Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Un nuevo paradigma basado en recuperación de información para sistemas de recomendación

Tesis para obtener el título de:

Maestro en Ciencias de la Computación

Presenta: Victor Giovanni Morales Murillo

Asesores: Dr. David Eduardo Pinto Avendaño

Dra. Darnes Vilariño Ayala

Puebla, Pue.

Julio 2020

Agradecimientos

Agradezco a Dios por darme la oportunidad de vivir esta grata experiencia de estudiar una maestría en Ciencias de la Computación.

Agradezco a mi padre, madre, hermano, esposa e hijo por tener su apoyo incondicional y por brindarme siempre su amor en cada momento de mi vida.

Agradezco a mis asesores el Dr. David Pinto y la Dra. Darnes Vilariño, por guiarme y apoyarme durante mis estudios de maestría.

Agradezco al comité tutorial por su retroalimentación y sus observaciones para mi tesis de maestría.

Agradezco a mis profesores por su dedicación y esfuerzo para compartirme sus conocimientos.

Agradezco a mis compañeros y amigos por compartir parte de su tiempo durante mis estudios.

Agradezco a la secretaría de posgrado del Facultad de Ciencias de la Computación por apoyarme durante mis estudios.

Agradezco al Consejo Nacional de Ciencia y Tecnología (CONACYT) por la beca que me otorgo para mis estudios de maestría.

Dedicatoria

A mi hijo Victor Caleb Morales Pérez

Resumen

Los sistemas de recomendación representan un alto impacto económico, social y tecnológico porque son fundamentales para muchas compañías como Google, Facebook, Spotify, Netflix, Amazon y muchas más. Estos sistemas surgen por el exceso de información que existe en el Internet para sugerir información de ítems o artículos relevantes para el usuario con respecto a sus preferencias y necesidades. Los datos de los usuarios y de los ítems se obtienen de grandes volúmenes de información, sin embargo, el manejo de grandes volúmenes de información representa un gran problema para los sistemas de recomendación que utilizan matrices de similitud porque el costo de computó es grande y el tiempo de respuesta es alto. Por tal motivo, en el laboratorio de Ingeniería del Lenguaje y del Conocimiento (LKE) dirigido por el Dr. David Pinto se tiene el interés por generar un nuevo paradigma basado en recuperación de información para sistemas de recomendación que maneje altos volúmenes de datos y que presente mínimos tiempos de respuesta. Además, este proyecto surge por el interés de la Benemérita Universidad Autónoma de Puebla para recomendar sus productos y servicios. En esta tesis se utiliza un conjunto de datos estándar de películas obtenido del sitio web de MovieLens que es ampliamente usado en la educación, la investigación y la industria. El conjunto de datos se pre-procesa para implementar dos paradigmas de recuperación de información para los sistemas de recomendación. Se implementa un paradigma para la similitud entre usuarios representando a cada usuario como un documento y se implementa un paradigma para la similitud entre películas (ítems) representando a cada película como un documento. Esto permite la implementación del modelo de espacio vectorial de recuperación de información con el algoritmo de CosineScore para buscar usuarios o películas similares. Después, con base en el algoritmo de recuperación de información se realiza la implementación de las técnicas de recomendación de filtrado colaborativo basado en el usuario y el filtrado colaborativo basado en el ítem, se realizan cuatro experimentos con estas técnicas de recomendación para analizar y comparar los resultados utilizando las métricas de evaluación de precisión, recall y f-measure. También, se analizan los resultados en tiempos de ejecución con base en el tiempo de indización y de recomendación. Los resultados observados con las técnicas de recomendación aplicadas demostraron que es fundamental en el conjunto de datos el número de valoraciones de los usuarios y las películas para mejorar los resultados de precisión, recall y f-measure. Finalmente, al utilizar algoritmos de recuperación de información con las técnicas de filtrado colaborativo se evita trabajar con algoritmos mínimamente cuadráticos que tradicionalmente se aplican sobre matrices.

Contenido

Lista de figuras

Lista de tablas

Lista de algoritmos

Capítulo 1
Introducción

En el Internet el número de usuarios y la información está creciendo exponencialmente, en el estudio llamado *Digital around the world in 2019* (Kemp, 2019) se presenta que durante el 2018 se conectaron más de 360 millones de nuevos usuarios a Internet, esto representa más de un millón de nuevos usuarios que utilizan el Internet por día. Un usuario en Internet navega aproximadamente 6 horas y media al día e invierte la mayor parte de su tiempo en las redes sociales. Además, dos tercios de la población mundial cuentan con dispositivos móviles que han impulsado fuertemente el comercio electrónico.

Actualmente, existen altos volúmenes de información en el Internet que limitan la búsqueda de productos y servicios relevantes para los millones de usuarios que navegan en el ciberespacio. Por tal motivo, surgen los sistemas de recomendación (SR) que se encargan de sugerir productos o servicios relevantes para sus usuarios con base en las preferencias de cada usuario. Las principales compañías de tecnología como son Google, Netflix, Amazon, YouTube, Facebook, Spotify y otras, se fundamentan en sus propios sistemas de recomendación e invierten grandes cantidades de dinero en la actualización de estos sistemas.

En esta tesis se propone un nuevo paradigma basado en recuperación de información (RI) para sistemas de recomendación que maneje altos volúmenes de información con mínimos tiempos de respuesta en comparación de los modelos clásicos de recomendación basados en matrices de similitud.

1.1 Antecedentes

En la última década, los sistemas de recomendación representan un alto impacto económico, social y tecnológico a nivel mundial porque las principales compañías de tecnología como Google, Facebook, Twitter, Linkedln, Netflix, Amazon, Microsoft, Yahoo!, eBay, Pandora, Spotify y otras, utilizan los sistemas de recomendación dentro de sus principales servicios e invierten grandes cantidades de dinero para optimizar sus algoritmos de recomendación (Jannach, Resnich, Tuzhilin, & Zanker, 2016).

En el dominio de la transmisión de películas y series, el sistema de recomendación de Netflix es primordial para la compañía porque al menos el 75% de sus descargas y alquileres provienen de su sistema de recomendación (Jannach, Resnich, Tuzhilin, & Zanker, 2016). En el dominio del comercio electrónico, Amazon creó su propia técnica de recomendación llamada filtrado colaborativo basado en ítems para ofrecer a sus clientes una experiencia de compra personalizada (Linden, Smith, & York, 2003). En el dominio de la transmisión de música, Spotify puede competir contra tres compañías más grandes como Apple, Google y Amazon, porque el sistema de recomendación de Spotify es mejor que el de su competencia (Nelson, 2018).

Los sistemas de recomendación son fundamentales para cualquier tipo de compañía y es una área de investigación que continua creciendo, sin embrago, las investigaciones de los sistemas de recomendación continúan enfrentándose a grandes problemas como son el procesamiento masivo de datos, la reducción de los tiempos de respuesta, la recomendación más personalizada, la optimización de los algoritmos y mejorar la forma de evaluación de los sistemas de recomendación.

1.2 Planteamiento del problema

Los sistemas de recomendación se originan como resultado al exceso de información expuesta en diversos ámbitos, que limita a los usuarios a determinar los productos que son relevantes para ellos (Tovar, Montoya, & Martelo, 2017), por lo tanto, los SR se utilizan para sugerir una amplia gama de artículos o ítems relevantes para los usuarios, por ejemplo, páginas web, productos de consumo, películas, canciones, amigos, artículos de noticias, restaurantes y muchos más. Los datos de las características de los usuarios y de los artículos a recomendar se obtienen de grandes volúmenes de información, sin embargo, el manejo de grandes volúmenes de información representa un gran problema para los sistemas de recomendación que utilizan matrices de similitud porque el costo de computó es grande y el tiempo de respuesta es alto (Jannach, Resnich, Tuzhilin, & Zanker, 2016), por tal motivo, en los SR surge el interés por el manejo de grandes volúmenes de información y por minimizar los tiempos de respuesta.

En el laboratorio de Ingeniería del Lenguaje y del Conocimiento (LKE) dirigido por el Dr. David Eduardo Pinto Avendaño de la Benemérita Universidad Autónoma de Puebla (BUAP) se realizan investigaciones sobre el área de recuperación de información y en una investigación se implementó un sistema de recuperación de información como mecanismo de interacción con un robot humanoide (Ramos, 2017). Actualmente, el LKE tiene el interés por generar un nuevo paradigma basado en recuperación de información para sistemas de recomendación que maneje altos volúmenes de datos y que presente mínimos tiempos de respuesta. Además, este proyecto surge por el interés de la Benemérita Universidad Autónoma de Puebla para recomendar sus productos y servicios.

1.3 Objetivos

A continuación se presenta el objetivo general, los objetivos específicos y las preguntas de investigación.

1.3.1 Objetivo general

Generar un nuevo paradigma basado en recuperación de información para sistemas de recomendación.

1.3.2 Objetivos específicos

- Proponer un nuevo paradigma de similitud entre productos que sea altamente escalable para el manejo de grandes volúmenes de datos y con mínimos tiempos de respuesta.

- Proponer un nuevo paradigma de similitud entre perfiles de usuarios que sea altamente escalable para el manejo de grandes volúmenes de datos y con mínimos tiempos de respuesta.

- Implementar un modelo de recomendación que utilice los nuevos paradigmas propuestos para impactar en dos procesos, el tratamiento masivo de datos y la reducción drástica en los tiempos de respuesta.

- Evaluar los paradigmas propuestos en un conjunto de datos estándar.

1.3.3 Preguntas de investigación

- ¿Es posible con el paradigma basado en el modelo de recuperación de información reducir los tiempos de respuesta con respecto a los modelos basados en matrices de similitud para un sistema de recomendación?

- ¿Es posible con el paradigma basado en el modelo de recuperación de información procesar un conjunto mayor de volumen de información con respecto a los modelos basados en matrices de similitud para un sistema de recomendación?

- ¿Es posible con el paradigma basado en el modelo de recuperación de información obtener un mejor rendimiento con respecto a los modelos basados en matrices de similitud para un sistema de recomendación?

1.4 Organización de la tesis

La organización de la tesis es la siguiente. En el *capítulo 1*, se presenta la introducción que contienen el planteamiento del problema, los antecedentes y los objetivos. En el *capítulo 2*, se presentan los trabajos relacionados a esta tesis. En el *capítulo 3*, se presenta el marco teórico que contiene las técnicas de recomendación, las técnicas de recuperación de información y las métricas de evaluación para los sistemas de recomendación. En el *capítulo 4,* se presenta la propuesta del nuevo paradigma basado en recuperación de información para sistemas de recomendación. En el *capítulo 5*, se presentan los experimentos y los resultados. En el *capítulo 6*, se presentan las conclusiones. Finalmente, se presentan las referencias utilizadas.

Capítulo 2
Trabajos relacionados

En este capítulo se presentan los principales trabajos relacionados a los sistemas de recomendación organizados por las investigaciones de las principales compañías de tecnología como son Netflix, Amazon y Spotify. En el primer trabajo sobre Netflix, se presenta una breve historia de los sistemas de recomendación hasta terminar en el desafío de Netflix que plantea el problema de los sistemas de recomendación como un completado de matrices, sin embargo, se menciona que el futuro de los SR continua con grandes retos. Esto sirve para observar las ventajas y desventajas de usar matrices en los sistemas de recomendación para justificar la investigación de esta tesis. En el segundo trabajo sobre Amazon, se presenta un dominio de recomendaciones de productos en el comercio electrónico que es similar al dominio de esta tesis, que consiste en recomendar productos y servicios de la Benemérita Universidad de Puebla. Esto sirve para estudiar las técnicas de recomendación que se utilizaron en un dominio similar al de esta tesis. En el tercer trabajo sobre Spotify, se presentan la combinación de diferentes técnicas de recomendación para generar un sistema de recomendación híbrido. Esto sirve para analizar otras técnicas de recomendación y seleccionar las técnicas adecuadas que se utilizarán en esta tesis.

2.1 Más que un desafió de completar matrices en Netflix

La compañía de Netflix ofrece un servicio de streaming que les permite a sus miembros ver una gran variedad de series, películas y documentales galardonados en miles de dispositivos conectados a Internet, en la Figura 1 se muestra el sitio web de Netflix que tiene diferentes listas de recomendaciones para el usuario (Netflix, 2020). Actualmente, Netflix está activo en 190 países con más de 100 millones de suscripciones y diariamente se ven más de 125 millones de horas de video en su plataforma. El sistema de recomendación de Netflix es primordial para la compañía porque al menos el 75% de sus descargas y alquileres provienen de su sistema de recomendación, además, en estudios realizados por la compañía se observó que las personas son malas para elegir entre muchas opciones. Por tal motivo, Netflix busca continuamente optimizar su sistema de recomendación para aumentar la satisfacción de sus clientes y, por lo tanto, los ingresos generales de la compañía (Postmus & Bhulai, 2018).

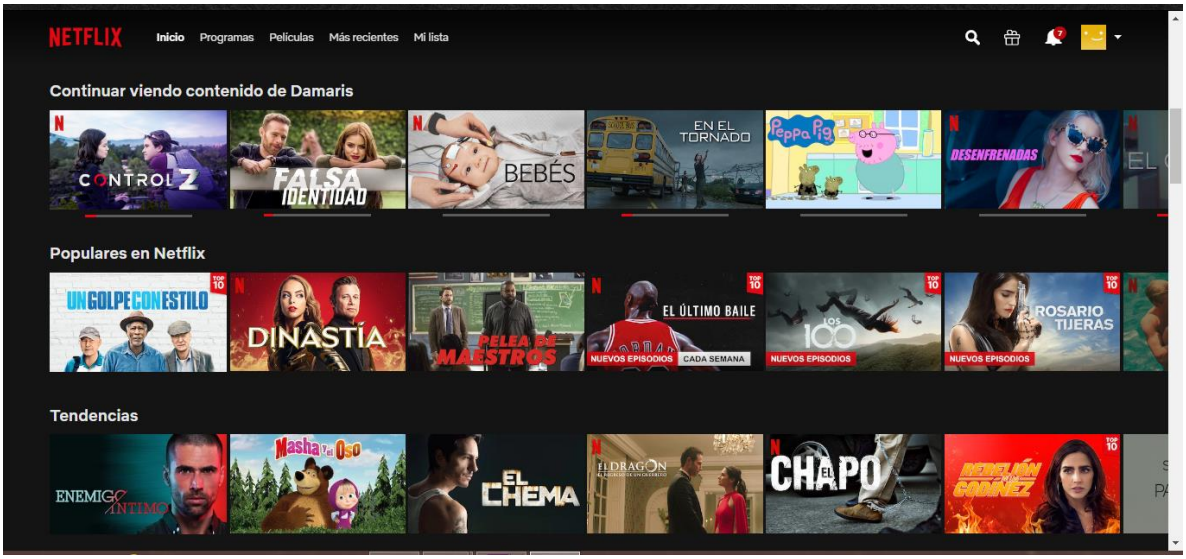

Figura 1. Sitio web de Netflix (Netflix, 2020).

En el trabajo titulado *Recommender Systems—Beyond Matrix Completion* de (Jannach, Resnich, Tuzhilin, & Zanker, 2016) se presenta que Netflix organizó una competencia llamada Netflix Prize para mejorar sus algoritmos de recomendación y otorgó un premio de un millón de dólares al equipo ganador, sin embargo, se menciona que el éxito de los sistemas de recomendación depende de más que un desafío de Netflix.

El Netflix Prize formuló el problema de recomendación como un problema de completado de una matriz que consiste en tres pasos que se muestran en la Figura 2. El primer paso es dada una matriz dispersa M, cree una matriz M' ocultando

aleatoriamente un subconjunto H de los rankings conocidos. El segundo paso es predecir los valores (H*) de los rankings ocultos utilizando M'. El tercer paso es evaluar la diferencia entre los rankings que se predijeron y que son verdaderos utilizando la métrica de evaluación del error cuadrático medio (RMSE). El conjunto de datos de Neflix Prize contenía 100 millones de calificaciones reales de sus clientes y el reto consistía en superar el sistema de recomendación de Netflix en un 10% en RMSE. Más de 5,000 equipos se inscribieron a la competencia, el premio fue otorgado en el 2009 y se obtuvieron grandes avances en las tareas de predicción de los rankings utilizando el aprendizaje automático.

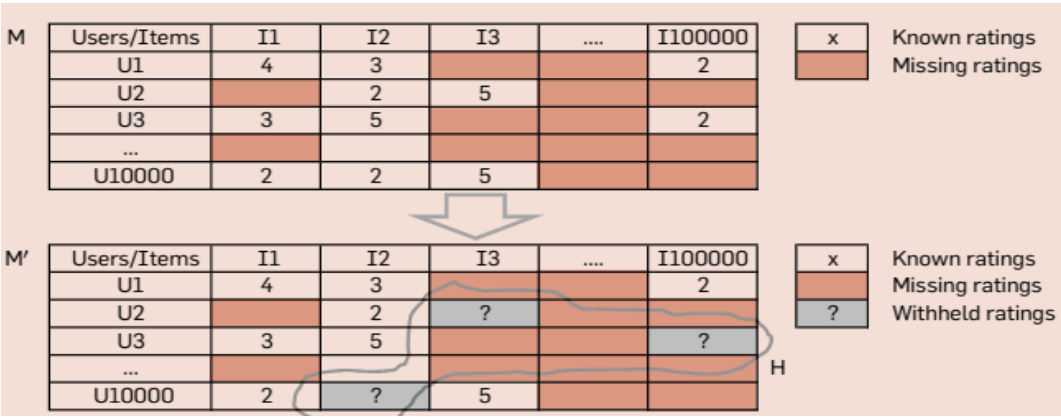

Figura 2. Problema de recomendación de Netflix Prize formulado como un problema de completado de matriz (Jannach, Resnich, Tuzhilin, & Zanker, 2016).

Los retos a los que se enfrenta el problema de recomendación visto como una tarea de completación de matrices son los siguientes. (1) Las métricas de evaluación de los sistemas de recomendación que son precision (precisión), recall (exhaustividad), f-measure (valor-f), RMSE (raíz del error cuadrático medio) y MAE (error absoluto medio) pueden ser engañosas, porque los SR predicen el pasado y no el futuro, es decir, la evaluación de los SR se realiza con las predicciones de los ratings ocultos de los usuarios y no con los ratings faltantes como se mostró en la Figura 1. La alternativa es realizar pruebas del tipo A/B con clientes reales en vivo, sin embargo, para las compañías no es fácil hacer público ese tipo de pruebas por la privacidad de los datos. (2) No todos los ítems se deben predecir, solo los que son más relevantes para el usuario con un rating de 4 o 5 estrellas. En el Netflix Prize, el RMSE fue calculado para todos los ítems que incluían los menos relevantes de 1, 2 y 3 estrellas, esto representa para una matriz que contienen 1 millón de ítems un alto procesamiento de datos y por lo cual, altos tiempos de respuesta. La alternativa para este reto es la recuperación de información para obtener un top de *N* ítems más relevantes para el usuario y posteriormente predecir su ranking. Sin embargo, las métricas de evaluación de RI continúan presentando el problema del reto anterior de predecir el pasado y no el futuro. (3) La novedad y la diversidad de los ítems sugeridos por los sistemas de recomendación es un gran reto porque se debe encontrar un equilibrio entre los ítems

novedosos y familiares para el usuario, con esto se buscan recomendaciones más personalizadas para cada usuario. Una alternativa son los sistemas de recomendación híbridos que combinan diferentes técnicas de recomendación, sin embargo, el dominio donde se aplicarán los SR es fundamental para seleccionar las técnicas de recomendación. (4) El contexto es importante porque los algoritmos de contexto pueden adaptar las recomendaciones tomando en cuenta los factores de ubicación, tiempo, estado de ánimo y la presencia de otras personas.

En esta tesis se aborda el problema desde el punto de vista de la recuperación de información para procesar altos volúmenes de información y obtener mínimos tiempos de respuesta. Además, se compararán diferentes técnicas de recomendación para mejorar la novedad y diversidad de las recomendaciones. En el futuro se pueden agregar algoritmos de contexto y realizar pruebas A/B.

2.2 Recomendaciones en el comercio electrónico con la técnica de Amazon

La compañía de Amazon es una de las 500 mayores empresas de Estados Unidos de América y es un líder global en el comercio electrónico. En la actualidad, Amazon ofrece gran variedad de productos, desde libros hasta raquetas de tenis o diamantes. Amazon tiene presencia directa en Estados Unidos, Reino Unido, Alemania, Francia, Italia, España, Japón, Canadá, China y en la mayoría de los países del mundo. La innovación tecnológica es la base para la expansión de Amazon y permite a sus clientes tener diferentes categorías de productos adaptados a sus necesidades y a un precio más bajo. En la Figura 3, se muestra el sitio web de Amazon que muestra las recomendaciones en la parte de *enlaces principales para ti* y en *Amazon del usuario* (Amazon, 2020).

Figura 3. Sitio web de Amazon (Amazon, 2020).

En el trabajo titulado *Amazon.com Recommendations Item-to-Item Collaborative Filtering* de (Linden, Smith, & York, 2003) se menciona que Amazon ofrece una experiencia de compra personalizada para sus clientes gracias a su sistema de recomendación. En Amazon.com, se utilizan algoritmos de recomendación para personalizar la tienda en línea para cada cliente, la tienda cambia radicalmente en función de los intereses del cliente. Por tal motivo, Amazon creó su propia técnica de recomendación llamada filtrado colaborativo basado en el ítem, que consiste en calcular la similitud entre los ítems utilizando los rankings de los usuarios que a diferencia del filtrado colaborativo basado en el usuario que consiste en calcular la similitud entre los usuarios. Se observó que en el filtrado colaborativo basado en el usuario se presentaba un mayor costo de cómputo en el procesamiento masivo de datos y en el tiempo de respuesta porque calculan la similitud entre todos sus usuarios y se predicen todos los ítems faltantes de cada usuario, sin en cambio, en el filtrado colaborativo basado en el ítem sólo se calcula la similitud entre los ítems que le gustan al usuario con los ítems que fueron calificados por otros usuarios, como se muestra en el algoritmo de Amazon de la Figura 2.

```
For each item in product catalog, I1
  For each customer C who purchased I1
    For each item I2 purchased by
       customer C
      Record that a customer purchased I1
       and I2
  For each item I2
    Compute the similarity between I1 and I2
```

Figura 4. Algoritmo de Amazon.com titulado filtrado colaborativo basado en el ítem (Linden, Smith, & York, 2003).

A pesar que se reduce el costo de cómputo en el filtrado colaborativo basado en el ítem, el costo de cómputo sigue siento alto porque se genera una matriz con las similitudes de los ítems y en esa matriz se buscan los ítems similares a los ítems que le gustan al usuario. Esta compañía creo esta técnica porque en el 2003 tenían más de 29 millones de clientes y varios millones de artículos, por tal motivo, requerían de una técnica específica para ellos y con un menor costo de cómputo.

En esta tesis el dominio es la recomendación de productos y servicios de la Benemérita Universidad Autónoma de Puebla, el cual, es similar al dominio Amazon.com que recomienda sus productos, por tal motivo, el filtrado colaborativo basado en ítems se analizará e implementará con un nuevo paradigma basado en recuperación de información y no con matrices de similitud, con la finalidad, de procesar altos volúmenes de información y minimizar los tiempos de respuesta.

2.3 Sistema de recomendación de Spotify

La compañía de Spotify ofrece un servicio de música, podcasts y vídeos digitales en streaming que da acceso a millones de canciones y otros contenidos de artistas de todo el mundo. En la Figura 5, se muestra el reproductor web de música de Spotify que muestra un radar de recomendaciones generales para el usuario y una lista de recomendaciones por cada día de la semana (Spotify, 2020).

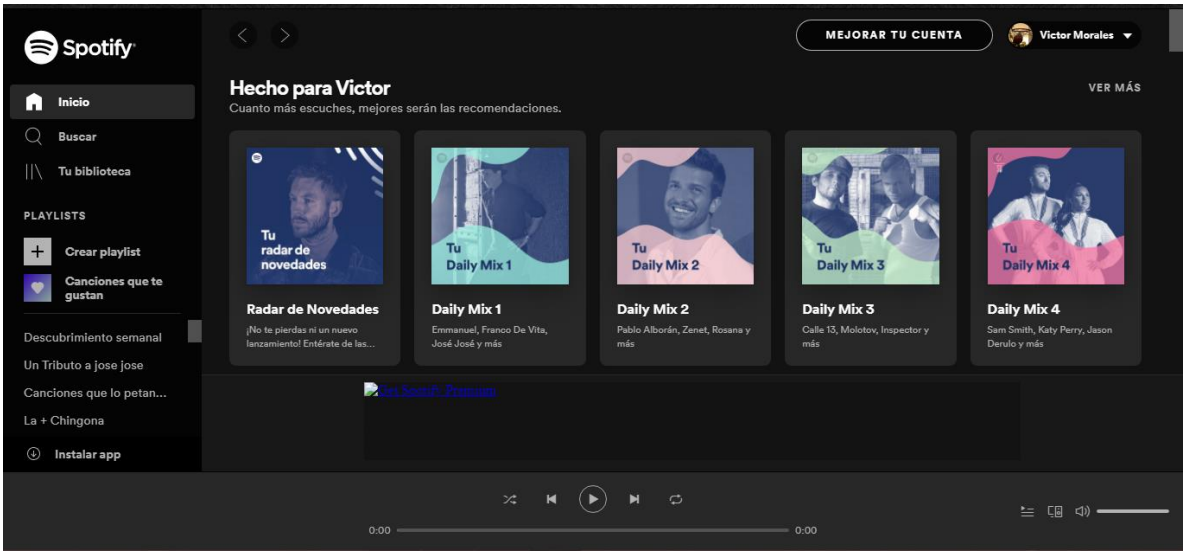

Figura 5. Reproductor web de música de Spotify (Spotify, 2020).

En el trabajo titulado *Spotify's Collaborative Filtering* de (Nelson, 2018) se menciona que en la industria de música continua existiendo Spotify después de unas de las compañías más grande del mundo que son Apple, Google y Amazon, porque Spotify tiene un mejor sistema de recomendación que el de su competencia. El sistema de recomendación de Spotify está formado por tres modelos de recomendación, en el primer modelo, se utiliza el procesamiento del lenguaje natural con las palabras de las descripciones de las canciones, en el segundo modelo se recomiendan canciones no tan populares, estos dos primeros modelos utilizan la técnica de recomendación llamada filtrado basado en contenido. En el tercer modelo, se utilizan las técnicas del filtrado colaborativo basado en el usuario y basado en el ítem, que generan una lista de reproducción llamada Discover Weekly para cada usuario, esta lista se genera con lo que escuchan otros usuarios similares y con otras canciones similares a las que escucha el usuario. Spotify tiene aproximadamente 140 millones de usuarios y por cada usuario se revisan más de 40 millones de canciones para encontrar las canciones que probablemente les gusten y que ya no tengan en su biblioteca de música. En el trabajo de Nelson, la técnica de filtrado colaborativo se puede aplicar con el uso del álgebra lineal para potenciar una función de costo de aprendizaje automático y cómo implementar un modelo de mínimos cuadrados alternos, puede minimizar la función de

costo para encontrar la lista de reproducción perfecta. Finalmente, se menciona que el sistema de recomendación de Spotify es el mejor que existe en la industria de la transmisión de música.

En esta tesis, se analizaran tres técnicas de recomendación: filtrado basado en el contenido, filtrado colaborativo basado en el usuario y filtrado colaborativo basado en el ítem. La primer técnica de filtrado basado en contenido tradicionalmente se implementa con técnicas de recuperación de información. Las técnicas de filtrado colaborativo tradicionalmente se implementan con matrices de similitud pero en este trabajo se implementaran con técnicas de recuperación de información para generar un nuevo paradigma de recomendación basado en RI.

Capítulo 3
Marco teórico

En este capítulo se presentan los conceptos, las técnicas de los sistemas de recomendación y de recuperación de información, y las métricas de evaluación, estos fundamentos sirven para seleccionar las técnicas de recomendación adecuadas para el dominio de esta tesis que es la recomendación de productos y servicios de la Benemérita Universidad Autónoma de Puebla. Además, se presentan las técnicas de recomendación que pueden ser implementadas con técnicas de recuperación de información para proponer un nuevo paradigma basado en recuperación de información para los sistemas de recomendación.

En la historia la primer técnica de recomendación surgió en la década de 1960 con el nombre de filtrado basado en contenido y utiliza técnicas de recuperación de información porque consiste en crear un perfil del usuario con los atributos textuales de sus preferencias de ítems, después, se realiza una consulta con el perfil del usuario para recuperar los documentos que coinciden con el perfil del usuario (Jannach, Resnich, Tuzhilin, & Zanker, 2016). La segunda técnica que surge es el filtrado colaborativo basado en el usuario, que consiste en aprovechar la opinión de otras personas para recomendar ítems a usuarios similares, esta técnica utiliza las valoraciones de los usuarios a los ítems para encontrar usuarios similares. La tercera técnica que surge es el filtrado colaborativo basado en el ítem que es desarrollada por la compañía de Amazon.com que consiste en encontrar productos similares con las valoraciones de los usuarios hacia los productos, que a diferencia del filtrado basado en contenido que busca la similitud de los productos con las características textuales de los productos. Muchas áreas de computación como recuperación de información, el aprendizaje automático, la interacción humano-computadora y otras áreas diferentes de computación como es el marketing continúan realizando investigaciones sobre los sistemas de recomendación porque es una área muy importante en la ciencia de los datos.

3.1 Definición de sistema de recomendación

Los sistemas de recomendación surgen por el exceso de información en el Internet que limita la búsqueda de productos relevantes para el usuario. El concepto de sistema de recomendación lo han presentado diferentes dominós, a continuación se presentaran algunas de estas definiciones.

El sistema de recomendación se define como un medio para ayudar y aumentar el proceso social de usar las recomendaciones de otros para tomar decisiones cuando no hay suficiente conocimiento personal o experiencia de las alternativas de ciertos productos o artículos como pueden ser películas, canciones, productos comerciales, páginas web, etcétera. (Resnick & Varian, 1997). En el comercio electrónico el sistema de recomendación se define como una herramienta que ayuda a los usuarios a buscar registros de conocimientos relacionados con el interés o preferencias de los usuarios (Schafer, Konstan, & Riedl, 1999). En el trabajo de (Isinkaye, Folajimi, & Ojokoh, 2015) el sistema de recomendación es una estrategia de toma de decisiones para usuarios en entornos de información complejos. Finalmente en el artículo de (Jannach, Resnich, Tuzhilin, & Zanker, 2016) se presentan las áreas que han contribuido en las investigaciones de sistemas de recomendación como son recuperación de información, aprendizaje automático, interacción humano-computadora y otras áreas como física y marketing. Estas áreas tienen en común el concepto de sistema de recomendación, que se encarga de almacenar perfiles de usuario o modelos de usuario para realizar recomendaciones personalizadas o adaptadas a la situación del usuario, con diferentes personas que generalmente reciben diferentes sugerencias de artículos.

Los sistemas de recomendación están compuestos por tres tipos de información: datos de contexto, datos de entrada y el algoritmo de recomendación. Los datos de contexto o datos base representan la información que el sistema tiene antes que inicie el proceso de recomendación. Los datos de entrada representan la información que el usuario comunica al sistema para generar una recomendación y los algoritmos de recomendación combinan los datos de contexto con los datos de entrada para generar las recomendaciones (Gómez, y otros, 2019). Usando estos tres componentes, se pueden identificar tres técnicas de recomendación principales el filtrado basado en contenido, el filtrado colaborativo basado en el usuario y el filtrado colaborativo basado en el ítem, sin embargo, existen otros técnicas como son el filtrado basado en el conocimiento, el filtrado basado en la demografía, el filtrado basado en la utilidad y los sistemas de recomendación híbridos. (Borges & Lorena, 2010). En esta tesis estudiaremos el filtrado basado en contenido, el filtrado colaborativo y los sistemas de recomendación híbridos.

El sistema de recomendación utiliza una función de utilidad que estima el interés que tiene un usuario por sus ítems recomendables, su definición formal es, sea U un conjunto de todos los usuarios e I el conjunto de todos los posibles ítems recomendables. Definimos $\bar{r}$ como una función que estima la utilidad de un ítem i para el usuario u, es decir:

$$\bar{r}: U \times I \rightarrow R, \qquad donde\ R\ es\ un\ conjunto\ ordenado.$$

Generalmente se crea un ranking de ítems basándose en el interés estimado y se seleccionan los n mejores ítems (Escamilla & Marcellin, 2017).

3.2 Técnicas de recomendación

En el trabajo titulado *Estado del arte en los sistemas de recomendación* de (Escamilla & Marcellin, 2017) se presentan las técnicas de recomendación como se muestra en la Figura 6, siendo la raíz los sistemas de recomendación se desglosan tres técnicas, el filtrado basado en contenido, el filtrado colaborativo e híbridos. Se observa que del filtrado colaborativo se desglosan otras técnicas basadas en modelos, basadas en confianza y basadas en memoria. La basada en modelos contiene las redes bayesianas, Slope One, Cloustering y Factorización de matrices. La basada en memoria contiene técnicas basadas en el usuario y basadas en el ítem. A continuación se describen estas técnicas.

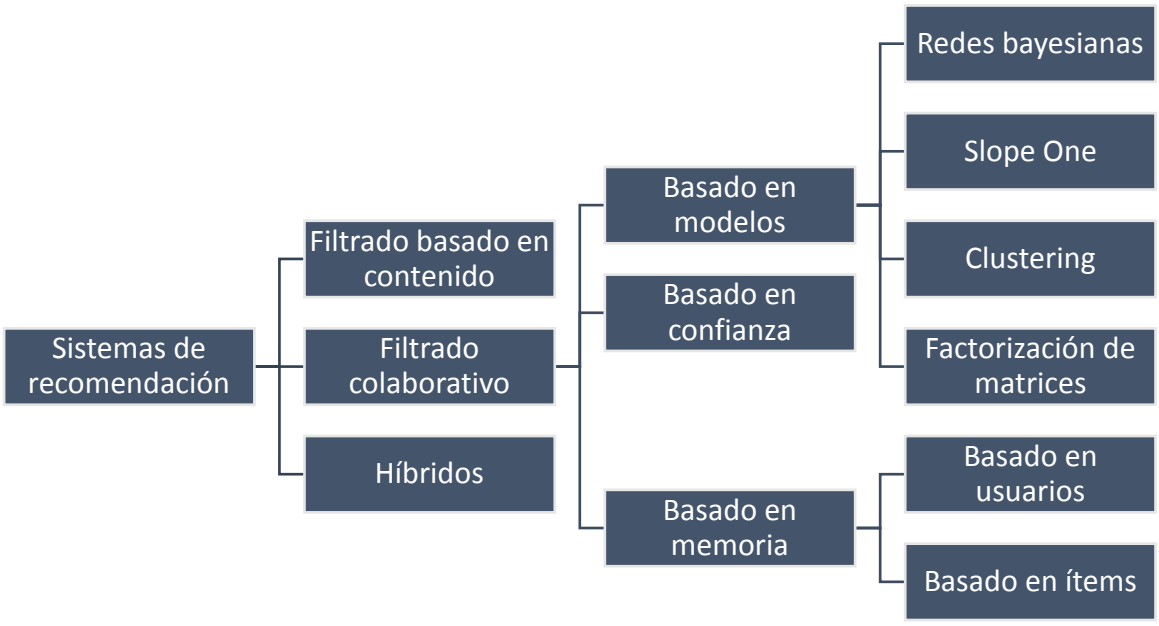

Figura 6. Técnicas de recomendación (Escamilla & Marcellin, 2017).

3.2.1 Filtrado basado en contenido

En el filtrado basado en contenido se crean perfiles de usuario de forma explícita o implícita. La forma explícita es por medio de formularios y la solicitud de información al usuario de sus preferencias, mientras que en la forma implícita se construye el perfil del usuario con base en el historial de los ítems preferidos del usuario. La representación del perfil del usuario es una consulta de texto con los atributos de los ítems favoritos del usuario. Estos sistemas de recomendación dependen la descripción de los ítems por medio de un conjunto de atributos llamados metadatos. Los metadatos pueden ser comentarios, etiquetas, descripciones textuales o contenido multimedia. Por tal motivo, utilizan técnicas de recuperación de información para representar a los ítems como documentos y el perfil del usuario como una consulta. La idea principal de esta técnica es comparar el perfil del usuario con los atributos de los ítems, los ítem más similares al perfil del usuario son los que se recomiendan. En la Figura 7 se muestra el procedimiento del filtrado basado en contenido que consiste en lo siguiente, (1) extraer los atributos de los ítems, (2) comparar atributos de los ítems con el perfil del usuario y (3) recomendar los ítems que sean más similares al perfil del usuario.

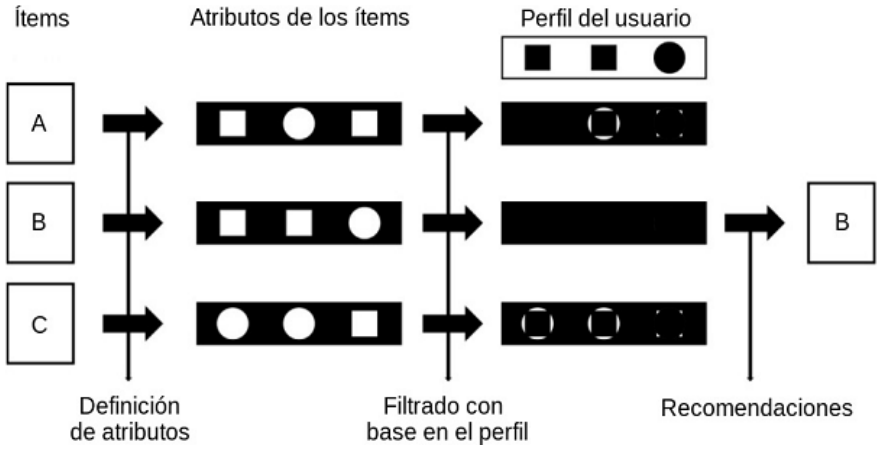

Figura 7. Filtrado basado en contenido (Escamilla & Marcellin, 2017).

La ventaja de esta técnica es que presenta buenas estimaciones en sus resultados porque son específicos el perfil del usuario y los atributos de los ítems, otra ventaja es que no presenta el problema de arranque en frío que no depende de las valoraciones de los usuarios hacia los ítems. La desventaja de esta técnica es que no puede distinguir entre contenidos de alta y baja calidad. También no presentan ninguna sorpresa o novedad en las nuevas recomendaciones en comparación de la técnica de filtrado colaborativo. Esta técnica está sujeta a especialización, en el que sólo se recomiendan artículos de las categorías evaluadas por el usuario en el pasado. Además, para ciertos tipos de medios como video, audio e imágenes, el análisis y la evaluación del contenido es más difícil, lo que hace complejo el uso de este enfoque o hace necesario el uso de algún tipo de descripción textual del elemento (Borges & Lorena, 2010). Finalmente, es alto el costo del mantenimiento de la información textual de los metadatos de los productos.

3.2.2 Filtrado colaborativo

El conjunto de técnicas más populares para el desarrollo de sistemas de recomendación son las del filtrado colaborativo. Esta técnica aprovecha la opinión de otros usuarios para realizar las recomendaciones, es decir, dado un usuario activo se buscan sus usuarios más similares con base en las valoraciones de sus productos y se recomiendan los productos de sus usuarios más similares que no ha visto el usuario activo.

Una ventaja de esta técnica en comparación del filtrado basado en contenido es que no requiere de descripciones textuales o de metadatos de los ítems, los perfiles de usuario se reducen a tripletas de *usuario*, *ítem* y *valoración*, en donde, la valoración representa el nivel del interés del usuario para un ítem, en la Figura 8 se muestra la matriz de valoraciones de los usuarios por los ítems, *u1, …, u5* son los usuarios, *A, …, D* son los ítems, los valores nulos significan que un usuario no valoró un cierto ítem. Estas valoraciones se obtienen de forma explícita por medio de los *likes* o una valoración numérica de estrellas que le puede otorgar un usuario a un ítem.

	A	**B**	**C**	**D**
u_1	3	2	2	1
u_2	3	1	2	5
u_3	∅	5	∅	2
u_4	5	4	2	4
u_5	2	3	4	2

Figura 8. Matriz de valoraciones (Escamilla & Marcellin, 2017).

La principal ventaja de este enfoque en comparación al enfoque basado en contenido es que es independiente de la representación de los artículos o ítems, es decir, no siempre recomendará productos similares a los gustos del usuario si no puede recomendar productos novedosos de otras categorías del agrado del usuario y puede emplearse directamente para diferentes tipos de artículos que las computadoras no pueden interpretar fácilmente, como imágenes o videos (Borges & Lorena, 2010). Las desventajas principales de esta técnica son: 1) El problema de arranque en frío es cuando no se puede hacer predicciones para elementos que aún no han sido evaluados por ningún usuario. 2) El problema de escasez es cuando la cantidad de artículos o ítems es mucho mayor que la cantidad de usuarios lo que dificulta en una matriz de evaluaciones encontrar elementos evaluados por un número suficiente de personas. 3) El problema de no tener un número suficiente de usuarios provoca que algunas personas no puedan beneficiarse con este enfoque porque tienen una opinión que no es diferente o similar a la de otros usuarios, lo que hace que las predicciones del sistema generalmente sean malas para este usuario en particular. 4) El problema de recomendar

elementos demasiados similares a los que ya consume el usuario (Borges & Lorena, 2010).

Basado en memoria

Los algoritmos trabajan con la matriz de valoraciones para estimar el interés de un usuario hacia un ítem. Se usan funciones de agregación para calcular la estimación de los ítems desconocidos para el usuario con una valoración nula. Esta estimación se puede calcular con la técnica de filtrado colaborativo basada en el usuario o con el filtrado colaborativo basado en el ítem. La principal desventaja de estas técnicas es que utilizan matrices de similitud para los usuario o los ítems lo que produce un alto costo de cómputo y cada vez que se ingresa un nuevo usuario o ítem se debe recalcular la similitud para todos los usuarios o ítems.

Filtrado colaborativo basado en el usuario

Esta técnica es la más utilizada del filtrado colaborativo que consiste en usar la función de agregación sobre los usuarios. Dado un usuario *u* se buscan sus usuarios más similares y las valoraciones desconocidas para el usuario *u* en la matriz de valoraciones se estiman con las valoraciones de sus usuarios similares. Tradicionalmente, se genera una matriz de similitud para los usuario, sin embargo, si una aplicación tienen 1 millón de usuarios, el tamaño de la matriz va ser de 1 millón por 1millon, lo que representa un alto costos de computó. Después, de obtener la matriz de similitud de los usuarios, se predicen las valoraciones desconocidas de los usuarios en la matriz de valoraciones con una función de agregación, este procedimiento es el que realiza el algoritmo *k*-vecinos. En la Figura 9, se muestra un ejemplo del algoritmo *k*-vecinos que obtiene tres usuarios similares o vecinos cercanos con respecto del usuario 4, en el ítem 7 dos vecinos lo calificaron con 4 y 5, entonces, el promedio es 4.5 que es la estimación que se predice que valorar el usuario 4 al ítem 7.

Figura 9. Ejemplo del algoritmo *k*-vecinos para el filtrado colaborativo basado en el usuario (Escamilla & Marcellin, 2017).

Filtrado colaborativo basado en el ítem

A diferencia del filtrado colaborativo basado en el usuario el filtrado colaborativo basado en el ítem estima las funciones de similitud y de agregación sobre los ítems. En la Figura 10 se observa el algoritmo *k*-vecinos basado en el ítem. Se observa que se va a estimar para el usuario 4 el ítem 1, entonces se buscan los ítems más similares al ítem 4 que tengan valoraciones similares por otros usuarios. Los ítems o vecinos más similares son el ítem 5, 9 y 10, el ítem 9 no se toma en cuenta porque el usuario 4 no lo calificó y sólo se promedian el ítem 10 y 5, el resultado para el ítem 1 y del usuario 4 es una predicción con un valor de 5.

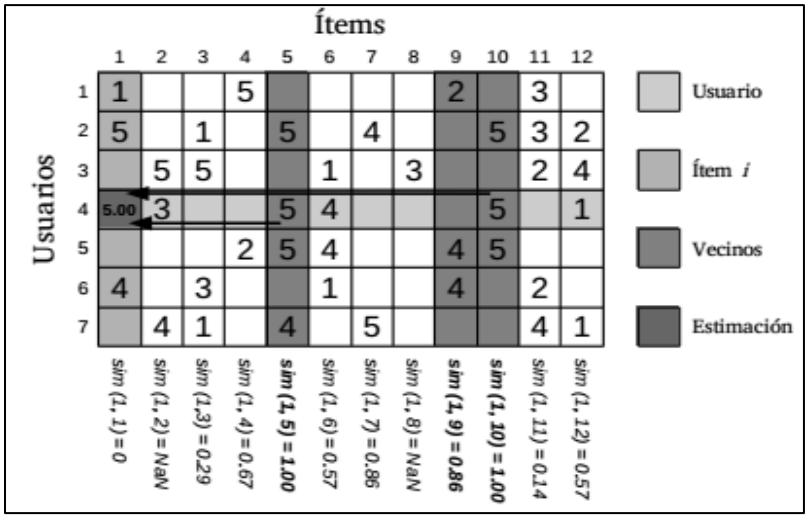

Figura 10. Ejemplo de algoritmo *k*-vecinos basado en ítems (Escamilla & Marcellin, 2017).

Las funciones de similitud más usadas para el filtrado colaborativo basado en el usuario ó en el ítem son la correlación de Pearson (COR), el coseno (COS), la correlación de rankeo de Spearman (SRC) y el error cuadrático medio (MAE). Las funciones de agregación más usadas son la media y la media ponderada (Escamilla & Marcellin, 2017).

Basado en confianza

Esta técnica también es conocida como de recomendaciones sociales. Esta técnica es similar al filtrado colaborativo basado en el usuario, lo que cambia es que en lugar de utilizar la noción de vecinos y similitud, se remplaza los vecinos por amigos y la similitud por confianza, en donde el propio usuario selecciona a sus amigos y la función de confianza es para estimar la relación de amigos de mis amigos. Esta técnica surge porque las personas confían en sus amigos y en los amigos de sus amigos en diferentes niveles.

Basado en modelos

A diferencia de los sistemas basados en memoria, esta técnica construye un modelo previamente a las recomendaciones con base en las calificaciones de los usuarios. Se plantea el problema como un problema de datos perdidos o clasificación y se utilizan algoritmo de aprendizaje automático como son redes neuronales, redes bayesianas, clustering o algoritmos de factorización de matrices.

Slope One

En esta técnica se estima una valoración desconocida con base en una función lineal con otra valoración conocida. Su forma es:

$$\bar{r}_{ui} = f(r_{uj}) + \delta_{ij}$$

El problema se reduce a buscar δ_{ij} para cada ítem. Para encontrar δ_{ij} se requiere de un conjunto de usuarios $\widehat{U}_{ij}$ que hayan calificado ambos ítems i y j, y se calcula el promedio de las diferencias:

$$\delta_{ij} = \frac{1}{|\widehat{U}_{ij}|} \sum_{u\ in\ \widehat{U}_{ij}} (r_{uj} - r_{ui})$$

En la Figura 11, se muestra un ejemplo de la técnica Slope One, en donde se estima la valoración del usuario *B* al ítem *J*, aplicando esta técnica se obtiene una estimación de 2.5.

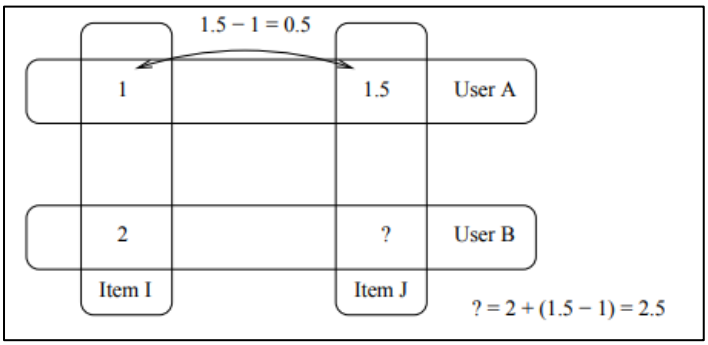

Figura 11. Ejemplo de técnica de Slope One (Lemire & Maclachlan, 2005).

Redes bayesianas

Esta técnica es similar a la técnica basada en memoria porque se estiman las valoraciones de $\bar{r}_{ui}$ como una función de agregación de las valoraciones que tienen de otros usuarios, sin embargo, en esta técnica se realiza una suma ponderada por un factor de probabilidad.

Clustering

Esta técnica también se conoce como *k*-medias que consiste en dividir el conjunto de usuarios en *k* subconjuntos de tal forma que los elementos de cada conjunto estén lo más cerca posible en relación a una medida de distancia dada. Cada *cluster* C_j está definido por N_j elementos y un centroide λ_j. El centroide λ_j es un punto en donde se minimiza la suma de las distancias de éste con todos los elementos que pertenecen al *cluster* C_j. Se pueden tomar los valores del centroide para estimar los valores de los ítems que el usuario no ha valorado.

Factorización de matrices

En esta técnica se asume que existen factores latentes que pueden explicar porque un usuario valora con una específica calificación a un ítem dado. Por ejemplo, en el dominio de las películas los factores latentes pueden ser los géneros, los actores, los directores, la historia, etc. Entonces, la valoración que otorga un usuario a un ítem con base en los *f* factores, se exprese de la siguiente forma:

$$r_{ui} = q_i \, p_u^T$$

Los símbolos q_i y p_u son vectores en $\mathbb{R}^f$ donde cada entrada se refiere a cada factor latente. En el caso de q_i, cada elemento refleja el grado de apego al factor dado y cada elemento de p_u refleja la relevancia de ese factor para los gustos del usuario. Al realizar esto para cada usuario y para cada ítem, se puede construir una matriz *R*.

$$R = QP^T$$

Para cada fila de Q corresponde a cada vector de q_i y para las filas de *P* son los vectores de p_u. El problema se transforma en tratar de estimar las matrices Q y P tomando como base la matriz de calificaciones usando *descomposición en valores singulares* (SVD).

3.2.3 Filtrado híbrido

Las técnicas de filtrado híbrido combinan varias técnicas de recomendación para optimizar los sistemas de recomendación para evitar las limitaciones y las desventajas de utilizar una sola técnica de recomendación. El objetivo es que al combinar algoritmos de recomendación se presente una mejor precisión y efectividad en las recomendaciones, que en lugar de utilizar un solo algoritmo. Las aproximaciones más utilizadas para combinar técnicas de recomendación se presentan a continuación.

Hibridación ponderada

La hibridación ponderada consiste en combinar las estimaciones de dos o más sistemas de recomendación utilizando una combinación lineal entre las estimaciones, así se puede ponderar la importancia o relevancia que tendrán las estimaciones hechas por cada sistema de recomendación que se están combinando al asignar distintos pesos (coeficientes) a cada una de ellas.

Hibridación en cascada

La hibridación en cascada crea un sistema de recomendación estrictamente jerárquico, es decir, en donde la primera técnica de recomendación es la raíz y la más fuerte, la segunda técnica de recomendación refina los resultados de la primera técnica y se repite el procedimiento si existe una tercera técnica de recomendación. La última técnica de recomendación es utilizar la más débil.

Hibridación por selección

Se calcula la estimación de cada uno de los sistemas de recomendación a combinar y con base en un conjunto de condiciones se selecciona el valor para la estimación global.

Hibridación por mezcla

Se mezclan las recomendaciones de varios sistemas de recomendación en una lista de recomendaciones global. Se puede utilizar el algoritmo de votación de Hare para ordenar las listas de recomendación dadas por cada técnica de recomendación (Escamilla & Marcellin, 2017).

3.3 Recuperación de información

Los sistemas de recomendación son similares a los sistemas de recuperación de información (SRI) porque ambos tienen como objetivo obtener información relevante de un conjunto de datos para el usuario, sin embargo, los SRI tienen un uso específico para buscar documentos relevantes con base en una consulta explicita del usuario, no almacenan perfiles de usuario y no recolectan información implícita del usuario. A pesar de sus diferencias, los sistemas de recomendación tradicionalmente han utilizado técnicas de recuperación de información en la técnica de filtrado basado en el contenido, en donde, se construye un perfil del usuario que representa la consulta en RI y se obtienen los ítems más semejantes al perfil del usuario, estos ítems representan los documentos en RI. Además, en el trabajo de (Costa & Roda, 2010) se presenta como un problema de recomendación que puede expresarse como un problema de

recuperación de información porque los documentos de RI pueden representar a los usuarios y a los ítems de un sistema de recomendación dependiendo de la técnica de recomendación.

La recuperación de información tiene como objetivo desarrollar modelos y algoritmos para recuperar información de repositorios de documentos. Tradicionalmente, los algoritmos de RI trabajan sobre texto, sin embargo, está creciendo el interés por recuperar información sobre imágenes, voz y video. RI plantea el problema de recuperación ad-hoc, que consiste en un usuario que ingrese una consulta con la descripción de la información requerida y el SRI devolverá como resultado una lista de documentos (Manning & Schütze, Foundations of statistical natural language processing, 2000). La definición académica de RI es encontrar material (generalmente documentos) de naturaleza no estructurada (generalmente texto) que satisfaga una necesidad de información desde grandes colecciones (generalmente almacenadas en computadoras). En un principio un grupo pequeño de personas utilizaban RI como son los bibliotecarios y los asistentes legales, pero, hoy día esto ha cambiado porque miles de millones de personas utilizan RI con los motores de búsqueda web o al buscar correos electrónicos, RI se está convirtiendo en la forma dominante de acceso a la información superando la forma tradicional de búsqueda de información en una base de datos estructurada.

RI también abarca otros tipos de problemas de información y de datos, más allá de su definición, por ejemplo, en esta tesis se utiliza RI para el problema de sistemas de recomendación. Otro uso de la RI es el de agrupar o clasificar documentos de acuerdo a su contenido, esta característica de RI sirve en los sistemas de recomendación para agrupar documentos de usuarios o ítems similares. Otra característica principal de RI es su escalabilidad porque un motor de búsqueda realiza la consulta entre miles de millones de documentos almacenados en millones de computadoras, es fundamental reunir documentos para indizar para desarrollar sistemas eficientes que funcionen a esta enorme escala. La escalabilidad de RI se utiliza en esta tesis para procesar altos volúmenes de datos y minimizar los tiempos de repuesta de los sistemas de recomendación que usan matrices de similitud.

La notación básica que se utiliza en RI es la siguiente. Un *documento* es conjunto de palabras, un *término* está compuesto por una o más palabras, además los términos se indizan en una estructura de datos para aplicar algoritmos de RI, generalmente el tipo de estructuras de datos que se utilizan son diccionarios o tablas hash. Una *colección* o *corpus* es un conjunto de documentos de los cuales se indizarán sus términos a un diccionario. Existen dos principales modelos de recuperación de información, el modelo booleano y el modelo de rankings de documentos que a continuación se describen (Manning, Raghavan, & Schütze, An Introduction to Information Retrieval, 2009).

3.3.1 Modelo booleano

Este modelo de recuperación de información utiliza consultas de términos expresadas de forma booleana utilizando los operadores AND, OR y NOT. Además, el modelo booleano asume que cada documento es un conjunto de palabras. En un principio, se utilizaban matrices binarias como estructura de datos que representaban la incidencia de los términos en los documentos como se muestra en la Figura 12. Las columnas representan los documentos de las obras de Shakespeare y los renglones representan los términos de las obras Shakespeare. El valor de 1 significa que un documento contiene un término y si el valor es 0 el documento no contiene al término. Un ejemplo de este modelo es al consultar "Brutus AND Caesar AND NOT Calpurnia", entonces se toman los vectores de Brutus, Caesar y Calpurnia, se aplica el complemento al vector de Calpurnia y después se aplica el AND para Brutus, Caesar y el complemento de Calpurnia: 110100 AND 110111 AND 101111 = 100100. El resultado es el documento de Antony and Cleopatra y el documento de Hamlet.

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	…
Antony	1	1	0	0	0	1	
Brutus	1	1	0	1	0	0	
Caesar	1	1	0	1	1	1	
Calpurnia	0	1	0	0	0	0	
Cleopatra	1	0	0	0	0	0	
mercy	1	0	1	1	1	1	
worser	1	0	1	1	1	0	
…							

Figura 12. Matriz binaria de incidencias términos-documentos en el modelo booleano de RI (Manning, Raghavan, & Schütze, An Introduction to Information Retrieval, 2009).

El utilizar matrices como estructura de datos en grandes colecciones de documentos con un conjunto desde millones hasta billones de palabras resulta ser muy costos computacionalmente. Además, la matriz estaría llena principalmente de valores de 0. Un ejemplo es una colección de documentos con 1,000 documentos y un conjunto de palabras con 1 millón de palabras, entonces la matriz sería del tamaño mil por un millón que es igual a mil millones. Por tal motivo, se cambió el tipo de estructura de datos a un diccionario que representa un índice invertido que se muestra en la Figura 13, en donde, las llaves son los términos y los valores son listas que contienen los documentos en los que aparece cada término ordenados de forma ascendente. Para procesar la consulta en un índice invertido se utiliza un algoritmo *merge* para cada operador *AND*, *OR* y *NOT*.

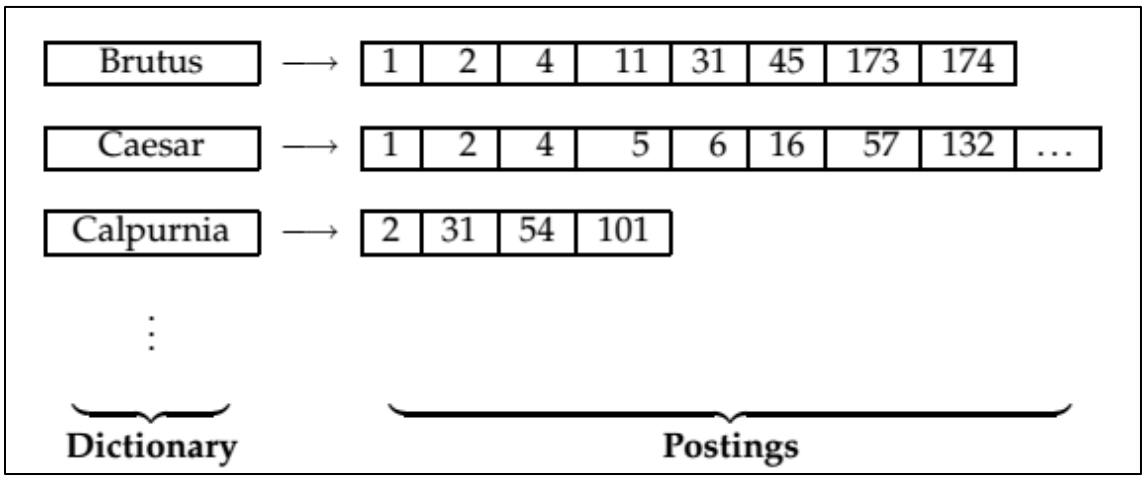

Figura 13. Índice invertido en el modelo booleano de RI (Manning, Raghavan, & Schütze, An Introduction to Information Retrieval, 2009).

Los resultados de estos modelos son precisos y exactos, por tal motivo, se usan en sistemas de información comercial, sin embargo, para conjuntos de documentos grandes y heterogéneos, los resultados suelen ser vacíos o enormes y difíciles de manejar.

3.3.2 Modelo de ranking de documentos

Este modelo utiliza consultas de texto libre con una o más palabras sin la necesidad de construir una expresión precisa con los operadores AND, OR o NOT. Con este modelo el sistema presenta como resultado una lista ordenada con los primeros K documentos que satisfacen mejor la consulta. Generalmente los usuarios prefieren las consultas booleanas porque son más transparentes con respecto a sus necesidades, sin embargo, las consultas de texto libre del modelo de ranking de documentos presenta mejores resultados porque un problema de los modelos booleanos es el equilibrio entre los operadores AND y OR, es decir, al usar un operador AND la búsqueda tiene una alta precisión pero una baja recuperación de documentos, y al usar un operador OR se obtienen búsquedas de alta recuperación pero baja precisión (Manning, Raghavan, & Schütze, An Introduction to Information Retrieval, 2009).

Modelo espacio vectorial

El modelo espacio vectorial representa un conjunto de documentos como vectores que comparten un espacio vectorial en común, en donde cada termino representa una dimensión. Este modelo es fundamental para operaciones de recuperación de información como son la puntuación de documentos en una consulta, la clasificación de documentos y la agrupación de documentos. También es importante representar la consulta como un vector en el mismo espacio vectorial que el conjunto de documentos para poder buscar los vectores de los documentos más cercanos al vector de la consulta utilizando el ángulo del coseno. Esta representación se muestra en la Figura 14, en

donde se denota $\vec{v}(d)$ como el vector de un documento *d*, en un principio se buscó obtener la diferencia de las magnitudes para calcular la similitud de los vectores, sin embargo existe un problema, dos vectores con un contenido muy similar pueden tener una gran diferencia por la longitud de los vectores. La forma estándar para compensar el problema de la longitud de los vectores es calcular la similitud con el ángulo del coseno con la siguiente formula:

$$sim(d1, d2) = \frac{\vec{v}(d1) \cdot \vec{v}(d2)}{|\vec{v}(d1)||\vec{v}(d2)|}$$

El numerador representa el producto punto de los vectores $\vec{v}(d1)$ y $\vec{v}(d2)$ y el denominador representa el producto de su distancia euclidiana.

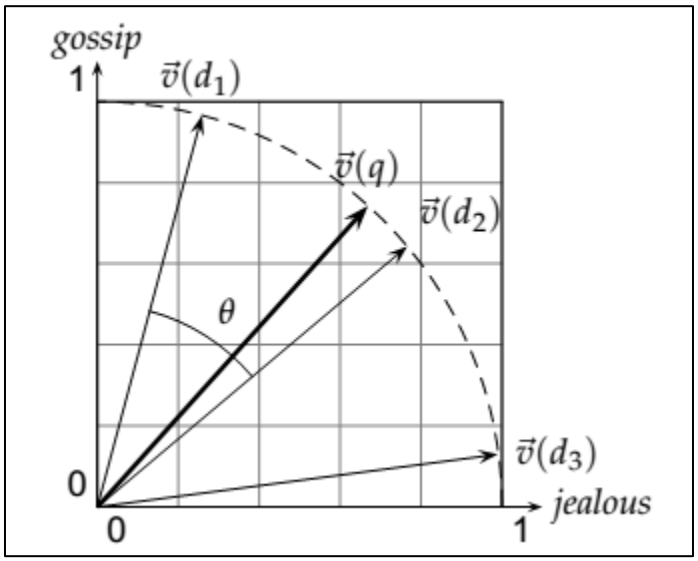

Figura 14. Representación del modelo de espacio vectorial (Manning, Raghavan, & Schütze, An Introduction to Information Retrieval, 2009).

El modelo espacio vectorial representa a la consulta como un documento para aplicar la similitud del coseno con el vector de la consulta $\vec{v}(q)$ y el vector de un documento $\vec{v}(d)$, esta representación se muestra en la siguiente formula.

$$sim(q, d) = \frac{\vec{v}(q) \cdot \vec{v}(d)}{|\vec{v}(q)||\vec{v}(d)|}$$

Además se utiliza la similitud del coseno como una medida para la puntuación del documento para esa consulta, por tal motivo se pueden obtener los primero *k* documentos más similares a la consulta.

Al representar los documentos como vectores que contienen términos representados por componentes, se calcula el peso de cada componente de un documento usando la frecuencia del término y la frecuencia inversa del documento *TF-IDF*. *TF-IDF* asigna un peso al termino *t* en el documento *d*, el peso es más alto cuando la frecuencia de *t*

es alta en un pequeño numeró de documentos, el peso es bajo cuando la frecuencia de *t* es baja en un documento o si aparece en muchos documentos. La fórmula de *TF-IDF* es la siguiente.

$$tf - idf_{t,d} = tf_{t,d} \times idf_t$$

Existen diferentes variantes para la fórmula de TF-IDF que se muestran en la Figura 15.

Term frequency		Document frequency		Normalization	
n (natural)	$\mathrm{tf}_{t,d}$	n (no)	1	n (none)	1
l (logarithm)	$1 + \log(\mathrm{tf}_{t,d})$	t (idf)	$\log \frac{N}{\mathrm{df}_t}$	c (cosine)	$\frac{1}{\sqrt{w_1^2+w_2^2+\ldots+w_M^2}}$
a (augmented)	$0.5 + \frac{0.5 \times \mathrm{tf}_{t,d}}{\max_t(\mathrm{tf}_{t,d})}$	p (prob idf)	$\max\{0, \log \frac{N-\mathrm{df}_t}{\mathrm{df}_t}\}$	u (pivoted unique)	$1/u$ (Section 6.4.4)
b (boolean)	$\begin{cases} 1 & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$			b (byte size)	$1/CharLength^{\alpha}$, $\alpha < 1$
L (log ave)	$\frac{1+\log(\mathrm{tf}_{t,d})}{1+\log(\mathrm{ave}_{t \in d}(\mathrm{tf}_{t,d}))}$				

Figura 15. Variantes de la fórmula de *TF-IDF* (Manning, Raghavan, & Schütze, An Introduction to Information Retrieval, 2009).

En esta tesis se calcula *TF-IDF* con la normalización del coseno calculando tf y $idf_{t,d}$ con las siguientes formulas:

$$tf = 1 + \log(tf_{t,d})$$

$$idf_{t,d} = \log(\frac{N}{df_t})$$

3.4 Métricas de evaluación

Es importante seleccionar las métricas de evaluación adecuadas para un sistema de recomendación con base en sus técnicas de recomendación y su implementación. Existen sistemas de recomendación que tienen como objetivo predecir la valoración que le otorgaría un usuario a un ítem y se recomiendan los ítems que presentan un mayor valoración. El otro tipo de sistemas de recomendación no tienen como objetivo predecir las valoraciones de los ítems, su único objetivo es recomendar ítems relevantes para el usuario. A continuación presentaremos las principales métricas de evaluación utilizadas para los dos tipos de sistemas de recomendación previamente mencionados (Herlocker, Konstan, Terveen, & Riedl, 2004).

3.4.1 Precisión, recall y f-measure

La precisión, el recall y la f-measure son las medidas utilizadas en el dominio de recuperación de información para medir la exactitud de un sistema de recuperación de información con respecto a los documentos relevantes que recupera para la consulta de un usuario. La métrica de precisión representa la cantidad de documentos relevantes recuperados por la consulta de un sistema de recuperación de información. La fórmula de la precisión es la siguiente:

$$Precisión = \frac{|\{documentos\ relevantes\} \cap \{documentos\ recuperados\}|}{|\{documentos\ recuperados\}|}$$

La métrica de recall representa la cantidad de documentos relevantes recuperados satisfactoriamente del total de documentos relevantes. La fórmula del recall es la siguiente:

$$Recall = \frac{|\{documentos\ relevantes\} \cap \{documentos\ recuperados\}|}{|\{documentos\ relevantes\}|}$$

La métrica de f-measure es la medida armónica entre precisión y recall. La fórmula de f-measure es la siguiente:

$$F - measure = 2 \times \frac{Precisión \times Recall}{Precisión + Recall}$$

3.4.2 RMSE y MAE

Compañías como Netflix requieren predecir las valoraciones o calificaciones que le otorgarían sus usuarios a ciertas películas, es decir, se predice si un usuario valoraría o calificaría con 1 o 5 estrellas una película. Entonces, para validar este tipo de sistemas se utilizan las métricas de RMSE y MAE.

La métrica de Root Mean Squared Error (RMSE) es la diferencia entre las valoraciones estimadas y las valoraciones reales. La fórmula de RMSE es la siguiente:

$$RMSE = \sqrt{E((\hat{\theta} - \theta)^2)}$$

La métrica del error absoluto medio, en inglés Mean Absolute Error (MAE) está definido por la siguiente fórmula.

$$MAE = \frac{\sum_{i=1}^{N} |p_i - v_i|}{N}$$

Capítulo 4
Paradigma basado en RI para los SR

En este capítulo se describe el procedimiento para la implementación de dos paradigmas basados en recuperación de información para los sistemas de recomendación, con el objetivo de procesar altos volúmenes de datos y obtener mínimos tiempos de respuestas en comparación con los sistemas de recomendación que utilizan matrices de similitud, el primer paradigma es para la similitud de los usuarios y el segundo paradigma es para la similitud entre los ítems. Para las pruebas de estos paradigmas se utilizó un conjunto de datos estandar llamado MovieLens del dominio de películas, este conjunto de datos es muy utilizado en las investigaciones de sistemas de recomendación. Se realizó un pre-procesamiento a este conjunto de datos para obtener la representación de los datos para los dos paradigmas. Se realizó la implementación de estos dos paradigmas con el lenguaje de programación de Python, se implementó una clase de recuperación de información con el modelo espacio vectorial en donde se mide la similitud de documentos con una consulta a través del ángulo del coseno. Se importó esta clase a otras dos clases que contienen el modelo de recomendación con base en dos técnicas de recomendación el filtrado colaborativo basado en el ítem y el filtrado colaborativo basado en el usuario. Se realizaron cuatro experimentos, en los primero tres experimentos se utilizó la técnica de filtrado colaborativo basado el ítem y en el cuarto experimento se utilizó la técnica de filtrado colaborativo basado en el usuario. Finalmente, se obtienen los resultados y se analizan para obtener las conclusiones de este trabajo de investigación. A continuación se describen cada uno de estos procedimientos.

4.1 Conjunto de datos

El conjunto de datos que se utilizó es de MovieLens, este conjunto de datos es para el dominio de las películas y fue lanzado desde 1997. En el 2014, obtuvo más de 140,000 descargas y más de 7,500 referencias en google académico, porque los conjuntos de datos de MovieLens son ampliamente usados en la educación, la investigación y la industria. Estos conjuntos de datos fueron creados por la actividad de los miembros en el sistema de recomendación de películas MovieLens, en donde solicitan a sus usuarios que valoren sus películas para recomendarles películas con base en el perfil del usuario. Los conjuntos de datos de MovieLens describen las preferencias de los usuarios hacia las películas con tuplas del tipo: <usuario, película, valoración, marca de tiempo> en donde la valoración se expresa en un rating de 0 a 5 estrellas para cada película.

La popularidad de este conjunto de datos refleja el incremento de las investigaciones de personalización y recomendación, en donde los conjuntos de datos tienen un valor fundamental para la validación de las investigaciones. Además, los datos de MovieLens son flexibles para los sistemas de recomendación y otras técnicas del área de ciencia de los datos. Dado a que las preferencias de las películas está sujeta a gustos personales, es un excelente dominio para probar tecnologías de recomendación (Maxwell & Konstan, 2015). En la Figura 13, se muestra el sitio web de MovieLens que presenta cuatro tipos de conjuntos de datos, (1) el recomendado para nuevas investigaciones, (2) el recomendado para educación y desarrollo, (3) los conjuntos de datos sintéticos y (4) otros conjuntos de datos. En esta tesis utilizamos el segundo tipo que es recomendado para la educación y desarrollo.

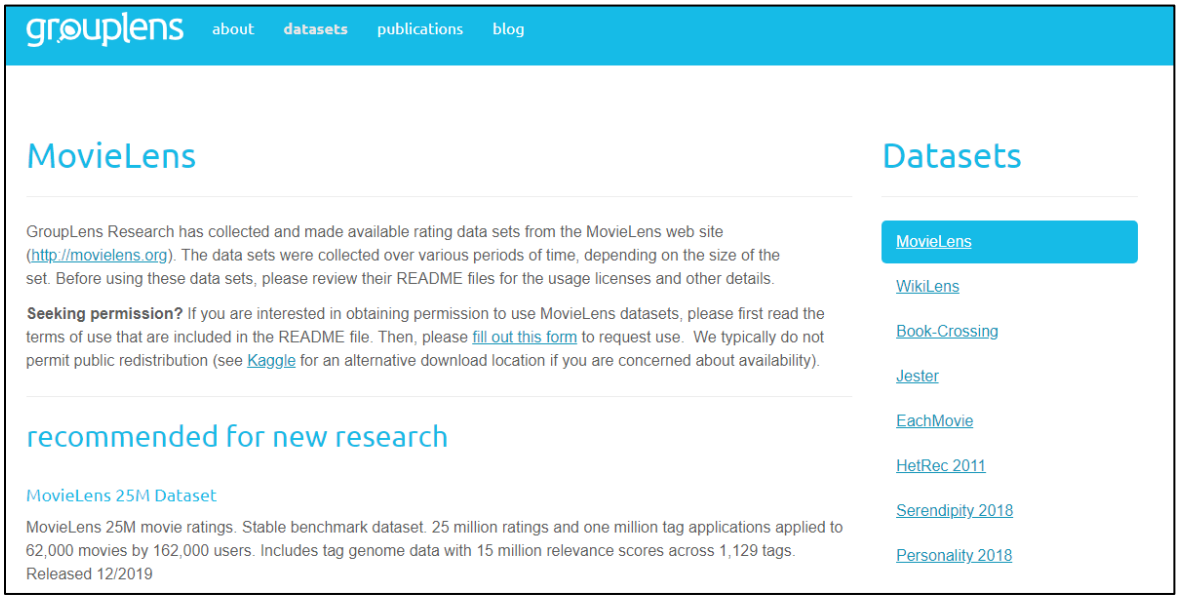

Figura 16. Sitio web de MovieLens.

En la Figura 17, se muestran los dos conjunto de datos uno pequeño y uno completo, en este trabajo utilizamos el pequeño que contiene 100,000 ratings, 3,600 etiquetas aplicadas a 9,000 películas por 600 usuarios. El nombre del archivo es ml-latest.zip con el tamaño de 1 MB. Se seleccionó este conjunto de datos para evaluar por primera vez los dos paradigmas propuestos y por su practicidad. En trabajos futuros se puede evaluar el sistema de recomendación de esta tesis con otro conjunto de datos para comparar resultados.

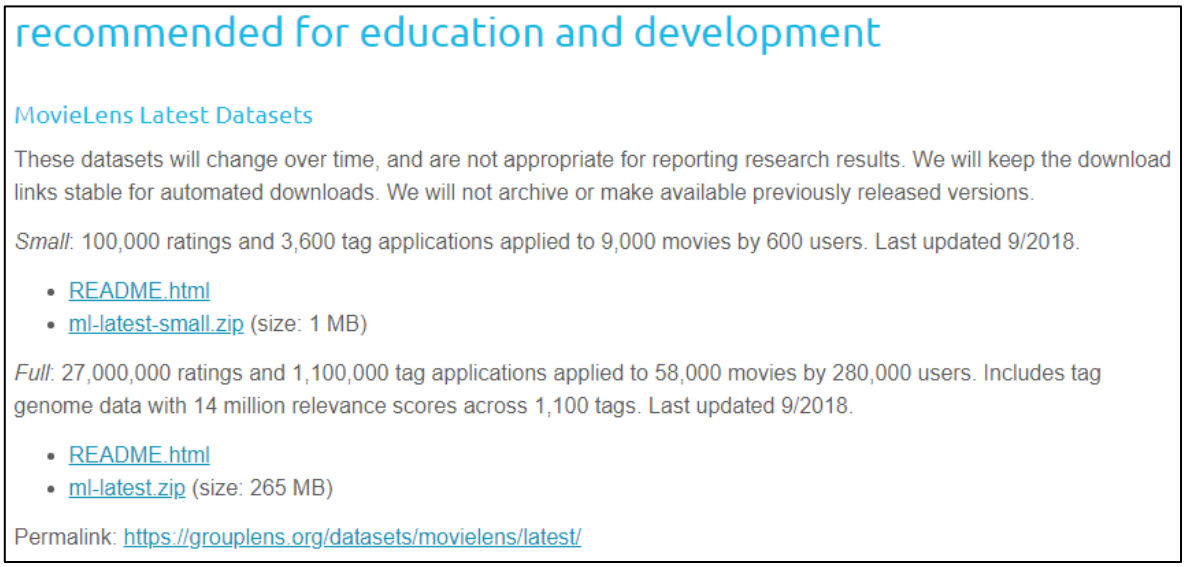

Figura 17. Conjuntos de datos de MovieLens recomendados para la educación y el desarrollo.

Los archivos que contiene ml-latest-small.zip son links.csv, movies.csv, ratings.csv, README.txt y tag.csv, estos archivos se muestran en la Figura 18.

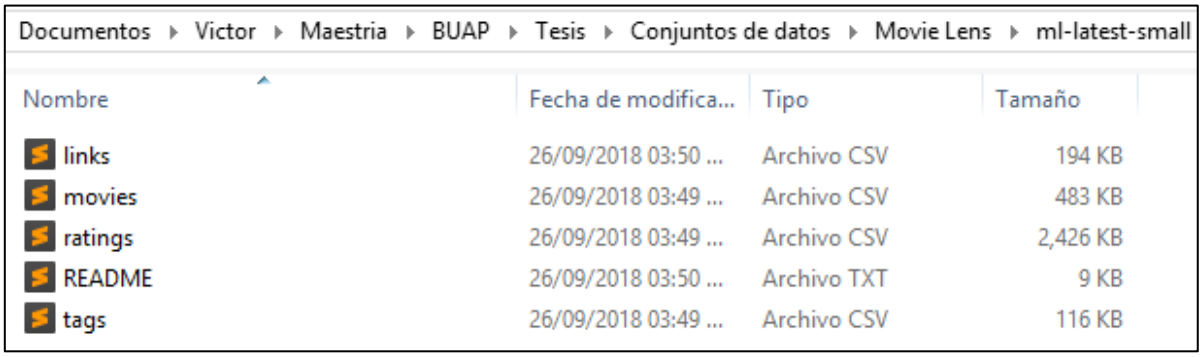

Figura 18. Archivos del ml-latest-small.zip de MovieLens.

Se utilizó el archivo de movies.csv que contiene 9,742 películas con descripciones textuales breves formadas por el id de la película, el titulo (año) y los géneros, el contenido de este archivo se muestra en la Tabla 1.

ID documento	ID película	Titulo	Géneros
1	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	2	Jumanji (1995)	Adventure|Children|Fantasy
3	3	Grumpier Old Men (1995)	Comedy|Romance
4	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
5	5	Father of the Bride Part II (1995)	Comedy
6	6	Heat (1995)	Action|Crime|Thriller
7	7	Sabrina (1995)	Comedy|Romance
8	8	Tom and Huck (1995)	Adventure|Children
9	9	Sudden Death (1995)	Action
10	10	GoldenEye (1995)	Action|Adventure|Thriller
…	…	…	….
9742	193609	Andrew Dice Clay: Dice Rules (1991)	Comedy

Tabla 1. Ejemplo del archivo movies.csv.

Otro archivo que se utilizo es el de ratings.csv que contiene 100,836 tuplas del tipo <usuario, película, valoración, marca de tiempo>, este archivo se muestra en la Tabla 2.

ID documento	ID usuario	ID película	Valoración	M. de Tiempo
1	1	1	4.0	964982703
2	1	3	4.0	964981247
3	1	6	4.0	964982224
4	1	47	5.0	964983815
5	1	50	5.0	964982931
…	…	…	…	…
100836	610	170875	3.0	1493846415

Tabla 2. Ejemplo del archivo ratings.csv.

El último archivo que se utilizó es el de tag.csv que contiene 3,683 etiquetas (géneros, actores, directores u otras características) que agregaron los usuarios a las películas.

ID documento	ID usuario	ID película	Etiqueta	M. de tiempo
1	2	60756	funny	1445714994
2	2	60756	Highly quotable	1445714996
3	2	60756	will ferrell	1445714992
4	2	89774	Boxing story	1445715207
5	2	89774	MMA	1445715200
…	…	…	…	…
3683	610	168248	Heroic Bloodshed	1493844270

Tabla 3. Ejemplo del archivo tags.csv.

4.2 Paradigmas basados en RI para los sistemas de recomendación

Los sistemas de recomendación tradicionalmente han utilizado las técnicas de recuperación de información para el filtrado basado en contenido, en donde se representan los ítems con descripciones textuales (metadatos) y se genera un perfil del usuario que cuadré con los productos más similares con base en los atributos textuales de los productos. Los problemas de esta técnica de recomendación son que es una técnica complicada de implementar y no presenta novedad en las recomendaciones.

Las técnicas de recomendación de filtrado colaborativo si presentan novedad y son las más usadas en los trabajos de investigación pero presentan un problema al utilizar matrices de similitud que es un alto costo de cómputo. Por tal motivo, en esta tesis se propone implementar dos paradigmas, uno para la similitud de los ítems y otro para la similitud de los usuarios, se utilizan las técnicas de filtrado colaborativo basado en el usuario y basado en el ítem con técnicas de recuperación de información para procesar altos volúmenes de datos y obtener mínimos tiempos de respuesta.

Expresando el problema de recomendación como un problema de recuperación de información, se representa a cada usuario como un documento y a cada ítem (en este una película es un ítem) como otro documento. El documento del usuario contiene las valoraciones que el realizó hacia sus películas que le gustaron, en la Figura 19 se muestra

en el documento del usuario que valoró la película 3 con una calificación de 5. El documento de la película contiene las valoraciones que le realizaron los usuarios a los que les gusto esa película, en la Figura 19, en el documento de la película se muestra que el usuario 3 valoró esa película con una calificación de 5. El documento del usuario contiene el id del usuario asignado por MovieLens y el documento de la película contiene el id de la película que le asignó MovieLens, sin embargo, ambos sólo son representativos y no se incluyen en la indización de los documentos.

Esta representación de datos que se realiza en este trabajo de tesis para el usuario y la película (ítem) es fundamental porque permite utilizar técnicas de recuperación de información para calcular la similitud entre usuarios y entre ítems, tomando en cuenta el nivel de interés que tiene un usuario hacia un ítem por medio de la frecuencia de los términos del documento.

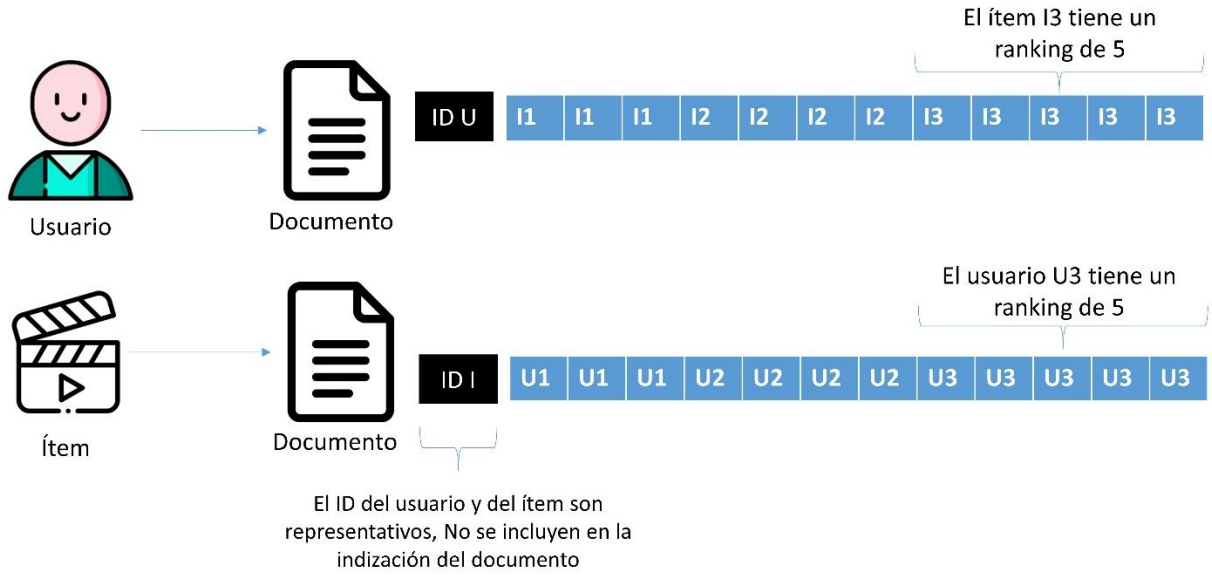

Figura 19. Representación de datos del usuario y del ítem (en este caso los ítems son películas) en el paradigma basado en RI para los SR.

En la Figura 20, se muestra el paradigma de similitud entre usuarios que consiste en insertar un documento de un usuario activo al algoritmo de recuperación de información para obtener como salida los 10 usuarios más similares para el usuario activo. De la misma forma, se muestra el paradigma de similitud entre películas que consiste en insertar un documento de una película al algoritmo de recuperación de información para obtener como salida las 10 películas más similares para esa película.

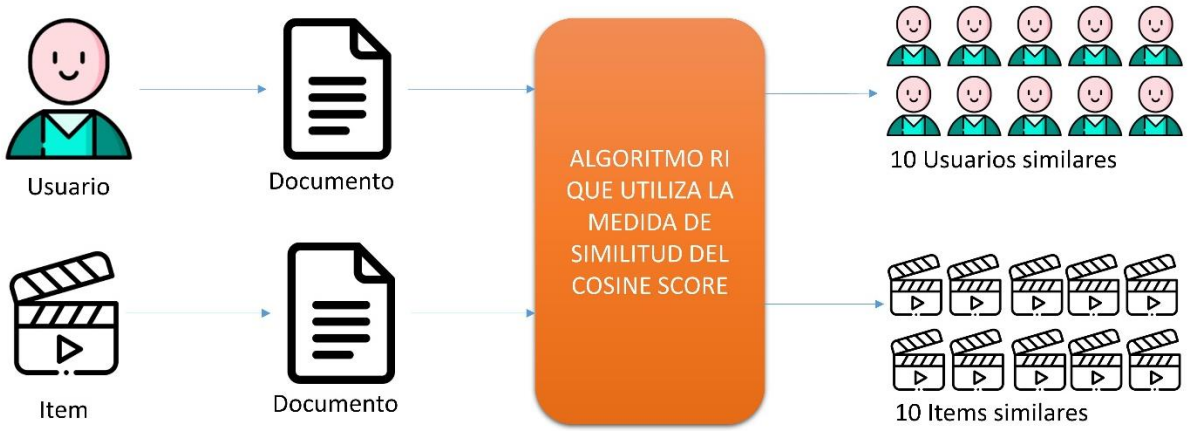

Figura 20. Paradigmas de similitud entre usuarios y entre ítems utilizando un algoritmo de RI.

4.3 Pre-procesamiento del conjunto de datos

El conjunto de datos de MovieLens se pre-proceso para obtener la representación de los usuarios y las películas como documentos. En Python se codificó una clase llamada preproceso, en esta clase se implementó el Algoritmo 1 como un método de la clase que genera la representación de los usuarios y las películas como documentos con base en el número de valoraciones de los usuarios y las películas. El algoritmo recibe como datos de entrada el nombre de tres archivos, el primero contiene al archivo ratings.csv de MovieLens, el segundo y el tercer archivo son nombres de archivos donde se almacenaran las representaciones de los usuarios y las películas como documentos. Después, se obtiene por cada usuario el número de películas que valoró y se almacenan en el diccionario llamado *noRankingsUsuarios*, este procedimiento se repite para las películas, se obtiene por cada película el número de usuarios que valoraron esa película y se almacenan en el diccionario llamado *noRankingsPeliculas*. Después, se inicializan los diccionarios *documentosUsuarios* y *documentosPeliculas* en donde se almacenaran las representaciones de los usuarios y las películas como documentos, también se inicializa la variable *noRankings* con un valor de 10 que significa que solo se tomaran en cuenta las películas y usuarios con más de diez valoraciones, otros valores que se le asignaron a *noRankings* son 20, 30, 40, 50, 100 y 150. Después, se recorre línea por línea el archivo de la variable *archivo_rankings* donde se obtienen los datos de cada línea almacenándolos en las variables *idUsuario*, *idPelicula* y *ranking*, se valida el número de valoraciones del usuario con el valor del diccionario *noRankingsUsuarios[idUsuario]* que sea mayor a *noRankings* y se valida el número de valoraciones de las película con el valor del diccionario *noRankingsPeliculas[idPelicula]* que sea mayor a *noRankings*. Si se cumple esta condición se agrega a *documentosUsuarios[idUsuario]* el *idPelicula* con una frecuencia denotada por el *ranking* y también se agrega a *documentosPeliculas[idPelicula]* el *idUsuario* con una frecuencia denotada por el *ranking*. Finalmente, se recorren los diccionarios

documentosUsuarios y *documentosPeliculas* para guardar su contenido en los archivos *archivo_usuarios* y *archivo_peliculas*.

```
procedure preprocesamiento(archivo_rankings, archivo_usuarios, archivo_peliculas)
        noRankingsUsuarios←Obtener número de rankings por usuario
        noRankingsPeliculas←Obtener número de rankings por película
        documentosUsuarios←Diccionario
        documentosPeliculas←Diccionario
        noRankings←10
        for each linea in archivo_rankings do
                idUsuario← línea.getUsuario()
                idPelicula←linea.getPelicula()
                ranking←linea.getRanking()
                if noRankingsUsuarios[idUsuario]>noRankings and
                noRankingsPelicula[idPelicula]>noRankings then
                        documentosUsuarios[idUsuario]←Agregar(getTerminos(idUsuario,r
                        anking))
                        documentosPeliculas[idPelicula]←Agregar(getTerminos(idPelicula,r
                        anking))
                end if
        end for
        for each documentoUsuario in documentosUsuarios do
                Escribir documentoUsuario en archivo_usuarios
        end for
        for each documentoPelicula in documentosPeliculas do
                Escribir documentoPelicula en archivo_peliculas
        end for
end procedure
```

Algoritmo 1. Pre-procesamiento para representar los usuarios y a las películas como documentos.

En el Algoritmo 2 se presenta el método *getTerminos* que se usó en el Algoritmo 1, el Algoritmo 2 devuelve la frecuencia de un término que es el id de un usuario o una película con base en el *rating*. Este algoritmo recibe dos datos de entrada el término que es un *id* del usuario o de la película de tipo *String* y el *rating* que es la valoración de una película o un usuario, se inicializa la variable cadena como vacía, se valida si el rating tiene un valor mayor de cero y menor de uno, si se cumple la condición se asigna el valor de *termino* a *cadena* y se devuelve el valor de *cadena* como resultado. En caso contrario se concatenara el término con base a la valoración del rating. Es decir, para el usuario con el id igual a 10 que valoró con un ranking de 5 a la película "Toy Story" con un id igual a 1, el resultado de *getTerminos(1,5)* será igual a "1 1 1 1 1" representado así el *id* de la película 1 con una frecuencia de 5 en el documento del usuario con *id* igual a 10.

```
procedure getTerminos(termino,rating)
        cadena←Vacia
        if rating > 0 and rating < 1 then
                cadena←termino
        else
                for i in range(rating)
                        if i = rating then
                                cadena←concatenar(cadena,termino)
                        else
                                cadena←concatenar(cadena,termino,",")
                end for
        end if
        return cadena
end procedure
```

Algoritmo 2. Método getTerminos que se utilizó en el algoritmo de preprocesamiento.

Los resultados obtenidos por el pre-procesamiento del conjunto de datos de MovieLens son 7 conjuntos de datos de la representación de usuarios como documentos y 7 conjuntos de datos de la representación de películas como documentos. Los conjuntos de datos se clasifican por el número de valoraciones de los usuarios y de las películas, por ejemplo, si el número de valoraciones es mayor a 10 significa que un usuario tiene más de 10 películas valoradas y una película fue valorada por más de 10 usuarios.

No. Valoraciones	**Archivos de usuarios**	**Archivos de películas**
>10	movies_users_frecuencias_10.txt	movies_items_frecuencias_10.txt
>20	movies_users_frecuencias_20.txt	movies_items_frecuencias_20.txt
>30	movies_users_frecuencias_30.txt	movies_items_frecuencias_30.txt
>40	movies_users_frecuencias_40.txt	movies_items_frecuencias_40.txt
>50	movies_users_frecuencias_50.txt	movies_items_frecuencias_50.txt
>100	movies_users_frecuencias_100.txt	movies_items_frecuencias_100.txt
>150	movies_users_frecuencias_150.txt	movies_items_frecuencias_150.txt

Tabla 4. Resultados del pre-procesamiento del conjunto de datos.

4.4 Algoritmo de recuperación de información

En esta tesis se utilizó el modelo de espacio vectorial que representa a los documentos como un conjunto de términos de un vector y la consulta de la misma forma se representa por un conjunto de términos de otro vector. Entonces, se van a recuperar los *k* vectores que tengan una menor distancia del ángulo del coseno con respecto a la consulta. En este trabajo *k* tiene un valor de 10. El algoritmo para calcular la distancia del ángulo del coseno entre los vectores se llama *CosineScore* y se muestra en el Algoritmo 3. En este algoritmo *N* es igual al número de documentos, se incializan dos listas de tamaño *N*, la primera lista llamada *Scores[ ]* es para almacenar los pesos de los documentos, la segunda lista llamada *Length[ ]* es para almacenar el número de términos de cada documento. Para cada termino *t* que aparece en la consulta *q*, se calculan los sus pesos con respecto a la consulta y a los documentos donde aparece el termino utilizando *TF-IDF* en $\mathbf{wf_{t,d}}$ **y** $\mathbf{w_{t,q}}$**,** se almacena el resultado en Scores de cada documento y se recorre la lista de Scores para dividir sus valores entre el tamaño del documento.

```
procedure CosineScore(q)
        float Scores[N]
        Inicialize Length[N]
        for each query term t
        do calculate w_{t,q} and fetch postings list for t
                for each pair(d,tf_{t,d}) in postings list
                do Scores[d]+=wf_{t,d} × w_{t,q}
        Read the array Length[d]
        for each d
        do Scores[d]=Scores[d]/Lenght[d]
        return Top K components of Scores[]
end procedure
```

Algoritmo 3. CosineScore para calcular la similitud de los documentos con una consulta con base en el ángulo del coseno.

La implementación de este algoritmo se realizó en el lenguaje de programación de Python porque es un lenguaje interpretado, dinámico, multiplataforma y usa el paradigma orientado a objetos. Se desarrolló una clase llamada *recuperación_de_información*, en esta clase se implementaron los métodos *crear_indice_posicional*, *mostrar_indice*, *normalizar*, *cosine_Score*, *frecuencia_terminos_consulta* y *ordenar_diccionario*. Estos métodos permiten crear un índice posicional como estructura de datos y utilizar el modelo de espacio vectorial de recuperación de información por medio del algoritmo de *CosineScore* para encontrar documentos de usuario o películas similares.

4.4 Experimentos

En esta tesis se realizaron 4 experimentos, en los primero tres experimentos se implementó la técnica de filtrado colaborativo basado en el ítem y en el cuarto experimento se implementó la técnica de filtrado colaborativo basado en el usuario.

4.4.1 Experimento I

El primer experimento consiste por cada película que le gusto al usuario se obtienen 10 películas similares, se almacenan las primeros N películas similares en una lista general de recomendaciones ordenada de forma ascendente y se recomiendan las 10 primeras películas de lista general. En este experimento se realizaron 5 sub-experimentos con respecto al valor de N, los valores de N para los sub-experimentos son 1, 2, 3, 4 y 5. En la Figura 21 se muestra este experimento.

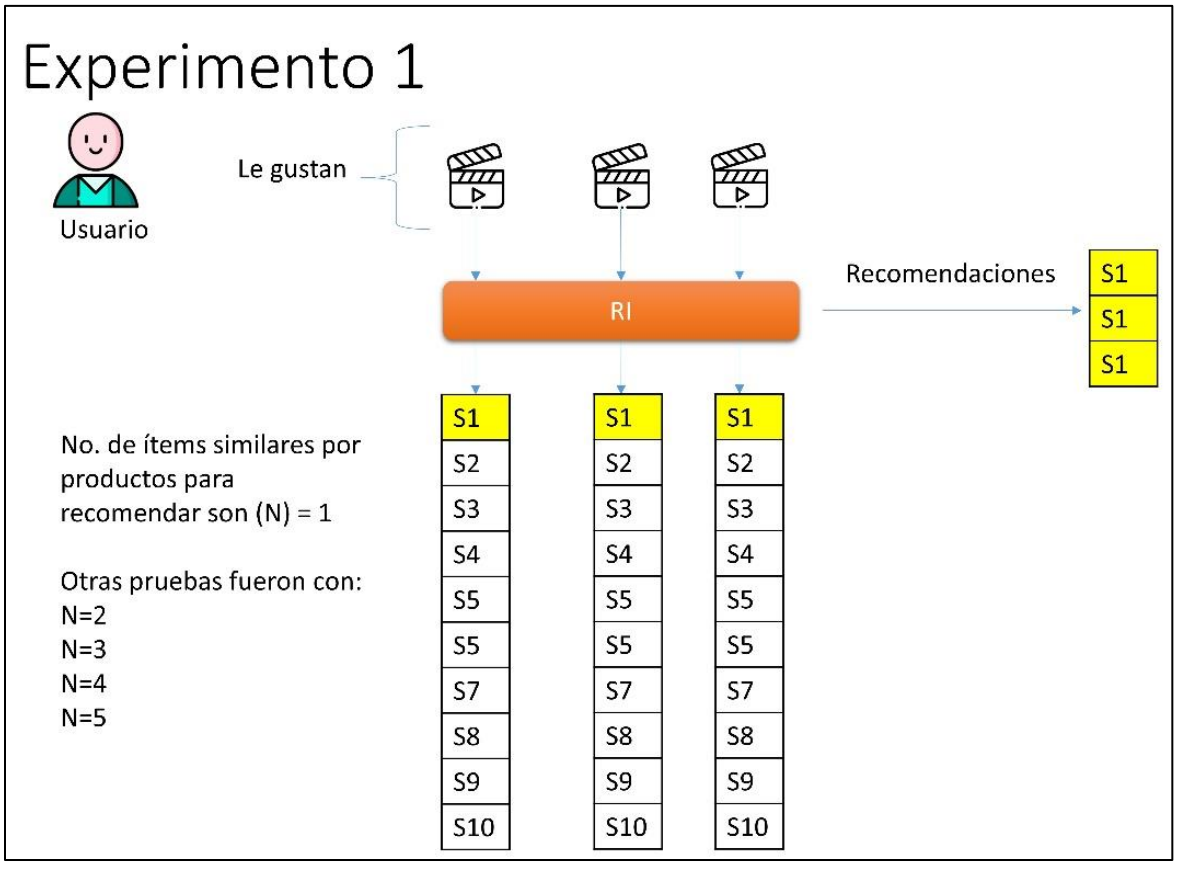

Figura 21. Experimento 1.

El algoritmo que se implementó para el experimento 1 se muestra en el Algoritmo 4, este algoritmo recibe como parámetro de entrada el usuario al que se le realizaran las

recomendaciones. La variable *N* representa el número de películas similares que se seleccionarán por cada película, en este caso *N* es igual a 1, *K* representa el número de recomendaciones que se realizaran y las k recomendaciones se almacenaran en un diccionario llamado *TopKrecomendaciones*. En *peliculas_usuario* se obtiene una lista de las películas que le gustan al usuario, por cada película que le gusta al usuario se obtiene una lista de sus películas similares y se eliminan las películas que ya vio el usuario. Se agregan las *N* primeras peliculas similares en una lista general de recomendaciones llamada *TopKrecomendaciones* y se repite el proceso para la siguiente película que le gusta al usuario. Al terminar el ciclo de las películas que le gustan al usuario se ordena *TopKrecomendaciones* de forma descendente y se retorna como resultado.

```
procedure Experimento1(usuario)
        N←1
        K←10
        TopKrecomendaciones←Diccionario vacio
        peliculas_usuario←Lista de películas que le gustan al usuario
        for each pelicula in peliculas_usuario do
                peliculas_similares←peliculas.CosineScore(pelicula)
                peliculas_similares←EliminarPeliculasVistas(peliculas_similares)
                for each i in range (N)
                        id_pelicula←peliculas_similares[ i ].getID()
                        valor_similitud←peliculas_similares[ i ].getValue()
                        TopKrecomendaciones[id_pelicula]+=valor_similitud
                end for
        end for
        TopKrecomendaciones←Ordenar de forma descendente
        return TopKrecomendaciones
end procedure
```

Algoritmo 4. Experimento 1.

4.4.2 Experimento II

El segundo experimento consiste en construir un perfil del usuario que contiene los atributos de las películas que en un principio fueron atributos textuales, por ejemplo, el título de una película, su género y sus etiquetas, sin embargo, no existía suficiente información textual en el conjunto de datos, por tal motivo, los atributos de los ítems se cambiaron por los usuarios que valoraron esas películas, entonces, el perfil del usuario contienen los usuarios similares de un usuario y se realiza una consulta para buscar los productos que tengan esos mismos usuarios en la representación de las películas, En la Figura 22 se muestra este experimento.

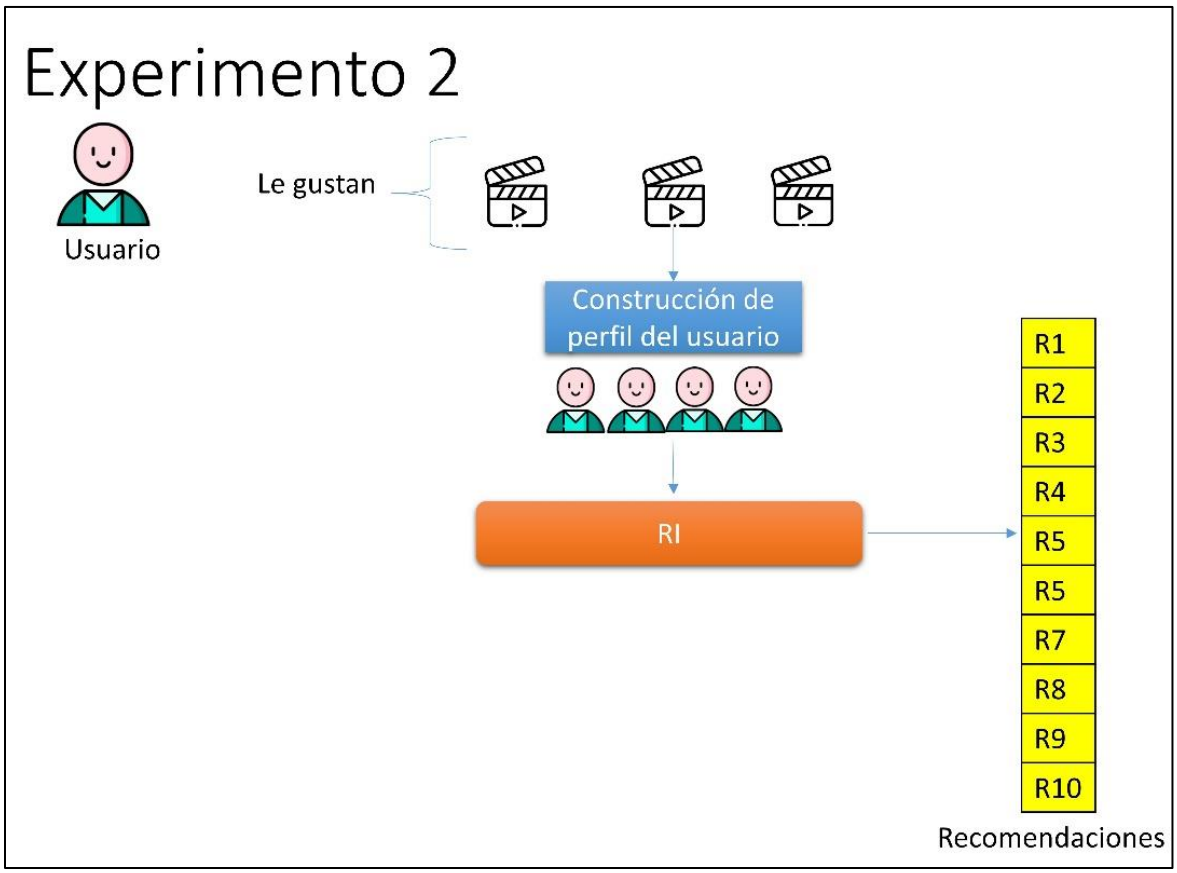

Figura 22. Experimento 2.

El algoritmo que se implementó para el experimento 2 se muestra en el Algoritmo 5. Este algoritmo recibe como datos de entrada al usuario al que se le va a realizar la recomendación, la variable *K* representa el numeró de recomendaciones que se le realizaran al usuario en este caso son diez, en *TopKrecomendaciones* se inicializa el diccionario en donde se almacenaran las recomendaciones, en *peliculas_usuario* se obtienen las películas que le gustan al usuario y se inicializa la variable de perfil con una cadena vacía de texto. Para cada película que le gusta al usuario se le extrae sus atributos y se asignan a *atributosPelicula*, la variable *atributosPelicula* se concatena con la variable de perfil y se repite el ciclo para otra película que le guste al usuario, al finalizar este ciclo se obtiene el perfil del usuario con los 10 principales atributos de sus películas, en este caso son los 10 usuarios más similares. Se buscan las películas que cuadren mejor con el perfil a través de *peliculas.CosineScore(perfil)*, se almacenan los resultados en *TopKrecomendaciones*, después se eliminan de *TopKrecomendaciones* las películas que ya vio el usuario, se ordenan las recomendaciones de *TopKrecomendaciones* de forma descendente y se retornan como resultado las primeras *K* recomendaciones de *TopKrecomendaciones*.

```
procedure Experimento2(usuario)
        K←10
        TopKrecomendaciones←Diccionario vacio
        peliculas_usuario←Lista de películas que le gustan al usuario
        perfil←Cadena de texto vacia
        for each pelicula in peliculas_usuario do
                atributosPelicula←getAtributosPelicula(pelicula)
                perfil←Concatenar(perfil,atributosPelicula)
        end for
        TopKrecomendaciones ←peliculas.CosineScore(perfil)
        TopKrecomendaciones ←EliminarPeliculasVistas(TopKrecomendaciones)
        TopKrecomendaciones←Ordenar de forma descendente
        return TopKrecomendaciones
end procedure
```

Algoritmo 5. Experimento 2.

4.4.3 Experimento III

El tercer experimento consiste por cada película que le gusta al usuario se recomiendan sus 10 películas más similares y se evalúan las recomendaciones independientemente, se repite el proceso para las otras películas que le gustan al usuario y se promedian los resultados de las evaluaciones. Este experimento se muestra en la Figura 23.

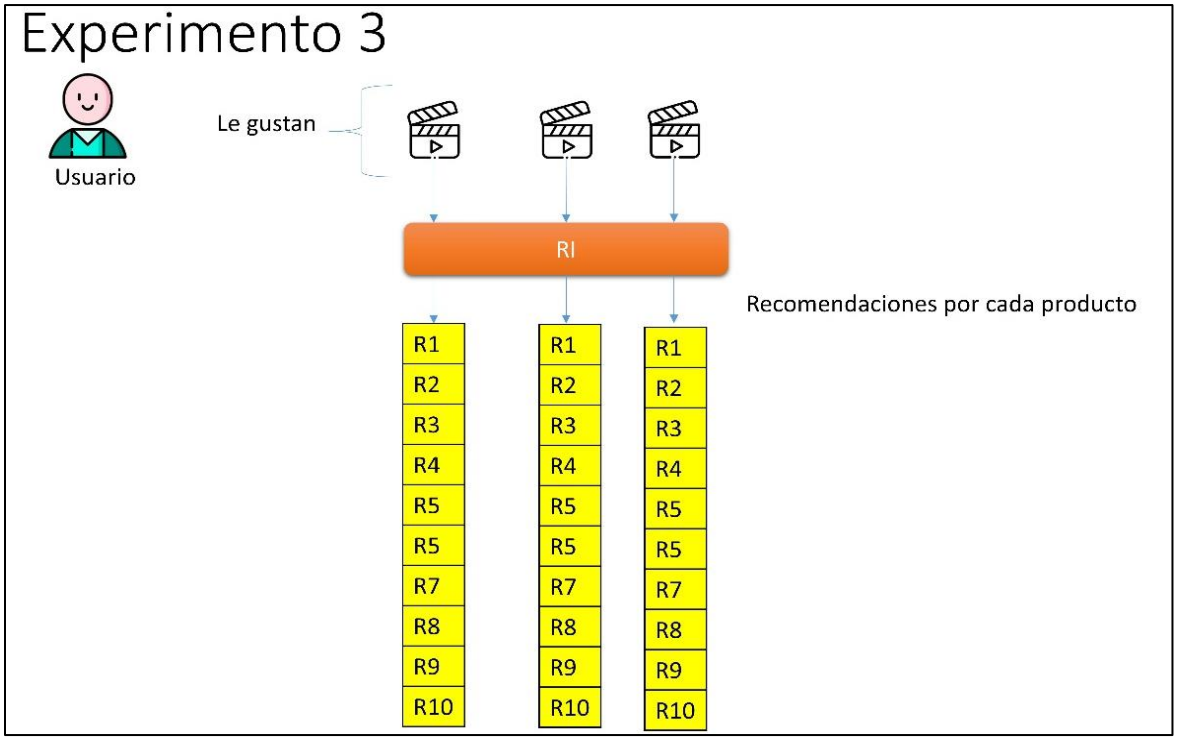

Figura 23. Experimento 3.

El algoritmo que se implementó para el experimento 3 se muestra en el Algoritmo 6. Este algoritmo recibe como datos de entrada al usuario al que se le va a realizar la recomendación, la variable *K* representa el numeró de recomendaciones que se le realizaran al usuario en este caso son diez, en *TopKrecomendaciones* se inicializa el diccionario en donde se almacenaran las recomendaciones y en *peliculas_usuario* se obtienen las películas que le gustan al usuario. Para cada película que le gusta al usuario se extraen sus atributos, estos atributos se agregan como consulta en *peliculas.CosineScore(atributosPelicula)* para obtener las películas más similares a esos atributos, las cuales, se almacenan en *TopKrecomendaciones*. En *TopKrecomendaciones* se eliminan las películas que ya vio el usuario y se ordenan las recomendaciones de forma descendente. Se evalúan las recomendaciones de *TopKrecomendaciones* para obtener la precisión, el recall y f-measure, se repite el ciclo con otra película que le guste al usuario. Al finalizar se promedian los resultados y se retornan.

```
procedure Experimento3(usuario)
        K←10
        TopKrecomendaciones←Diccionario vacio
        peliculas_usuario←Lista de películas que le gustan al usuario
        for each pelicula in peliculas_usuario do
                atributosPelicula←getAtributosPelicula(pelicula)
                TopKrecomendaciones←peliculas.CosineScore(atributosPelicula)
                TopKrecomendaciones←EliminarPeliculasVistas(TopKrecomendaciones)
                TopKrecomendaciones←Ordenar de forma descendente
                evaluacion←EvaluarRecomendaciones(TopKrecomendaciones)
        end for
        return evaluacion
end procedure
```

Algoritmo 6. Experimento 3.

4.4.4 Experimento IV

El cuarto experimento utiliza otra técnica de recomendación a diferencia de los tres experimentos anteriores en donde se buscaban productos similares con base en los atributos de las películas. En este experimento su utiliza la técnica de recomendación llamada filtrado colaborativo basado en el usuario. Este experimento consiste en obtener los 10 usuarios más similares de un usuario con base en la representación de los usuarios. Después, se agregan las películas que les gustan a sus usuarios similares y que no ha visto el usuario a una lista de recomendaciones general, en donde se pondera la película recomendada con un valor de 1 por cada usuario similar que realice la recomendación. Esto quiere decir, que si tres usuarios similares recomendaron la misma película entonces la película tendrá una ponderación de 3.En la Figura 24 se muestra este experimento.

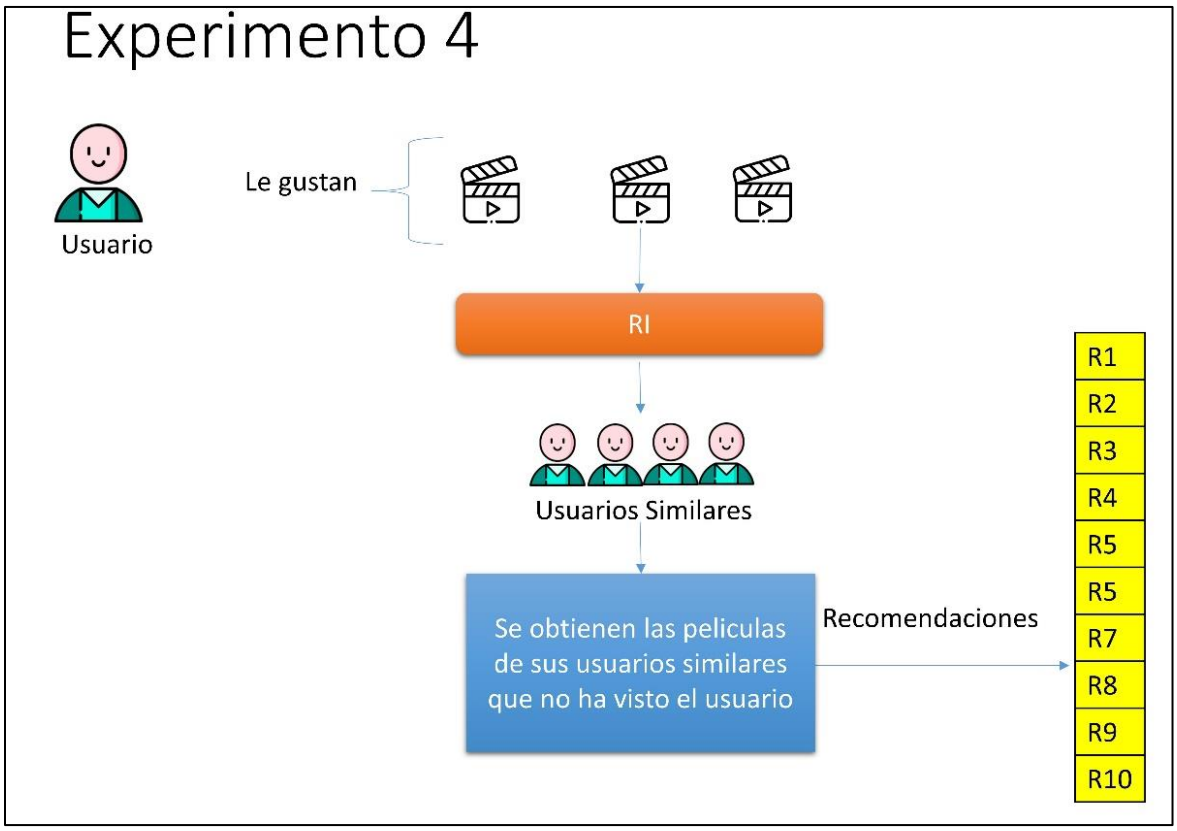

Figura 24. Experimento 4.

El algoritmo que se implementó para el experimento 4 se muestra en el Algoritmo 7. Este algoritmo recibe como datos de entrada al usuario al que se le va a realizar la recomendación, la variable *K* representa el numeró de recomendaciones que se le realizaran al usuario en este caso son diez, *N* representa el número de usuarios similares en este caso son 10, en *TopKrecomendaciones* se inicializa el diccionario en donde se almacenaran las recomendaciones y en *peliculas_usuario* se obtienen las películas que le gustan al usuario. Se obtienen los 10 usuarios similares con *usuarios.CosineScore(peliculasUsuario)* y se almacenan en u*suariosSimilaresN*. Por cada usuario similar se obtienen sus películas en *peliculasUS*, por cada película de *peliculasUS* se valida que no la vio el usuario y se agrega en *TopKrecomendaciones* con una ponderación de 1. Se repite el procedimiento para el resto de los usuarios similares. Se ordenan las recomendaciones de *TopKrecomendaciones* de forma descendente y se retornan las 10 primeras recomendaciones de *TopKrecomendaciones*.

```
procedure Experimento3(usuario)
        K←10
        N←10
        TopKrecomendaciones←Diccionario vacio
        peliculasUsuario←Lista de películas que le gustan al usuario
        usuariosSimilaresN←usuarios.CosineScore(peliculasUsuario)
        for each usuarioSimilar in usuariosSimilaresN do
                peliculasUS←getPeliculas(usuarioSimilar)
                for each pelicula in peliculasUS do
                        if pelicula not in peliculasUsuario then
                                TopKrecomendaciones[pelicula]+=1
                         end if
                end for
        end for
        TopKrecomendaciones←Ordenar de forma descendente
        return TopKrecomendaciones
end procedure
```

Algoritmo 7. Experimento 4.

4.5 Aspectos técnicos

Los experimentos se implementaron con el lenguaje de programación de Python, este lenguaje de programación es de alto nivel, interpretado y multipropósito. En la actualidad es uno de los lenguajes de programación más empleados para el desarrollo de software porque funcionan en diversas plataformas y sistemas operativos. Unas de las principales razones porque se elige este lenguaje de programación son porque es un lenguaje potente, flexible, utiliza una sintaxis clara y concisa, además no requiere dedicar tiempo a su compilación porque es interpretado (Fernández, 2012).

En el Anexo A se presentan los principales métodos que se desarrollaron para la implementación de los algoritmo de los experimentos. Se crearon tres clases principales, una contiene el algoritmo de recuperación de recuperación con el modelo espacio vectorial, la segunda clase contiene la implementación de tres experimentos y la tercera clase contiene la implementación de un experimento. En la tercer clase se implementó solo el experimento 3 porque la evaluación de realiza de manera independiente por cada película que le gusta a un usuario, que a diferencia de los otros experimentos que solo retornan una lista de recomendaciones y se evalúa esa lista.

En las pruebas se realizó un pre-procesamiento por cada usuario, en donde se utilizaba un 50% de sus valoraciones para el entrenamiento y el otro 50% para las pruebas. Es importante mencionar, que por cada usuario se realizó una indización a la estructura de datos del tipo índice posicional. Las pruebas se realizaron en una laptop con 8 GB de RAM, un procesador Intel Core i5 y con un sistema operativo Windows 10.

Capítulo 5
Resultados

En este capítulo final se realiza un análisis de los resultados obtenidos de los 4 experimentos que se describieron en el capítulo anterior, en el primer experimento se realizaron 5 sub-experimentos y cada sub-experimento contiene 7 pruebas. En los experimentos 2, 3 y 4, solo se realizaron las 7 pruebas por cada experimento. En total se realizaron 42 pruebas con base en el número de valoraciones que los usuarios tienen sobre las películas y las valoraciones que tienen las películas por cada usuario. En los experimentos 1, 2 y 3 se utilizó la técnica de recomendación llamada filtrado colaborativo basado en el ítem y en el experimento 4 se utilizó la técnica de recomendación llamada filtrado colaborativo basado en el usuario.

A continuación se presentan los resultados por cada experimento y después se presenta la comparación de los resultados con base en las métricas de evaluación de precisión, recall y f-measure, y la comparación de los tiempos de ejecución en segundos en la indización por usuario y en la recomendación por usuario. Además, en la comparación de resultados se presentan las ventajas y desventajas de los resultados obtenidos.

5.1 Resultados experimento I

Los resultados del primer experimento se presentan en la Tabla 5 que contiene los resultados de los 5 sub-experimentos del experimento 1. Estos sub-experimentos están clasificados por el valor de *N* de la columna 1, *N* significa el número de películas similares que se agregan a la lista general de recomendaciones por cada película que le gusto al usuario, se puede consultar el capítulo anterior para observar la descripción de este experimento. Las columnas 2, 3 y 4 contienen las métricas de evaluación de precisión, recall y f-measure, la columna 5 contiene el tiempo de indización por usuario en segundos (seg.) y la columna 6 contiene el tiempo de recomendación por usuario en segundos (seg).

Se observa que al aumentar el valor de *N* mejora el resultado de f-measure que es la medida armónica entre precisión y recall, esto quiere decir, que en el experimento 1 los resultados mejoran al recomendar un mayor número de películas similares por cada película que le gusta al usuario. Por tal motivo, el sub-experimento 5 presenta el mejor resultado con f-measure, por tal motivo este sub-experimento se utilizará en la comparación de resultados con otros experimentos. Sin embargo, no se presenta una gran diferencia entre los valores de f-measure de los sub-experimentos porque se utiliza la misma técnica de recomendación para cada sub-experimento y solo varia el valor de N. Los tiempos en segundos de indización y recomendación por usuario son similares para cada sub-experimento porque ejecutan el mismo número de veces el algoritmo CosineScore, lo único que varía son las películas similares que retornan por cada película que corresponde al valor de *N*.

N	**Precisión**	**Recall**	**F-measure**	**Tiempo indización por usuario (seg)**	**Tiempo recomendación por Usuario (seg)**
1	0.257142857	0.12857143	0.16428571	0.37285714	1.27428571
2	0.247142857	0.15428571	0.18285714	0.37714286	1.25857143
3	0.251428571	0.18714286	0.20285714	0.37285714	1.28714286
4	0.254285714	0.21142857	0.21428571	0.37714286	1.28428571
5	0.255714286	0.21857143	0.21857143	0.37	1.26
Promedio:	0.253142857	0.18	0.19657143	0.374	1.27285714

Tabla 5. Resultados experimento 1 con el resumen de sus sub-experimentos.

En los resultados individuales por cada sub-experimento se realizaron 7 pruebas con base en el número de valoraciones por usuario y por película, es decir, si el número de valoraciones es mayor a 10, entonces solo se tomaran en cuentas los usuario que tengan más de 10 películas valoradas y las películas que fueron valoradas por más de 10 usuarios. Esto significa que se eliminaran los usuarios y películas que no cumplan con esa condición, por lo tanto, se reduce el número de datos.

El contenido de las columnas de los sub-experimentos y para el resto de los experimentos es el siguiente. En la columna 1 se muestra el número de valoraciones (No. Vals.) de los usuarios y de las películas, en la columan2 se muestra el número de usuarios evaluados (No. Usu. Eva.), en la columna 3 se muestra la precisión, en la columna 4 se muestra el recall, en la columna 5 se muestra la f-measure, en la columna 6 se muestra el tiempo de indización por usuario en segundos (seg.) y la columna 7 se muestra el tiempo de recomendación por usuario en segundos (seg).

5.1.1 Resultados sub-experimento 1

En los resultados del sub-experimento 1 que se muestra en la Tabla 6, se observa que al crecer el número de valoraciones los resultados de precisión, recall y f-measure aumentan. Con más de 150 valoraciones se obtuvieron los mejores resultados con una precisión de 0.82, un recall de 0.46, un f-measure de 0.58, un tiempo de indización por usuario de 0.01 segundos y un tiempo de recomendación por usuario de 0.02 segundos. Los tiempos de ejecución se minimizan con un mayor número de valoraciones porque los usuarios y las películas que no tienen más de 150 valoraciones son eliminados del conjunto de datos. Al reducir el número de valoraciones con más de 10 valoraciones por usuarios y películas se obtuvieron los más bajos resultados con una precisión del 0.06, un recall de 0.02, un f-measure de 0.03, un tiempo de indización de 0.95 segundos y un tiempo de recomendación de 3.33 segundos.

No. Vals.	**No. Usu. Eva.**	**Precisión**	**Recall**	**F-measure**	**Tiempo indización por usuario (seg)**	**Tiempo recomendación por Usuario (seg)**
>10	566	0.06	0.02	0.03	0.95	3.33
>20	457	0.07	0.02	0.03	0.63	2.13
>30	357	0.1	0.04	0.05	0.46	1.59
>40	301	0.11	0.04	0.05	0.31	1.07
>50	248	0.16	0.05	0.07	0.19	0.66
>100	134	0.48	0.27	0.34	0.06	0.12

>150	67	0.82	0.46	0.58	0.01	0.02
Prom.		0.25714286	0.12857143	0.16428571	0.37285714	1.27428571

Tabla 6. Resultados experimento 1 del sub-experimento 1 con n=1.

5.1.2 Resultados sub-experimento 2

En los resultados del sub-experimento 2 que se muestra en la Tabla 7 presenta un comportamiento similar al sub-experimento anterior, se observa que al crecer el número de valoraciones los resultados de precisión, recall y f-measure aumentan. Con más de 150 valoraciones se obtuvieron los mejores resultados con una precisión de 0.77, un recall de 0.58, un f-measure de 0.66, un tiempo de indización por usuario de 0.01 segundos y un tiempo de recomendación por usuario de 0.01 segundos. Los tiempos de ejecución se minimizan con un mayor número de valoraciones porque los usuarios y las películas que no tienen más de 150 valoraciones son eliminados del conjunto de datos. Al reducir el número de valoraciones con más de 10 valoraciones por usuarios y películas se obtuvieron los más bajos resultados con una precisión del 0.06, un recall de 0.02, un f-measure de 0.03, un tiempo de indización de 0.96 segundos y un tiempo de recomendación de 3.21 segundos.

No. Vals.	**No. Usu. Eva.**	**Precisión**	**Recall**	**F-measure**	**Tiempo indización por usuario (seg)**	**Tiempo recomendación por Usuario (seg)**
>10	566	0.06	0.02	0.03	0.96	3.21
>20	457	0.08	0.03	0.04	0.68	2.2
>30	357	0.09	0.04	0.05	0.42	1.54
>40	301	0.1	0.04	0.05	0.3	1.07
>50	248	0.16	0.06	0.08	0.21	0.67
>100	134	0.47	0.31	0.37	0.06	0.11
>150	67	0.77	0.58	0.66	0.01	0.01
Prom.		0.24714286	0.15428571	0.18285714	0.37714286	1.25857143

Tabla 7. Resultados experimento 1 del sub-experimento 2 con n=2.

5.1.3 Resultados sub-experimento 3

En los resultados del sub-experimento 3 que se muestra en la Tabla 8 presenta un comportamiento similar a los anteriores sub-experimentos, se observa que al crecer el número de valoraciones los resultados de precisión, recall y f-measure aumentan. Con más de 150 valoraciones se obtuvieron los mejores resultados con una precisión de 0.77, un recall de 0.78, un f-measure de 0.77, un tiempo de indización por usuario de 0.01 segundos y un tiempo de recomendación por usuario de 0.01 segundos. Los tiempos de ejecución se minimizan con un mayor número de valoraciones porque los usuarios y las películas que no tienen más de 150 valoraciones son eliminados del conjunto de datos. Al reducir el número de valoraciones con más de 10 valoraciones por usuarios y películas se obtuvieron los más bajos resultados con una precisión del 0.07, un recall de 0.03, un f-measure de 0.04, un tiempo de indización de 0.95 segundos y un tiempo de recomendación de 3.43 segundos.

No. Vals.	**No. Usu. Eva.**	**Precisión**	**Recall**	**F-measure**	**Tiempo indización por usuario (seg)**	**Tiempo recomendación por Usuario (seg)**
>10	566	0.07	0.03	0.04	0.95	3.43
>20	457	0.09	0.04	0.05	0.65	2.19
>30	357	0.09	0.03	0.04	0.42	1.51
>40	301	0.1	0.04	0.05	0.31	1.09
>50	248	0.14	0.06	0.08	0.21	0.67
>100	134	0.5	0.33	0.39	0.06	0.11
>150	67	0.77	0.78	0.77	0.01	0.01
Prom.		0.25142857	0.18714286	0.20285714	0.37285714	1.28714286

Tabla 8. Resultados experimento 1 del sub-experimento 3 con n=3.

5.1.4 Resultados sub-experimento 4

En los resultados del sub-experimento 4 que se muestra en la Tabla 9 presenta un comportamiento similar a los anteriores sub-experimentos, se observa que al crecer el número de valoraciones los resultados de precisión, recall y f-measure aumentan. Con más de 150 valoraciones se obtuvieron los mejores resultados con una precisión de 0.78, un recall de 0.95, un f-measure de 0.85, un tiempo de indización por usuario de 0.01 segundos y un tiempo de recomendación por usuario de 0.01 segundos. Los

tiempos de ejecución se minimizan con un mayor número de valoraciones porque los usuarios y las películas que no tienen más de 150 valoraciones son eliminados del conjunto de datos. Al reducir el número de valoraciones con más de 10 valoraciones por usuarios y películas se obtuvieron los más bajos resultados con una precisión del 0.08, un recall de 0.03, un f-measure de 0.04, un tiempo de indización de 0.97 segundos y un tiempo de recomendación de 3.44 segundos.

No. Vals.	**No. Usu. Eva.**	**Precisión**	**Recall**	**F-measure**	**Tiempo indización por usuario (seg)**	**Tiempo recomendación por Usuario (seg)**
>10	566	0.08	0.03	0.04	0.97	3.44
>20	457	0.09	0.04	0.05	0.69	2.31
>30	357	0.09	0.03	0.04	0.42	1.44
>40	301	0.1	0.04	0.05	0.29	1.01
>50	248	0.14	0.06	0.08	0.21	0.67
>100	134	0.5	0.33	0.39	0.05	0.11
>150	67	0.78	0.95	0.85	0.01	0.01
Prom.		0.25428571	0.21142857	0.21428571	0.37714286	1.28428571

Tabla 9. Resultados experimento 1 del sub-experimento 4 con n=4.

5.1.5 Resultados sub-experimento 5

En los resultados del sub-experimento 5 que se muestra en la Tabla 10 presenta un comportamiento similar a los anteriores sub-experimentos, se observa que al crecer el número de valoraciones los resultados de precisión, recall y f-measure aumentan. Con más de 150 valoraciones se obtuvieron los mejores resultados con una precisión de 0.79, un recall de 0.99, un f-measure de 0.87, un tiempo de indización por usuario de 0.01 segundos y un tiempo de recomendación por usuario de 0.01 segundos. Los tiempos de ejecución se minimizan con un mayor número de valoraciones porque los usuarios y las películas que no tienen más de 150 valoraciones son eliminados del conjunto de datos. Este sub-experimento presenta mejores resultado en f-measure que los demás sub-experimentos, por tal motivo, se selecciona para comparar los resultados del experimento I con otros experimentos. Al reducir el número de valoraciones con más de 10 valoraciones por usuarios y películas se obtuvieron los más bajos resultados con una precisión del 0.08, un recall de 0.03, un f-measure de 0.04, un tiempo de indización de 0.9 segundos y un tiempo de recomendación de 3.28 segundos.

No. Vals.	No. Usu. Eva.	Precisión	Recall	F-measure	Tiempo indización por usuario (seg)	Tiempo recomendación por Usuario (seg)
>10	566	0.08	0.03	0.04	0.9	3.28
>20	457	0.09	0.04	0.05	0.67	2.21
>30	357	0.09	0.04	0.05	0.42	1.43
>40	301	0.1	0.04	0.05	0.32	1.11
>50	248	0.14	0.06	0.08	0.21	0.67
>100	134	0.5	0.33	0.39	0.06	0.11
>150	67	0.79	0.99	0.87	0.01	0.01
Prom.		0.25571429	0.21857143	0.21857143	0.37	1.26

Tabla 10. Resultados experimento 1 del sub-experimento 5 con n=5.

5.2 Resultados experimento II

En los resultados del experimento 2 que se muestra en la Tabla 11, se observa que al crecer el número de valoraciones los resultados de precisión, recall y f-measure aumentan. Con más de 150 valoraciones se obtuvieron los mejores resultados con una precisión de 0.79, un recall de 1.00, un f-measure de 0.88, un tiempo de indización por usuario de 0.01 segundos y un tiempo de recomendación por usuario de 0.006 segundos. Los tiempos de ejecución se minimizan con un mayor número de valoraciones porque los usuarios y las películas que no tienen más de 150 valoraciones son eliminados del conjunto de datos. Al reducir el número de valoraciones con más de 10 valoraciones por usuarios y películas se obtuvieron los más bajos resultados con una precisión del 0.05, un recall de 0.02, un f-measure de 0.02, un tiempo de indización de 0.94 segundos y un tiempo de recomendación de 0.06 segundos.

En comparación del experimento anterior, a pesar que tienen un comportamiento similar por el incremento de sus resultados con base en las valoraciones de los usuarios y las películas, este experimento presentó mejores resultados en f-measure, sin embargo, la diferencia es mínima porque utilizan la misma técnica de recomendación. En sus tiempos de ejecución si se presenta una diferencia notable porque sus tiempos no son tan variantes y elevados, esto se debe a que el algoritmo de CosineScore solo se ejecuta una vez por cada usuario y en el anterior experimento se ejecuta por cada película que vio el usuario.

No. Vals.	No. Usu. Eva.	Precisión	Recall	F-measure	Tiempo indización por usuario (seg)	Tiempo recomendación por Usuario (seg)
>10	566	0.05	0.02	0.02	0.94	0.06
>20	457	0.07	0.03	0.04	0.62	0.04
>30	357	0.1	0.03	0.04	0.43	0.03
>40	301	0.15	0.05	0.07	0.32	0.03
>50	248	0.2	0.07	0.1	0.24	0.03
>100	134	0.5	0.33	0.39	0.06	0.01
>150	67	0.79	1	0.88	0.01	0.006
Prom.		0.26571429	0.21857143	0.22	0.37428571	0.02942857

Tabla 11. Resultados experimento 2.

5.3 Resultados experimento III

En los resultados del experimento 3 que se muestra en la Tabla 12, se observa que al crecer el número de valoraciones los resultados de precisión, recall y f-measure aumentan. Con más de 150 valoraciones se obtuvieron los mejores resultados con una precisión de 0.79, un recall de 1.00, un f-measure de 0.88, un tiempo de indización por usuario de 0.01 segundos y un tiempo de recomendación por usuario de 0.03 segundos. Los tiempos de ejecución se minimizan con un mayor número de valoraciones porque los usuarios y las películas que no tienen más de 150 valoraciones son eliminados del conjunto de datos. Al reducir el número de valoraciones con más de 10 valoraciones por usuarios y películas se obtuvieron los más bajos resultados con una precisión del 0.08, un recall de 0.03, un f-measure de 0.04, un tiempo de indización de 0.85 segundos y un tiempo de recomendación de 3.45 segundos.

En comparación del experimento anterior, a pesar que tienen un comportamiento similar por el incremento de sus resultados con base en las valoraciones de los usuarios y las películas, este experimento presento mejores resultados en f-measure, sin embargo, la diferencia es mínima porque utilizan la misma técnica de recomendación. En sus tiempos de ejecución se incrementan con una diferencia notable porque sus tiempos son variantes y elevados, esto se debe a que el algoritmo de CosineScore se ejecuta por cada película que vio el usuario.

No. Vals.	No. Usu. Eva.	Precisión	Recall	F-measure	Tiempo indización por usuario (seg)	Tiempo recomendación por Usuario (seg)
>10	566	0.08	0.03	0.04	0.85	3.45
>20	457	0.1	0.04	0.05	0.6	2.23
>30	357	0.12	0.04	0.06	0.41	1.64
>40	301	0.16	0.05	0.07	0.29	1.3
>50	248	0.22	0.08	0.11	0.21	0.94
>100	134	0.51	0.34	0.4	0.06	0.19
>150	67	0.79	1	0.88	0.01	0.03
Prom.		0.28285714	0.22571429	0.23	0.34714286	1.39714286

Tabla 12. Resultados experimento 3.

5.4 Resultados experimento IV

En los resultados del experimento 4 que se muestra en la Tabla 13, se observa que al crecer el número de valoraciones los resultados de precisión, recall y f-measure aumentan. Con más de 150 valoraciones se obtuvieron los mejores resultados con una precisión de 0.79, un recall de 1.00, un f-measure de 0.88, un tiempo de indización por usuario de 0.01 segundos y un tiempo de recomendación por usuario de 0.005 segundos. Los tiempos de ejecución se minimizan con un mayor número de valoraciones porque los usuarios y las películas que no tienen más de 150 valoraciones son eliminados del conjunto de datos. Al reducir el número de valoraciones con más de 10 valoraciones por usuarios y películas se obtuvieron los más bajos resultados con una precisión del 0.34, un recall de 0.11, un f-measure de 0.16, un tiempo de indización de 0.85 segundos y un tiempo de recomendación de 0.02 segundos.

En comparación del experimento anterior, a pesar que tienen un comportamiento similar por el incremento de sus resultados con base en las valoraciones de los usuarios y las películas, este experimento presento mejores resultados en precisión, recall y f-measure con una diferencia notable. En sus tiempos de ejecución se reducen y no son tan variantes porque el algoritmo de CosineScore solo se ejecuta una vez por cada usuario. La mejoría de los resultados se debe a que este experimento utiliza otra técnica de recomendación basada en el usuario y no el ítem como es el caso de los anteriores experimentos.

No. Vals.	No. Usu. Eva.	Precisión	Recall	F-measure	Tiempo indización por usuario (seg)	Tiempo recomendación por Usuario (seg)
>10	566	0.34	0.11	0.16	0.85	0.02
>20	457	0.37	0.13	0.19	0.55	0.02
>30	357	0.38	0.12	0.18	0.39	0.02
>40	301	0.4	0.14	0.2	0.27	0.02
>50	248	0.44	0.17	0.24	0.19	0.01
>100	134	0.6	0.4	0.48	0.05	0.008
>150	67	0.79	1	0.88	0.01	0.005
Prom.		0.47428571	0.29571429	0.33285714	0.33	0.01471429

Tabla 13. Resultados experimento 4.

5.2 Comparación de resultados

La comparación de resultados en general por cada experimento (Exp.) se realiza con base en las métricas de precisión, recall, f-measure, y en los tiempos en segundos de indización y recomendación por usuario. Los resultados por cada experimento se muestran en la Tabla 14, en donde están ordenados de forma ascendente por las métricas de precisión, recall y f-measure.

Exp.	Precisión	Recall	F-measure	Tiempo indización por usuario (seg)	Tiempo recomendación por Usuario (seg)
I	0.255714286	0.21857142	0.218571429	0.37	1.26
II	0.265714286	0.21857143	0.22510013	0.37428571	0.02942857
III	0.282857143	0.22571429	0.2349978	0.34714286	1.39714286
IV	0.474285714	0.29571429	0.33662244	0.33	0.01471429

Tabla 14. Resumen de los resultados de los experimentos.

En el experimento I, se realizaron 5 sub-experimentos y se seleccionó el sub-experimento 5 que presento mejores resultados para comparar el experimento I con los otros experimentos. Se observa en la Tabla 14 que el experimento I presenta los más bajos resultados con una precisión de 0.255, un recall de 0.218 y un f-measure de .0218. En tiempo de indización por usuario en segundos es similar para los cuatro experimentos, sin embargo, en el tiempo de recomendación por usuario en segundos, este experimento es el segundo lugar que presenta un mayor tiempo, esto se debe a que este experimento ejecuta un mayor número de veces el algoritmo CosineScore.

En el experimento II, se observa una breve mejoría en los resultados con una precisión de 0.265, un recall de 0.218 y un f-measure de 0.225. En tiempo de indización por usuario en segundos es similar para los cuatro experimentos, sin embargo, en el tiempo de recomendación por usuario en segundos, este experimento es el segundo lugar que presenta un menor tiempo, esto se debe a que este experimento ejecuta solo una vez el algoritmo CosineScore por usuario.

En el experimento III, se observa una breve mejoría en los resultados con una precisión de 0.282, un recall de 0.225 y un f-measure de 0.234. En tiempo de indización por usuario en segundos es similar para los cuatro experimentos, sin embargo, en el tiempo de recomendación por usuario en segundos, este experimento es el primer lugar que presenta un mayor tiempo, esto se debe a que este experimento ejecuta varias veces el algoritmo CosineScore por cada producto de un usuario.

En el experimento IV, se observa una mejoría significante en los resultados con una precisión de 0.474, un recall de 0.295 y un f-measure de 0.336. En tiempo de indización por usuario en segundos es similar para los cuatro experimentos pero este es el menor tiempo, sin embargo, en el tiempo de recomendación por usuario en segundos, este experimento es el primer lugar que presenta un menor tiempo, esto se debe a que este experimento ejecuta solo una vez el algoritmo CosineScore por cada usuario.

En la gráfica de la Figura 25 se puede visualizar de mejor manera el orden ascendente de los resultados de los experimentos por precisión, recall y f-measure. Se observa que en los resultados de los experimentos I, II y III son similares porque comparte la misma técnica de recomendación que está basada en la similitud de los ítems, sin embargo, se realizaron diferentes experimentos para observar cual presentaba mejores resultados que es el caso del experimento III. Al comparar los resultados del experimento III con el experimento IV, se presentan mejores resultados en el experimento IV que a diferencia de los otros experimentos se implementó una técnica de recomendación basada en la similitud de los usuarios.

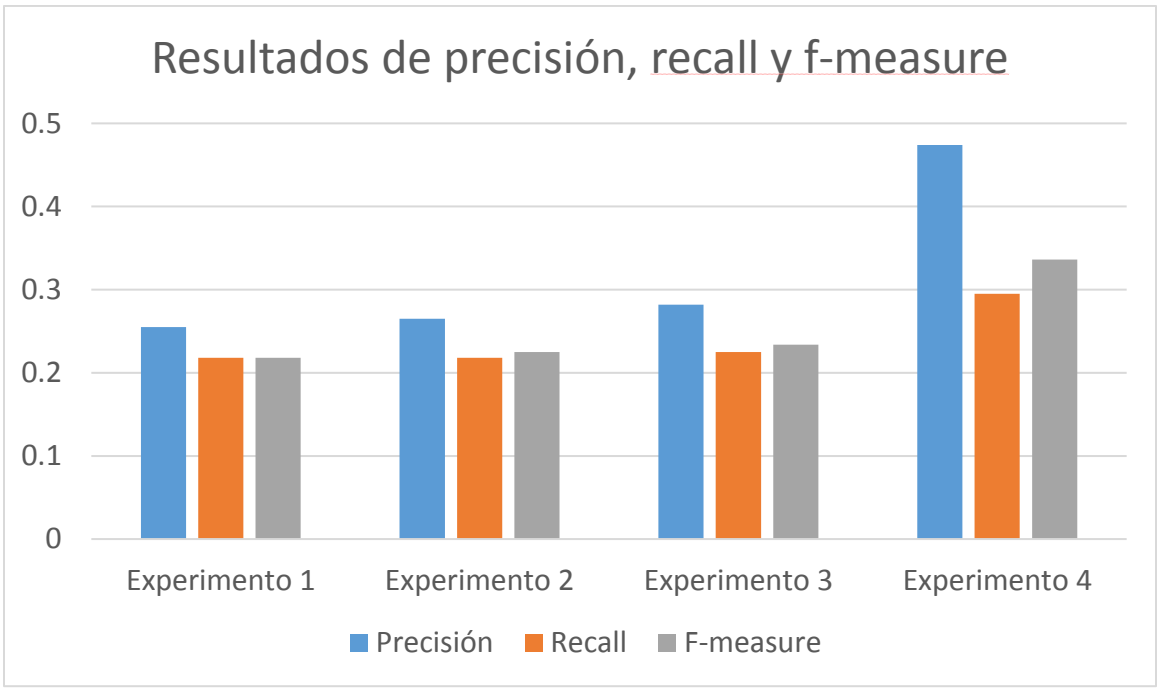

Figura 25. Gráfica de precisión, recall y f-measure.

En la gráfica de la Figura 26 se puede visualizar de mejor manera los resultados de los tiempos de ejecución. Al comparar primero los resultados de los experimentos I, II y III que utilizaron la misma técnica de recomendación basada en la similitud de los ítems se observa que tienen resultados similares en el tiempo de indización por usuario, sin embargo, en el tiempo de recomendación por usuario, el experimento II presenta el menor tiempo porque solo ejecuta el algoritmo de CosineScore una sola vez por cada perfil del usuario, sin en cambio, en los experimentos I y III el algoritmo de CosineScore se ejecuta por cada película que le gusta a un usuario, por tal motivo, el experimento I y III tienen tiempos de recomendación similares. Al comparar los mejores tiempos de ejecución del experimento II y el experimento IV, se observa que tienen resultados similares pero sobresale el experimento IV con un menor tiempo de recomendación por usuario. Ambos experimentos ejecutan solo una vez el algoritmo CosineScore, sin embargo, el experimento II realiza una tarea extra de construir un perfil para el usuario por lo que se eleva un poco su tiempo de recomendación.

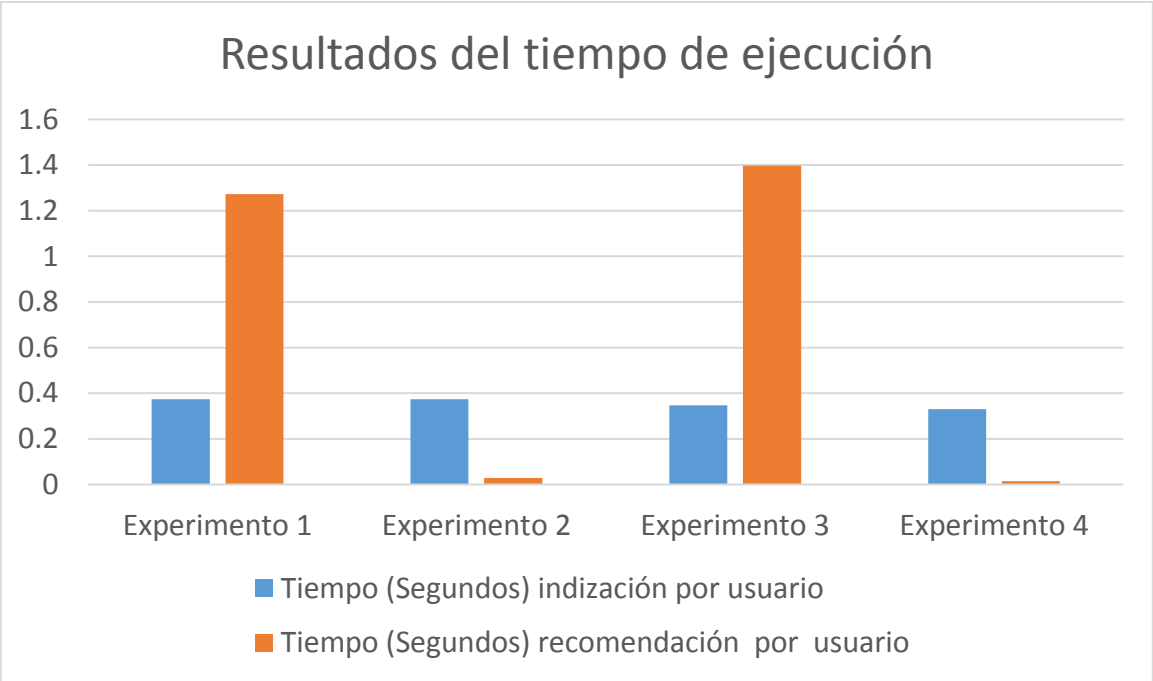

Figura 26. Resultados en tiempo de ejecución.

Al analizar los resultados en el dominio de recomendaciones de películas se muestra que la técnica de filtrado colaborativo basada en el usuario presentan mejores resultados de precisión, recall, f-measure y en los tiempos de indización y recomendación por usuario en comparación del filtrado colaborativo basado en el ítem. Se observa que ambas técnicas de recomendación tienen un comportamiento similar con base en el número de valoraciones de los usuarios y de las películas. Esto se comprueba con las gráficas de las Figuras 27, 28 y 29, se observa que los resultados de precisión, recall y f-measure crecen con un mayor número de valoraciones y los resultados bajan con un menor número de valoraciones.

De la misma forma, en los tiempos de indización por usuario se reducen con el incremento de las valoraciones de los usuarios y las películas porque se eliminan del conjunto de datos los usuarios y las películas que no cumplen con el mínimo de valoraciones, por tal motivo al tener un menor número de valoraciones de los usuarios y las películas los tiempos de ejecución se incrementaran. Esto se comprueba con las gráficas de las Figura 30.

En el caso del tiempo de recomendación por usuario que se presenta en la gráfica de la Figura 31, el experimento I y III presentan un comportamiento similar, entre mayor valoraciones el tiempo disminuye y entre menores valoraciones el tiempo aumenta. Sin embargo, en el experimento II y IV se observa que el tiempo de recomendación no varía con respecto al número de valoraciones y se mantiene uniforme. Esto se debe porque el experimento I y III ejecutan el algoritmo Cosine Score por cada pelicula del

usuario, sin embargo, los experimentos II y IV solo ejecutan una vez el algoritmo CosineScore.

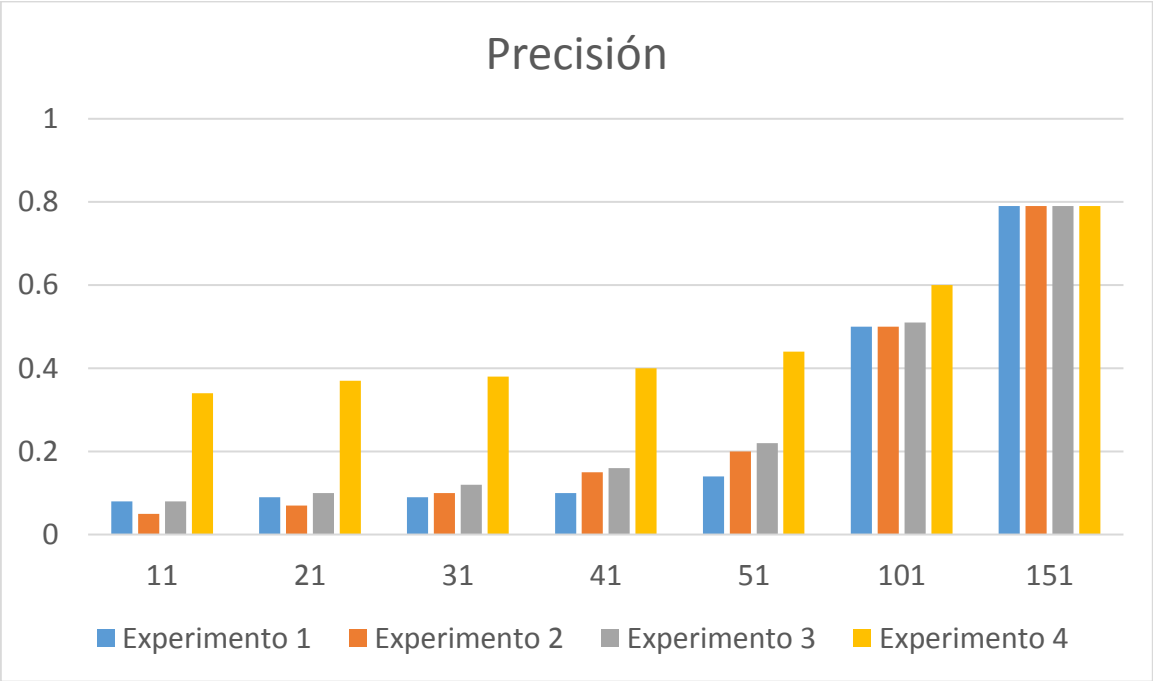

Figura 27. Comparación por cantidad de valoraciones en precisión.

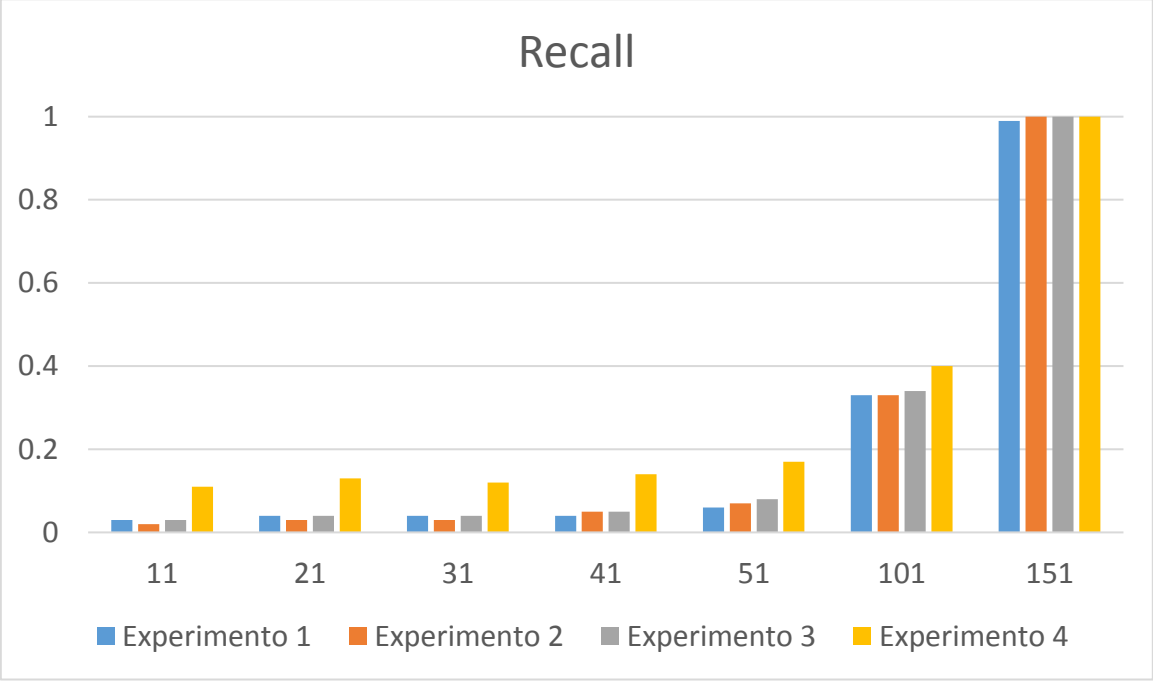

Figura 28. Comparación por cantidad de valoraciones en recall.

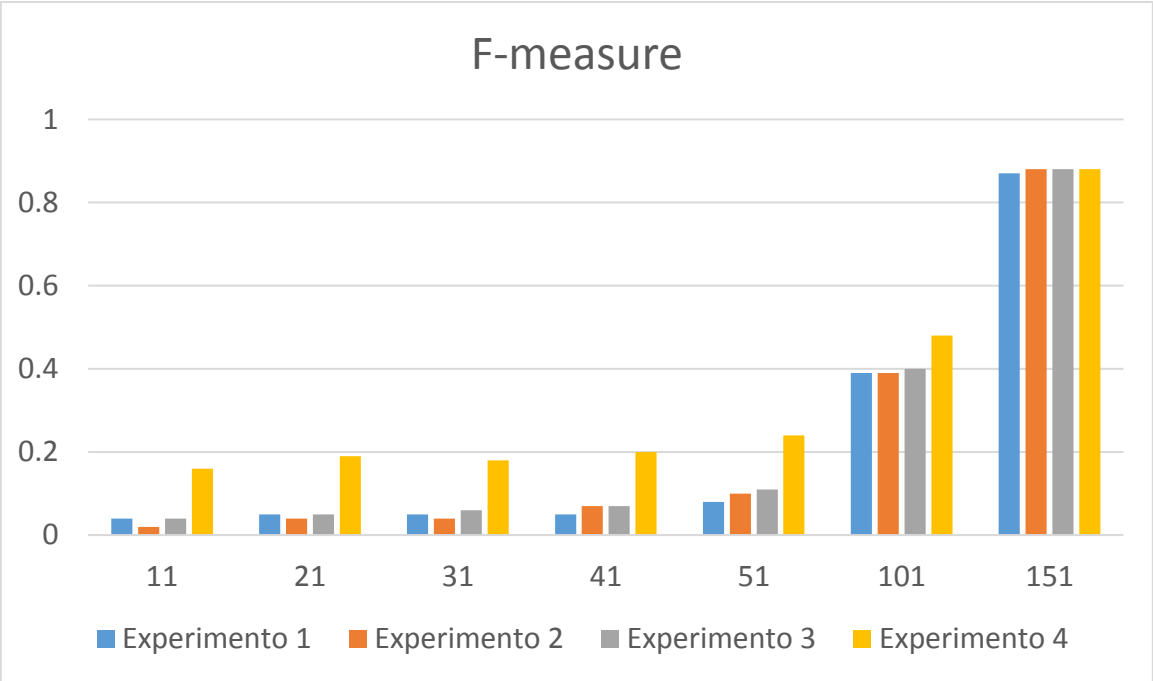

Figura 29. Comparación por cantidad de valoraciones en f-measure.

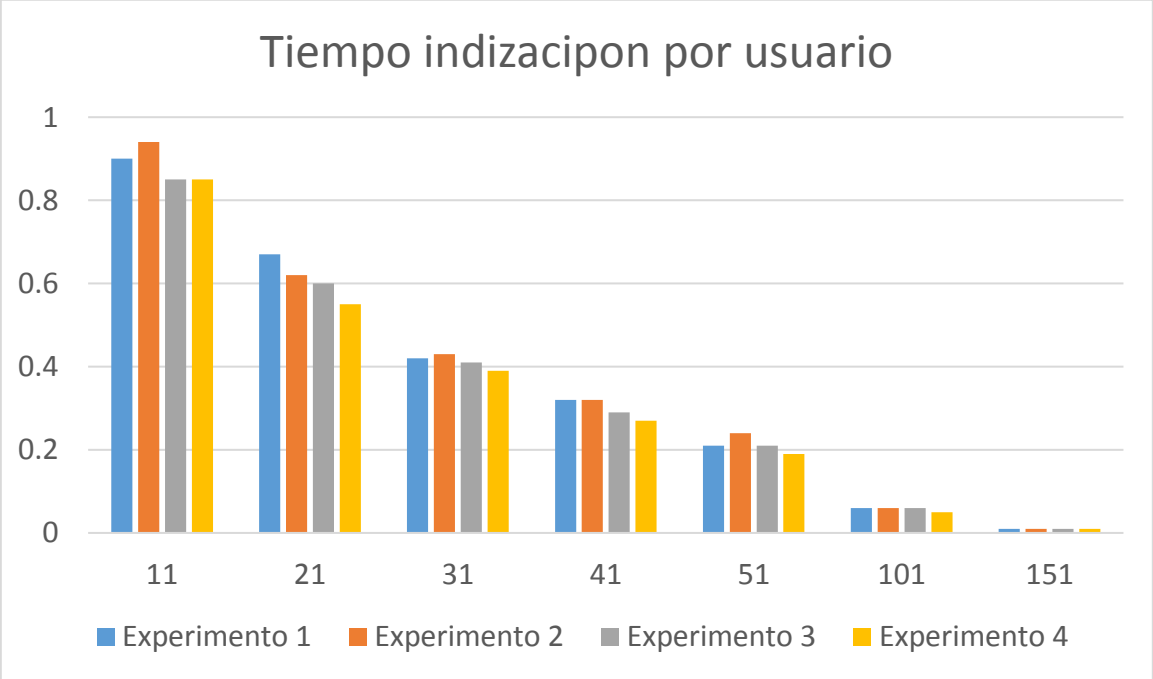

Figura 30. Comparación por cantidad de valoraciones en el tiempo de indización por usuario.

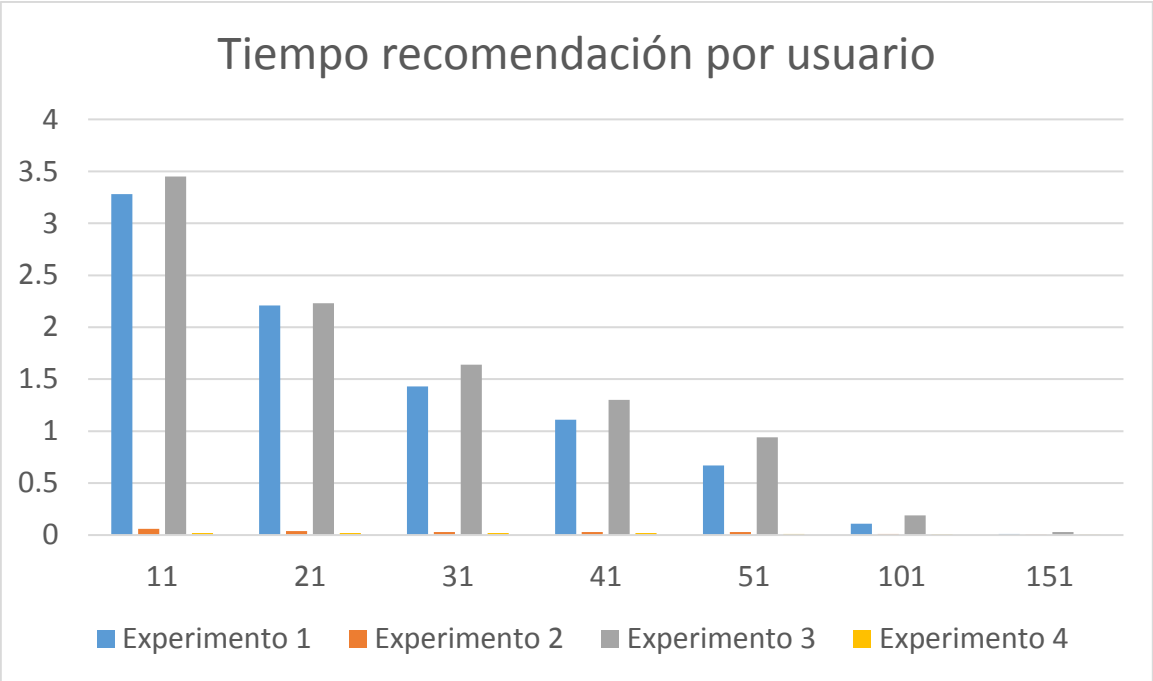

Figura 31. Comparación por cantidad de valoraciones en el tiempo de recomendación por usuario.

Capítulo 6
Conclusiones

En este trabajo de tesis de maestría titulado *Un nuevo paradigma basado en recuperación de información para sistemas de recomendación* se logró el objetivo general y los objetivos específicos al utilizar un conjunto de datos estándar de películas obtenido del sitio web de MovieLens que es ampliamente usado en la educación, la investigación y la industria. El conjunto de datos se pre-procesó para implementar dos paradigmas de recuperación de información para los sistemas de recomendación, el paradigma para la similitud entre usuarios representando a cada usuario como un documento y el paradigma para la similitud entre películas (ítems) representando a cada película como un documento. Se utilizó el modelo de espacio vectorial con el algoritmo de CosineScore para buscar documentos de usuarios o películas similares. Después, con base en el algoritmo de RI se implementaron las técnicas de recomendación de filtrado colaborativo basado en el usuario y el filtrado colaborativo basado en el ítem, se realizaron cuatro experimentos con estas técnicas de recomendación para analizar y comparar los resultados utilizando las métricas de evaluación de precisión, recall y f-measure. Con base en la investigación, el desarrollo y los resultados de este trabajo de tesis se obtienen las siguientes aportaciones y conclusiones.

- Los sistemas de recomendación son fundamentales para muchas compañías y representan un alto impacto económico, social y tecnológico a nivel internacional. Estos sistemas surgen por el exceso de información que existe en el Internet para recomendar ítems relevantes a los usuarios con base en sus preferencias.
- Dos principales retos a los que se enfrentan los sistemas de recomendación son el procesamiento masivo de datos y la minimización de los tiempos de respuesta al utilizar matrices de similitud, por tal motivo, en esta tesis se generaron dos paradigmas basados en recuperación de información para los sistemas de recomendación.

- Se implementó un paradigma para la similitud de los usuarios, representando a cada usuario como un documento.
- Se implementó un paradigma para la similitud de los ítems que en este trabajo son películas, representando a cada película como un documento.
- Se implementó el modelo de espacio vectorial de recuperación de información con el algoritmo de CosineScore para encontrar usuarios similares o películas similares.
- Se implementó la técnica de recomendación llamada filtrado colaborativo basado en el ítem utilizando el algoritmo de recuperación de información.
- Se implementó la técnica de recomendación llamada filtrado colaborativo basado en el usuario utilizando el algoritmo de recuperación de información.
- Se realizaron 4 experimentos, los tres primeros utilizando el filtrado colaborativo basado en el ítem y el último utilizando el filtrado colaborativo basado en el usuario.
- Se demostró que el filtrado colaborativo basado en el usuario presenta mejores resultados que el filtrado colaborativo basado en el ítem en el dominio de las películas.
- En el análisis de los resultados se concluye que los resultados de precisión, recall y f-measure mejoran al aumentar el número de valoraciones de los usuarios y las películas. Y al bajar el número de valoraciones de los usuarios y las películas los resultados decrementan.
- En el análisis de los resultados del tiempo de indización por usuario se demostró que al aumentar las valoraciones de los usuarios y las películas el tiempo de ejecución disminuye y al aumentar el número de valoraciones de los usuarios y las películas el tiempo de ejecución incrementa.
- En el análisis de los resultados del tiempo de recomendación por usuario se demostró que en dos experimentos el tiempo dependía de las valoraciones, sin embargo, en otros dos experimentos no se presentaba esta dependencia.
- Se concluye que para la técnica del filtrado colaborativo basado en el usuario y basado en el ítem es fundamental el número de valoraciones de una película y un usuario para presentar óptimos resultados en precisión, recall y f-measure.
- Se concluye que al utilizar algoritmos de recuperación de información en las técnicas de recomendación de filtrado colaborativo se evita trabajar con algoritmos mínimamente cuadráticos que se aplican sobre matrices tradicionalmente. Por tal motivo, este trabajo es novedoso e innovador en los sistemas de recomendación.
- Como trabajo futuro ambas técnicas del filtrado colaborativo basado en el ítem y basado en el usuario implementadas con algoritmos de RI se pueden utilizar en un sistema de recomendación híbrido para sugerir productos y servicios de la Benemérita Universidad Autónoma de Puebla.

Referencias

Amazon. (18 de Mayo de 2020). *Amazon.es*. Obtenido de Amazon.es: https://www.amazon.es/Acerca-Amazon-Descubre-Nuestra-Empresa-Nuestra-Tecnologia/b?ie=UTF8&node=1323175031

Borges, H., & Lorena, A. (2010). A Survey on Recommender Systems for News Data. *Smart Information and Knowledge Management. Studies in Computational Intelligence, 260*, 129-151.

Costa, A., & Roda, F. (2010). Recommender Systems by means of Information. *ACM International Conference Proceeding Series*.

Escamilla, O., & Marcellin, S. (2017). Estado del arte en los sistemas de recomendación. *Research in Computing Science, 135*(1), 25-39.

Fernández, A. (2012). *Python 3 al descubierto.* Madrid: RC Libros.

Gómez, P., Guarda, T., Cedeño, J., Benavides, A., Alejandro, C., Mosquera, G., . . . Benavides, V. (2019). Sistemas de Recomendación: un enfoque a las técnicas de filtrado. *Iberian Journal of Information Systems and Technologies*(18), 286-293.

Herlocker, J., Konstan, J., Terveen, L., & Riedl, J. (2004). Evaluating Collaborative Filtering Recommender Systems. *ACM Transactions on Information Systems, 22*(1), 5-53.

Isinkaye, F., Folajimi, Y., & Ojokoh, B. (2015). Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal, 16*(3), 261-273.

Jannach, D., Resnich, P., Tuzhilin, A., & Zanker, M. (2016). Recommender Systems - Beyond Matrix Completion. *Communications of the ACM, 59*(11), 94-102.

Kemp, S. (2019). *Digital 2019* . New York: We are social y Hootsuite.

Lemire, D., & Maclachlan, A. (2005). Slope One Predictors for Online Rating-Based Collaborative Filtering. *Proceedings of the 2005 SIAM International Conference on Data Mining*, (págs. 471-475).

Linden, G., Smith, B., & York, J. (2003). Amazon.com Recommendations Item-to-Item Collaborative Filtering. *IEEE Internet Computing, 7*(1), 76-80.

Manning, C., & Schütze, H. (2000). *Foundations of statistical natural language processing.* London, England: MIT Press.

Manning, C., Raghavan, P., & Schütze, H. (2009). *An Introduction to Information Retrieval.* Cambridge, England: Cambridge University Press.

Maxwell, F., & Konstan, J. (2015). The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems, 5*(4), 19.

Nelson, S. (2018). *Spotify's Collaborative Filtering.* SALT LAKE CITY, UT: THE UNIVERSITY OF UTAH: DEPARTMENT OF MATHEMATICS College od Science.

Netflix. (18 de Mayo de 2020). *¿Qué es Netflix?* Obtenido de https://help.netflix.com/es/node/412

Postmus, S., & Bhulai, S. (2018). Recommender system techniques applied to Netflix movie. *Research Paper Business Analytics, 1*(1), 1-31.

Ramos, O. (2017). *Un sistema de recuperación de información como mecanismo de interacción con un robot humanoide.* Puebla: Benémerita Universidad Autónoma de Puebla.

Resnick, P., & Varian, H. R. (1997). Recommender systems. *Communications of the ACM*, 56-58.

Schafer, B., Konstan, o., & Riedl, o. (1999). Recommender Systems in E-Commerce. *1st ACM Conference on Electronic Commerce*, 158-166.

Spotify. (18 de Mayo de 2020). *¿Qué es Spotify?* Obtenido de https://support.spotify.com/es/using_spotify/getting_started/what-is-spotify/

Tovar, L., Montoya, J., & Martelo, R. (2017). SISTEMA ECLÉCTICO DE FILTRADO DE INFORMACIÓN BASADO EN INTELIGENCIA COMPUTACIONAL PARA RECOMENDACIÓN DE ATRACTIVOS TURÍSTICOS DEL CARIBE COLOMBIANO. *Revista Loginn, 1*(2), 6-17.

Anexo A
Sistema de recomendación

En este Anexo A se presentan los principales métodos que se implementaron con los algoritmos de los experimentos utilizando el lenguaje de programación de Python. El primer código fuente que se implemento es el de recuperación de información que contiene el algoritmo CosineScore, se codifico una clase llamada recuperación_de_informacion en la cual tienen dos principales métodos uno llamado crear_indice_posicional( ) y otro llamado cosine_score( ), estos métodos son los que se muestran a continuación. El método crear_indice_posicional( ) es para construir un índice posicional con base en las representaciones de las películas o los usuarios, este método se muestra en la Tabla 15.

```
def crear_indice_posicional(self,archivo1):
        docID=0
        archivo = open(archivo1,"r")
        for linea in archivo.readlines():
            linea = self.normalizar(linea)
            palabras = linea.split(" ")
            posiciones={}
            docID=docID+1
            no_palabras=0
            pos=0
            aux = palabras[:]
            aux.pop(0)
            self.datos_contexto[str(docID)] = " ".join(aux)
            self.datos_id[str(palabras[0])]=docID
            self.datos_docID[str(docID)]=palabras[0]
```

```
            for h in aux:
                pos=pos+1
                if h in posiciones:
                    posiciones[h] = posiciones[h] +","+ str(pos)
                else:
                    posiciones[h]=str(pos)
                    if h in self.df_t:
                        self.df_t[h]=self.df_t[h]+1
                    else:
                        self.df_t[h]=1
            self.lenght[str(docID)]=pos
            for x in posiciones:
                if x in self.indice:
                    self.indice[x]=self.indice[x] + str(docID) + ":" + posiciones[x] + ";"
                else:
                    self.indice[x]=str(docID) + ":" + posiciones[x]+";"
        archivo.close()
        self.N=docID
```

Tabla 15. Método para crear índice posicional.

El método CosineScore ( ) es para calcular la similitud entre una consulta y los documentos más cercanos a esa consulta, en este trabajo se utilizó para buscar usuarios o películas similares. Este método se muestra en la Tabla 16.

```
def cosine_score(self, consulta, NRL):
        freqTC = self.frecuencia_terminos_consulta(consulta)
        for TC in freqTC:
            if TC in self.indice:
                documentos = self.indice[TC].split(";")
                df_t = len(documentos)-1
                for d in documentos:
                    if d!="":
                        doc=d.split(":")
                        docID=doc[0]
                        posiciones=doc[1].split(",")
                        freqTD  = len(posiciones)
                        W_tq= (1 + math.log10(freqTC[TC]) ) * math.log10(self.N/df_t)
                        W_td= (1 + math.log10(freqTD) )      * math.log10(self.N/df_t)
                        if docID in self.score:
                            self.score[docID] = self.score[docID] + (W_tq * W_td)
                        else:
                            self.score[docID] = W_tq * W_td
        for x in self.score:
            self.score[x] = self.score[x] / self.lenght[x]
        return (self.ordenar_diccionario(self.score, 0,NRL))
```

Tabla 16. Método del CosineScore.

A continuación se presentan los métodos para cada experimento. El primer experimento se implementó en el método experimento1( ) que se muestra en la Tabla 17.

```
def experimento1(self, id_usuario):
        N_prductos_similares=1
        numero_recomendaciones=10
        topN_recomendaciones = dict( )
        topN_recomendaciones.clear( )
        docID = str(self.r_usuarios.datos_id[id_usuario])
        documento = self.r_usuarios.datos_contexto[docID]
        print ("DocID Usuario: " + docID )
        print ("ID Usuario: "+id_usuario)
        print("Documento Usuario: "+documento+"\n")
        terminos = documento.split(" ")
        terminos2=list(set(terminos))
        terminos2.sort(key=int)
        for id_producto in terminos2:
            docID_producto = self.r_productos.datos_id[id_producto]
            documento_producto = self.r_productos.datos_contexto[str(docID_producto)]
            productos_similares = self.r_productos.cosine_score(documento_producto,0)
            recomendaciones = dict()
            recomendaciones.clear()
            for i in productos_similares:
                docID = productos_similares[i][0]
                id_pelicula = self.r_productos.datos_docID[docID]
                if id_pelicula not in terminos:
                    recomendaciones[ docID ] = productos_similares[i][1]
                else:
                    pass
            recomendaciones                                                    =
self.r_productos.ordenar_diccionario(recomendaciones,0,N_prductos_similares)
            for r in recomendaciones:
                docID=recomendaciones[r][0]
                similitud=recomendaciones[r][1]
                if docID in topN_recomendaciones:
                    topN_recomendaciones[docID]=topN_recomendaciones[docID]+similitud
                else:
                    topN_recomendaciones[docID]=similitud
        topN_recomendaciones                                                   =
self.r_productos.ordenar_diccionario(topN_recomendaciones,0,numero_recomendaciones
)
        return topN_recomendaciones
```

Tabla 17. Método del experimento 1.

El segundo experimento se implementó en el método experimento2( ) que se muestra en la Tabla 18.

```
def experimento2(self, id_usuario):
        long_pu = 10
        docID = str(self.r_usuarios.datos_id[id_usuario])
        documento = self.r_usuarios.datos_contexto[docID]
        perfil = dict()
        perfil.clear()
        print ("DocID: " + docID )
        print ("ID Usuario: "+id_usuario)
        print("Documento: "+documento)
        terminos = documento.split(" ")-
        peliculas_del_usuario = dict()
        for id_pelicula in terminos:
            if id_pelicula in self.r_productos.datos_id:
                docID_pelicula = str(self.r_productos.datos_id[id_pelicula])
                peliculas_del_usuario[docID_pelicula] = id_pelicula
                aux = self.r_productos.datos_contexto[docID_pelicula].split(" ")
                for a in aux:
                    if (a!=id_usuario):
                        if a in perfil:
                            perfil[a]+=1
                        else:
                            perfil[a]=1
        perfil_usuario = self.r_productos.ordenar_diccionario(perfil,False,long_pu)
        consulta = ""
        c=False
        con = long_pu
        for i in perfil_usuario:
            subcadena=perfil_usuario[i][0]
            peso = int(perfil_usuario[i][1])
            if c:
                consulta = consulta +" "+subcadena
            else:
                consulta = subcadena
            c=True
        print ("\nPerfil del Usuario: "+consulta)
        resultados = self.r_productos.cosine_score(consulta,0)
        recomendaciones = dict()
        for i in resultados:
            docID = resultados[i][0]
            id_pelicula = self.r_productos.datos_docID[docID]
```

```
            if id_pelicula not in terminos:
              recomendaciones[ docID ] = resultados[i][1]
            else:
               print("Eliminando:"+str(i)+"   ID_Pelicula:"+id_pelicula)
        recomendaciones = self.r_productos.ordenar_diccionario(recomendaciones,0,10)
        self.r_productos.mostrar_indice(recomendaciones)
        return (recomendaciones)
```

Tabla 18. Método del experimento 2.

El tercer experimento se implementó en el método experimento3( ) que se muestra en la Tabla 19.

```
def experimento3(self, id_usuario, test):
        N_prductos_similares=10
        numero_recomendaciones=10
        topN_recomendaciones = dict( )
        topN_recomendaciones.clear( )
        presicion_usuario=0
        recall_usuario=0
        productos_test=len(test)
        productos_evaluados=0
        resultados=dict()
        resultados.clear()
        docID = str(self.r_usuarios.datos_id[id_usuario])
        documento = self.r_usuarios.datos_contexto[docID]
        print ("DocID Usuario: " + docID )
        print ("ID Usuario: "+id_usuario)
        print("Documento Usuario: "+documento+"\n")
        terminos = documento.split(" ")
        terminos2=list(set(terminos))
        terminos2.sort(key=int)
        self.r_productos.mostrar_indice(test)
        for id_producto in terminos2:
            docID_producto = self.r_productos.datos_id[id_producto]
            documento_producto = self.r_productos.datos_contexto[str(docID_producto)]
            print ("DocID Producto:",docID_producto)
            print ("ID  Producto:",id_producto)
            print ("Documento Producto:",documento_producto)
            productos_similares = self.r_productos.cosine_score(documento_producto,0)
            recomendaciones = dict()
            recomendaciones.clear()
            for i in productos_similares:
                docID = productos_similares[i][0]
                id_pelicula = self.r_productos.datos_docID[docID]
                if id_pelicula not in terminos:
                    recomendaciones[ docID ] = productos_similares[i][1]
```

```
            else:
                pass
        recomendaciones                                                    =
self.r_productos.ordenar_diccionario(recomendaciones,1,N_prductos_similares)
        if len(recomendaciones)>0:
            productos_evaluados+=1
            r_correctas=0
            for r in recomendaciones:
                docID=recomendaciones[r][0]
                id_prod=self.r_productos.datos_docID[docID]
                if id_prod in test:
                    r_correctas+=1
            presicion = r_correctas  / len(recomendaciones)
            presicion_usuario = presicion_usuario + presicion
            recall = r_correctas  / len(test)
            recall_usuario = recall_usuario + recall
            print ("Presicion por producto:",presicion," Recall por producto:",recall)
    presicion_usuario = presicion_usuario / productos_evaluados
    recall_usuario = recall_usuario / productos_evaluados
    resultados["presicion"]=presicion_usuario
    resultados["recall"]=recall_usuario
    resultados["productos_test"]=productos_test
    resultados["productos_evaluados"]=productos_evaluados
    print ("Presicion usuario:",presicion_usuario," Recall usuario:",recall_usuario, "
NoProductos_test:",productos_test," NoProductosEvaluados:",productos_evaluados)
    return resultados
```

Tabla 19. Método del experimento 3.

El cuarto experimento se implementó en el método experimento4( ) que se muestra en la Tabla 20.

```
def experimento4(self, id_usuario):
    docID = str(self.r_usuarios.datos_id[id_usuario])
    documento = self.r_usuarios.datos_contexto[docID]
    print ("ID Usuario: "+id_usuario)
    print ("DocID: " + docID )
    print("Documento: "+documento)
    terminos = documento.split(" ")
    aux = terminos[:]
    aux.pop(0)
    d = dict()
    for t in terminos:
        if t in d:
            d[t]=1
        else:
            d[t]=1
```

```
        c=0
        cadena=""
        if len(d)==1:
            cadena = t
        else:
            for t in d:
                c+=1
                if (c==1):
                    cadena = cadena + t
                else:
                    cadena = cadena +" "+t
        nv = 11 # No de vecinos
        vecindario = self.r_usuarios.cosine_score(documento,nv)
        consulta = ""
        for i in range(len(vecindario)):
            id_vecino = self.r_usuarios.datos_docID[vecindario[i][0]]
            if id_usuario != id_vecino :
                terminos_vecino = self.r_usuarios.datos_contexto[vecindario[i][0]].split(" ")
                for tv in terminos_vecino:
                    if not tv in terminos:
                        if tv in self.r_productos.datos_id:
                            docID_tv = str(self.r_productos.datos_id[tv])
                            if docID_tv in self.resultados_colaborativo:
                                self.resultados_colaborativo[ docID_tv ]+=1#vecindario[i][1]
                            else:
                                self.resultados_colaborativo[ docID_tv ]=1#vecindario[i][1]
        NR = 10
        recomendaciones_colaborativo                                                    =
self.r_productos.ordenar_diccionario(self.resultados_colaborativo, 1,NR)
        return (recomendaciones_colaborativo)
```

Tabla 20. Método del experimento 4.