluaciones muestra un valor positivo sin un valor en particular. Esto permite separar las medidas de similitud en componentes divisibles fáciles de acumular para posteriores modificaciones, como ejemplo se muestra la similitud cosenoidal implícita en la Ecuación 4.2. En esta ecuación los elementos principales son el número de usuarios que han calificado tanto a los ítems i como al ítem j, $|U_i \cup U_j|$ y el número de usuarios que han calificado al ítem i, $|U_i|$ (y al ítem j, $|U_j|$). Esto permite realizar el cálculo de la similitud con

Algoritmo 5: Item_KNN_SP (construcción modelo) [13]

Entrada: R
Salida : S

```
1  ∀  u ∈ U :   I[u] ← ∅
2  ∀  i ∈ I :   U[i] ← ∅
3  for (u, i, r) ∈ R do
4  |  U[i] = U[i] ∪ (u, r)
5  |  I[u] = I[u] ∪ (i, r)
6  end
7  S ← 0
8  for i ∈ U do
9  |  for (u, r) ∈ i do
10 |  |  for j ∈ I[u] do
11 |  |  |  if i > j then
12 |  |  |  |  S_{i,j} = sim(i, j)
13 |  |  |  else
14 |  |  |  |  S_{j,i} = sim(i, j)
15 |  |  |  end
16 |  |  end
17 |  end
18 end
```

base en una matriz M de tamaño $[m \times m]$ que contiene en cada celda el número de usuarios en común entre dos ítems y un vector N de tamaño m con el número de usuarios que han calificado cada ítem.

$$cos(i,j) = \frac{\vec{\boldsymbol{u}} \cdot \vec{\boldsymbol{v}}}{||\vec{\boldsymbol{u}}||_2 \times ||\vec{\boldsymbol{v}}||_2} \approx \frac{|U_i \cap U_j|}{\sqrt{|U_i|} \times \sqrt{|U_j|}} \approx \frac{M_{ij}}{\sqrt{N_i} \times \sqrt{N_j}} \qquad (4.2)$$

El Algoritmo 6 muestra el funcionamiento del método, como entrada

Algoritmo 6: Dynamic-Index [43]

Entrada: R, I
Salida : M, N, I_r, I_n

```
1  M ← 0, N ← 0
2  ∀  u ∈ U :  I_r[u] ← ∅, I_n[u] ← ∅
3  for (u, i) ∈ R do
4  |  for j ∈ I_r[u] do
5  |  |  M_{i,j} += 1
6  |  end
7  |  if i ∈ I then
8  |  |  for j ∈ I_n[u] do
9  |  |  |  M_{i,j} += 1
10 |  |  end
11 |  |  N_i += 1
12 |  |  I_r[u] = I_r[u] ∪ i
13 |  else
14 |  |  I_n[u] = I_n[u] ∪ i
15 |  end
16 end
```

se tiene un conjunto de recomendaciones implícitas R, y el conjunto de ítems recomendables I los cuales pueden variar por disponibilidad o por ser artículos estacionarios. Como salida se tiene a la matriz M y al vector N recientemente mencionados e índices invertidos recomendables y no recomendables para cada usuario I_r y I_n, respectivamente [43]. El índice $I_r[u]$ permite identificar si es posible recomendar a un usuario u determinado ítem si esta en la lista. Por otro lado el índice $I_n[u]$ sirve para tener una relación de los ítems que el usuario ha recomendado y que no son recomendables. Otra función de este índice es que permiten calcular una relación existente en dos ítems. Esto se logra al buscar para

cada usuario u y cada ítem i en la lista de recomendación todos los ítems j ya encontrados con el usuario u e incrementar en uno el valor en la matriz $M_{i,j}$. En la Línea 2 se puede ver como se declaran los índices I_r e I_n en los cuales se cuenta con una lista vacía para cada usuario. En las Líneas 4 a 6 se llena la matriz M con base en una tupla (u, i), habiendo ítems j en el vector $I_r[u]$, se podrán contar los usuarios en común. En las Líneas 8 a 10 se continua el llenado de la matriz M pero ahora considerando también a los ítems j no recomendables. Una vez contabilizadas las relaciones de ítems en la matriz M se procede en la Línea 11 a aumentar el número de usuarios que han mostrado preferencia por i en N_i si el ítem es recomendable. Finalmente, se agrega al ítem i a su respectivo conjunto ya sea al conjunto de ítems recomendables en la Línea 12 o al conjunto de ítems no recomendables en la Línea 14.

4.3. Implementación de algoritmos

Afortunadamente, en los últimos años la reproducibilidad de los artículos de investigación esta aumentando gracias a un aumento en la cantidad de códigos fuente disponibles para las propuestas planteadas [20]. Aún así, con el fin de hacer una comparación justa y a pesar de que algunas implementaciones estén disponibles en otros lenguajes de programación, se implementarán todos los algoritmos del estado del arte en el mismo lenguaje de programación. Estos métodos ya han sido descritos en la sección anterior sin embargo, esto no aplica para el

método Dynamic-Index ya que una cualidad importante es el uso de la biblioteca Map-Reduce [21] que permite una paralelización eficiente en procesadores y la réplica de esta biblioteca no comprende los objetivos de esta investigación. Como se ha visto en el estado del arte, nos hemos apoyado en los métodos ya mencionados para contrastar los resultados de estos métodos con las propuestas algorítmicas planteadas.

El lenguaje de programación a utilizar será C ya que es un lenguaje portable en muchas plataformas además de ser bastante eficiente computacionalmente. En nuestro caso vamos a analizar la viabilidad de los métodos **Item-KNN-SP** [13] y **Dynamic-Index** [43] los cuales ya ha comprobado tener buenos resultados por si solos. En este experimento se hará uso de la metodología mostrada en la Sección 2.3 para la paralelización con GPU. El Framework utilizado para esto será Cuda C, un conjunto de bibliotecas y controladores para operar la GPU con el lenguaje C.

El almacenamiento en memoria utilizado en los métodos **Item-KNN-GPU** e **Item-KNN-SP-GPU** consiste en el uso de Almacenamiento de Filas Comprimidas CRS (por sus siglas en inglés) [51]. Esta forma de almacenar la información consiste en desplazar los elementos E de una matriz M y colocarlos en serie. Las coordenadas de los elementos se almacenan en vectores adicionales C y F. El vector C contiene los índices de columna de cada elemento. El vector F contiene un valor por cada fila mas una casilla de terminación. Finalmente, el vector E

$$M = \begin{bmatrix} - & 7 & 5 & - \\ - & - & - & - \\ - & - & 2 & 4 \\ 2 & 4 & 3 & 5 \end{bmatrix} \quad \rightarrow \quad E = \begin{bmatrix} 7 \\ 5 \\ 2 \\ 4 \\ 2 \\ 4 \\ 3 \\ 5 \end{bmatrix}, \quad C = \begin{bmatrix} 1 \\ 2 \\ 2 \\ 3 \\ 0 \\ 1 \\ 2 \\ 3 \end{bmatrix}, \quad F = \begin{bmatrix} 0 \\ 2 \\ 2 \\ 4 \\ 8 \end{bmatrix}$$

Figura 4.2: Conversión de información para matrices dispersas al formato CRS. Basado en [51].

contiene los valores de los datos de la matriz. Como se puede observar en la Figura 4.2, al transformar una matriz disperza mediante CRS, los vectores resultantes estarán ahora en función del número de elementos de la matriz. A manera de ejemplo, note el primer elemento de la matriz se encuentra en la posición (0,1) y tiene valor 7. Despues de la transformación, su valor sera colocado en la primer posición del vector E, similarmente su componente en y se colocará en C. Esto aplica a todos los elementos de la matriz recorridos con una lectura en filas. La componente en x, por otro lado tiene una construcción distinta y se encuentra en el vector F. F_i indica la cantidad de elementos que hay que desplazar en E y C para llegar al primer elemento de la fila i, ya que la primer fila no tiene elementos anteriores, se coloca F_0 como 0. El uso de CRS en la implementación de métodos de sistemas de recomendación se debe a la gran dispersión de datos encontrada en estos sistemas.

Para la implementación de los métodos en GPU se definieron los

tamaños de malla y de bloque en función de las dimensiones de entrada y de salida, es decir, arreglos ya sea de dos dimensiones para el caso de las matrices así como de una dimensión para el casos de los arreglos. A manera de ejemplo vamos a discutir el Algoritmo 3. Este se puede dividir en dos tareas fundamentales, primero el cálculo de las similitudes para cada par de ítems, la segunda es la delimitación de los mejores vecinos para cada ítem. Para el cálculo de las similitudes, al tener una matriz como salida, se ha optado por definir un tamaño de bloque que cubra todos los hilos disponibles en forma de matriz de dos dimensiones. Por otro lado, para la delimitación de los mejores vecinos para cada ítem aún cuando se trabaja con matrices como en la tarea anterior, el resultado se obtiene mediante consideraciones aplicables en una dimensión, los ítems. Es por eso que en esta tarea se ha separado por bloques de una sola dimensión. En ambos casos, se creó una malla lo suficientemente grande para que los bloques ejecutados puedan cubrir el vector o la matriz correspondiente. En la Figura 4.3 se puede ver la cantidad de hilos activados de acuerdo a la definición del bloque.

4.4. Análisis de complejidad

En la Tabla 4.2 se muestran las implementaciones utilizadas en esta investigación así como la complejidad de entrenamiento C_E y la complejidad de explotación o recomendación C_R. Como se puede ver, el segmento con la mayor complejidad ha sido el cómputo del entrenamiento

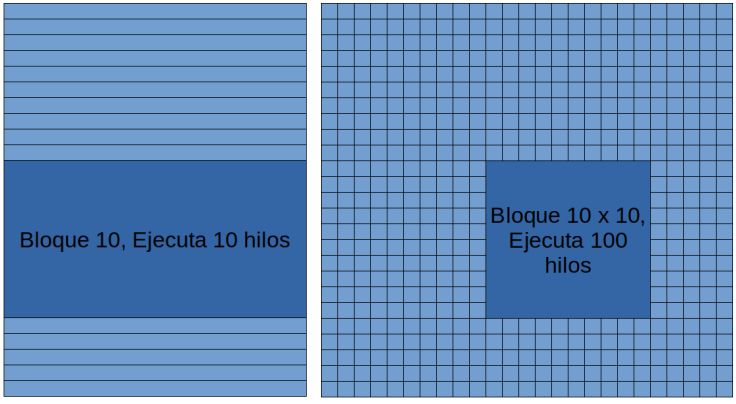

Figura 4.3: Configuración de bloques de programación para arreglos de una y dos dimensiones.

para el método Item-KNN ya que en este método se utiliza una matriz completa de ítems a ítems y cada elemento en la matriz se calcula con toda la información de los usuarios. Esta técnica no es necesariamente mala ya que con esta técnica se logra una fácil paralelización del método. Adicionalmente, el tiempo de recomendación es el mejor ya que el se encuentran listos los vecinos de la matriz.

En el caso de Dynamic-Index, la complejidad de entrenamiento aparenta ser mejor ya que la matriz de similitud se calcula de acuerdo a M y N. M y N son dos elementos explicados previamente y construidos iterativamente con una complejidad de Rp, de forma que cada casilla de similitud se obtiene en complejidad constante al iterar en todos los pares de ítems con complejidad m^2. Sin embargo, el método está pensado solamente para evaluaciones implícitas. Item-KNN-SP no tiene ese problema ya que se puede trabajar con evaluaciones explicitas. Además,

se mejora el tiempo de entrenamiento con respecto a Item-KNN. Este método igual que con dynamic-index es que proponen el uso de vecindades basadas en un número de ítems dependientes tanto del ítem con el que se comparan como con que el usuario los halla evaluado. Esto aumenta la complejidad de otorgar recomendaciones ya que se tiene que buscar un conjunto de vecinos especifico para cada par (ítem, usuario).

Con la versión Item-KNN-SP-GPU-V2 se mejoran los tiempos de recomendación hasta casi alcanzar la implementación inicial en GPU de item-KNN. La razón por la cual no es posible igualar las complejidades del método inicial Item-KNN-GPU en Item-KNN-SP-GPU-V2 es porque se ha decidido utilizar al igual en otras implementaciones de nuestra propuesta matrices dispersas con el fin de evaluar la mayor cantidad de conjuntos de datos disponibles. Vale la pena resaltar que los resultados pueden ser mejorables si se decide utilizar mayores cantidades de memoria para los experimentos, ejemplo de esto son las matrices estándares y las tablas de hash. Sin embargo, se ha optado por abordar la mayor cantidad de conjuntos de datos disponibles y en esta investigación no se proponen métodos para seccionar los bloques de información que recibe la tarjeta gráfica. A continuación se presentan los términos que se utilizan para definir la complejidad de los algoritmos.

- R: número de evaluaciones de los usuarios.
- n: número de usuarios en el sistema de recomendación.

- m: número de ítems en el sistemas de recomendación.
- p: número máximo de ítems que un solo usuario ha evaluado.
- q: número máximo de usuarios que han evaluado un ítem.
- v: número de vecinos utilizados
- h: número de hilos en la tarjeta GPU

Tabla 4.2: Análisis de complejidad de diversos métodos implementados.

Experimento	C_E (Entrenamiento)	C_R (Recomendación)
Item-KNN	O(nm^2)	O(mv)
Item-KNN-GPU	O($\frac{nm^2}{h}$)	O($\frac{mv}{h}$)
Dynamic-Index	O($Rp + m^2$)	O(m^2v)
Item-KNN-SP	O(m^2q)	O($mvp \times log(q)$)
Item-KNN-SP-GPU	O($\frac{m^2q}{h}$)	O($\frac{mvp}{h} \times log(q)$)
Item-KNN-SP-GPU-V2	O($\frac{m^2(q+v)}{h}$)	O($\frac{mv}{h} \times log(p)$)

Capítulo 5

Experimentación y análisis de resultados

En este capítulo se presentan los resultados experimentales de la investigación así como un análisis de resultados. En las Secciones 5.1, 5.2 y 5.3 se describen los conjuntos de los datos utilizados, el equipo utilizado y la validación utilizada los resultados obtenidos, respectivamente. Las Secciones 5.4, 5.5 y 5.6 son secciones enfocadas a describir los experimentos y sus resultados. En la primera sección de pruebas se muestran los tiempos de ejecución del método Item-KNN tanto en CPU como con el uso de GPU. Las pruebas muestran los tiempos de ejecución de este método para dos versiones. En la segunda sección de pruebas se muestra una comparación entre dos métodos del estado del arte, Item-

KNN-SP y Dynamic-Index. En la tercera sección experimental se evalúa Item-KNN-SP al comparar su implementación estándar con una implementación basada en GPU. Adicionalmente, se hace una modificación al método con la intención de mejorar el rendimiento del programa. Se contrastan los resultados tanto de rendimiento como la calidad de las recomendaciones. Por último, en la Sección 5.8 se contrasta la mejora teórica con la mejora alcanzada.

5.1. Conjuntos de datos utilizados en la experimentación

Se han utilizado los conjuntos de datos más comunes en los estudios sobre los sistemas de recomendación. Los conjuntos de datos utilizados son MovieLens [34], Netflix [10], MyAnimeList [52] y Amazon [49]. Todos los conjuntos de datos son de caracter público. En la Figura 5.1 se muestra el esquema de los conjuntos de datos en un modelo entidad relación. La información relevante del usuario y las características del ítem, es información que no es utilizada en los sistemas de recomendación de filtrado colaborativo. Cada usuario y cada ítem tienen un identificador. El valor de la evaluación número con valores de 1 hasta 10 siendo 1 una mala evaluación y 10 una evaluación excelente. El conjunto de datos MovieLens es un conjunto de datos para la recomendación de películas creados por GroupLens para la investigación de los sistemas

de recomendación. Los subconjuntos de MovieLens de 100 mil registros **ML100k**, un millón de registros **ML1M** y diez millones de registros **ML10M** fueron utilizados para esta investigación. Dichos conjuntos cuentan con evaluaciones a 1682 ítems hechas por 1682 usuarios para el caso de ML100k; 3706 ítems y 6040 usuarios en el caso de ML1M y 10681 ítems y 69878 usuarios en el caso de ML10M. El conjunto de datos **Netflix** se hizo público por los creadores de la empresa de servicios de transmisión de vídeo. Netflix cuenta con recomendaciones con un total de 100 millones de evaluaciones a 17,000 películas por 480,000 usuarios. **MyAnimeList** tiene en total 7,813,000 recomendaciones a 12,292 películas y series animadas japonesas *anime* por 76516 usuarios. Los administradores de la página www.myanimelist.net hicieron público el conjunto de datos con el mismo nombre con el objetivo de mejorar su sistema de recomendación. Otro conjunto de datos utilizado en los experimentos es el conjunto de datos de **Amazon**, el cual cuenta con las evaluaciones de más de 40 millones de usuarios en más de 15 millones de ítems con un aproximado de 233 millones de evaluaciones de productos. Este conjunto de datos se ha obtenido por medio del rastreo de datos de Amazon.

Los conjuntos de datos de Netflix y de Amazon fueron utilizados para la creación de subconjuntos más pequeños $\text{Amazon}_{0.1\,\%}$ y $\text{Netflix}_{1\,\%}$ con la intención de obtener una mayor concentración de datos. En el caso de $\text{Amazon}_{0.1\,\%}$ El conjunto obtenido contiene 1000 ítems y 10000 usuarios en un total de 250578 registros. Para realizar esta muestra,

identificamos los 1000 ítems más utilizados de todo el conjunto de datos de las recomendaciones de Amazon. Sobre todos los registros que contienen a dichos ítems se conservaron solo los registros con los 10000 usuarios más activos. El sufijo 0.1 % en el conjunto de datos representa el porcentaje aproximado de datos utilizados para crear el subconjunto. Para el caso de Netflix$_{1\,\%}$ se utilizó el mismo procedimiento con 1000 ítems y 1300 usuarios.

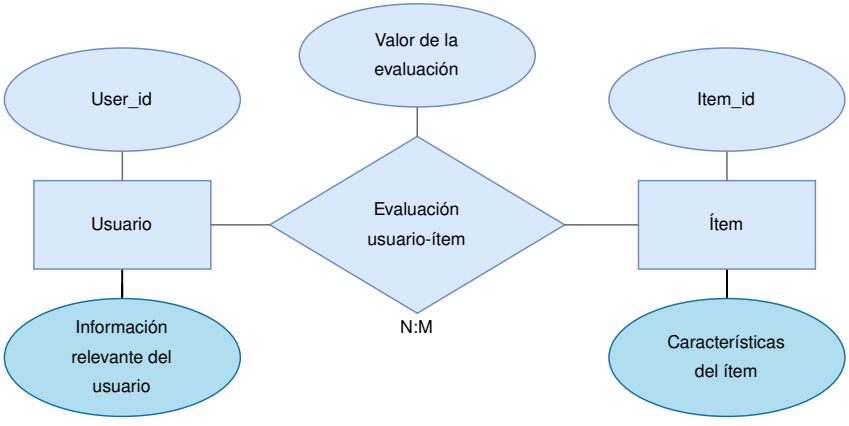

Figura 5.1: Modelo Entidad Relación de un conjunto de datos para un sistema de recomendación típico.

En la Tabla 5.1 se pueden ver las principales características de los conjuntos de datos. En esta tabla se utiliza el concepto de Densidad. Esta característica muestra la concentración de elementos ya calificados, R, de acuerdo al número de elementos existentes en una matriz de tamaño $[n \times m]$ de usuarios por ítems. Esto se puede apreciar en la Ecuación 5.1.

$$Densidad = \frac{R}{n \times m} \times 100\,\% \tag{5.1}$$

Tabla 5.1: Resumen cuantitativo de diversos conjuntos de datos para implementación. La tabla se muestra en orden ascendente de acuerdo al número de recomendaciones.

Nombre	Evaluaciones	Usuarios	Ítems	Densidad [%]
ML100K	100,000	943	1,682	6.30
$\text{Amazon}_{0.1\,\%}$	250,000	10,000	1,000	2.50
ML1M	1,000,000	6,040	3,706	4.47
$\text{Netflix}_{1\,\%}$	1,043,000	1,300	1,000	82.2
MyAnimeList	7,813,000	76,516	12,292	0.83
ML10M	10,000,000	69,878	10,681	1.34

5.2. Hardware utilizado y configuración

El equipo utilizado es una computadora con procesador Ryzen 7 3750h con 4 núcleos de procesamiento y trabaja a un ciclo de reloj de 2.3 GHz. Cuenta con un dispositivo de almacenamiento principal de estado sólido SSD conectado al puerto PCI express de 256 GB y otro dispositivo SSD conectado al puerto SATA de 512 GB y se cuenta con 16 GB de RAM. Este equipo cuenta con una tarjeta gráfica Nvidia GTX 1650. Esta tarjeta de vídeo tiene 4 GB de RAM, así como 896 núcleos y funciona con un ciclo de reloj de 1.49 GHz [69]. El equipo trabaja con Linux Ubuntu versión 20.04. El gestor de bases de datos es PostgreSQL 9.6. Los programas fueron implementados en el lenguaje C.

5.3. Validación del algoritmo propuesto

Para la experimentación se utilizó en todos los casos la validación cruzada con 10 agrupaciones. Los resultados presentados en las siguientes secciones muestran el promedio de 30 ejecuciones.

5.3.1. Parámetros de evaluación

Para evaluar nuestro sistema de recomendación se utiliza en primer instancia el tiempo de ejecución, sin embargo, para contrastar nuestros resultados vamos a evaluar también la utilidad de las recomendaciones mostradas. Para los experimentos se definen tres tiempos, el de entrenamiento, de explotación y de respuesta del programa. El tiempo de entrenamiento mide el tiempo de ejecución desde que se tienen los datos ya cargados en la RAM hasta que se obtiene una matriz de vecindades para cada ítem. El tiempo de explotación o de recomendación parte desde que se tiene la matriz de similitudes hasta que se obtiene una lista de recomendaciones para cada usuario. Dado que se hace una recomendación por usuario, el resultado mostrado es el tiempo medio de las recomendaciones. El tiempo de ejecución del programa se utiliza para medir el tiempo desde la obtención de datos desde la base de datos hasta la obtención de recomendaciones para todos los usuarios.

Para evaluar la respuesta del sistema utilizaremos la precisión, la exhaustividad y la normalización de la ganancia acumulada discontinua.

La precisión se define como el porcentaje de la división del número de valores positivos acertados entre aquellos que son considerados como positivos [2]. En sistemas de recomendación la precisión mide la razón del número de ítems en el conjunto de ítems recomendados a los que los usuarios han mostrado interés [64]. Para determinar los ítems de interés al usuario utilizaremos el conjunto de ítems que el usuario ha calificado con la máxima calificación con el fin de mantener solamente evaluaciones muy buenas. En la Ecuación 5.2 se muestra la fórmula de la precisión, donde Rec_u representa el conjunto de ítems recomendados para el usuario u y T_u representa el conjunto de ítems de interés para dicho usuario.

$$Precisión = \frac{1}{|U|} \sum_{u \in U} \frac{|Rec_u \cap T_u|}{|Rec_u|} \tag{5.2}$$

La exhaustividad mide el número de positivos acertados entre el total de verdaderos (número de ítems a los que el usuario u ha mostrado interés T_u) para cada usuario. La Ecuación 5.3 indica como se calcula la exhaustividad. De manera similar a la precisión, el resultado general de la exhaustividad es la media de la evaluación para cada usuario.

$$Exhaustividad = \frac{1}{|U|} \sum_{u \in U} \frac{|Rec_u \cap T_u|}{|T_u|} \tag{5.3}$$

La normalización de la ganancia acumulada discontinua (NDCG por

sus siglas en inglés) tiene por objetivo evaluar el orden de los ítems recomendados dada una lista ordenada de ítems. NDCG está basada en la ganancia acumulativa discontinua DCG. DCG se muestra en la Ecuación 5.5 [39], es una función que considera la posición de cada elemento recomendado y, en caso de ser de interés para el usuario, este tendrá una ponderación positiva de acuerdo al orden en que se encuentre. En la Ecuación 5.5, *Rec* es la lista de los elementos recomendados y de interés para el usuario siendo Rec_p un elemento de *Rec* recomendado en la posición p con valor de 1 si es este un ítem de interés para el usuario y 0 en caso contrario ($Rec_p \in \{0, 1\}$). El cálculo de DCG utiliza una relación logarítmica para acentuar una ponderación mayor en las primeras casillas. Esto permite estimar no solo que la lista de recomendación sea buena, sino también que tenga un orden adecuado. La suma de los puntajes de ítems relevantes determinan la calidad de la lista de recomendaciones. $NDCG$, como se puede apreciar en la Ecuación 5.4, se calcula al obtener la relación entre DCG_u con una ganancia acumulativa discontinua ideal $IDCG_u$. $IDCG_u$ es el puntaje máximo obtenible de DCG_u para el usuario u y se calcula de acuerdo a la Expresión 5.6.

$$NDCG = \frac{1}{|U|} \sum_{u \in U} \frac{DCG_u}{IDCG_u} \tag{5.4}$$

Donde:

$$DCG_u = \sum_{\rho=1}^{N} \frac{Rec_\rho}{\log_2(\rho + 1)} \tag{5.5}$$

$$IDCG_u = \sum_{\rho=1}^{N} \frac{RMAX_\rho}{\log_2(\rho+1)} \tag{5.6}$$

Los elementos Rec_p de Rec pueden tener valores de 1 en caso de existir un ítem relevante o 0 en caso contrario. $RMAX$ representa el caso ideal en una lista de recomendaciones. Para obtener $RMAX$ se considera el número de ítems relevantes α_u para el usuario u, en las primeras α_u posiciones de $RMAX$ se asignará el valor de 1; en caso de existir elementos restantes en $RMAX$ estos se poblarán con el valor 0. Así, un escenario perfecto para un usuario que solo ha calificado 2 ítems debería ser $IDCG_u = 1 + \frac{1}{log_2(3)} + \frac{0}{log_2(4)} + ... = 1.63$, independientemente de la cantidad de ítems que se hallan recomendado.

5.4. Experimentación y resultados de la paralelización de Item-KNN

Para esta sección se describen dos pruebas experimentales al implementar el método previamente descrito, **Item-KNN** en su versión estandar en CPU y su versión en GPU, **Item-KNN-GPU**. Estas pruebas se evalúan con los conjuntos de datos mostrados en la Tabla 5.2. En la primer prueba se analiza un aspecto técnico de la tarjeta gráfica, el número de hilos utilizado en el esquema de paralelización para el caso de la explotación. En la segunda prueba se hizo una comparación simple de los tiempos de entrenamiento y de explotación. Se compara tanto la im-

plementación basada en CPU como la implementación basada en GPU con la configuración que genera mejores resultados para cada conjunto de datos. En todos los casos se utilizan 50 vecinos y se recomiendan 10 ítems. Se utiliza la similitud cosenoidal como función de similitud ya que es la función de similitud que se utilizó en los métodos previamente descritos.

Prueba uno. Para seleccionar los mejores tiempos de explotación, variamos la distribución de hilos por bloque de forma que los multiprocesadores de la GPU utilicen el total de sus núcleos para realizar sus tareas, es decir, mantendremos siempre el número de hilos como múltiplos de 32. La variación en el número de hilos determina la cantidad de memoria que tiene que ser desplazada a un bloque así como el tiempo de inicialización de los hilos antes de empezar éstos a realizar las operaciones. Los resultados mostrados en las Figuras 5.2, 5.3 y 5.4 no indican un comportamiento generalizable a los conjuntos de datos, ya que las velocidades fueron mejores con 512 hilos, 64 hilos y 32 hilos para los conjuntos de datos ML1M, ML100k y $\text{Amazon}_{0.1\,\%}$, respectivamente. Al contrastar la cantidad de ítems en cada conjunto de datos, vemos que a mayor cantidad de ítems, mayor es el número de hilos por bloque que generan un mejor rendimiento. La diferencia entre los resultados de las elecciones de bloque más tardadas con respecto a las más rápidas indican que los tiempos de ejecución pueden aumentar 20.7 %, 263 % y 20.0 % para los conjuntos de datos ML1M, ML100k y $\text{Amazon}_{0.1\,\%}$, respectivamente. La selección del número de hilos por bloque resulta ser

muy relevante en conjunto de datos más chico, ML100k. En este caso utilizar un número grande de hilos por bloque implica un desperdicio de tiempo ya que algunos bloques incluirán el calculo de hilos que exceden las dimensiones de la matriz de similitud. Estos hilos serán simplemente descartados mientras que el proceso asigna recursos para todos los hilos en el bloque.

Tabla 5.2: Resumen cuantitativo de los conjuntos de datos para implementación de algoritmos.

Nombre	núm. hilos con mejor rendimiento
ML 1M	512
ML 100K	64
$\text{Amazon}_{0.1\,\%}$	32

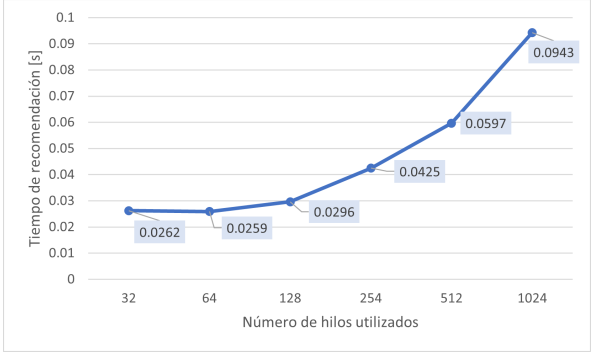

Figura 5.2: Comparación de tiempos de explotación con distintos tamaños de bloque utilizados en ML100K, algoritmo: ItemKNN_GPU (50 vecinos).

Prueba dos. En esta prueba se compararon los tiempos de ejecución de las implementaciones en CPU y en GPU. En la Figura 5.5 se muestra una gráfica con escala logarítmica con los resultados de nues-

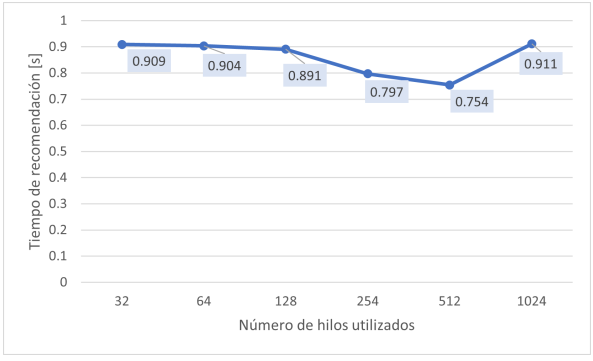

Figura 5.3: Comparación de tiempos de explotación con distintos tamaños de bloque utilizados en ML1M, algoritmo: ItemKNN_GPU (50 vecinos).

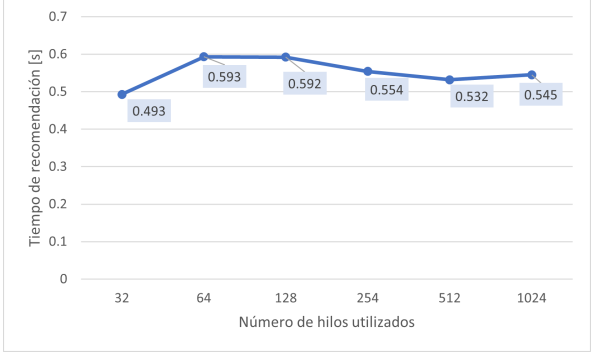

Figura 5.4: Comparación de tiempos de explotación con distintos tamaños de bloque utilizados en $Amazon_{0.1\,\%}$, algoritmo: ItemKNN_GPU (50 vecinos).

tros experimentos en relación al entrenamiento. Se aprecian mejoras de tiempos desde 46 hasta 104 veces más rápido para los tiempos de construcción del modelo al comparar el uso de la implementación basada en CPU con la implementación basada en GPU. Para los tiempos de obtención de recomendaciones, se tienen mejoras desde 4x para el ca-

so de Amazon0.1 % hasta 13x para el caso de ML100k, como se puede apreciar en la gráfica con escala logarítmica de la Figura 5.6. En ambas figuras se ha optado por una gráfica de tipo logarítmica por las grandes diferencias en los resultados obtenidos. La notable diferencia de los tiempos de respuesta se debe a cantidad de cómputo necesario en las faces del algoritmo que pudieron ser paralelizadas. Similarmente, se puede ver que el costo computacional adicional de la paralelización no impacta significativamente los tiempos de respuesta de la implementación basada en GPU. El argumento anterior aplica también al contrastar la cantidad de mejora del entrenamiento con respecto a la explotación del método. El trabajo realizado durante la paralelización es menor en la explotación del método. Esto hace más significativo el costo de los procesos adicionales necesarios para la paralelización en contraste con el entrenamiento del método. Dichos procesos adicionales consisten en el envío de los datos desde la RAM de la computadora hasta la RAM de la GPU; el proceso de administración y control de los hilos por parte de la GPU y el tiempo de desplazamiento de los resultados de la RAM de la GPU a la RAM de la computadora.

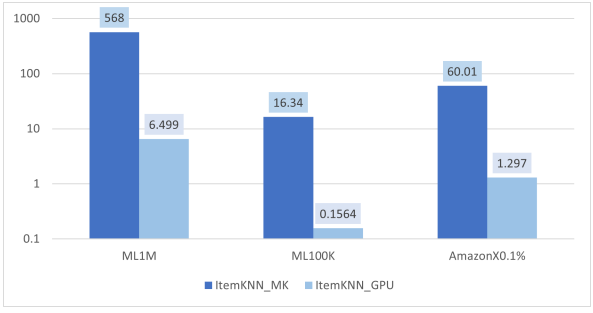

Figura 5.5: Comparación de tiempos de entrenamiento para algoritmos Item-KNN e Item-KNN-GPU.

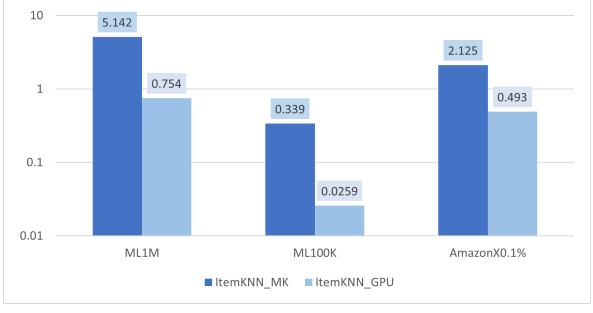

Figura 5.6: Comparación de tiempos de realización de recomendaciones para algoritmos Item-KNN e Item-KNN-GPU.

5.5. Comparación de métodos Item-KNN-SP y Dynamic Index

En esta sección se muestran los tiempos de ejecución de dos métodos del estado del arte. Se ha utilizado la implementación de Jeunen [43] para investigar los tiempos de entrenamiento en los métodos **Item-KNN-SP** y **Dynamic-Index**. Se muestran en la Tabla 5.3 todos los conjuntos de datos almacenables en memoria permitidos por el método. Estos son

ML100k, Amazon$_{0.1\,\%}$, ML1M, Netflix$_{1\,\%}$, MyAnimeList y ML10M. Las pruebas han sido realizadas utilizando un solo hilo en CPU. Los resultados muestran tiempos de entrenamiento similares, sin embargo, al descartar de los resultados los subconjuntos de datos creados por nosotros Amazon$_{0.1\,\%}$ y Netflix$_{1\,\%}$, vemos resultados superiores en el método Item-KNN-SP. Esto confirma que este método funciona mejor con conjuntos de datos de baja densidad, es decir, ambientes más cercanos a la realidad. Esta conclusión da lugar que la siguiente sección de pruebas se base en este método.

Tabla 5.3: Comparación de los tiempos de entrenamiento de los métodos para retroalimentaciones implícitas ItemKNN-SP y Dynamic-Index.

Dataset	Item-KNN-SP [s]	Dynamic-Index [s]
ML100K	1.115	1.306
Amazon$_{0.1\,\%}$	0.529	0.223
ML1M	35.183	32.682
Netflix$_{1\,\%}$	67.685	43.766
MyAnimeList	102.536	239.896
ML10M	473.958	467.528

5.6. Experimentación y resultados de la paralelización de Item-KNN-SP

En esta sección se tiene el eje focal de la investigación, encontrar una mejora substancial con respecto al mejor método encontrado en el estado del arte. Para esto se ha seleccionado Item-KNN-SP con base en

la experimentación de la sección anterior. En esta etapa de pruebas se compara la implementación basada en CPU versus la implementación basada en GPU **Item-KNN-SP-GPU**. Estos métodos se describen a profundidad en el capítulo anterior. Las pruebas se han realizado con la función de similitud cosenoidal de acuerdo con el estado del arte. Se han utilizado 50 vecinos en todos los casos. En el caso del tiempo de explotación se muestra el tiempo de recomendación (conjunto de 10 ítems) por usuario. Se han utilizado todos los conjuntos de datos almacenables en la memoria RAM de la tarjeta gráfica. Debido a esto, no fue posible evaluar el conjunto de datos ML10M.

Durante la realización de los experimentos se observó un área de mejora en lo que respecta la selección de los vecinos basados en ítems, por lo que se ha optado por mostrar también los resultados al cambiar la función de similitud. En el estado del arte se propone utilizar $N_u(i)$, es decir, los mejores k ítems vecinos similares al ítem i siempre y cuando los halla evaluado el usuario u. En esta tercera implementación se utiliza $N(i)$, es decir, los mejores k vecinos similares al ítem i. Debido al cambio en la vecindad considerada, llamaremos a esta implementación **Item-KNN-SP-GPU-V2**. Esta diferencia implica lo siguente. En **Item-KNN-SP-GPU**, se hace una búsqueda de ítems de los N vecinos evaluados por el usuario u, los cuales serán utilizados en su totalidad por la formula 2.13. Por otro lado, en **Item-KNN-SP-GPU-V2**, se toman solo los mejores N vecinos, si un ítem no tiene vecinos cercanos evaluados por el usuario u no se buscarán otros vecinos con los que

hacer la evaluación, sino que se descartará este ítem como una potencial recomendación. Es por esto que se pierde información relevante ya que cualquier ítem vecino que no halla sido evaluado por el usuario u evitará el ingreso de algún ítem (quizá no tan similar) que si lo halla evaluado el usuario u en el cálculo de la recomendación. En cuanto a la velocidad de recomendación en **Item-KNN-SP-GPU-V2**, deja de ser necesario encontrar un conjunto de vecinos distinto para cada par (ítem, usuario) reduciendo el tiempo de recomendación significativamente. Esta diferencia afecta directamente la complejidad de la explotación del método.

Los tiempos de ejecución de las implementaciones del método en CPU Item-KNN-SP, del método Item-KNN-SP-GPU y de la variante propuesta Item-KNN-SP-GPU-V2 se muestran en la Tabla 5.4. Similarmente, en la tabla 5.6 se han registrado los parámetros de evaluación basado en la utilidad. Los resultados indican que al contrastar los resultados de la implementación Item-KNN-SP-GPU con Item-KNN-SP se redujo el tiempo de respuesta tanto en el entrenamiento como en la explotación de los métodos. En la tabla 5.5 se puede ver la relación entre los métodos para los tiempos obtenidos. La mejora obtenida para el tiempo de entrenamiento ha sido desde 3.2x para el caso de $\text{Amazon}_{0.1\,\%}$ hasta las 23x en el caso de ML1M. En el caso de la explotación tenemos mejoras de entre 18.2x hasta 30x. Por otro lado, la implementación Item-KNN-SP-GPU-V2 comparado con Item-KNN-SP-GPU muestra mejoras en el tiempo explotación de recomendaciones desde 3.5x hasta 10.8x. Es-

ta gran mejora, como se explico anteriormente, se debe a que hay una reducción en la complejidad de la explotación al no tener que considerar los elementos evaluados por el usuario al momento de obtener las vecindades de los ítems. Estas versiones paralelizadas ofrecen tiempos muy similares en el entrenamiento de los métodos con diferencias de hasta +/- 4 % para todos los conjuntos de datos. La mejora más grande se puede ver al contrastar los tiempos de explotación de la implementación Item-KNN-SP-CPU con la implementación Item-KNN-SP-GPU-V2 ya que se pueden ver mejoras desde 82.3x en el caso de $\text{Amazon}_{0.1\,\%}$ y de hasta 278.5x en el caso de ML100k. El aspecto más destacable de la implementación Item-KNN-SP-GPU-V2 es el aumento en la calidad de las recomendaciones aún cuando en esta implementación la selección de vecinos supone un trabajo menor. Item-KNN-SP-GPU-V2 suele utilizar menos vecinos para el calculo de las recomendaciones, lo que se ve en los resultados es que en todos los conjuntos de datos se observó una mejora en los resultados de Precisión, Exhaustividad y NDCG. Esto puede observarse en la tabla 5.6. La obtención de estos resultados se debe a que el uso de más elementos calificados por el usuario suponen un deterioro de la calidad de las recomendaciones si esto implica alejarse de una similitud natural entre los ítems. Los resultados obtenidos resultan contra intuitivos ya que se esperaría que una selección personalizada de ítems vecinos para cada usuario generaría mejores resultados.

Tabla 5.4: Comparación de métodos ItemKNN-SP, ItemKNN-SP-GPU y ItemKNN-SP-GPU-V2 en tiempos de entrenamiento, explotación y parametros de evaluación basados en utilidad.

Implementación	Conjunto de datos	Tiempo de entrenamiento [s]	Tiempo de explotación [ms]
Item-KNN-SP-CPU	ML100k	2.982	28.129
	ML1M	73.968	124.124
	Amazon	3.700	5.997
	Netflix	23.291	18.443
	MyAnimeList	1008.342	195.826
Item-KNN-SP-GPU	ML100k	0.183	1.093
	ML1M	3.206	6.826
	Amazon	1.144	0.200
	Netflix	1.182	0.780
	MyAnimeList	47.722	8.175
Item-KNN-SP-GPU-V2	ML100k	0.180	0.101
	ML1M	3.131	0.650
	Amazon	1.191	0.038
	Netflix	1.177	0.224
	MyAnimeList	47.466	1.182

Tabla 5.5: Relación entre tiempos RT de entrenamiento y explotación para los métodos ItemKNN-SP, ItemKNN-SP-GPU y ItemKNN-SP-GPU-V2.

		Item-KNN-SP-GPU		Item-KNN-SP-GPU-V2	
Implementación	Conjunto de datos	*RT* de entrenamiento	*RT* de explotación	*RT* de entrenamiento	*RT* de explotación
Item-KNN-SP-CPU	ML100k	16.295	25.735	16.567	278.505
	ML1M	23.072	18.184	23.624	190.96
	Amazon	3.2343	29.985	3.1066	157.816
	Netflix	19.705	23.645	19.788	82.3348
	Anime	21.129	23.954	21.243	165.673
Item-KNN-SP-GPU	ML100k	1.00	1.00	1.017	10.822
	ML1M	1.00	1.00	1.024	10.501
	Amazon	1.00	1.00	0.960	5.2631
	Netflix	1.00	1.00	1.004	3.4821
	Anime	1.00	1.00	1.005	6.9162

Tabla 5.6: Comparación de métodos ItemKNN-SP, ItemKNN-SP-GPU y ItemKNN-SP-GPU-V2 en tiempos de entrenamiento, explotación y parametros de evaluación basados en utilidad.

Implementación	Conjunto de datos	Precisión	Exhaustividad	NDCG
Item-KNN-SP-CPU	ML100k	0.041	0.129	0.088
	ML1M	0.045	0.108	0.085
	Amazon	0.177	0.712	0.682
	Netflix	0.180	0.125	0.148
	MyAnimeList	0.001	0.006	0.003
Item-KNN-SP-GPU	ML100k	0.041	0.129	0.088
	ML1M	0.045	0.108	0.085
	Amazon	0.177	0.712	0.682
	Netflix	0.180	0.125	0.148
	MyAnimeList	0.001	0.006	0.003
Item-KNN-SP-GPU-V2	ML100k	0.058	0.190	0.140
	ML1M	0.062	0.146	0.116
	Amazon	0.178	0.715	0.683
	Netflix	0.189	0.132	0.156
	MyAnimeList	0.001	0.009	0.005

5.7. Conclusiones de la mejora teórica vs la mejora observada

Como se ve en la sección de hardware utilizado, la tarjeta gráfica utilizada cuenta con 896 núcleos. Estos núcleos se reparten a lo largo de 14 multiprocesadores. Al esperar una carga de trabajo similar y distribuida uniformemente en cada hilo se esperaría una reducción de tiempo de 896x. Los resultados obtenidos muestran una mejoría de 104x en el mejor experimento para la mejora del método Item-KNN y mejoras de tan solo 23x en los mejores experimentos al usar el método Item-KNN-SP con la versión Item-KNN-SP-GPU. Debido a que Item-KNN-SP-GPU-V2 cambia los resultados obtenidos al tomar seleccionar otro conjunto de vecinos, no será tomado en cuenta en este análisis. Las posibles causas de este descenso en la mejora observada son los procesos emergentes al paralelizar cualquier proceso en GPU y la cantidad de hilos desocupados en el desarrollo de las operaciones. Los procesos emergentes anteriormente mencionados consisten en el desplazamiento de la información hacia la tarjeta gráfica; el proceso de administración y control de los hilos por parte de organizador de tareas de la GPU y posteriormente el desplazamiento a la RAM de la computadora. Tenga en cuenta que las operaciones de desplazamiento de información son mucho mas lentas que una operación aritmética que pueda realizarse en un ciclo de reloj del procesador, ya que se trata de un desplazamiento físico de información.

Con lo que respecta a la cantidad de hilos desocupados, el organizador de tareas distribuye información para que todos los hilos trabajen a la vez. Pese a esto, algunos hilos terminan su ejecución de manera temprana al tener una condición de finalización sin la posibilidad de seguir ejecutando tareas hasta que todo su bloque de hilos este listo para comenzar con una nueva tarea. De forma menos impactante se encuentran también todos aquellos hilos que se encuentran más allá de los límites de la memoria utilizada. Para todos estos casos la condición de parada se encuentra justo al inicio de la creación de dicho hilo. Este efecto se hizo muy notorio en el experimento de la Figura 5.2 para el conjunto de datos ML100k, donde al aumentar la cantidad de hilos por bloque de tareas era más evidente el aumento en el tiempo de ejecución. Al usar conjuntos de datos con más información como fueron los casos de ML1M y de $\text{Amazon}_{0.1\,\%}$, no se notó este efecto.

Capítulo 6

Conclusiones y trabajo a futuro

6.1. Conclusiones

En esta investigación se emplearon métodos del estado del arte para la mejora del tiempo de ejecución de los sistemas de recomendación de filtrado colaborativo basado en vecindarios. Los resultados obtenidos al mejorar los sistemas de recomendación mediante el uso de GPU son favorables y predecibles dado que hay una gran cantidad de aplicaciones y estudios que lo han demostrado. Como se puede observar en nuestros resultados experimentales, se ha logrado una mejora significativa, lo que nos permite confirmar la validez de nuestras hipótesis iniciales al obtener

un progreso notable sin afectar la calidad de las recomendaciones.

Dentro de los hallazgos más significativos, se destaca que el método item-KNN muestra una mayor ventaja al ser paralelizado, con mejoras de tiempo de hasta 104 veces. Esto se debe a que las estructuras de datos utilizadas en este algoritmo como las matrices permiten un trabajo óptimo en la GPU. La GPU, al tener cientos de hilos trabajando con currentemente, requiere estructuras de datos simples para la lectura rápida de memoria. Al contrastar esto con métodos como los índices de tipo (usuario, lista de ítems) se observa una mayor cantidad de trabajo por parte de los hilos de procesamiento para acceder a la información aún cuando se reduce la cantidad de memoria utilizada. Por otro lado, se siguen observando mejoras al comparar item-KNN con los métodos item-KNN-SP en sus versiones paralelizadas ya que este último necesita menos tiempo para generar las recomendaciones.

En este trabajo de investigación se aportan dos nuevos algoritmos de paralelización para filtrado colaborativo en sistemas de recomendación. Estos algoritmos superan los resultados observados en el estado del arte, lo cual constituye un avance significativo en el campo. Se espera que con este trabajo de investigación se sienten bases para futuros estudios de optimización en sistemas de recomendación. La presente investigación muestra un marco de trabajo simple y conciso, con el objetivo de mejorar el rendimiento de los algoritmos de recomendación y obtener mejoras tangibles en este aspecto. Este enfoque busca abrir nuevas oportunida-

des de investigación en términos de optimización y rendimiento de los sistemas de recomendación. Vale la pena destacar que estos nuevos algoritmos son validos para los casos en donde se cuenta con suficiente memoria en RAM en la GPU para almacenar la totalidad del conjunto de datos así como una matriz de usuarios e ítems.

6.2. Limitaciones

Durante esta investigación, fueron encontradas algunas limitaciones que afectaron el progreso y los resultados de esta investigación. Una de ellas fue el desconocimiento de metodologías de desarrollo de software, lo cual ralentizo la implementación de algoritmos. Por otro lado, algunos conjuntos de datos considerados en las etapas iniciales no se pudieron evaluar debido a la gran cantidad de memoria necesaria. Finalmente, hubo retrasos en la etapa de pruebas debido a que el equipo utilizado para implementar los algoritmos era el mismo que con el que se realizaban las pruebas. Con el fin de mejorar en posteriores estudios, se recomienda el uso de metodologías ágiles de desarrollo de software en el cual los entregables se obtienen con mayor frecuencia; Respecto a la limitación de memoria, se sugiere considerar propuestas de separación del conjunto de datos para generar soluciones parciales al problema; Finalmente, en lo que respecta a la etapa de pruebas, se recomienda hacer un diseñar estrategias de prueba en etapas más tempranas para utilizar los horarios no laborales para ejecutar bloques más pequeños de pruebas

o utilizar servicios de computo en la nube para realizar pruebas en un número virtualmente ilimitado de maquinas.

6.3. Trabajo a futuro

Como trabajo a futuro se propone lo siguiente.

- Implementar esquemas de paralelización a sistemas de recomendación que requieran más información. Como ya se ha mencionado, el filtrado colaborativo ofrece buenos resultados solo con el uso de evaluaciones de usuarios a ítems. Sin embargo, las propuestas híbridas que contemplan también la información del ítem y del usuario han demostrado obtener resultados superiores con respecto a la calidad de las recomendaciones. [40], [62].

- Utilizar sistemas gestores de bases de datos, no solo para el almacenamiento de la información, sino también para la implementación de procedimientos que permitan ejecutar los sistemas de recomendación dentro del mismo sistema gestor. Así mismo, se propone comparar distintos gestores de bases de datos con el fin de aprovechar en medida de lo posible las capacidades de estos.

- Estudiar tecnologías paralelizables a través de varios dispositivos. Creemos que tecnologías como contenedores en servidores almacenados en una nube constituyen un baja barrera de acceso y

permiten una alta escalabilidad en los sistemas de recomendación. Sin embargo, esta es solo una de muchas soluciones disponibles para obtener un mayor alcance en el uso de los sistemas de recomendación.

Bibliografía

[1] Abdollahpouri, H., Adomavicius, G., Burke, R., Guy, I., Jannach, D., Kamishima, T., Krasnodebski, J., and Pizzato, L. Multistakeholder recommendation: Survey and research directions. *User Modeling and User-Adapted Interaction 30*, 1 (2020), 127–158.

[2] Adomavicius, G., and Tuzhilin, A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering 17*, 6 (2005), 734–749.

[3] Aggarwal, C. C., et al. *Recommender systems*, vol. 1. Springer, 2016.

[4] Airbnb. url : https://www.airbnb.com, 2021.

[5] Amazon. url : https://www.amazon.com.mx, 2021.

[6] Atasu, K., Parnell, T., Dünner, C., Vlachos, M., and Pozidis, H. High-performance recommender system training using co-clustering on cpu/gpu clusters. In *2017 46th International Conference on Parallel Processing (ICPP)* (2017), IEEE, pp. 372–381.

[7] Bagchi, S. Performance and quality assessment of similarity measures in collaborative filtering using mahout. *Procedia Computer Science 50* (2015), 229–234.

[8] Barbaglia, L., Consoli, S., Manzan, S., Recupero, D. R., Saisana, M., and Pezzoli, L. T. Data science technologies in economics and finance: A gentle walk-in. In *Data Science for Economics and Finance*. Springer, Cham, 2021, pp. 1–17.

[9] Bell, R. M., and Koren, Y. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *Seventh IEEE international conference on data mining (ICDM 2007)* (2007), IEEE, pp. 43–52.

[10] Bennett, J., Lanning, S., et al. The netflix prize. In *Proceedings of KDD cup and workshop* (2007), vol. 2007, New York, p. 35.

[11] Billsus, D., Pazzani, M. J., et al. Learning collaborative information filters. In *Icml* (1998), vol. 98, pp. 46–54.

[12] Burke, R., Felfernig, A., and Göker, M. H. Recommender systems: An overview. *Ai Magazine 32*, 3 (2011), 13–18.

[13] Chae, D.-K., Lee, S.-C., Lee, S.-Y., and Kim, S.-W. On identifying k-nearest neighbors in neighborhood models for efficient and effective collaborative filtering. *Neurocomputing 278* (2018), 134–143.

[14] Chen, P.-Y., Wu, S.-y., and Yoon, J. The impact of online recommendations and consumer feedback on sales. *ICIS 2004 Proceedings* (2004), 58.

[15] Chen, R., Hua, Q., Chang, Y.-S., Wang, B., Zhang, L., and Kong, X. A survey of collaborative filtering-based recommender systems: From traditional methods to hybrid methods based on social networks. *IEEE Access 6* (2018), 64301–64320.

[16] Chen, T., and Guestrin, C. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (2016), pp. 785–794.

[17] Cheng, H.-T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., Chai, W., Ispir, M., et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems* (2016), pp. 7–10.

[18] Coppel. url : https://www.coppel.com, 2021.

[19] Cui, Z., Xu, X., Fei, X., Cai, X., Cao, Y., Zhang, W., and Chen, J. Personalized recommendation system based on collaborative filtering for iot scenarios. *IEEE Transactions on Services Computing 13*, 4 (2020), 685–695.

[20] Dacrema, M. F., Cremonesi, P., and Jannach, D. Are we really making much progress? a worrying analysis of recent neural recommendation approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems* (2019), pp. 101–109.

[21] Dean, J., and Ghemawat, S. Mapreduce: simplified data processing on large clusters. *Communications of the ACM 51*, 1 (2008), 107–113.

[22] Deshpande, M., and Karypis, G. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS) 22*, 1 (2004), 143–177.

[23] Duckduckgo. url : https://www.duckduckgo.com, 2021.

[24] Facebook. url : https://www.facebook.com, 2021.

[25] Felix, E. Propuesta de algoritmo de filtrado colaborativo basado en vecindarios para sistemas de recomendación, fase experimental. url : https://github.com/emmanuelfv/recomenderSystemsGPU, 2023.

[26] Gantner, Z., Rendle, S., Freudenthaler, C., and Schmidt-Thieme, L. Mymedialite: A free recommender system library. In *Proceedings of the fifth ACM conference on Recommender systems* (2011), pp. 305–308.

[27] Geuens, S., Coussement, K., and De Bock, K. W. A framework for configuring collaborative filtering-based recommendations derived from purchase data. *European Journal of Operational Research 265*, 1 (2018), 208–218.

[28] Goldberg, K., Roeder, T., Gupta, D., and Perkins, C. Eigentaste: A constant time collaborative filtering algorithm. *information retrieval 4*, 2 (2001), 133–151.

[29] Google. url : https://www.google.com, 2021.

[30] Greenacre, M. *Correspondence analysis in practice*. chapman and hall/crc, 2017.

[31] Guide, D. Cuda c++ programming guide. *NVIDIA, July* (2020).

[32] Guo, G., Zhang, J., and Yorke-Smith, N. Trustsvd: Collaborative filtering with both the explicit and implicit influence of user trust and of item ratings. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2015), vol. 29.

[33] Guo, H., Tang, R., Ye, Y., Li, Z., and He, X. Deepfm: a factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247* (2017).

[34] Harper, F. M., and Konstan, J. A. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis) 5*, 4 (2015), 1–19.

[35] Heckel, R., Vlachos, M., Parnell, T., and Dünner, C. Scalable and interpretable product recommendations via overlapping co-clustering. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)* (2017), IEEE, pp. 1033–1044.

[36] Hill, M. D., and Marty, M. R. Amdahl's law in the multicore era. *Computer 41*, 7 (2008), 33–38.

[37] Hwang, W.-S., Parc, J., Kim, S.-W., Lee, J., and Lee, D. "told you i didn't like it": Exploiting uninteresting items for effective collaborative filtering. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)* (2016), IEEE, pp. 349–360.

[38] Jain, G., Mahara, T., and Tripathi, K. N. A survey of similarity measures for collaborative filtering-based recommender system. *Soft computing: theories and applications* (2020), 343–352.

[39] Jalili, M., Ahmadian, S., Izadi, M., Moradi, P., and Salehi, M. Evaluating collaborative filtering recommender algorithms: a survey. *IEEE access 6* (2018), 74003–74024.

[40] Jankiewicz, P., Kyrashchuk, L., Sienkowski, P., and Wójcik, M. Boosting algorithms for a session-based, context-aware recommender system in an online travel domain. In *Proceedings of the Workshop on ACM Recommender Systems Challenge* (2019), pp. 1–5.

[41] Jannach, D., and Adomavicius, G. Recommendations with a purpose. In *Proceedings of the 10th ACM Conference on Recommender Systems* (2016), pp. 7–10.

[42] Jannach, D., de Souza P. Moreira, G., and Oldridge, E. Why are deep learning models not consistently winning recommender systems competitions yet? a position paper. In *Proceedings of the Recommender Systems Challenge 2020*. 2020, pp. 44–49.

[43] Jeunen, O., Verstrepen, K., and Goethals, B. Efficient similarity computation for collaborative filtering in dynamic environments. In *Proceedings of the 13th ACM Conference on Recommender Systems* (2019), pp. 251–259.

[44] Koren, Y., and Bell, R. Advances in collaborative filtering. In *Recommender systems handbook*. Springer, 2015, pp. 77–118.

[45] Linden, G., Smith, B., and York, J. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing 7*, 1 (2003), 76–80.

[46] Linkedin. url : https://www.linkedin.com, 2021.

[47] Lu, J., Wu, D., Mao, M., Wang, W., and Zhang, G. Recommender system application developments: a survey. *Decision Support Systems 74* (2015), 12–32.

[48] Lü, L., Medo, M., Yeung, C. H., Zhang, Y.-C., Zhang, Z.-K., and Zhou, T. Recommender systems. *Physics reports 519*, 1 (2012), 1–49.

[49] McAuley, J., Pandey, R., and Leskovec, J. Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining* (2015), pp. 785–794.

[50] MercadoLibre. url : https://www.mercadoLibre.com.mx, 2021.

[51] Merrill, D., and Garland, M. Merge-based sparse matrix-vector multiplication (spmv) using the csr storage format. *ACM SIGPLAN Notices 51*, 8 (2016), 1–2.

[52] Miller, J., and Southern, G. Recommender system for animated video. *Issues Inform Syst 15*, 2 (2014), 321–7.

[53] Naumov, M., Mudigere, D., Shi, H.-J. M., Huang, J., Sundaraman, N., Park, J., Wang, X., Gupta, U., Wu, C.-J., Azzolini, A. G., et al. Deep learning recommendation model for personalization and recommendation systems. *arXiv preprint arXiv:1906.00091* (2019).

[54] Netflix. url : https://www.netflix.com, 2021.

[55] Ning, X., Desrosiers, C., and Karypis, G. A comprehensive survey of neighborhood-based recommendation methods. In *Recommender systems handbook*. Springer, 2015, pp. 37–76.

[56] Oldridge, E., Perez, J., Frederickson, B., Koumchatzky, N., Lee, M., Wang, Z., Wu, L., Yu, F., Zamora, R., Yilmaz, O., et al. Merlin: A gpu accelerated recommendation framework.

[57] Pan, R., Zhou, Y., Cao, B., Liu, N. N., Lukose, R., Scholz, M., and Yang, Q. One-class collaborative filtering. In *2008 Eighth IEEE International Conference on Data Mining* (2008), IEEE, pp. 502–511.

[58] Piñeres, M. F. C., Hernández, J., and Builes, J. A. J. Diseño de un sistema de recomendación en repositorios de objetos de aprendizaje basado en la percepción del usuario: caso rodas. *Ciencia e Ingeniería Neogranadina 21*, 1 (2011), 51–72.

[59] Ricci, F., Rokach, L., and Shapira, B. Recommender systems: introduction and challenges. In *Recommender systems handbook*. Springer, 2015, pp. 1–34.

[60] Rich, E. User modeling via stereotypes. *Cognitive science 3*, 4 (1979), 329–354.

[61] Rossetti, M., Stella, F., and Zanker, M. Contrasting offline and online results when evaluating recommendation algorithms. In *Proceedings of the 10th ACM conference on recommender systems* (2016), pp. 31–34.

[62] Schifferer, B., Titericz, G., Deotte, C., Henkel, C., Onodera, K., Liu, J., Tunguz, B., Oldridge, E., De Souza Pereira Moreira, G., and Erdem, A. Gpu accelerated feature engineering and training for recommender systems. In *Proceedings of the Recommender Systems Challenge 2020*. 2020, pp. 16–23.

[63] Scholar, G. url : https://scholar.google.com, 2021.

[64] Silveira, T., Zhang, M., Lin, X., Liu, Y., and Ma, S. How good your recommender system is? a survey on evaluations in recommendation. *International Journal of Machine Learning and Cybernetics 10*, 5 (2019), 813–831.

[65] Spotify. url : https://www.spotify.com, 2021.

[66] Springer. url : https://www.springer.com, 2021.

[67] Stanley I. Grossman S., J. J. F. G. *Álgebra lineal*, vol. 7. Mc Graw Hill, 2012.

[68] Takács, G., and Tikk, D. Alternating least squares for personalized ranking. In *Proceedings of the sixth ACM conference on Recommender systems* (2012), pp. 83–90.

[69] tomshardware. url : https://www.tomshardware.com/news/nvidia-gtx-1650-ti-specs-listed-benchmark, 2020.

[70] Trivago. url : https://www.trivago.com.mx, 2021.

[71] Twitter. url : https://www.twitter.com, 2021.

[72] Wang, R., Fu, B., Fu, G., and Wang, M. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*. 2017, pp. 1–7.

[73] Wang, Z., Liu, Y., and Chiu, S. An efficient parallel collaborative filtering algorithm on multi-gpu platform. *The Journal of Supercomputing 72*, 6 (2016), 2080–2094.

[74] Yang, X.-S., and Deb, S. Cuckoo search via lévy flights. In *2009 World congress on nature & biologically inspired computing (NaBIC)* (2009), Ieee, pp. 210–214.

[75] Youtube. url : https://www.youtube.com, 2021.