Instituto Tecnológico y de Estudios Superiores de Occidente

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación del 29 de noviembre de 1976.

Departamento de Electrónica, Sistemas e Informática
MAESTRÍA EN SISTEMAS COMPUTACIONALES

Diseño de un sistema de recomendación usando algoritmos de aprendizaje máquina

---

Trabajo recepcional para obtener el grado de
Maestro en Sistemas Computacionales

Presenta: **BENJAMÍN ABRAHAM GONZÁLEZ DELGADO**
CVU: 722356

Director **DR. ISMAEL MORENO NÚÑEZ**
Codirector **DR. RIEMANN RUIZ CRUZ**

Tlaquepaque, Jalisco. 11 de octubre de 2018.

AGRADECIMIENTOS

Agradezco al CONACYT por otorgarme la beca de posgrado con número 603434, al ITESO por sus apoyos económicos y la educación de calidad otorgada, así como a Unima Diagnósticos por apoyarme económica y laboralmente durante todo el tiempo del estudio de la maestría.

Quiero agradecer especialmente a mi asesor de tesis, el Doctor Ismael Moreno Núñez por todo el apoyo, la paciencia y las buenas sugerencias que me otorgó para la realización de mi trabajo de obtención de grado.

DEDICATORIA

La presente tesis se la dedico a mi familia que gracias a sus consejos y apoyo incondicional he podido crecer personal y profesionalmente. A mi madre Ana Lilia por ofrecerme su cariño y atención en todo momento, a mi padre Benjamin por ser mi mejor amigo, a mis hermanos por su apoyo y paciencia, y a mi prometida Brenda por ser una motivación constante para mi y en especial por su amor.

RESUMEN

Este trabajo está constituido por una sección inicial donde se presenta una breve introducción a la problemática de los sistemas de recomendación susceptibles a *cold-start* y pobre eficiencia ante el procesamiento de matrices dispersas.

El objetivo principal de este trabajo es proponer y desarrollar un sistema de recomendación que incorpore contenido para solventar el problema del *cold-start* en los métodos de filtrado colaborativo, a la vez que permita una formulación e implementación con matrices dispersas para un procesamiento eficiente.

En la sección de antecedentes se abordan los primeros enfoques de los sistemas de recomendación basados en filtrado y distribución de la información. También se incluye una revisión de los primeros modelos de filtrado colaborativo y una pequeña reseña de los sistemas de recomendación usados en algunas empresas.

Se presenta un capítulo de marco teórico donde se explican los algoritmos para extracción de características, las formulaciones de sistemas de recomendación utilizados y la forma en que se calculan algunas de las métricas más usadas para evaluar los sistemas de recomendación.

Se presenta el desarrollo del trabajo donde se explica a detalle cada paso realizado para la formulación e implementación de un sistema compuesto que integra dos enfoques, uno de filtrado colaborativo SVD y uno de contenido. Posteriormente se muestran los resultados de cada experimento realizado para validar el sistema de recomendación propuesto. Finalmente se presentan las conclusiones del trabajo.

Los resultados demuestran que se resuelven de manera particular los objetivos de procesamiento de datos no estructurados y se aborda el problema del *cold-start* incorporando el contenido derivado de los datos no estructurados. Usando un enfoque de matrices dispersas la implementación resultante es modular y escalable.

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ACRÓNIMOS Y ABREVIATURAS

BC	Basado en contenidos
DF	*DataFrame*
FC	Filtrado colaborativo
FPC	Fracción de pares concordantes
IF	Filtrado de información
IR	Recuperación de información
MAE	Error medio absoluto
RMSE	Raíz del error cuadrático medio
SGD	Gradiente descendente estocástico (*Stochastic Gradient Descent*)
SQL	Lenguaje estructurado de consulta
SR	Sistema de recomendación
SVD	Descomposición de valor singular

1. INTRODUCCIÓN

Resumen: *En este capítulo se presentan brevemente los antecedentes de los sistemas de recomendación en el comercio digital, a su vez se expone la justificación del presente trabajo, la definición del problema a solucionar y se cierra el capítulo con el objetivo principal y los objetivos específicos que se desarrollaron.*

1.1. Antecedentes

Desde la aparición de la Web como una plataforma de negocios y de ventas de artículos, surgió la necesidad de ofrecer al cliente productos de acuerdo a sus necesidades particulares y preferencias. A partir de esto, se comenzó a explorar con sistemas de recomendación en las plataformas de *E-commerce* [1].

El primer sistema de recomendación creado fue Tapestry [2], que introdujo el término de *collaborative filtering* donde se utilizan los *ratings* de usuarios con patrones de compra similares para predecir los faltantes.

En la actualidad la mayoría de las empresas Web que se dedican al E-commerce cuentan con sistemas de recomendación propios y utilizan ratings de usuarios, análisis de patrones de compra y navegación, o preferencias señaladas explícitamente por el usuario [3].

1.2. Justificación

Los sistemas de recomendación tienen como finalidad primordial incrementar las ganancias para un negocio a partir de las ventas de sus productos o servicios, teniendo como principal ventaja que el cliente encuentra de manera rápida un elemento que se adapte a sus preferencias. Los SR pueden implementarse desde distintos enfoques, como el filtrado colaborativo que usa los ratings de un grupo de usuarios o los basados en contenidos que utilizan información adicional de cada artículo para proporcionar sugerencias incluso para productos que no han sido calificados previamente por un grupo de usuarios.

Cada uno de los enfoques tiene ventajas y desventajas que los llevan a ser elegidos para distintos escenarios. En este trabajo se pretende explotar las ventajas que proporcionan los SR de filtrado colaborativo y los basados en contenidos para atender las desventajas de cada uno y generar un enfoque más robusto ante el problema de *cold-start* y matrices dispersas.

Para lograr este propósito se propone un enfoque híbrido que produce una recomendación final a partir de una combinación lineal de un método colaborativo y un método basado en contenido.

1.3. Problema

Dentro de los sistemas de recomendación uno de los problemas más conocidos es el llamado *cold-start* que se presenta cuando el número de *ratings* disponibles es muy pequeño lo que dificulta la aplicación de métodos tradicionales como filtros colaborativos, debido a que no se cuenta con datos suficientes para ofrecer buenas estimaciones [3].

Este problema se ha abordado con diferentes enfoques, entre los que se encuentran los sistemas de recomendación *content-based* y *knowledge-based* [4], aunque los datos requeridos por estos no siempre se encuentran disponibles. El primero de ellos involucra información extra de los artículos y usuarios, como descripciones de producto o palabras clave [5], mientras que el segundo aporta información del

usuario como gustos seleccionados de forma explícita, datos geográficos y demográficos, entre otros. Al involucrar datos de diversas fuentes se dificulta integrarlos en una sola matriz de utilidad o comprimirlos de forma óptima para usarlos como datos de entrada para un modelo de recomendación. Para esto se ha propuesto combinar distintos modelos utilizando lo mejor de cada uno en modelos híbridos, R. Burke explica en [6] distintas maneras de integrar varios enfoques.

Otra deficiencia de los modelos de filtros colaborativos y que heredan ciertos modelos híbridos radica en los cálculos matriciales como similitud y estimación de ratings directamente sobre la matriz de utilidad inicial, que demanda grandes recursos computacionales, y en muchas ocasiones se realizan sobre datos poco útiles o representativos como en matrices muy dispersas con gran cantidad de elementos nulos o vacíos.

Una de las alternativas más recientes para abordar estos problemas ha sido el uso de modelos de variables latentes como los mostrados en [7] que permiten representar a través de vectores los patrones que identifican las relaciones, las cuales pueden ser claramente observables como géneros de películas o más abstractos que no representan una variable observable o física.

Estos métodos y en particular la factorización de matrices presenta una mejor escalabilidad y puede integrar datos de distintos tipos para crear matrices más densas y de menor dimensión.

Aunque los métodos de variables latentes se han implementado para solucionar problemas de densidad de matrices en los sistemas de recomendación, queda como tema de interés la integración de los métodos de matrices dispersas con enfoques de factores latentes para comprimir la información implícita y explícita obtenida de los usuarios, junto con un buen desempeño en manejo de memoria y precisión de resultados.

1.4. Objetivos

1.4.1. Objetivo General:

El objetivo de este trabajo es desarrollar un sistema de recomendación basado en contenidos que permita proporcionar sugerencias de productos o servicios que podrían ser útiles a un usuario o cliente. Este sistema debe ser eficiente con grandes volúmenes de datos y no vulnerable al problema de *cold-start*.

1.4.2. Objetivos Específicos:

1. Analizar, diseñar e implementar un sistema de recomendación que permita generar sugerencias hacia un cliente particular a partir de la descripción del contenido de los artículos.

2. Crear un sistema de recomendación basado en contenidos usando un enfoque de *Data Mining*, específicamente a partir de los datos no estructurados de *reviews* y características de los artículos.

3. Integrar una herramienta modular en el lenguaje de programación Python que permita realizar el proceso completo de creación de un sistema de recomendación basado en contenidos.

4. Lograr que el sistema creado sea eficiente para procesar grandes volúmenes de datos.

5. Establecer una metodología que reduzca la vulnerabilidad del sistema de recomendación al problema de *cold-start*.

1.5. Novedad científica, tecnológica o aportación

Los sistemas de recomendación actuales tienen problemas cuando se intenta proporcionar una recomendación para usuarios o productos nuevos. Las soluciones que integran datos complementarios como los sistemas basados en contenidos suelen dejar de lado la eficiencia para grandes volúmenes de datos.

Se propone crear un sistema que aborde ambos puntos que pueda generar predicciones con un buen nivel de precisión para usuarios y productos nuevos, a su vez que tenga un rendimiento adecuado para *datasets* grandes.

2. ESTADO DEL ARTE O DE LA TÉCNICA

Resumen: *En este capítulo se presenta una descripción de los trabajos previos relacionados con el presente trabajo. Se inicia con los primeros métodos de filtrado y recuperación de la información que son la base del filtrado colaborativo. Se enuncian modelos basados en contenido y recomendación no personalizada. Y se culmina la sección con los modelos actuales de factorización matricial que sirven como base junto con los BC para la propuesta del trabajo.*

2.1. Filtrado y recuperación de la información

Algunos de los primeros trabajos relacionados a los sistemas de recomendación son los sistemas de filtrado de información (*Information filtering*) y de recuperación de información (*IR, Information retrieval).*

El filtrado de información es la manera de identificar los procesos de entrega de información a quienes la necesitan, están diseñados para soportar tanto datos estructurados como semiestructurados, a diferencia de una base de datos típica que sólo soporta estructurados [8].

En [8] se mencionan algunas de las características:

- Manejan textos de forma primaria, e imágenes, voz y video de forma secundaria.
- Implican el uso de una gran cantidad de datos.
- Normalmente utilizan *streams* de datos de entrada provenientes de servidores remotos (como sitios de noticias) o enviados directamente (email).
- El filtrado se basa en las descripciones individuales o preferencias de grupo llamadas perfiles.

En la Figura 1 se representa el modelo general de un sistema de filtrado de información, por la parte izquierda del diagrama se muestra la forma en que la información puede ser consultada, el proceso inicia con los productores de texto, este texto se distribuye y organiza. Normalmente esta organización se hace a partir del indexado de palabras clave del contenido del texto. En la parte derecha de la imagen se ejemplifica la forma en que un usuario o un grupo genera la necesidad de información continua que se representa a partir de perfiles de interés.

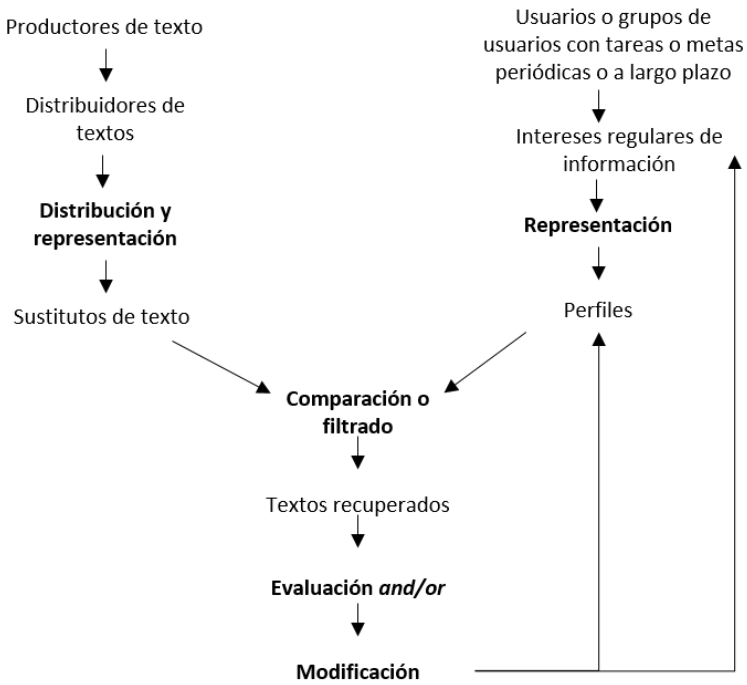

Figura 1 Modelo general de filtrado de información [8]

Cuando el perfil existe se envía de regreso al usuario la información filtrada a partir de una evaluación de contenido del texto.

En la Figura 2 por su parte es mostrado el modelo de recuperación de información que, a diferencia del filtrado, el usuario se genera a sí mismo una necesidad de información que ejecuta en forma de consulta a una base de datos de texto de igual manera indexado. El resultado de la consulta que cumple con los criterios de evaluación se regresa al usuario que interactúa con la información.

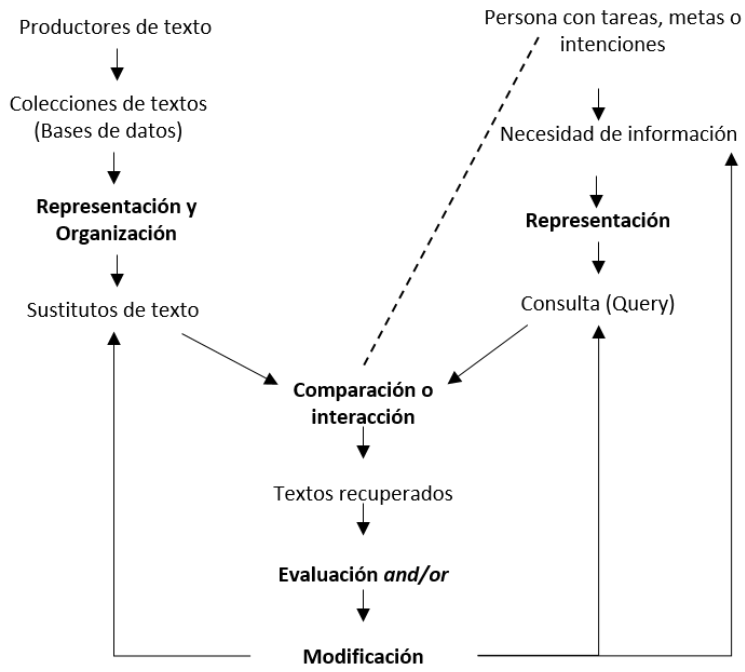

Figura 2 Modelo general de recuperación de información [8]

Algunas de las características de los sistemas de recuperación de información mencionada en [8] son las siguientes:

- Se basa en usuarios simples del sistema que realizan consultas únicas.
- Los textos se encuentran en bases de datos estáticas, mientras que en el filtrado existe una selección por eliminación de las entradas por el *stream* de datos.

La comparación de textos se ha abordado desde distintas perspectivas: booleana, a partir de vectores espaciales y probabilística.

El modelo booleano [9] busca coincidencia exacta entre una consulta y una o más representaciones de textos. Mientras que los otros dos buscan la mejor coincidencia.

El modelo de vectores trata a las consultas y el texto como vectores en un espacio multidimensional, se realiza una comparación entre vectores, usando por ejemplo la medida de similitud coseno [10] que

asume que entre mayor similitud exista entre un texto y una consulta es más probable que este texto sea de relevancia.

El modelo probabilístico se basa en el Principio de clasificación probabilística [11], que realiza la clasificación de textos en una base de datos a partir de la probabilidad de relevancia de una consulta dada una evidencia disponible. La forma más típica de encontrar esa evidencia es a partir de la distribución estadística de los términos contenidos en textos relevantes y no relevantes.

2.2. Tapestry y los filtros colaborativos

Tapestry es un sistema experimental de correo electrónico desarrollado en el centro de investigación de Xerox en Palo Alto [2]. La principal creencia de los creadores de este sistema está basada en que el filtrado de información puede ser más efectivo cuando los humanos forman parte en el proceso. Tapestry está diseñado para soportar filtros basados en contenido y filtros colaborativos.

Los filtros colaborativos del sistema funcionan a partir de las reacciones de los usuarios a documentos que leen, mismas que se guardan a través de anotaciones.

En [2] se define la arquitectura del sistema Tapestry de la siguiente manera:

- Clasificador: Encargado de leer documentos de fuentes externas como correo electrónico, noticias y las agrega al almacenamiento de documentos.
- Almacenamiento de documentos: Proporciona almacenamiento para todos los documentos del sistema.
- Almacenamiento de anotaciones: Almacena las anotaciones relacionadas a los documentos
- Filtrado: Consultas realizadas por el usuario, los documentos filtrados van a la caja pequeña de cada usuario.
- Caja pequeña: Colas de documentos de interés para un usuario en particular
- *Remailer:* Envía contenido periódicamente al usuario de la caja pequeña vía correo electrónico.
- *Appraiser:* Clasifica de forma personalizada los documentos de un usuario, da prioridad y categoriza.
- Lector/Navegador: Es la interfaz de acceso a los servicios de Tapestry, desde aquí pueden manejarse los documentos y las consultas de cada usuario.

El filtrado de información utilizado por Tapestry es un modelo binario para aceptar o rechazar un documento.

El sistema Tapestry está basado en un modelo cliente servidor y un lenguaje de consulta propio *Tapestry Query Language (TQL)* basado en SQL, pero con la funcionalidad extra de soportar *sets* en las consultas.

2.3. Aplicaciones actuales de los sistemas de recomendación

Algunas de las más grandes empresas tecnológicas de la actualidad basan su negocio en la utilización de sistemas de recomendación.

Amazon utiliza los algoritmos para incrementar sus ventas, usando publicidad personalizada. Utilizan un algoritmo llamado *Item-to-Item Collaborative Filtering* [12]. Este algoritmo busca artículos similares, no usuarios similares. De esta manera personaliza los productos ofrecidos al usuario a través de la Web y por medio de campañas de correo electrónico.

Para determinar que artículos son parecidos se crea una tabla utilizando la información de los artículos que los clientes suelen comprar juntos. Para determinar la similitud entre dos elementos utilizan el método de medición de coseno [10]. Cada vector corresponde a un artículo y las dimensiones corresponden a los clientes que lo han comprado previamente.

Netflix utiliza los sistemas de recomendación como una manera de atraer y mantener clientes al servicio de películas *online* [13]. El problema de recomendación tiene como finalidad predecir la calificación que un usuario le daría a un contenido en un rango de 1 a 5 estrellas. Netflix utiliza varios algoritmos de recomendación para presentar su contenido, entre estos se encuentran:

- *Personalized Video Ranker, PVR*: Este algoritmo ordena el catálogo completo de videos por género de forma personalizada para el usuario.
- *Top-N Video Ranker:* Proporciona algunas recomendaciones personalizadas combinadas con popularidad.
- *Video-video Similarity:* Genera recomendaciones a partir de los elementos similares a un video con buena calificación.

Spotify utiliza como su sistema de recomendación principal los filtros colaborativos. Si dos usuarios escuchan la misma canción sus gustos probablemente son similares, de igual manera si dos canciones son escuchadas por el mismo grupo de usuarios probablemente suenan de forma similar [14].

2.4. Modelos de recomendación no personalizada

Los modelos de recomendación no personalizada presentados por Porya et. al en [4] son los más simples, ya que no toman en cuenta las preferencias de los usuarios. Las recomendaciones producidas por el sistema se generan de la misma manera para cada usuario, pueden realizarse manualmente por el vendedor, están basadas en la popularidad de un artículo o a partir de los artículos más nuevos. Se han utilizado principalmente dos tipos de algoritmos para estos sistemas: Recomendación por opinión agregada y asociación básica de productos [4].

La recomendación por opinión agregada utiliza el promedio de las calificaciones de los usuarios. Generalmente estas calificaciones se proporcionan en rangos numéricos, que podrían ser estrellas con

valores de 0 a 5. Este tipo de recomendaciones se utilizan en sitios de restaurantes, hoteles, películas entre otros.

Una de las desventajas de estas recomendaciones es que por estar basada en promedio sufre gran variación con valores extremos, una mala calificación podría tener un gran peso sobre otras.

Por otro lado, la recomendación por asociación de productos proporciona un poco más de contexto, se puede identificar en sitios de e-commerce como Amazon, a partir del mensaje "personas que han comprado el artículo1 también han comprado el artículo2". Matemáticamente la recomendación para un rating binario en este caso puede representarse como el porcentaje superior a un umbral de X-compradores que también han comprado un *artículo* Y. En la siguiente ecuación se describe un rating binario:

$$\hat{r} = \begin{cases} 1 & si \ \dfrac{(X \ AND \ Y)}{X} > umbral \\ 0 & en \ otro \ caso \end{cases} \tag{2.1}$$

2.5. Modelos de recomendación basados en filtros colaborativos

Los filtros colaborativos basan sus predicciones en las calificaciones de los usuarios o el comportamiento de los mismos en un sistema. Estos asumen fundamentalmente que la opinión de otros usuarios puede utilizarse para proveer una predicción a un usuario activo. Es decir, si algunos usuarios comparten preferencias sobre ciertos elementos, entonces sus preferencias pueden ser las mismas en elementos distintos. Las tareas de recomendación pueden verse como un problema de *information retrieval* en el cual su dominio son los elementos consultados para formar un perfil de preferencia de un usuario [15].

El modelo de filtros colaborativos fue introducido por el recomendador de artículos *GroupLens Usenet* [16], el de música *The Ringo* [17] y videos *BellCore*.[18].

El filtrado colaborativo tiene dos enfoques principales como se mencionan en [3], basado en usuarios y basado en artículos. Ambos parten de una matriz de ratings o de utilidad como la mostrada en la Figura 4, que representa la forma en que los usuarios demuestran ciertas preferencias sobre los *artículos.* Los elementos que no tienen una calificación definida dentro de la matriz pueden representar artículos para los que el usuario no ha asignado un rating o bien que no conoce.

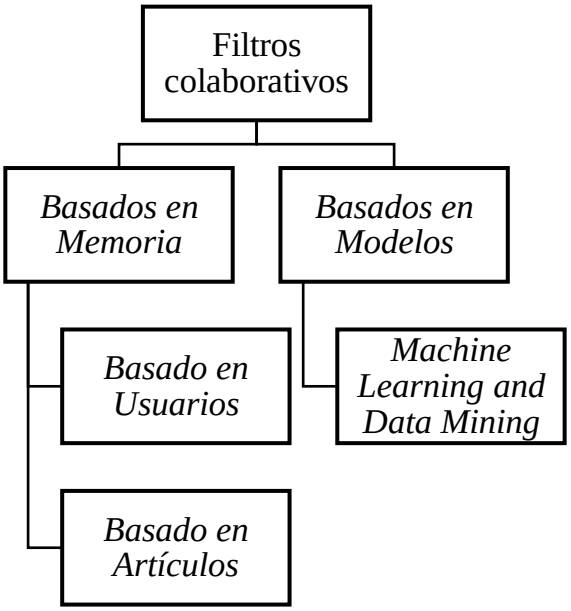

Figura 3 Modelos de filtros colaborativos

La finalidad de los filtros colaborativos es estimar esos ratings faltantes. Para el caso del modelo basado en usuarios se calcula una matriz de similitud entre todos los usuarios mientras que para el modelo basado en artículos se realiza el cálculo de similitudes entre todos los artículos como en la Figura 5. Diferentes funciones para el cálculo de similitudes se mencionan en [3]. Esta matriz de similitud es utilizada posteriormente para tomar los elementos más parecidos al elemento a calcular y generar un valor a partir de los valores de ratings de estos.

	i_1	i_2	i_3	i_4
u_1	1			
u_2			3	
u_3	2			
u_4				3

	u_1	u_2	u_3	u_4
i_1	1		2	
i_2				
i_3		3		
i_4				3

Figura 4 Matriz de utilidad, basado en usuarios (izq.) y en artículos (der.)

El modelo de filtros colaborativos tradicional presenta grandes deficiencias cuando se utilizan matrices muy dispersas ya que no existe información suficiente para el cálculo de similitud lo que provoca que los valores obtenidos sean muy bajos y generen ratings muy cercanos a cero. Los artículos con mayor cantidad de ratings tienden a ser los más recomendados por ser los únicos con calificaciones suficientes para ofrecer valores de similitud altos. La representación matricial de estos modelos resulta muy ineficiente al tener una gran cantidad de elementos vacíos o con valores nulos y el tiempo de cómputo por su orden cuadrático

al procesar pares de elementos, aumenta en gran medida con matrices más grandes. Entre otras de las deficiencias de este enfoque es en el problema de *cold-start* tanto para usuarios como para productos.

Para el caso de usuarios nuevos al no tener ratings para *artículos* registrados es imposible estimar su similitud con otros usuarios lo que impide completar el vector de ratings. A su vez en el caso de nuevos productos requieren ser calificados por varios usuarios para determinar su similitud con otros *artículos* y evitar que la estimación sea imprecisa al tener pocos datos para evaluar.

	u_1	u_2	u_3	u_4
u_1	1	0.1	0.2	0.5
u_2	0.1	1	0.5	0.4
u_3	0.2	0.5	1	0.8
u_4	0.5	0.4	0.8	1

	i_1	i_2	i_3	i_4
i_1	1	0.2	0.1	0.3
i_2	0.2	1	0.5	0.6
i_3	0.1	0.5	1	0.1
i_4	0.3	0.6	0.1	1

Figura 5 Matriz de similitud, usuarios (izq.) y artículos (der.)

2.6. Modelos de recomendación basados en contenidos

En los sistemas de recomendación basados en contenidos, los atributos de un artículo son usados para realizar recomendaciones. Las descripciones de un elemento son etiquetadas como calificaciones que se utilizan como datos de entrenamiento para una clasificación por usuario o un problema de regresión. Los atributos son las descripciones de los artículos comprados mientras que la variable de clase (dependiente) son las calificaciones previas que el usuario ha otorgado a esos artículos. El modelo creado se utiliza posteriormente para predecir la calificación que un usuario otorgaría a un elemento distinto.

Algunas de las ventajas de estos sistemas aparecen cuando se trata de artículos nuevos, que no cuentan con calificaciones previas suficientes para utilizar un sistema de filtro colaborativo.

A su vez presentan desventajas como que puede realizar recomendaciones demasiado obvias, fenómeno que reduce la diversidad de artículos recomendados. También se considera un sistema deficiente para nuevos usuarios, debido a que necesita conocer los intereses previos del mismo [3].

Los componentes de un sistema de recomendación basado en contenidos pueden resumirse en los siguientes puntos presentados en [3]:

- Pre-procesamiento y extracción de características: Dependiendo del dominio se extraen características de los elementos que se representan como un vector de palabras clave.

- Aprendizaje basado en contenidos de perfiles de usuario: Se construye un modelo de usuario para predecir sus intereses, para esto es necesario la retroalimentación del usuario en forma de productos calificados (retroalimentación explícita) o su actividad (retroalimentación implícita). Esos valores junto con los atributos del artículo son utilizados para construir el modelo.
- Filtrado y recomendación: En este paso el modelo se utiliza para hacer recomendaciones de artículos para usuarios específicos, este paso debe ser eficiente para poder realizarse en tiempo real.

Si bien el sistema basado en contenidos mejora las recomendaciones cuando se encuentra presente el problema de cold-start, sólo representa una solución para el caso de nuevos productos ya que puede extraer sus atributos y usarlos para hacer predicciones. Mientras que nuevos usuarios no pueden relacionarse con este tipo de sistemas al no tener información de gustos previos. Además, al no utilizar los ratings de otros usuarios se reduce la diversidad y novedad en las recomendaciones.

Bajo el enfoque basado en contenidos se presenta el problema de una matriz de utilidad dispersa, tomando como referencia una matriz de utilidad de n documentos contra m términos, donde $n \ll m$. Se tiene que los términos en común de los documentos son muy poco frecuentes por lo que la mayoría de las columnas contienen elementos nulos con una matriz de una dimensión muy grande y con poca información relevante.

2.7. Modelos de Variables Latentes

Los modelos de variables latentes son una alternativa a los filtros colaborativos, que caracterizan al mismo tiempo artículos y usuarios para ofrecer una recomendación. Las variables latentes pueden tener sentido semántico claramente identificable como el caso de las películas que se correlaciona a algún género o bien pueden ser variables latentes abstractas no identificables. Los modelos de variables latentes como *Singular Value Decomposition (SVD)*, transforman productos y usuarios en un espacio de variable latente haciéndolos directamente comparables y permitiendo caracterizar ambos para inferir los *ratings* faltantes.

Estos modelos suelen estimar de manera efectiva la estructura general que relaciona todos los artículos, sin embargo, no presentan los mejores resultados para asociaciones fuertemente relacionadas de un set pequeño de elementos. En [19] se presenta un sistema que combina las ventajas de los métodos de variables latentes con las de filtros colaborativos (*Neighborhood methods*).

Los métodos de variables latentes abordan el problema de las matrices dispersas al trabajar con matrices de dimensión mucho menor a la matriz de utilidad original, lo que disminuye el uso de memoria y complejidad computacional.

2.8. Métodos de factorización de matrices

La factorización de matrices ha sido la base para muchos métodos de variables latentes. En su forma básica caracteriza artículos y usuarios a través de factores en forma de vectores.

Estos métodos combinan una buena escalabilidad con una gran exactitud al momento de la predicción.

Una de las principales fortalezas de la factorización de matrices es que permite la incorporación de información adicional como retroalimentación implícita de usuarios a través de historial de compras, historial y patrones de búsqueda e incluso la interacción con interfaces gráficas. Al agregar información adicional permite representar los datos de entrada en una matriz más densa.

Estos métodos son los más usados en la actualidad y han demostrado una mejoría considerable en las estimaciones al agregar factores como *biases*, variables temporales y atributos del producto como lo exponen en Koren et al. en [7] .

3. MARCO TEÓRICO/CONCEPTUAL

Resumen: *En este capítulo se presentan las bases teóricas y conceptuales sobre los Sistemas de Recomendación que sustentan el desarrollo de este trabajo. Se inicia explicando los SR de FC, BC y los tipos de ratings existentes. El ciclo completo para el desarrollo de un SR así como las ecuaciones que representan a distintos modelos se explican en este capítulo. Finalmente se detalla la forma en que puede evaluarse el desempeño de un SR.*

3.1. Introducción a los sistemas de recomendación.

Los SR consisten en el uso de distintos algoritmos para ofrecer una recomendación a una entidad conocida como "usuario" sobre un elemento llamado "artículo". Los datos de entrada para las recomendaciones pueden ser explícitos o implícitos. Los primeros son expresados directamente por el usuario calificando ciertos artículos, mientras que los segundos se basan en suposiciones sobre la actividad previa del usuario, como pueden ser cantidad de visitas, patrones de compra, por nombrar algunos.

El principio básico que rige los algoritmos de recomendación es la dependencia significativa que puede existir entre un usuario y un artículo [3]. Esas dependencias pueden ser aprendidas a partir de los datos en forma de matriz de ratings o de contenidos.

Los modelos para SR más utilizados son los filtros colaborativos (FC) y los basados en contenidos (BC).

Existen dos aproximaciones para lograr los objetivos finales de un SR:

1. Problema de predicción: Esta aproximación permite hacer una predicción de la combinación usuarios-artículos. Para m usuarios y n artículos existe una matriz $m\ x\ n$ incompleta donde los valores especificados (observados) son utilizados para el entrenamiento.
2. Problema de *ranking*: En la práctica no es necesario estimar todos los ratings para generar recomendaciones. En su lugar puede ser mejor determinar los top-k artículos para un usuario. En este caso los valores absolutos de la predicción no son importantes.

3.1.1. Filtrado Colaborativo

Los modelos CF utilizan los ratings para generar las recomendaciones. La idea básica de este modelo radica en que los ratings observados pueden tener una alta correlación entre varios usuarios y artículos. Esto se obtiene a partir de una medida de *similarity* que es utilizada para hacer inferencias.

El reto principal de estos modelos son las matrices dispersas (*sparse*) pues suelen existir una pequeña porción de elementos observados por los usuarios ante un amplio universo de artículos. Mientras que su principal ventaja es su fácil implementación y que puede generar recomendaciones buenas en los casos promedio.

Dentro del filtrado colaborativo existen dos aproximaciones, los modelos basados en memoria que explotan las correlaciones usuario-usuario y a nivel de artículo-artículo, por otro lado, los basados en modelos que utilizan técnicas de optimización para entrenar un modelo de la misma manera que se entrena un clasificador supervisado.

3.1.1. Basados en Contenidos

En los métodos de recomendación basados en contenido los atributos de los artículos son utilizados para hacer recomendaciones. El término contenido se refiere a los atributos que describen al artículo. En estos métodos se combinan los ratings conocidos de un usuario sobre ciertos artículos con las descripciones de estos para encontrar artículos similares.

Estos modelos tienen como ventaja que pueden generarse recomendaciones para nuevos artículos cuando no se cuenta con ratings disponibles (cold-start), esto a partir de que el usuario activo puede haber calificado otros artículos con atributos similares.

Sus desventajas radican en que pueden generar predicciones muy obvias debido al uso de palabras clave, además si bien es efectivo para nuevos artículos, no lo es para nuevos usuarios ya que requiere del historial del usuario activo.

3.1.2. Tipos de ratings

Los usuarios pueden expresar sus preferencias de manera explícita o implícita, y esto tiene un impacto al decidir qué tipo de SR utilizar. Los ratings pueden representarse de las siguientes formas:

- Con una escala de valores *continuos* con un número infinito de posibilidades. Esta representación es muy poco usada ante la forma de intervalos debido a que carece de relación semántica.
- En *intervalos* siendo la forma más frecuente, por ejemplo, para representar en una escala de 5 puntos que va desde disgusto por completo a una fuerte afinidad, se pueden utilizar valores dentro del set {1,2,3,4,5} o {-2,-1,0,1,2}.
- Los ratings *ordinales* representan categorías como “Fuerte disgusto”, “Disgusto”, “Neutral”, “Afinidad” y “Fuerte afinidad”. Su diferencia con los intervalos es que no necesariamente la distancia es la misma en el valor de cada par. Suelen ser equidistantes para evitar el *bias* de elección forzada que ocurre cuando no existe un valor neutral [3].
- Los ratings *binarios* que se usan para representar afinidad o rechazo por un artículo pueden representarse con valores de 0 y 1.
- Un caso especial son los ratings *unarios*, que ocurren en los sistemas donde hay un mecanismo para expresar una afinidad a un artículo, pero no para expresar el rechazo, son más comunes dentro de los ratings implícitos que se generan en situaciones como realizar una compra o ver un artículo.

3.2. Filtrado colaborativo basado en Neighborhood.

Estos algoritmos fueron los primeros en desarrollarse para filtrado colaborativo, existen dos tipos principales, el FC basado en usuarios y el basado en artículos. La matriz de ratings se representa con la

matriz R_{mxn} que contiene m usuarios y n artículos. El rating para un usuario u para un artículo j se representa por r_{uj}.

Sólo una pequeña porción de los ratings se encuentra especificados en la matriz dando origen a una matriz dispersa, las entradas de la matriz son los datos de entrenamiento, mientras que los ratings faltantes son los datos de prueba. La idea general de este modelo es utilizar la similitud usuario-usuario o artículo-articulo para generar recomendaciones.

3.2.1. FC Basado en usuarios

En esta aproximación se utilizan los usuarios similares al usuario objetivo para estimar sus ratings faltantes. Para determinar los vecinos más cercanos al usuario objetivo se calcula su función de similitud con todos los otros usuarios.

Una medida que captura la similitud entre dos usuarios *u* y *v*, es el coeficiente de correlación de Pearson. El primer paso es calcular el rating promedio de cada usuario, para *u* se encuentra definido con la ecuación 3.1.

$$\boldsymbol{\mu_u = \frac{\sum_{k \in I_u} r_{uk}}{|I_u|} \quad \forall u \in \{1 \dots m\}} \tag{3.1}$$

Donde I_u representa el set de índices de los artículos evaluados por el usuario u.

El coeficiente de correlación de Pearson se encuentra determinado por la ecuación 3.2.

$$Sim(u,v) = Pearson(u,v) = \frac{\sum_{k \in I_u \cap I_v}(r_{uk} - \mu_u) \cdot (r_{vk} - \mu_v)}{\sqrt{\sum_{k \in I_u \cap I_v}(r_{uk} - \mu_u)^2} \cdot \sqrt{\sum_{k \in I_u \cap I_v}(r_{vk} - \mu_v)^2}} \tag{3.2}$$

El coeficiente de correlación se calcula entre el usuario objetivo y todos los demás usuarios, entonces se seleccionan los top-k usuarios con el coeficiente mayor siendo $P_u(j)$ el set de los k usuarios más cercanos al usuario objetivo que han calificado el artículo j, se puede generar el rating a partir del promedio ponderado de los ratings con los coeficientes entre el usuario objetivo y los usuarios de mayor coeficiente.

Un problema con esta aproximación es que los ratings pueden tener diferentes escalas, un usuario puede tender a calificar la mayoría de los artículos con valores muy altos mientras que otro de forma negativa, para evitarlo se deben centrar los ratings por usuario de acuerdo con su media representada por 3.3.

$$s_{uj} = r_{uj} - \mu_u \; \forall u \in \{1 \dots m\} \tag{3.3}$$

La ecuación general para predicción de ratings con el modelo basado en neighborhood es la siguiente [3]:

$$\hat{r}_{uj} = \boldsymbol{\mu_u} + \frac{\sum_{v \in P_u(j)} Sim(u,v) \cdot s_{vj}}{\sum_{v \in P_u(j)} |Sim(u,v)|} = \boldsymbol{\mu_u} + \frac{\sum_{v \in P_u(j)} Sim(u,v) \cdot \left(r_{vj} - \boldsymbol{\mu_v}\right)}{\sum_{v \in P_u(j)} |Sim(u,v)|} \tag{3.4}$$

3.2.2. FC Basado en artículos

En los modelos basados en artículos las similitudes se calculan sobre las columnas de la matriz de ratings, las filas deben estar centradas con una media de cero. Tal como en el modelo de usuarios el valor promedio de ratings por artículo debe sustraerse de cada rating para crear una matriz centrada en la media, este rating ajustado se representa por s_{uj}. El set de usuarios que han especificado un rating para el artículo *i*, se representa como U_i. A pesar de que la correlación de Pearson se puede usar en columnas, la función de similitud coseno suele generar mejores resultados [3]. La función coseno con ajuste entre artículos *i* y *j*, se define como:

$$AdjustedCosine(i,j) = \frac{\sum_{u \in U_i \cap U_j} s_{ui} \cdot s_{uj}}{\sqrt{\sum_{u \in U_i \cap U_j} s_{ui}^2} \cdot \sqrt{\sum_{u \in U_i \cap U_j} s_{uj}^2}} \tag{3.5}$$

Similar al FC basado en usuarios, en este caso se deben seleccionar los *top-k* artículos más parecidos tomando los resultados de la ecuación 3.5 para un artículo objetivo *t*. Los top-k artículos para los cuales el usuario ha otorgado una calificación anterior se denotan con $Q_t(u)$. La predicción de rating $\hat{r}_{ut}$ se determina con:

$$\hat{r}_{ut} = \frac{\sum_{j \in Q_t(u)} AdjustedCosine(j,t) \cdot r_{uj}}{\sum_{j \in Q_t(u)} |AdjustedCosine(j,t)|} \tag{3.6}$$

3.3. FC basado en Modelos.

En este tipo de sistemas se crean modelos de datos utilizando clasificadores supervisados o no supervisados.

En un problema de clasificación general, existe una matriz *mxn* donde las primeras (*n-1*) columnas son las características o variables independientes, mientras que la última columna son las variables de clase o dependientes. Todas las columnas se encuentran especificadas por completo. Se suele separar la totalidad de los datos disponibles en un *subset* de entrenamiento con *k* filas y un subset de pruebas con *m-k* filas. En los SR se utiliza una generalización de los problemas de clasificación llamada problema de llenado de matrices (*Matrix Completion Problem*), que tiene diferencias clave como:

- No existe una separación clara entre variables dependientes e independientes, cada columna puede ser una u otra dependiendo de las entradas que se consideren para el modelo predictivo.
- No existe una clara separación entre los datos de entrenamiento y pruebas, se consideran las entradas especificadas (observadas) como entrenamiento y las no especificadas (faltantes) como prueba.
- A diferencia de los problemas generales de clasificación, las filas no siempre son consideradas como las entradas y las columnas como las características, por ejemplo, en el FC basado en artículos las entradas son la transpuesta de las características.

Los SR de FC basados en modelos tienen algunas ventajas entre las que destacan:

- *Eficiencia espacial:* El modelo aprendido suele ser mucho más pequeño que la matriz de ratings original. El FC de neighborhood basado en usuarios tiene una complejidad espacial $O(m^2)$, y el basado en artículos de $O(n^2)$.
- *Velocidad de entrenamiento y predicciones:* De igual manera la fase de procesamiento en los métodos de neighborhood suele ser de complejidad cuadrática, mientras que el FC basado en modelos generalmente es más eficiente en esta fase y se pueden hacer predicciones con un modelo más compacto.
- *Previenen overfitting:*A partir de métodos de regularización puede evitarse el overfitting para generar modelos más robustos.

Los métodos para generar recomendaciones más eficientes en la actualidad suelen ser los de FC basados en modelos, más específicamente los modelos de variables latentes [3].

3.3.1. Modelos de factorización de matrices.

Una matriz de ratings es una representación con overfit de los gustos de usuario y las descripciones de los artículos.

La factorización matricial tiene como finalidad crear una representación más compacta de los gustos de usuario y las descripciones de los artículos, reduciendo así la complejidad computacional.

Este tipo de modelos trata de predecir los ratings a partir de la caracterización de usuarios y artículos con *k* variables inferidas a partir del entrenamiento sobre los datos. Las variables latentes encontradas pueden ser obvias como géneros de películas (comedia, drama, acción, para público infantil), o no interpretables.

Los modelos de factorización de matrices combinan una buena escalabilidad con predicciones precisas, además de ofrecer flexibilidad para incorporar otras fuentes de datos y crear una matriz de ratings más densa.

3.3.1.1. SVD

De manera general la finalidad del SVD en filtrado colaborativo es tomar una matriz de ratings y reducir sus dimensiones para todos los usuarios y artículos en dimensiones de gustos más pequeñas.

Para una matriz de ratings R de m usuarios por n artículos, es posible factorizarla en dos matrices Q y P^T, así como una matriz diagonal Σ, este proceso se representa en la Figura 6.

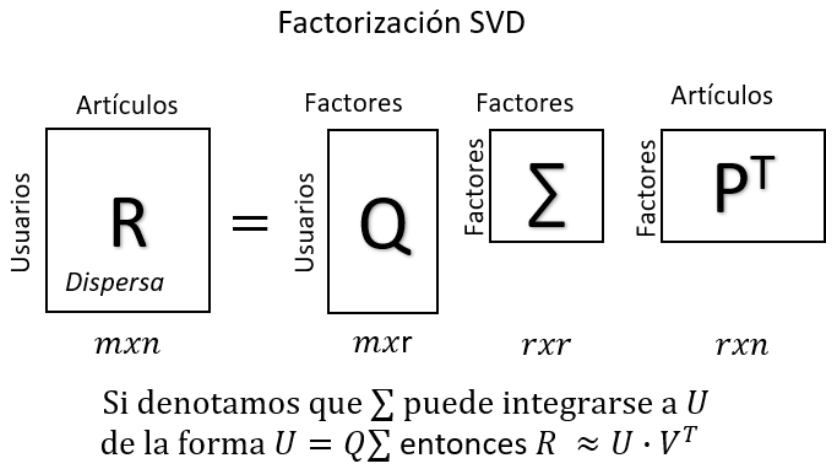

Figura 6. Factorización SVD en SR

Cada valor en la diagonal de Σ representa que tan importante es la nueva dimensión. Si los valores son ordenados de forma descendente tomando el valor absoluto pueden tomarse los k mayores y reconstruirse la matriz. La matriz reconstruida *R'* es la matriz *rank-k* más cercana a la R original. Esto significa la mejor aproximación con k-dimensiones. El valor de k es llamado también el rango de una matriz, y el valor adecuado se obtiene experimentalmente dependiendo del dominio.

En el SVD para filtrado colaborativo, cada gusto de usuario puede representarse como un vector mxk de gustos de usuario y uno kxn de descripción de artículos. Entonces para hacer una predicción sólo es necesario ejecutar un producto punto entre ambos vectores. Cada dimensión es ortogonal, es decir, linealmente independiente entre cada una.

Cada artículo i se asocia a un vector $q_i \epsilon \mathbb{R}^k$ y cada usuario u se asocia a un vector $p_u \epsilon \mathbb{R}^k$. El producto resultante de ambos captura la interacción entre el usuario u y el artículo i, representado en la ecuación 3.7 [7].

$$\hat{r}_{ui} = q_i^T p_u \tag{3.7}$$

En este caso los usuarios e artículos se encuentran en el espacio de características de dimensiones k. El vector p_u indica que tan relevante es una característica para un usuario mientras que el vector q_i indica que tanto de esa característica se encuentra presente en un artículo.

Debido a que la factorización matricial SVD de manera algebraica no está definida para matrices incompletas, es común el uso de algoritmos de aprendizaje para encontrar las variables latentes (q_i y p_u) que buscan minimizar el error cuadrático regularizado dado por la ecuación 3.8 [7]

$$\min_{q_i, p_u} \sum_{(u,i) \in K} (r_{ui} - q_i^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2) \tag{3.8}$$

Donde K es el set de pares (u, i) para los cuales el valor r_{ui} es conocido (set de entrenamiento). El sistema aprende de los ratings observados para predecir los ratings desconocidos. El factor λ previene el

overfitting al regularizar los parámetros aprendidos, el valor adecuado suele encontrarse con técnicas de cross-validation.

3.4. Sistemas basados en contenidos

Estos sistemas generalmente se basan en las descripciones de los artículos y la información disponible del usuario, para esto se recurre a transformaciones de información no estructurada a una forma más estandarizada. Por ejemplo, las descripciones se convierten a palabras clave (*keywords)*.

Los componentes principales de un SR BC suelen incluir una porción fuera de línea donde ocurre un preprocesamiento de la información disponible y un aprendizaje sobre la misma, así como una porción en línea donde ocurren las predicciones. Este modelo en línea es el que genera las recomendaciones para los usuarios. Los componentes del SR BC se enlistan como sigue:

1. Preprocesamiento y extracción de características
2. Aprendizaje de los perfiles de usuario
3. Filtrado y recomendación

3.4.1. Preprocesamiento y extracción de características (text mining).

En esta fase se concentra toda la información disponible de los artículos, aunque es posible cualquier tipo de representación de los datos, la más común es el manejo de palabras clave extraídas de las descripciones.

En algunos casos las descripciones son convertidas a un modelo de *Bag of Words (BoW)* [10]. Cada una de las palabras clave puede obtener un peso a partir del dominio del problema o su relevancia dentro de un texto, para asignar ese peso es común utilizar el valor TF-IDF. Estos modelos se describen de la siguiente manera:

- Bag of words: Es un modelo que cuenta la cantidad de veces que aparece una palabra en un documento. Es una forma de preparar el texto para clasificación de documentos. Por ejemplo, en el documento;

  'I love that place, the food is awesome'

  Se genera una tabla de frecuencias como en la Tabla 1:

Tabla 1. Ejemplo modelo Bag of Words

documento	I	love	that	place	the	food	is	awesome
1	1	1	1	1	1	1	1	1

- TF-IDF: Es un modelo que también permite juzgar un tópico por las palabras que contienen, en este caso las palabras tienen un peso asignado por la medida TF-IDF que indica relevancia y no simplemente frecuencia. Reemplaza al modelo de bag of words y en algunos casos presenta mejores resultados. Se representa con la ecuación 3.9.

$$w_{i,j} = tf_{i,j} x \log\left(\frac{N}{df_i}\right) \tag{3.9}$$

Donde para cada termino i en un documento j:
$tf_{i,j}$= Número de apariciones de i en j
df_i = Número de documentos que contienen i.
N = Número total de documentos

3.4.2. Aprendizaje de perfiles de usuario.

En esta etapa se puede ver como un problema de clasificación de texto cuando los ratings son valores discretos, o como una regresión para entidades numéricas.

Se asume que existe un set de documentos D_L etiquetados por el usuario denominado activo. Esos documentos de entrenamiento corresponden a las clasificaciones de los artículos, de los cuales se han procesado y extraído características, mismos que son usados para crear un modelo de aprendizaje supervisado. Se cuenta también con un set para pruebas D_U que no han sido evaluados por el usuario y que pueden ser potencialmente recomendados.

El modelo más simple es al igual que en el filtrado colaborativo el clasificador *nearest neighbor*, el primer paso es el cálculo de una medida de similitud.

Siendo $\bar{X} = (x_1 \dots x_d)$ y $\bar{Y} = (y_1 \dots y_d)$ un par de documentos, en los cuales las frecuencias normalizadas del término *i* están dadas por x_i y y_i, respectivamente, la similitud de la función coseno está dada por la ecuación 3.10.

$$\text{Cosine}(\bar{X}, \bar{Y}) = \frac{\sum_{i=1}^{d} x_i y_i}{\sqrt{\sum_{i=1}^{d} x^2{}_i}\sqrt{\sum_{i=1}^{d} y^2{}_i}} \tag{3.10}$$

Esta medida es utilizada en el dominio del texto por su habilidad para ajustarse a las distintas longitudes de los documentos [3].

3.4.3. Filtrado y recomendación.

Para cada documento en D_U se determinan sus k-nearest neighbors en D_L usando los valores de la similitud del coseno. El promedio de las evaluaciones para los vecinos de cada artículo en D_U es determinado. Ese promedio es el valor de cada rating para el artículo correspondiente en D_U.

Los documentos en D_U son ordenados de acuerdo con el valor de rating calculado y se recomiendan al usuario activo los mejores valorados.

3.5. Métricas de desempeño de un SR

Los SR pueden evaluarse de distintas formas de acuerdo con los objetivos, el más conocido es la precisión en las clasificaciones, pero también pueden evaluarse otros factores como la diversidad, novedad, sorpresa, robustez o escalamiento, algunas pueden ser cuantificadas objetivamente y otras subjetivas a la experiencia de usuario que son medidas en encuestas. Para este trabajo las métricas a evaluar son la precisión y escalamiento y fracción de pares concordantes.

Para medir la precisión de un SR se puede considerar el caso donde existe un rating conocido r_{ui}, que es estimado con el modelo como $\hat{r}_{ui}$. Entonces el error específico a esa entrada está dado por:

$$e_{ui} = \hat{r}_{ui} - r_{ui} \tag{3.11}$$

Posteriormente el error total es calculado promediando el error de las entradas en términos de valores absolutos o cuadráticos [3].

Los componentes principales de una evaluación de precisión son los siguientes:

- *Diseño de evaluación de precisión:* No todas las entradas de ratings conocidos pueden ser usadas para entrenamiento y para validar la precisión puesto que se sobreestimaría la precisión por *overfitting*. Es importante realizar la separación de los datos, si las entradas totales se representan con S, un pequeño set $E \subset S$debe tomarse para evaluación y el set $S - E$ para entrenamiento. También pueden aplicarse técnicas usadas en algoritmos de clasificación generales como la validación cruzada [20].
- *Métricas de precisión:*
    - *Precisión de ratings estimados:* Como se explica en la ecuación 3.11 el error de cada entrada es la diferencia entre el rating observado y estimado, el error total puede calcularse con la función MSE (*Mean Squared Error)*:

      $$MSE = \frac{\sum_{(u,i)\in E} e_{ui}^2}{|E|} \tag{3.12}$$

      Es común encontrar en la literatura el RMSE (*Root Mean Squared Error*) como evaluación de la precisión de los modelos de regresión, teniendo como ventaja que se encuentra en unidades de ratings:

      $$RMSE = \sqrt{\frac{\sum_{(u,i)\in E} e_{ui}^2}{|E|}} \tag{3.13}$$

      Una característica del RMSE es que tiende a penalizar desproporcionadamente los errores grandes por el término cuadrático. Una medida que no genera esta penalización es la MAE (*Mean Absolute Error*).

$$MAE = \frac{\sum_{(u,i)\in E}|e_{ui}|}{|E|} \tag{3.14}$$

La escalabilidad de un sistema puede ser determinada con las siguientes medidas:

- *Tiempo de entrenamiento:* La mayor parte de los SR requieren una fase de entrenamiento, los de FC por neighborhood requieren el pre-cálculo de los *topk* usuarios o artículos, mientras que la factorización de matrices requiere realizar la determinación de las variables latentes. En general unas cuantas horas son aceptables ya que ese entrenamiento suele realizarse fuera de línea.
- *Tiempo de predicción:* Una vez que el modelo se ha entrenado, es usado para determinar las recomendaciones para un usuario en particular, este tiempo es crucial ya que impacta la latencia para que el usuario reciba las respuestas.
- *Uso de memoria:* Cuando las matrices de ratings son grandes, es un reto mantener la matriz completa en memoria, por lo que deben diseñarse algoritmos que minimicen el uso de memoria.

Las métricas de precisión RMSE y MAE si bien muchas veces son consideradas como la medida de mayor importancia en un SR pueden evaluar negativamente a una predicción correctamente posicionada para un usuario si el valor calculado se encuentra fuera de la escala. Además, asumen únicamente valores de rating numéricos y no ordinales, para el caso donde las predicciones que se muestran al usuario activo en forma de topk artículos estas métricas no son las adecuadas.

Una medida orientada a ranking como se explica en [21] es la fracción de pares concordantes (FPC). En un set $\hat{R}$, se define el número de pares concordantes para un usuario u como la cuenta de los elementos posicionados correctamente para los ratings calculados $\hat{r}_u$, como se muestra en la ecuación 3.15.

$$n_c^u = \left|\{(i,j)|\hat{r}_{ui} > \hat{r}_{uj}\ and\ r_{ui} > r_{ui}\}\right| \tag{3.15}$$

$$n_d^u = \left|\{(i,j)|\hat{r}_{ui} >= \hat{r}_{uj}\ and\ r_{ui} < r_{ui}\}\right| \tag{3.16}$$

Los pares discordantes se calculan con la ecuación 3.16. Sumando para todos los usuarios se define $n_c = \sum_c n_c^u$ y $n_d = \sum_d n_d^u$. A partir de estas sumatorias se calcula la fracción de pares concordantes (FPC) como en la ecuación 3.17.

$$FPC = \frac{n_c}{n_c + n_d} \tag{3.17}$$

4. DESARROLLO METODOLÓGICO

Resumen: *En este capítulo se presenta en detalle el desarrollo metodológico que incluye la descripción de los datos utilizados para crear el modelo, su estructura y las variables que los componen. Así como el procesamiento que se realizó sobre estos para extraer contenido para el SR BC. Se explica en este capítulo la estructura del SR compuesto de un SR BC y uno de FC con SVD y su formulación. Finalmente se presenta la metodología usada para la validación.*

4.1. Datos para SR

Para el desarrollo y evaluación de la propuesta se utilizaron datos públicos de Yelp [22] una base de datos que contiene información de distintos usuarios y establecimientos, así como los ratings otorgados por usuarios acerca de esos establecimientos.

4.1.1. Descripción de los datos

El *dataset* original está compuesto por seis archivos en formato JSON:

- *business*; contiene información de los establecimientos como el nombre, un identificador, la ubicación, cantidad de estrellas y cantidad de evaluaciones, horario, así como una variedad de atributos y categorías que funcionan como descriptores del artículo.
- *review*; tiene información de las evaluaciones realizadas por un usuario a un establecimiento, se relacionan a partir del identificador de usuario e identificador de establecimiento y contiene el texto de la evaluación, la fecha y una calificación.
- *user*; incluye datos del usuario como identificador, primer nombre, cantidad de evaluaciones realizadas, fecha de inicio en Yelp, un mapeo de amigos y otros metadatos asociados al usuario.
- *checkin;* contiene la información del *Check In* por fecha y horario de todos los establecimientos con su identificador.
- tip; son consejos de los usuarios sobre un establecimiento, son más cortos que las evaluaciones y contienen sugerencias rápidas. No contienen un valor de evaluación.
- photos; este es un archivo auxiliar para un dataset complementario de imágenes, tiene la relación entre identificador de imagen, de establecimiento y una descripción corta de la fotografía, así como una etiqueta.

El tamaño del dataset completo es de 6.52 GB, para este trabajo son utilizados los archivos de business, user y review, para extraer características necesarias para la creación del modelo propuesto.

En la tabla Tabla 2 se representa la cantidad de elementos contenidos en cada archivo disponible, el tamaño total del archivo y el tamaño promedio de cada elemento. Esto nos da una idea de las dimensiones del dataset disponible y es útil para las métricas de rendimiento de la propuesta.

Tabla 2. Tamaño de elemento por archivo

Archivo	Cantidad de elementos	Tamaño total del archivo (MB)	Tamaño promedio del elemento (KB)
business	156 639	132.272	0.844
user	1 183 362	1 572.537	1.329
review	4 736 897	3 819.731	0.806

En la Figura 7 a partir de un mapa se grafican las ubicaciones de los negocios del archivo business, el archivo no tiene un dato explícito del país, pero a partir de los valores de latitud y longitud se genera la

figura que muestra marcadores en los países de EUA, Argentina, Alemania, Noruega y Reino Unido, siendo el primero de ellos la ubicación con mayor número de marcadores.

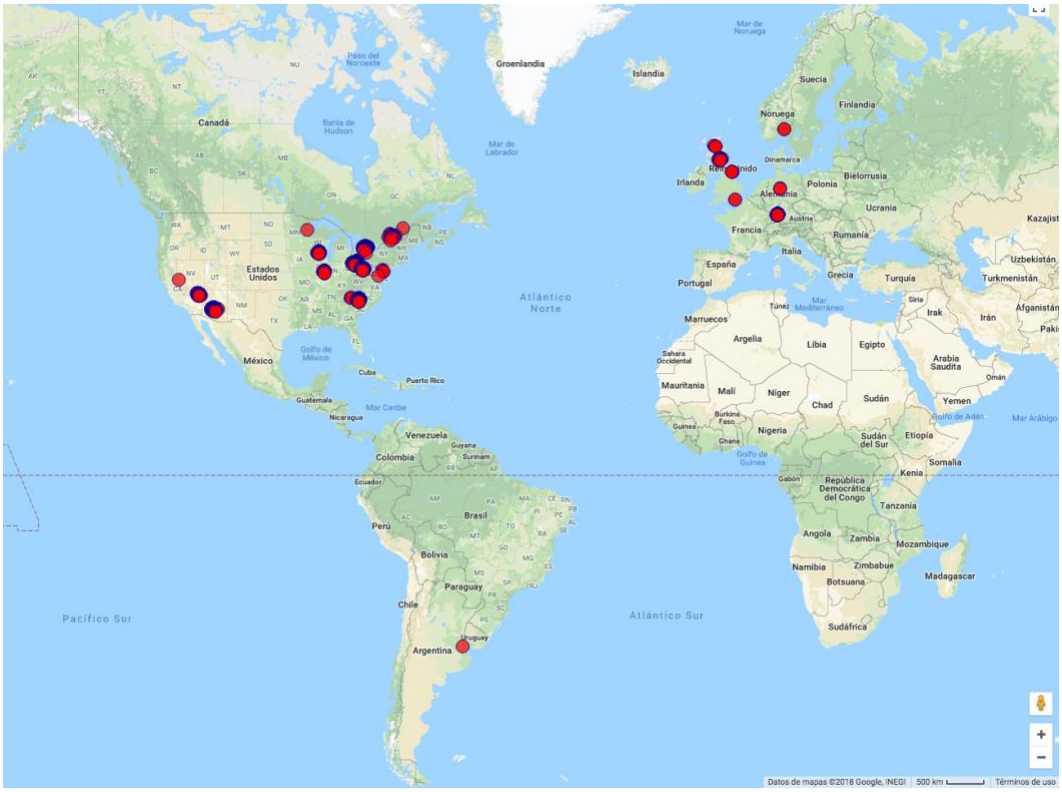

Figura 7. Ubicación de los negocios en el business.json

El archivo de reviews tiene texto en varios idiomas entre los que destacan: inglés, francés, alemán y español. No existe una variable para identificar el idioma del texto, en la sección de procesamiento de datos se describe una técnica que se utilizó para generar las etiquetas del idioma para cada review.

4.1.2. Estructura

Un aspecto importante para abordar el procesamiento de los datos es la estructura y el formato de los elementos de cada archivo disponible puesto que las etapas de transformación y extracción de características dependen del manejo de los mismos.

En la Figura 33 del Apéndice A, se representa la estructura para el dataset original de negocios, donde se mezclan datos de tipo llave-valor con otros elementos anidados que contienen a su vez otros elementos de tipo llave-valor, esto sucede para la característica *attributes*, que contiene descriptores específicos para cada negocio, como puede ser, si tiene estacionamiento, servicio de valet, si es a pie de calle, entre otros.

La característica de *categories* es otro caso especial, puesto que la información se encuentra contenida en un arreglo de cadenas de texto, donde cada cadena representa una categoría utilizada para anotar los negocios.

En la Figura 35 y Figura 34 del Apéndice A, se muestran las estructuras de review y user respectivamente. Para estos archivos no existen elementos anidados, únicamente arreglos de cadenas de texto para representar relaciones entre usuarios y años en los que un usuario ha sido elite.

Las características de cada variable se describen con más detalle en la siguiente sección.

4.1.3. Variables

En la Tabla 3, Tabla 4 y Tabla 5 se muestran las variables de cada archivo y el tipo de dato recomendado para asignarse, puesto que al manejar datos no estructurados, no se cuenta con tipos de datos ni esquemas definidos, es relevante conocer la naturaleza de cada característica para su asignación de tipo. A su vez cada variable contiene una descripción en las tres tablas, esta descripción es de utilidad para la selección de características en la etapa de procesamiento de datos.

Tabla 3. Variables disponibles de negocios

Variable	Tipo de Dato	Descripción
business_id	string	Identificador del negocio de 22 caracteres
name	string	Nombre del negocio
neighborhood	string	Nombre del vecindario
address	string	Dirección completa del negocio
city	string	Ciudad de ubicación
state	string	Código de estado en 2 caracteres, si es aplicable
postal code	string	Código postal
latitude	float	Latitud
longitude	float	Longitud
stars	float	Estrellas de rating redondeado al punto medio más cercano
review_count	integer	Número de reviews
is_open	integer	0 o 1 si el negocio está cerrado o abierto respectivamente
attributes	object	Atributos del negocio en forma llave-valor
categories	array	Categorías del negocio
hours	object	Horario en formato de 24 en llave-valor

Tabla 4. Variables disponibles de usuarios

Variable	Tipo de Dato	Descripción
user_id	string	Identificador del usuario de 22 caracteres
name	string	Primer nombre del usuario
review_count	integer	Número de reviews que el usuario ha escrito
yelping_since	string	Fecha en que el usuario entró a Yelp con formato YYYY-MM-DD
friends	array	Amigos del usuario con user_id
useful	string	Cantidad de votos recibidos en la categoría *useful*
funny	integer	Cantidad de votos recibidos en la categoría *funny*
cool	integer	Cantidad de votos recibidos en la categoría *cool*
fans	integer	Número de fans que tiene el usuario
elite	array	Años en los que ha sido élite

average_stars	float	Promedio de rating de todos sus reviews
others	various	Otros metadatos del usuario que no son relevantes para este trabajo.

Tabla 5. Variables disponibles de reviews

Variable	Tipo de Dato	Descripción
review_id	string	Identificador del review de 22 caracteres
user_id	string	Identificador único del usuario de 22 caracteres, mapea al usuario en el archivo user.json
business_id	string	Identificador único del negocio de 22 caracteres, mapea al negocio en el archivo business.json
stars	integer	Estrellas de rating
date	string	Fecha en formato YYYY-MM-DD
text	string	El review del usuario
useful	integer	Cantidad de votos recibidos en la categoría *useful*
funny	integer	Cantidad de votos recibidos en la categoría *funny*
cool	intefer	Cantidad de votos recibidos en la categoría *cool*

4.2. Procesamiento de datos

El procesamiento de datos previo a la implementación del SR tiene como finalidad crear un formato de datos de entrada propicio para crear un modelo de filtrado colaborativo y una recomendación por contenidos. Este procesamiento contiene etapas bien definidas para cualquier preparación de datos al crear modelos de aprendizaje automático. Estas etapas incluyen la transformación, la selección de características y la normalización de datos.

4.2.1. Transformación

Con la finalidad de poder lograr una selección de características y codificación de variables se transformó la estructura original de cada uno de los archivos, utilizando herramientas del lenguaje de programación Python como pandas [23].

Pandas permite utilizar DataFrames (DFs), que son estructuras tabulares de dos dimensiones con tamaño mutable y potencialmente heterogéneas, a las cuales se les puede aplicar operaciones aritméticas sobre filas o columnas. Estas características hacen que este tipo de estructura sea la elegida para manejar las fuentes de datos.

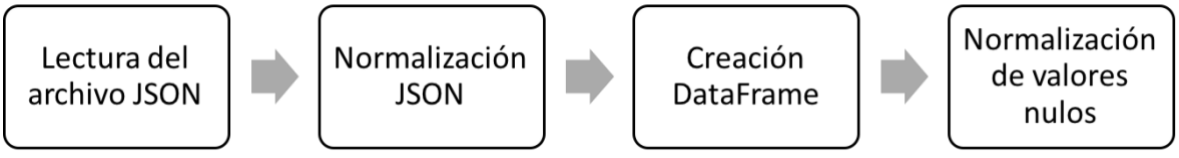

Figura 8. Proceso de transformación de datos

El proceso completo de transformación es el mostrado en la Figura 8, en cada uno de los pasos se ejecutó lo siguiente:

- Lectura del archivo JSON: Utilizando una librería de Python llamada *codecs* es posible leer un archivo en forma de *StreamReader*, es decir, no se almacena el archivo completo en memoria, si no que se lee de forma iterativa mientras existan líneas por leer.
- Normalización de JSON: Cada una de las líneas del archivo es procesada como un diccionario de Python y a partir de la función *json_normalize* de pandas es posible modificar las variables de objetos anidados para que se puedan adaptar al DF de dos dimensiones. En la Figura 9 se encuentra el resultado de la función json_normalize para un elemento del archivo business. La nomenclatura para objetos anidados tiene la forma atributoPadre.atributoHijo.*.
- Creación DF: Si bien el resultado de la función de normalización de json es una estructura de tipo DF, se ejecutó un procesamiento en lotes para evitar saturar la memoria y se concatenaron los DFs parciales en un DF total.
- Normalización de valores nulos: Por la naturaleza de los datos, es posible que algunos atributos o características tengan varias formas de representar un valor negativo al no existir para algún elemento, en este paso se normalizaron las distintas representaciones a un valor *False* de tipo booleano, esto se ejecutó principalmente para el caso de los negocios donde este tipo de casos es más común.

```
data = {
    "business_id": "YDf95gJZaq05wvo7hTQbbQ",
    "attributes": {
        "RestaurantsPriceRange2": 2,
        "BusinessParking": {
            "garage": False,
            "street": False,
            "validated": False,
            "lot": True,
            "valet": False
        },
        "BikeParking": True,
        "WheelchairAccessible": True
    },
    "categories": ["Shopping", "Shopping Centers"],
    "hours": {
        "Monday": "10:00-21:00"
    }
}
```

attributes.BikeParking	True
attributes.BusinessParking.garage	False
attributes.BusinessParking.lot	True
attributes.BusinessParking.street	False
attributes.BusinessParking.valet	False
attributes.BusinessParking.validated	False
attributes.RestaurantsPriceRange2	2
attributes.WheelchairAccessible	True
business_id	YDf95gJZaq05wvo7hTQbbQ
categories	[Shopping, Shopping Centers]
hours.Monday	10:00-21:00

Figura 9. Resultado de la función json_normalize para objeto anidado del archivo business

4.2.2. Selección de características

En la etapa de selección de características cada una de las fuentes de datos es utilizada de distinta forma para generar las entradas al SR.

Los datos de negocios contenidos en esta etapa en un DF serán la entrada para la recomendación por contenidos de artículos, mientras que los datos de usuarios generan un rating de contenidos para usuarios.

Por su parte los datos de reviews son utilizados para generar la matriz de ratings necesaria para la etapa de recomendación por factorización matricial SVD.

De cada uno de los DFs formados en la etapa de transformación se seleccionan únicamente las variables requeridas para la finalidad mencionada anteriormente.

Para los datos de negocios posterior a la transformación se generaron 100 variables, de las cuales 81 son descendientes de atributos, resultado del proceso de normalización de la estructura JSON, en la Tabla 13 del apéndice B se muestra la lista completa de atributos de negocios. La totalidad de atributos de cada negocio es la información de contenidos como tal, por lo tanto, todas estas características son seleccionadas junto con las categorías y el identificador del negocio para crear el DF de entrada para el SR por contenidos. Los datos de usuarios posterior a la transformación contienen 21 variables mostradas en la Tabla 14 del apéndice B, junto con las 8 variables de review. Para el caso de los datos de usuarios que tienen como finalidad generar la entrada para el SR por contenidos de usuarios, se seleccionaron las variables de *compliment* junto con *cool, funny y useful*, puesto que indican de forma indirecta características propias de cada usuario. En el caso de reviews se generaron dos DFs, el primero de ellos contiene únicamente los identificadores de negocios, usuarios y el valor de rating que son los datos necesarios para generar los ratings por factorización matricial, mientras que el segundo contiene sólo las descripciones textuales.

4.2.3. Filtrado por idioma y ubicación

Como se mencionó previamente, tenemos datos de distintos países e idiomas disponibles. Por la naturaleza del problema a resolver, la recomendación final debe estar restringida a lugares cercanos para un usuario en particular (Usuario activo). Utilizando la información disponible de latitud y longitud se utilizó un servicio de geo codificación inversa en Python [24], para generar la etiqueta del país en el que se encuentra cada uno de los negocios del dataset.

La función recibe los datos de latitud y longitud en forma de tuplas y genera un diccionario con los datos de ciudad, país y código de país correspondientes a la ubicación. En la Figura 10 se muestra el resultado de la función de codificación inversa para un elemento de los datos de negocios y la forma en que se integra a los datos disponibles, esto para lograr un filtrado por país de una forma más sencilla.

A partir de las etiquetas de país y ciudad se generaron la Tabla 6 y Tabla 7, que indican la cuenta de negocios en el dataset para países y ciudades respectivamente. Para esta solución en particular se trabajó únicamente con los datos de Estados Unidos, debido a que son mayoría en el dataset.

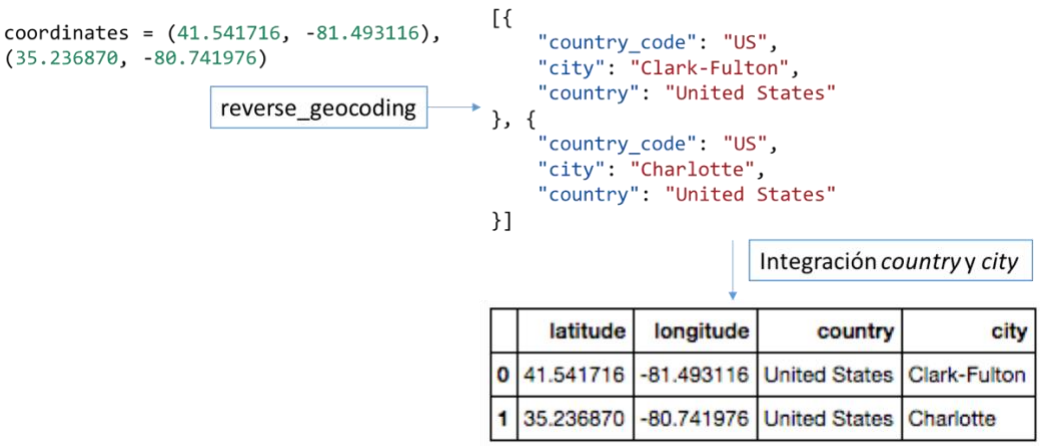

Figura 10. Proceso de codificación inversa para obtener la etiqueta de país y ciudad

Tabla 6. Cantidad de negocios por país

País	Número de negocios
United States	115546
Canada	33804
United Kingdom	4144
Germany	3098
Argentina	32
Norway	6
Austria	1
Belgium	1
Brazil	1
Chile	1
China	1
France	1
Italy	1
Mexico	1

Tabla 7. Ciudades con mayor cantidad de negocios

Ciudad	Número de negocios
Toronto	10414
Paradise	8186
Paradise Valley	7078
Spring Valley	6015
Phoenix	5891
Clark-Fulton	5464
Glendale	4161
Charlotte	4138
Las Vegas	4068
Montréal	3981

Gilbert	3689
Scottsdale	3380

Para el caso de los reviews en diferentes idiomas fue posible etiquetar el idioma correspondiente con un algoritmo que consiste en generar un conteo de *stop words* en distintos idiomas y marcar el que tenga mayor número de apariciones. En la Figura 11 se especifica el proceso realizado para generar esta clasificación. Una vez etiquetado el texto es sencillo filtrar todos los reviews en inglés, limitando de esta manera a lugares que pertenezcan a EUA y que estén valorados por personas que hablan inglés.

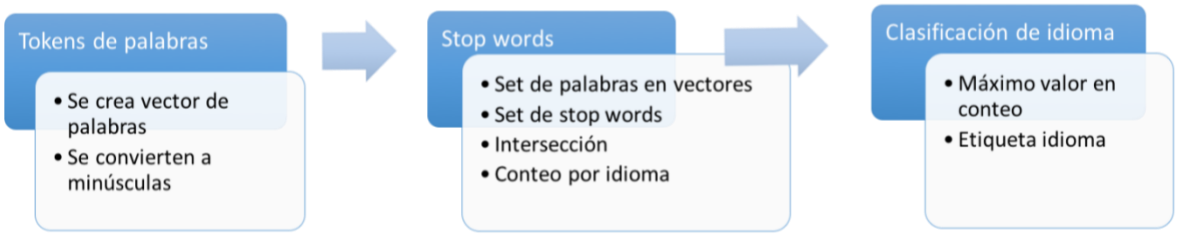

Figura 11. Proceso de clasificación de idoma en texto

4.3. Diseño del SR

La parte central de la propuesta consiste en utilizar los datos procesados de los negocios y usuarios para generar un rating artificial a partir del contenido y complementar la generación de ratings con factorización matricial SVD, atacando el problema de matrices dispersas y *cold start*.

4.3.1. Estructura

En la Figura 12 se muestra un diagrama del SR propuesto, donde el primer paso es el procesamiento de datos no estructurados de los cuales se genera la matriz de ratings y una matriz de contenidos para usuarios y para establecimientos.

A partir de las matrices de contenidos se generan ratings artificiales de contenido utilizando el algoritmo de nearest neighbour, mientras que a partir de la matriz de ratings dispersa se generan las predicciones de ratings por factorización matricial SVD. Con la finalidad de utilizar las ventajas de cada algoritmo de recomendación se genera una predicción de rating global con la suma ponderada de ambos enfoques.

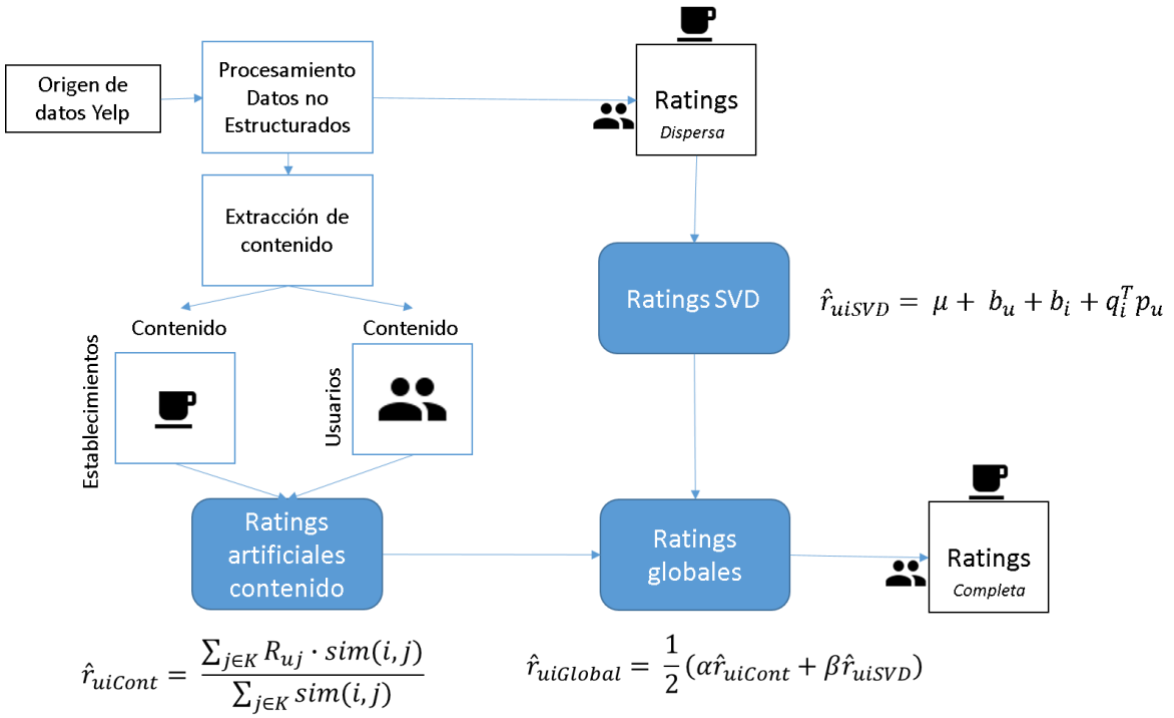

Figura 12. Diagrama del sistema de recomendación propuesto

4.3.2. Procesamiento de datos no estructurados

En la sección anterior se abordó a detalle el procesamiento de los datos no estructurados para cada una de las fuentes de datos. En el apéndice C, en la Figura 36, Figura 37 y Figura 38 se simplifica el proceso que se sigue para cada una de ellas. Los datos de json originales son transformados finalmente en DFs con las variables correspondientes a contenidos y ratings dependiendo del proceso, que funcionan como entradas para los algoritmos de recomendación.

En esta etapa se incluyen varios pasos de preparación y limpieza de datos, transformación, se agregan nuevas variables y se filtran a partir de restricciones de ubicación e idioma.

4.3.3. Matriz de ratings

Como se explicó en la sección de marco teórico la matriz de ratings es una representación de todos los valores observados entre usuarios y artículos, que tiene como principal característica su alto nivel de dispersión. En nuestro dataset donde las relaciones de la matriz de ratings corresponden a las valoraciones de cada usuario sobre un negocio, existe el grado muy alto de dispersión puesto que cada usuario sólo ha evaluado una pequeña cantidad del universo de negocios. La finalidad del SR es precisamente estimar los

valores de ratings de todos los negocios por usuario o una cantidad k de negocios con el mayor valor estimado.

En la Figura 13 se muestra el proceso realizado para crear la matriz de ratings con los datos del DF de ratings, que a partir de las columnas de business_id, user_id, y stars se ejecuta el método *pivot* de pandas [23] para generar un nuevo DF con los usuarios como filas y los negocios como columnas, donde cada elemento r_{ui} es el valor de rating, y los ratings conocidos se representan por los valores de estrellas mientas que los desconocidos como NaN. La representación resultante tiene un valor de $O(n^2)$ en espacio de memoria que lo hace ineficiente puesto que la mayoría de celdas no contienen valor.

Para asumir el problema de la complejidad espacial se ejecuta el método to_sparse() que permite modificar la forma en que se representa en memoria la matriz de ratings, almacenando únicamente los valores observados y sus índices en la matriz original. Este proceso es reversible a través el método to_dense() también de la librería pandas.

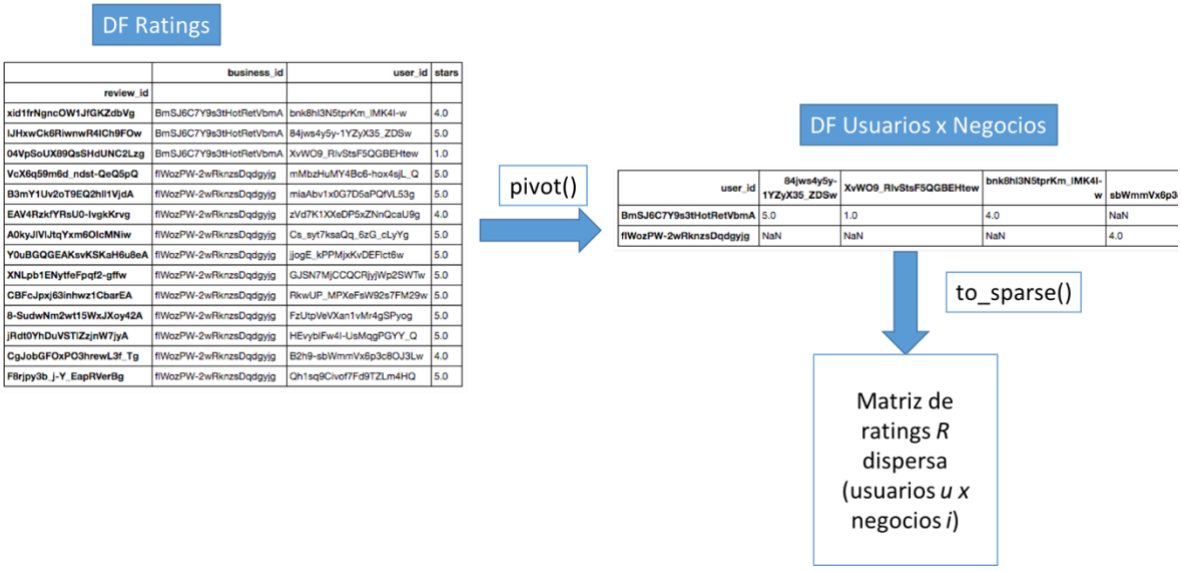

Figura 13. Creación de la matriz de ratings a partir del DF de ratings

Es posible describir que tan dispersa es una matriz a partir de su valor de densidad, que representa la relación del número de elementos observados entre la totalidad de los elementos de la matriz. En la Figura 14 se muestran matrices con diferentes valores de densidad donde los puntos negros son los valores conocidos y el espacio blanco los desconocidos. La matriz de ratings R generada tiene un valor de densidad de 0.0039 que es el valor de menor densidad de la figura.

La cantidad de reducción de espacio para una matriz dispersa depende de los elementos nulos que puedan omitirse y el algoritmo utilizado para generar la representación, en la Figura 15 se representan los disponibles en scipy.sparse [25]. En este caso la representación utilizada para almacenar la matriz de utilidad es el CSC porque permite realizar operaciones aritméticas y es eficiente para la naturaleza de los

datos, donde las columnas son menores a las filas. En la Tabla 15 del apéndice D, se describe a mayor detalle cada representación, con sus ventajas y desventajas.

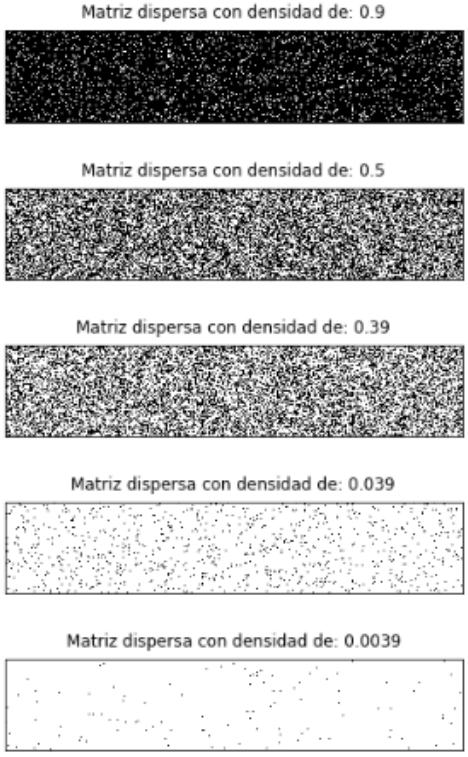

Figura 14. Diferentes valores de densidad en una matriz dispersa

En la Figura 16 se muestran los valores de compresión obtenidos en cada estructura de matrices dispersas de scipy para nuestra matriz de ratings, los valores de todas las compresiones se encuentran en la escala de MB mientras que la matriz densa está en la escala de 10x con fines de visualización, esto muestra claramente el gasto excesivo de uso de memoria cuando se trabaja con matrices completas.

Matriz densa

```
A = [  [0 1 2]
       [0 4 0]
       [2 0 1]]
```

Matriz CSR

```
A_csr =   (0, 1)  1
(0, 2)  2
(1, 1)  4
(2, 0)  2
(2, 2)  1
```

Matriz COO

```
A_coo =   (0, 1)  1
(0, 2)  2
(1, 1)  4
(2, 0)  2
(2, 2)  1
```

Matriz CSC

```
A_csc =   (2, 0)  2
(0, 1)  1
(1, 1)  4
(0, 2)  2
(2, 2)  1
```

Matriz BSR

```
A_bsr =   (0, 1)  1
(0, 2)  2
(1, 1)  4
(2, 0)  2
(2, 2)  1
```

Figura 15. Representaciones de matrices dispersas en scipy

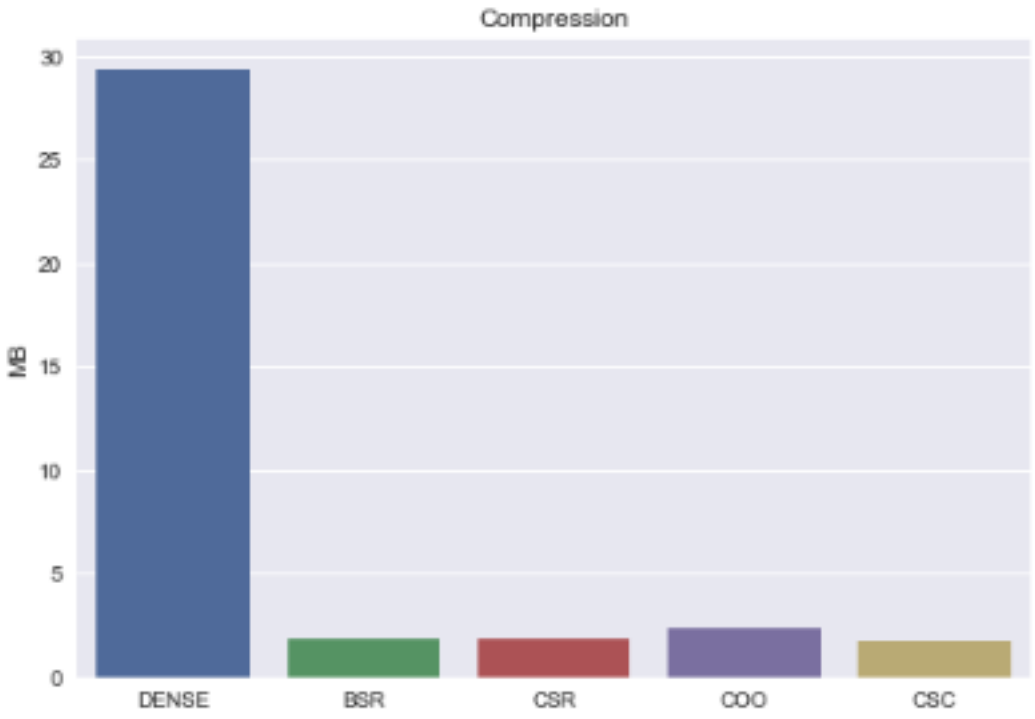

Figura 16. Valores de compresión de la matriz de ratings en los formatos de matrices dispersas.

4.3.4. Formulación de factores matriciales

Para la formulación de factores matriciales a partir de la matriz de ratings generada entre negocios y usuarios se utilizó un algoritmo de SVD. En la ecuación 3.8, se muestra la función a minimizar en el modelo, mientras que en la ecuación 3.7 la forma en que se genera el rating a partir de los factores entrenados. A estas ecuaciones se les agregan los factores conocidos como *biases*, cuya finalidad es amortiguar el efecto de los usuarios que tienden a otorgar mayores ratings o los artículos que reciben de manera general mejores evaluaciones. Entonces se produce una estimación que puede describir de manera más realista un rating. Para el caso de los usuarios el *bias* se representa por b_u, mientras que en los artículos como b_i (negocios), el valor de μ indica el promedio general de ratings[7]. Entonces la ecuación para predicción de rating resulta en la ecuación 4.1 mientras que la de minimización resulta como la mostrada en la ecuación 4.1, tomando en cuenta el valor de regularización para evitar el overfitting.

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u \tag{4.1}$$

$$\min_{q_i, p_u} \sum_{r_{(u,i} \in R_{train}} (r_{ui} - \hat{r}_{ui})^2 + \lambda\left(b_u{}^2 + b_i{}^2 + \|q_i\|^2 + \|p_u\|^2\right) \tag{4.2}$$

Estos factores son calculados a partir de un algoritmo de optimización como el *Stochastic Gradient Descent (SGD)* de la ecuación 4.3.

$$\begin{aligned} b_u &\leftarrow b_u + \gamma(e_{ui} - \lambda b_u) \\ b_i &\leftarrow b_i + \gamma(e_{ui} - \lambda b_i) \\ p_u &\leftarrow p_u + \gamma(e_{ui} \cdot q_i - \lambda p_u) \\ q_i &\leftarrow q_i + \gamma(e_{ui} \cdot p_u - \lambda q_i) \end{aligned} \tag{4.3}$$

Donde $e_{ui} = r_{ui} - \hat{r}_{ui}$. Estos pasos son ejecutados sobre todo el set de entrenamiento y repetidos durante n-épocas.

El entrenamiento descrito se produjo utilizando la librería surprise [26] en Python que permite ejecutar el algoritmo de SGD descrito en la ecuación 4.3 sobre un dataset de ratings.

Para el algoritmo se deben definir los parámetros:

- *n_factores*: Cantidad de factores k a calcular
- *n_epochs*: Cantidad de épocas en las que se ejecutará el entrenamiento
- *lr_all*: Tasa de aprendizaje (Es posible definir una tasa diferente por cada parámetro del aprendizaje).
- *reg_all*: El termino de regularización para todos los parámetros (Es posible definir un término diferente para cada parámetro).

La búsqueda de los valores de parámetros ideales se realizó por medio de fuerza bruta probando distintas combinaciones y valores de cada parámetro. Para esta búsqueda de parámetros se utilizó la funcionalidad de GridSearchCV de surprise que calcula la mejor combinación a partir de una métrica de precisión sobre

procedimientos de cross-validation. Los valores obtenidos de las ejecuciones se encuentran en la sección de resultados.

4.3.5. Generación ratings por SVD

La generación de ratings con el algoritmo de SVD se realiza como se describió en el punto anterior utilizando la librería de surprise sobre el DF de ratings de la Figura 13. Los pasos de este proceso se representan en la Figura 17, los cuales se describen a detalle en los siguientes puntos:

- El objeto Reader; tiene como finalidad leer y analizar un formato de datos de entrada que contengan ratings, asume un rating por línea con el orden usuario| artículo | rating.
- La carga de datos; se genera con el método Dataset.load_from_df que recibe el DF de ratings y el Reader definido previamente.
- Separación entrenamiento/pruebas; el set de datos de entrada se separa con un iterador de cross-validation que a partir del valor de *n-splits* divide el total de filas en *n* partes similares y se itera sobre todas las divisiones tomando una parte como pruebas y el resto como entrenamiento.
- Instancia SVD; se declara el modelo que se utilizará para el entrenamiento.
- Entrenamiento; A partir de la instancia del modelo se realiza el entrenamiento con el método SVD.fit(), que internamente calcula los vectores de la ecuación 4.3.
- Pruebas; Los vectores de la ecuación 4.3 se encuentran en memoria cache, para poder ejecutar las predicciones a partir de la ecuación 4.1. La predicción se ejecuta con el método SVD.test().
- Precisión; Los ratings obtenidos se comparan con los ratings originales sobre el set de pruebas y a partir de una métrica que puede ser RMSE de la ecuación 3.12 o MAE de la ecuación 3.14 se puede conocer la precisión del algoritmo en cada iteración del cross-validation.

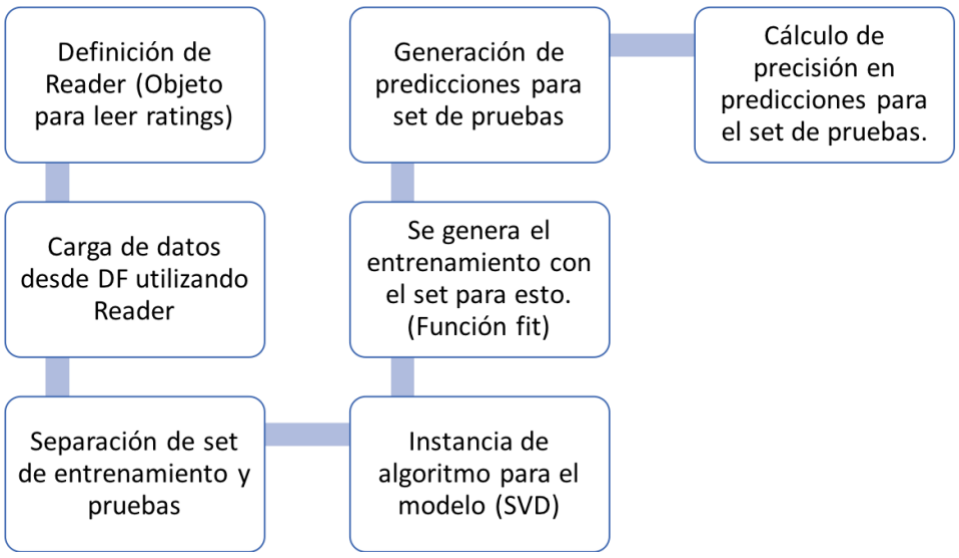

Figura 17. Proceso para generar ratings con algoritmo SVD

4.3.1. Generación ratings basados en contenidos

La sección del SR por contenidos se desarrolló a partir del proceso mostrado en la Figura 18, donde se toma inicialmente el DF de contenidos de cada negocio, este DF de contenidos es filtrado para que los negocios incluidos coincidan con los que forman parte del DF de ratings utilizado para el SR de FC.

Uno de los pasos más importantes del proceso es la normalización y codificación de variables que definen el contenido de cada artículo ya que a partir de esta información se infieren los perfiles de usuario. El modelo más utilizado para proporcionar valores numéricos a los vectores de palabras es el TF-IDF mencionado en el marco teórico, este modelo se prefiere cuando el contenido se obtiene a partir de descripciones textuales de cada artículo, puesto que asigna un valor numérico a cada palabra (término) a partir de su frecuencia en un documento y su relación con el vocabulario completo (frecuencia entre todos los documentos) representado por la ecuación 3.9. Cuando las descripciones están formadas a partir de términos clave (keywords), la frecuencia de un término en un documento carece de relevancia y por lo tanto el valor TF-IDF. En este caso el contenido de cada artículo implica únicamente la presencia o ausencia de el término clave.

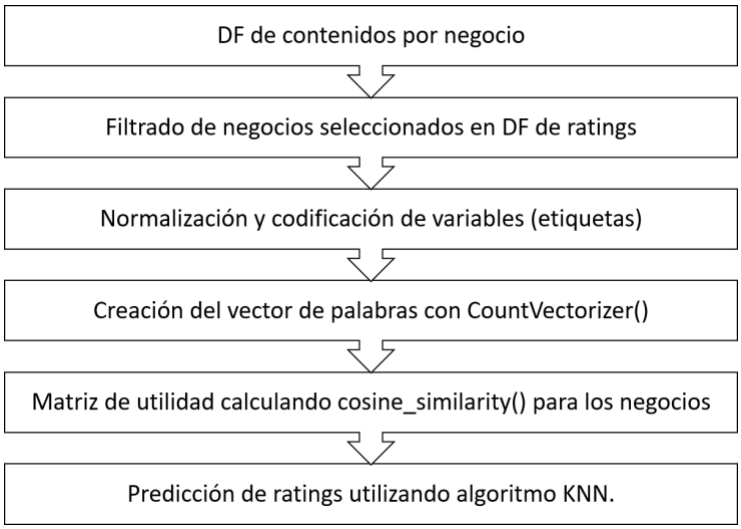

Figura 18. Proceso de generación de rating por contenidos

Para crear los vectores de términos clave, tomamos las variables enlistadas en la Tabla 13 del DF de contenidos de negocios. En su mayoría las variables son de tipo binario, pero algunas de ellas son de otros tipos, como objetos o numéricos. Con la finalidad de normalizar todas las variables de contenido y crear la matriz de utilidad de *m* artículos por *n* términos, se ejecutó el proceso siguiente:

- Se exploraron los posibles valores de cada variable no binaria, en la Tabla 8 se muestran algunas de ellas. Los posibles valores son descriptores del atributo en una cadena de texto o números que indican una escala.
- Para cada variable se seleccionó la mejor forma de obtener la palabra clave que incluyera el nombre del atributo y el valor. Por ejemplo, para el atributo *NoiseLevel* con posibles valores de

False, average, loud, quiet, very_loud, se concatenó el valor del atributo con la categoría para el artículo actual, resultando en palabras clave *NoiseLevel_average, NoiseLevel_loud* y de esa manera para el resto de las categorías del atributo.

Tabla 8. Variables con valores no binarios que definen el contenido de los negocios

attributes.RestaurantsAttire	N. business
False	1938
casual	1090
dressy	99
formal	1

attributes.NoiseLevel	N. business
False	1838
average	862
loud	144
quiet	201
very_loud	83

attributes.RestaurantsPriceRange2	N. business
0.0	692
1.0	655
2.0	1348
3.0	310
4.0	123

attributes.Alcohol	N. business
False	1716
beer_and_wine	195
full_bar	769
none	448

attributes.BYOBCorkage	N. business
False	2993
no	53
yes_corkage	32
yes_free	50

attributes.AgesAllowed	N. business
18plus	2
21plus	41
False	3084
allages	1

- Los atributos cuyo valor es False se consideran no definidos para ese artículo por lo que no se incluye en el proceso.

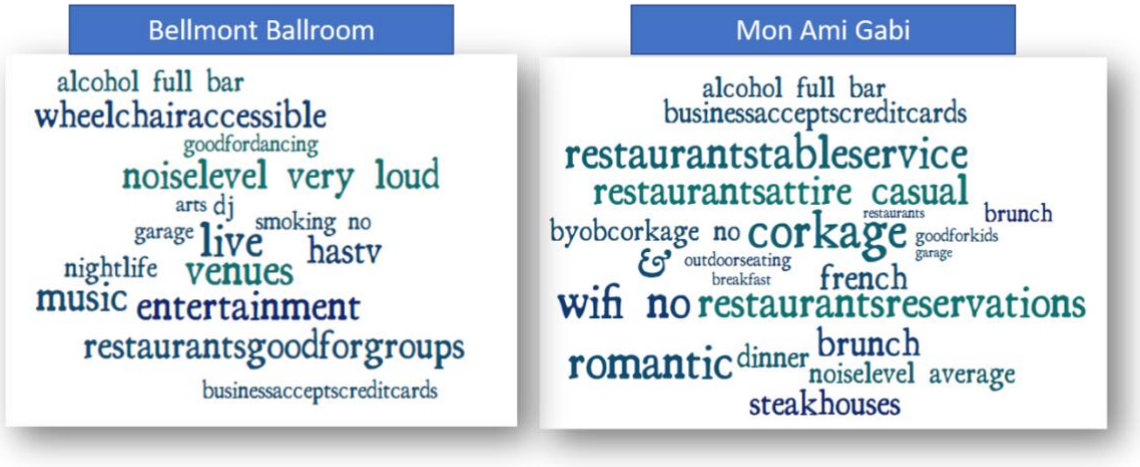

Figura 19. Términos clave del modelo BoW para dos negocios

- Cada palabra clave generada se asignó a una lista de tipo BoW. En la Figura 19, se ejemplifican los términos obtenidos en esta etapa para dos distintos negocios del set de datos.
- A partir de los términos del modelo BoW se generó la matriz de utilidad aplicando el método *CountVectorizer* de *sklearn.*

Con la matriz de utilidad calculada fue posible estimar el valor de similitud de la ecuación 3.10 entre todos los artículos con la función de *cosine_similarity* de sklearn, resultando una matriz cuadrada de *nxn* artículos donde s_{ij} representa la similitud entre el artículo *i* y *j*.

En la Figura 20 se muestra el valor de similitud coseno para dos artículos del set de datos y sus *k=10* vecinos más cercanos. El primer elemento con valor de 1 es el artículo propio, por lo que no debe tomarse en cuenta para la generación de recomendaciones. El elemento de la parte izquierda de la Figura 20, es un lugar de música, baile y entretenimiento en las vegas y los lugares con mayor similitud son a su vez, lugares para bailar, espectáculos en vivo o musicales por lo que se confirma que las palabras clave seleccionadas ofrecen la información suficiente para identificar artículos similares. De la misma forma el artículo del lado derecho es un restaurante francés y sus vecinos más cercanos incluyen lugares para cenar, comida a la carta y *buffet*, o lugares para desayunar.

similarity	name
1.000000	Belmont Ballroom
0.788241	The Joint at Hard Rock
0.782624	Boyz II Men
0.774597	Elton John - The Million Dollar Piano
0.753778	Mondays Dark With Mark Shunock
0.750000	Magic Mike Live
0.750000	Wine Amplified
0.729800	Human Nature
0.729800	Carnaval Court Bar & Grill
0.729800	Studio 54

similarity	name
1.000000	Mon Ami Gabi
0.796098	Wicked Spoon
0.742611	The Buffet
0.730297	Sterling Brunch
0.723568	Bacchanal Buffet
0.712697	Country Club
0.712697	Tableau
0.702439	Flavors The Buffet
0.702439	Spice Market Buffet
0.696311	Oyster Bay Seafood

Figura 20. Similitud coseno entre dos artículos y sus k=10 vecinos más cercanos.

Las predicciones de ratings son generadas con la matriz de ratings seleccionada para entrenamiento de la siguiente forma:

- Para el usuario activo *u*, se cuenta con un set de artículos de entrenamiento D_u y un set de pruebas D_L.
- Para estimar el rating $\hat{r}_{ui}$ se toman los k vecinos más cercanos a i y se estima con el promedio de la multiplicación de ratings del set D_u por su valor de similitud s_{ij}, de forma similar a la ecuación 3.6.

4.3.2. Formulación del rating compuesto

El SR compuesto se conforma de la combinación lineal de la salida del SR BC y el SR FC desarrollados previamente. Las salidas de ambos se combinan a partir de un set de pesos para proporcionar la predicción. La ecuación 4.4 muestra el cálculo de la matriz de ratings a partir del SR compuesto.

$$\hat{R} = \alpha \hat{R}_{Contenido} + \beta \hat{R}_{SVD} \tag{4.4}$$

Donde $\hat{R}$ representa la matriz de ratings completamente especificada, $\hat{R}_{Contenido}$ la matriz de ratings completamente especificada con el SR BC y $\hat{R}_{SVD}$ la calculada con el SR FC. Los ratings no observados de la matriz de ratings original son calculados con cada respectivo modelo.

De forma ideal los valores de α y β tienen el mismo valor, lo que implicaría que el peso de ambos SR en el rating final es el mismo. Debido a que no se puede asumir que cada sistema puede aportar de forma equivalente la proporción para minimizar el RMSE o el MAE de las ecuaciones 3.12 y 3.14 respectivamente, estos valores deben ser calculados.

Para calcular los valores de α y β se sigue la misma metodología de validación que los sistemas individuales de uso de los datos, donde se separa un set para entrenamiento y uno para pruebas. A partir del set de entrenamiento se calculan las salidas de cada SR que se convierte en variables independientes y se maneja a partir de ahí como un problema de regresión, que al buscar minimizar el error se encuentran los valores más adecuados de los pesos. Con los valores de pesos calculados se evalúa el error en las predicciones para el set de entrenamiento.

4.4. Metodología de validación

La efectividad del SR propuesto se evaluará a partir de diferentes métricas, con la finalidad de comprobar en distintos escenarios su precisión, escalabilidad y fracción de pares concordantes en las recomendaciones. Los datos utilizados para evaluar la precisión de manera individual y en composición del algoritmo de FC y el BC pertenecen a un mismo *subset* de pruebas que se obtiene como se muestra en la Figura 21, separando una parte de la totalidad del set de datos original.

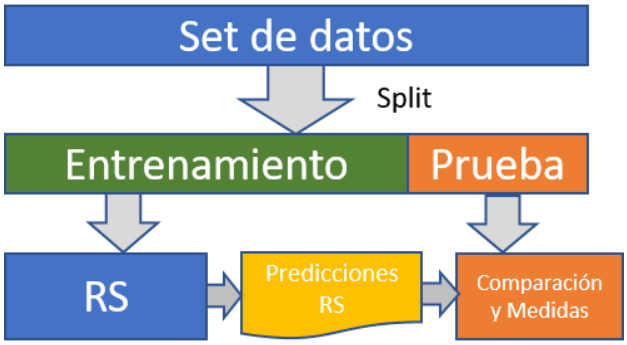

Figura 21. Separación de los datos para evaluar precisión.

A pesar de que esta separación de datos se realiza de forma aleatoria, existe la posibilidad de que se tomen ratings con mayor o menor facilidad para su predicción por lo que el proceso de separación, entrenamiento, predicción y medidas se ejecuta a través de una validación cruzada como la que se muestra en la Figura 22.

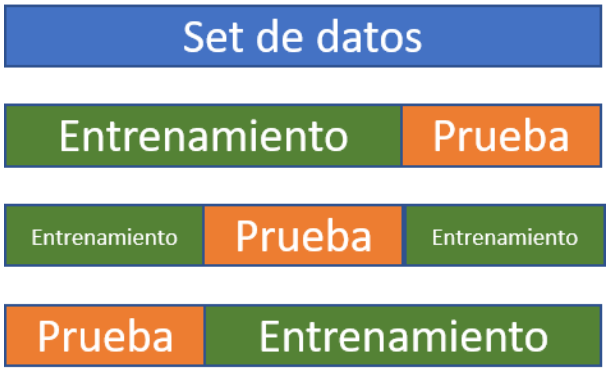

Figura 22. Proceso de cross-validation para k =3

El valor de precisión se calcula sobre el set de pruebas comparando los valores obtenidos de las predicciones con los valores de rating reales. Las métricas por obtener son el valor de RMSE y MAE de la ecuación 3.12 y 3.14 respectivamente.

La escalabilidad se evalúa a través de las métricas de tiempo y uso de memoria. Para evaluar la complejidad temporal se registra el tiempo de entrenamiento para cada algoritmo (FC y BC) y la suma de los entrenamientos de ambos para el algoritmo compuesto, se evalúa de la misma forma el tiempo de predicción. Para el uso de memoria, se registra la aplicación máxima de memoria para entrenamiento y predicciones para cada línea del SR. En la Figura 23 se representa el experimento completo para el cálculo

de precisión y escalabilidad del SR, así como las métricas que se capturan en la etapa correspondiente del proceso.

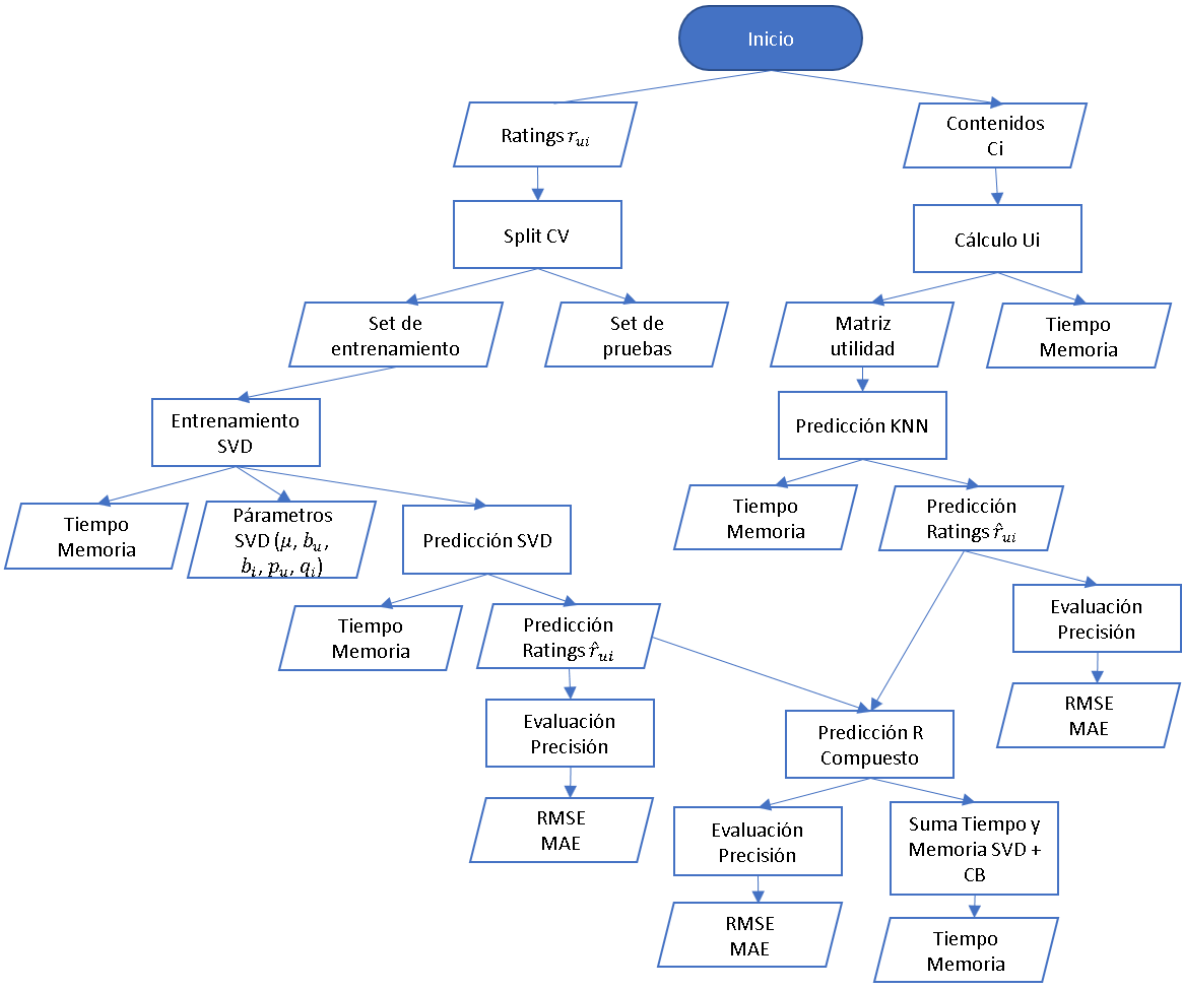

Figura 23. Metodología de validación de precisión y escalabilidad del SR propuesto

El valor de fracción de pares concordantes que indica si las predicciones fueron realizadas correctamente de acuerdo a un ranking de valores.

Una vez definiendo la forma en que se trabajarán los valores de precisión, escalabilidad y FPC para el sistema completo se enlistan los experimentos a ejecutar:

1. Evaluación de escalabilidad en tiempo y memoria para el SR BC con el uso de matrices dispersas en Python contra matrices densas.
2. Búsqueda de parámetros con mejor valor de precisión utilizando GridSearchCV de surprise para el algoritmo de FC.
3. Evaluación de escalabilidad temporal para el SR compuesto para distintos tamaños del set de datos de entrada.
4. Evaluación de precisión para el SR compuesto para distintos tamaños de entrada del set de datos.

5. Evaluación de precisión para cada algoritmo de recomendación y el sistema compuesto para distintos valores de densidad en la matriz de ratings.
6. Búsqueda de los valores α y β, del SR compuesto que ofrezcan la proporción más adecuada para combinar el algoritmo de FC y el BC.
7. Evaluación de FPC para el FC, BC y compuesto en el set de datos para distintos valores de densidad.

5. RESULTADOS Y DISCUSIÓN

Resumen: *En este capítulo se presentan los resultados obtenidos del desarrollo de este trabajo y una discusión sobre el impacto que tiene el SR compuesto sobre los problemas de cold-start y matrices dispersas.*

5.1. Resultados

Los resultados obtenidos a partir del desarrollo del trabajo responden a la metodología de validación descrita en la sección anterior, donde se evaluaron de manera detallada y comparativa las métricas de escalabilidad en tiempo y memoria, precisión y FPC, para los algoritmos individuales y su integración en un SR compuesto.

Las pruebas de escalabilidad fueron ejecutadas en JupyterHub: 0.7.2 con la versión de Python 3.5.2 en un ambiente Linux Ubuntu Server 16.04.4 LTS cuyo hardware cuenta con un procesador Xeon E5 2643-v3 a 3.4GHz y 32 núcleos y 4 GB de Ram DDR4.

La matriz de ratings filtrada utilizada para esta etapa tiene dimensiones de 12 016 usuarios por 3 128 negocios con un total de 148 388 ratings observados de 37 586 048 ratings totales, lo que indica una densidad de 0.0039.

La matriz de palabras clave utilizada en el SR BC cuenta con dimensiones de 3 128 negocios por 849 términos clave, donde cada elemento indica la cantidad de veces que aparece el término clave en cada negocio. De esta matriz contienen valor 42 204 celdas de 2 655 672 posibles, lo que indica una densidad de 0.0159.

5.1.1. Escalabilidad en tiempo y memoria en el SR BC utilizando matrices dispersas.

En esta prueba se ejecutó un proceso de entrenamiento y predicción para comprobar las mejoras en tiempo de entrenamiento y uso de memoria al aplicar la estructura de datos de matriz dispersa de scipy[25] para representar las matrices de ratings y palabras clave, necesarias para estimar los ratings del SR BC.

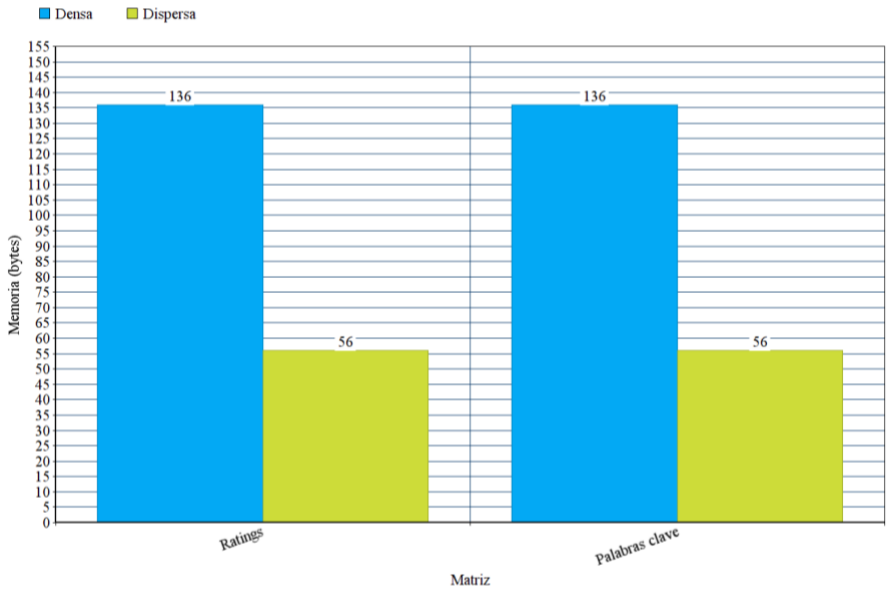

Figura 24. Uso de memoria por matrices densas y dispersas utilizadas para el SR BC

El uso de memoria para cada matriz en forma densa y dispersa resultante se representa en la Figura 24, donde se observa una reducción de un 58% de uso de memoria para el caso de las matrices dispersas contra las matrices densas, de igual manera para ratings y palabras clave.

En el SR BC el entrenamiento corresponde a la etapa fuera de línea del sistema que implica el cálculo de la matriz de similitud entre artículos a partir de los valores que representan las palabras clave. Esta matriz de similitud calculada con la función cosine_similarity de sklearn.pairwise [27] también presentó una mejora en tiempo con el uso de matrices dispersas sobre matrices densas, esto se puede observar en la Figura 25, con la etiqueta de tiempo de entrenamiento.

A su vez en el tiempo de predicción no se observa una diferencia considerable puesto que para generar los ratings se utiliza la ecuación 3.4 donde los ratings se centran a partir de la media y por lo tanto el vector de ratings correspondiente a un usuario pierde su representación dispersa previo a la predicción, a pesar de esto se comprueba que las predicciones son ejecutadas en poco tiempo.

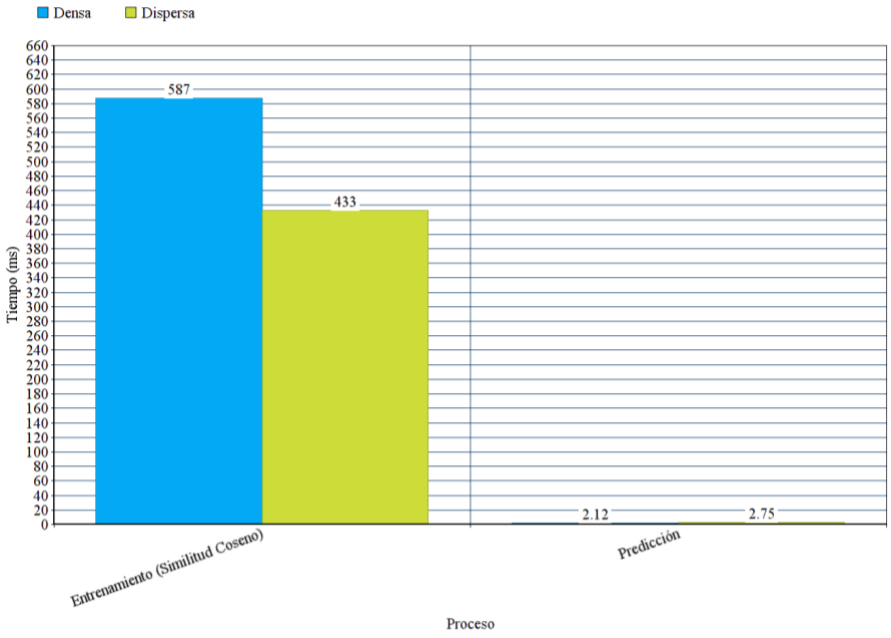

Figura 25. Tiempo de entrenamiento y predicción para matrices densas y dispersas para el SR BC

5.1.2. Búsqueda de parámetros con menor RMSE y MAE para el SR SVD

Para encontrar la combinación de valores de la ecuación 4.3, se ejecutó la combinación de parámetros mostrados en la Tabla 9 utilizando la función GridSearchCV de surprise que permite realizar el entrenamiento con todas las posibles combinaciones y seleccionar aquella en la que el error sea menor.

De las cuatro variables cada una cuenta con 4 posibles valores por lo que se ejecutaron 256 combinaciones de parámetros y se evaluaron sus valores de RMSE y MAE contra el set de entrenamiento. Cada

combinación se realizó con cross-validation a 3 segmentos, por lo que los valores de las métricas obtenidas corresponden a la media de todos los segmentos.

Tabla 9. Valores de parámetros utilizados para GridSearchCV en SR SVD

n_epochs	lr_all (γ)	reg_all (λ)	n_factors
5	0.002	0.02	12
20	0.005	0.1	40
50	0.01	0.4	60
100	0.1	0.6	100

Los mejores estimadores fueron con los parámetros lr_all: 0.002, n_epochs: 100, n_factors: 12, reg_all: 0.1 para la medida MAE, mientras que para la medida de RMSE fueron lr_all: 0.002, n_epochs: 50, n_factors: 12 y reg_all: 0.1. Para ambas métricas sólo difiere la cantidad de épocas utilizadas para llegar al valor más bajo. En la Tabla 10 y Tabla 11, se presentan las mejores cinco combinaciones de parámetros para ambas métricas.

Tabla 10. Valores MAE para las mejores combinaciones de parámetros

	mean_test_mae	param_lr_all	param_n_epochs	param_n_factors	param_reg_all	rank_test_mae
88	0.831241	0.010	20	12	0.10	2
128	0.832432	0.002	50	12	0.02	5
148	0.831461	0.005	50	12	0.10	3
208	0.830390	0.002	100	12	0.10	1
209	0.832333	0.002	100	40	0.10	4

Tabla 11. Valores RMSE para las mejores combinaciones de parámetros

	mean_test_rmse	param_lr_all	param_n_epochs	param_n_factors	param_reg_all	rank_test_rmse
84	1.055836	0.005	20	12	0.10	2
85	1.056816	0.005	20	40	0.10	5
128	1.056811	0.002	50	12	0.02	4
144	1.055822	0.002	50	12	0.10	1
145	1.056794	0.002	50	40	0.10	3

5.1.3. Escalabilidad temporal para entrenamiento y predicciones del SR compuesto para distintos tamaños del set de datos de entrada

La escalabilidad temporal del SR compuesto depende directamente de los algoritmos BC y FC implementados, así como del posterior entrenamiento para encontrar los valores de α y β que minimicen el error.

En este experimento realizamos un análisis a posteriori de la complejidad temporal para entrenamiento del SR compuesto modificando el tamaño de entrada del set de datos iniciando desde un número reducido de usuarios y negocios, hasta el valor máximo disponible de la matriz de ratings.

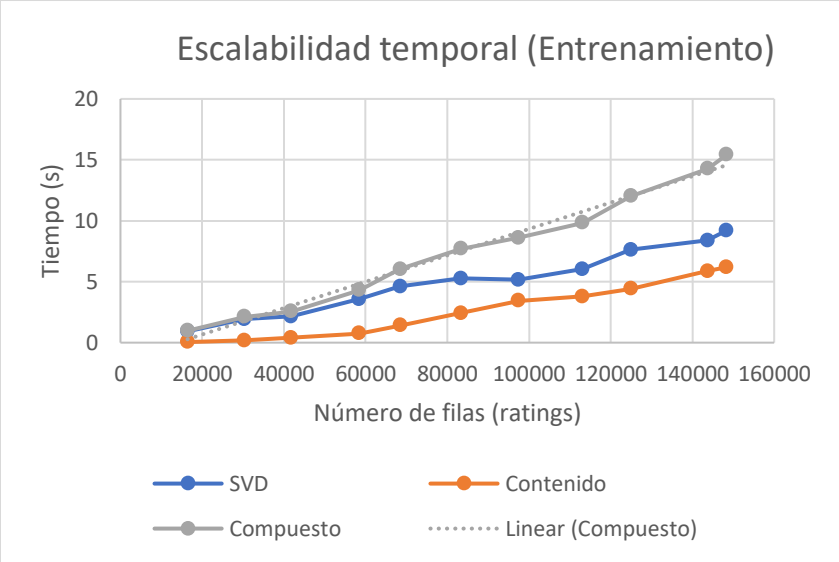

Figura 26. Escalabilidad temporal (Entrenamiento) para distintos tamaños de set de datos.

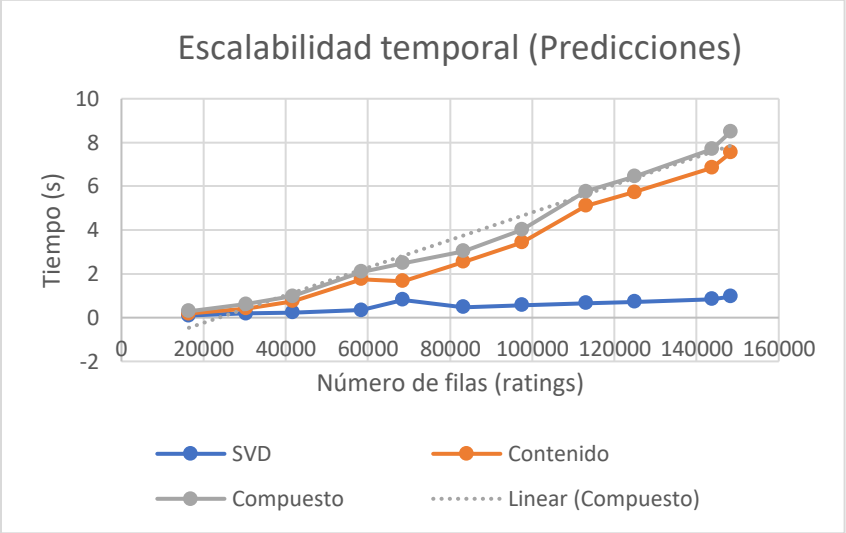

Figura 27. Escalabilidad temporal (predicciones) para distintos tamaños del set de datos.

5.1.4. Pesos α y β del SR compuesto con la proporción más adecuada para combinar linealmente el SR FC y el SR BC.

La combinación de pesos para integrar los SR BC y FC en un SR compuesto debe ser calculada, ya que la combinación ideal donde α y β tienen valor de 0.5 no puede ser asumida. Para este experimento se tomaron los ratings estimados con cada entrenamiento de los algoritmos individuales como variables independientes y el rating real como la variable dependiente. Con estos datos se calculó un modelo de regresión para minimizar el error, para esto se utilizó la librería de scipy que cuenta con distintos algoritmos para el cálculo de la regresión.

En la Tabla 12 se muestran los pesos calculados y los valores de RMSE y MAE para cada tamaño del set de datos de entrenamiento a partir de un modelo de regresión.

Tabla 12. Valores de pesos calculados con regresión para el SR Compuesto

Tamaño del set de datos	Valor de α	Valor de β	RMSE	MAE
15766	1.83	-0.05	0.4	0.49
29560	1.57	-0.01	0.41	0.5
41752	1.59	-0.05	0.42	0.5
58408	1.75	-0.19	0.42	0.51
68452	1.76	-0.25	0.41	0.5
83280	1.91	-0.41	0.42	0.5
97449	2.05	-0.56	0.41	0.5
113055	2.09	-0.64	0.4	0.49
124919	2.13	-0.69	0.4	0.49
143700	2.20	-0.79	0.38	0.48
148360	2.23	-0.82	0.38	0.48

5.1.5. Precisión obtenida con el SR compuesto para distintos tamaños del set de datos de entrada.

La precisión de un SR puede verse afectada por el tamaño del set de datos de entrada, en este experimento se mide la forma en que el SR compuesto desarrollado es afectado a partir de la variación del tamaño de la matriz de ratings para usuarios y negocios.

Se comparan los resultados del SR compuesto con los algoritmos individuales BC y SVD.

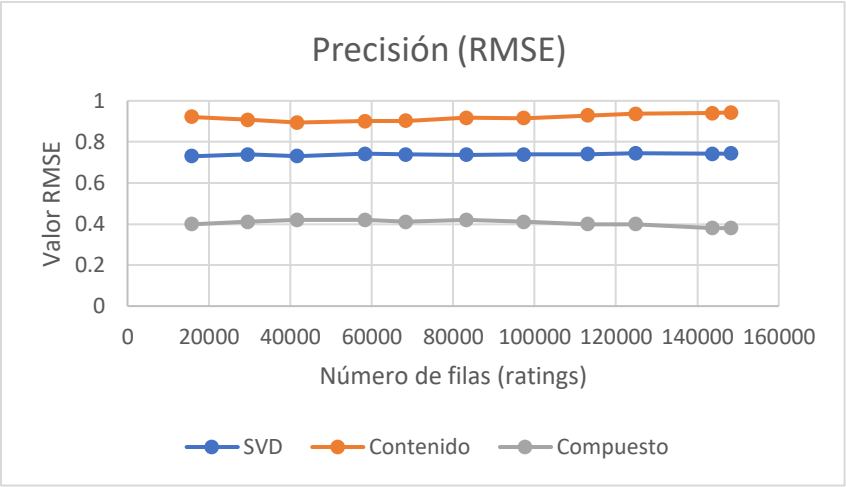

Figura 28. Valor RMSE para distintos tamaños del set de datos de entrada.

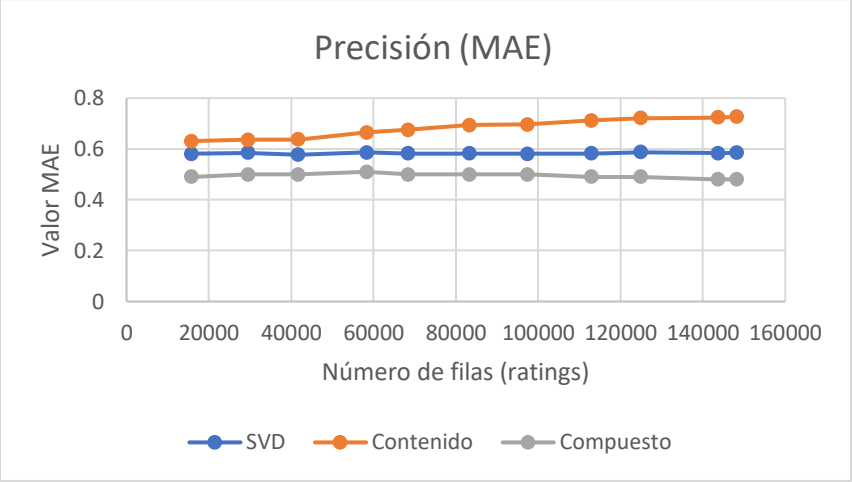

Figura 29. Valor MAE para distintos tamaños del set de datos de entrada.

5.1.6. Precisión lograda con algoritmos individuales y algoritmo compuesto para distintos valores de densidad en el set de datos de entrada.

Las matrices de ratings muy dispersas pueden no ser lo suficientemente informativas para extraer de manera adecuada los factores en algoritmos como el SVD y puede ejecutar predicciones con un error muy alto, por su parte los SR BC pueden ofrecer recomendaciones aún con pocos ratings.

En este experimento se demostró la robustez del SR compuesto a los cambios de densidad en la matriz de ratings de entrada.

En la Figura 30 y la Figura 31 se comparan las métricas RMSE y MAE respectivamente, para ocho niveles de densidad para los algoritmos individuales y el algoritmo de SR compuesto.

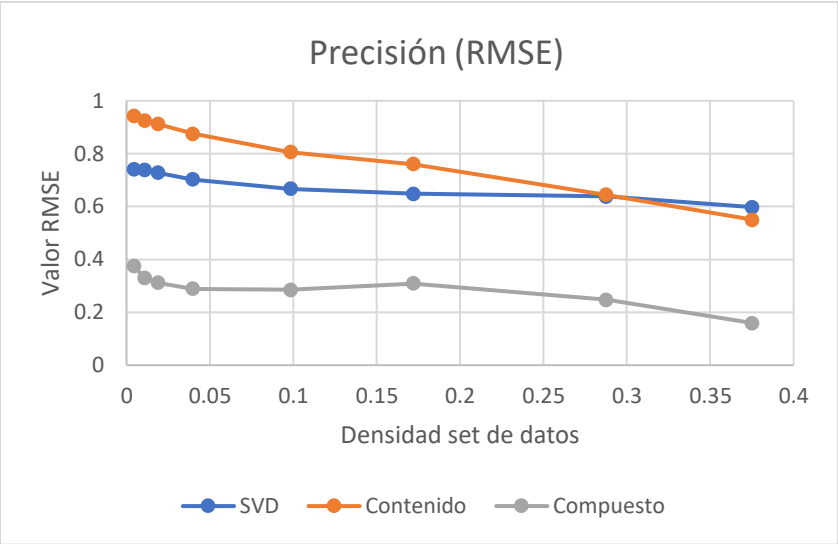

Figura 30. RMSE para distintos valores de densidad del set de datos

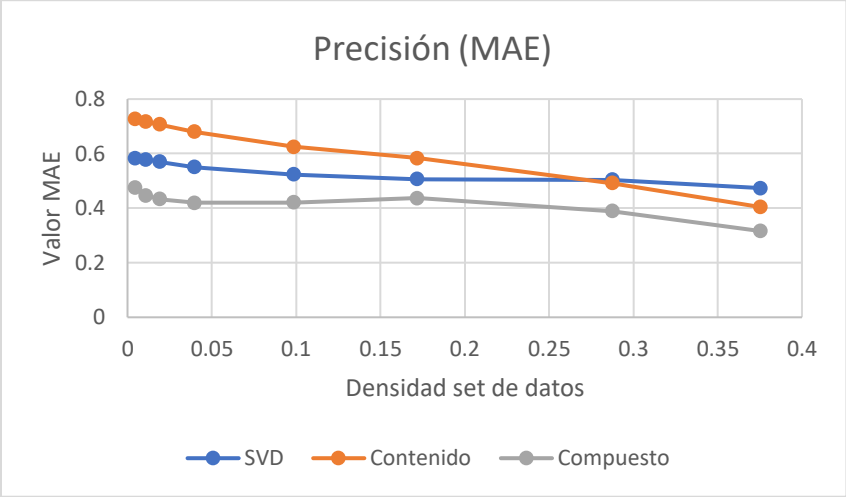

Figura 31. MAE para distintos valores de densidad del set de datos

5.1.7. Medida de FPC para los algoritmos individuales y el algoritmo compuesto, para distintos valores de densidad.

En muchas ocasiones las recomendaciones finales se presentan en forma de topk elementos y no como un valor numérico de predicción. En este experimento se calculó el valor FPC para evaluar la efectividad del SR propuesto para predicciones ordinales.

En la Figura 32 se presentan los resultados para cada algoritmo SVD, BC, y Compuesto en cuanto a FPC. Un valor mayor de FPC implica una proporción mayor de pares concordantes, es decir mejores predicciones.

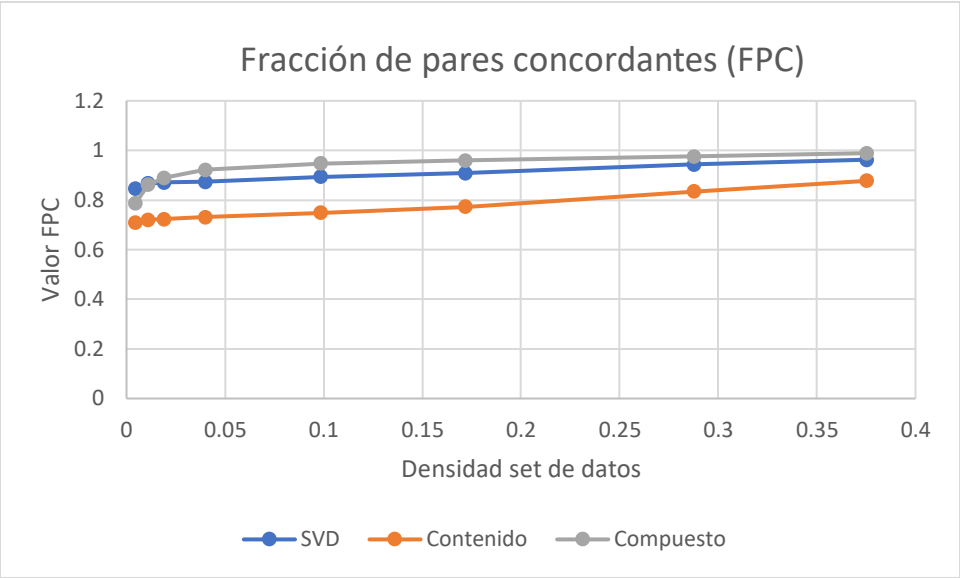

Figura 32. FPC para distintos valores de densidad

5.2. Discusión

A partir de los resultados generados se obtuvo la información de los siguientes puntos:

- Se pudo comprobar que la representación de datos como matrices dispersas de Python redujeron el uso de memoria en un 58% y un 26% en el tiempo de entrenamiento en lo que corresponde al SR BC. Para el tiempo de predicción, es decir, la etapa en línea del sistema no existe una mejoría significativa entre un modelo de datos dispersos y uno de datos densos.

- La ejecución de la selección de parámetros con GridSearchCV de la biblioteca de surprise, permitió obtener la mejor combinación de factores, dicho de otra forma, la que presenta un menor RMSE y MAE. En la Tabla 10 y Tabla 11 se mostró el orden entre las mejores combinaciones, para el número de factores, número de épocas, valor de regularización y factor de aprendizaje. Los parámetros de cada tabla no coinciden en el mejor ranking para ambos, pero de estos se seleccionan los valores correspondientes a la posición 3 de MAE y 1 de RMSE puesto que al ser la combinación con menor error y menor cantidad de épocas requeridas para su cálculo permite que sea más eficiente el entrenamiento.

- Se evalúo la escalabilidad temporal en el entrenamiento y las predicciones para los algoritmos individuales SVD, BC y Compuesto. Los tres algoritmos tienen una complejidad temporal lineal, lo que los hace igualmente escalables. Como era de esperarse para el entrenamiento el algoritmo compuesto tiene una pendiente positiva más pronunciada ya que abarca el entrenamiento de los dos algoritmos SVD y BC, junto con el cálculo de los coeficientes a partir de una regresión, este último paso incrementa de forma poco significativa el tiempo de entrenamiento total. También se muestra en la Figura 27 que el algoritmo SVD es el más eficiente en cuanto a predicciones,

puesto que, una vez calculados los factores, la predicción está denotada por una suma y un producto vectorial mostrado en la ecuación 4.1. La complejidad temporal de las predicciones para el SR compuesto se ve afectada principalmente por el SR BC ya que requiere la selección de los k vecinos para generar la predicción, entonces debe recorrer y ordenar los valores del vector de usuarios o artículos.

- Para la combinación de los algoritmos de SVD y BC se generó una regresión a partir de las estimaciones de ratings parciales. Se calcularon los valores de pesos para cada algoritmo, utilizando distintos tamaños del set de datos de entrenamiento y validación se obtuvieron los valores de la Tabla 12. En todos los casos el coeficiente de α correspondiente al algoritmo SVD tiene un mayor peso y β correspondiente al algoritmo BC se presenta como un valor de ajuste para la predicción final. Los valores de RMSE y MAE calculados para el SR compuesto sin duda alguna superan el mejor escenario de los obtenidos únicamente con un algoritmo, tomando como referencia los datos de la Tabla 10 y Tabla 11.

- A partir del experimento de precisión contra tamaño del set de datos de entrada para el algoritmo compuesto y los algoritmos individuales, se muestra que el algoritmo compuesto presenta valores más bajos en ambas medidas en todos los escenarios, lo que indica un mejor rendimiento. También se comprobó que el tamaño del set de datos de entrada no afecta drásticamente la precisión de los SR aquí utilizados para el caso de RMSE y MAE.

- A diferencia del experimento donde se varía el tamaño del set de datos de entrada y no representa un mayor cambio en la precisión para los SR presentados, el caso de la densidad es muy distinto. En la Figura 30 y Figura 31 se observa la clara relación entre densidad y error. Una menor densidad implica un error mayor y viceversa. También en este experimento el SR compuesto fue el que presentó un error más bajo para todos los niveles de densidad. El buen rendimiento aún para niveles bajos de densidad indica que el SR compuesto es favorable para corregir el problema del cold-start para artículos, ya que a partir del SR BC contenido ajusta las predicciones del algoritmo SVD.

- En lo que se refiere al experimento para calcular la FPC se utilizó de la misma forma distintos valores de densidad en los datos de entrenamiento y se calcularon los valores para los tres algoritmos. Aquí un valor mayor de FPC indica una mejor predicción, en la Figura 32 se muestra que esta métrica también se modifica en función de la densidad al igual que los valores RMSE y MAE. En este caso una mayor densidad es proporcional a un valor mayor de FPC, aunque esta métrica también pone con un valor mayor al SR compuesto, la diferencia no es tan significativa con el algoritmo SVD individual ni tan clara como con las otras métricas.

6. CONCLUSIONES

Resumen: *En este capítulo se presentan las conclusiones respecto a los resultados obtenidos y el trabajo futuro en relación con el SR compuesto usando algoritmos de aprendizaje máquina.*

6.1. Conclusiones

El objetivo principal 1.4.1 de este trabajo era desarrollar un sistema de recomendación que permitiera generar sugerencias de un artículo para un usuario o grupo de usuarios, abordando principalmente el reto de eficiencia temporal y espacial para el manejo de los datos y el problema del cold-start al que muchos SR tradicionales son susceptibles.

En este sentido la aportación principal de este trabajo es el uso conjunto de dos enfoques para la recomendación, por una parte, las predicciones que se realizan a través del enfoque de Factorización Matricial SVD que utiliza los ratings observados de usuarios respecto a artículos para entrenar un modelo con SGD que minimizando el error calcula las variables latentes para generar la predicción de los ratings no observados. Por otro lado, se encuentra el enfoque BC que utiliza descriptores obtenidos de la minería de datos no estructurados en forma de un espacio vectorial de palabras clave para estimar una métrica de similitud entre artículos y a partir de esta métrica obtener los vecinos más cercanos que sirvan para estimar los ratings no observados para cada relación de usuario y artículo. Ambos enfoques se integran en una suma ponderada donde los coeficientes son calculados a partir de una regresión para obtener la proporción en que cada SR aporta a la estimación final de los ratings para reducir el error.

El uso de estas aproximaciones tuvo como motivo explotar las cualidades que caracterizan a cada enfoque, como la escalabilidad del modelo SVD y la capacidad para recomendar artículos que no han sido evaluados previamente del BC, que representan el problema de cold-start para artículos.

El SR resultante presentó buenos resultados con respecto a los valores calculados RMSE y MAE que se mantuvieron siempre por debajo de los algoritmos individuales lo que indica su buen rendimiento para las predicciones.

El valor calculado de FCP calculado también favoreció al SR compuesto, colocándolo un poco por encima del valor SR SVD.

La complejidad del SR compuesto permanece con un orden lineal, lo que permite que sea escalable para procesar grandes volúmenes de datos, en parte esta eficiencia se ve beneficiada con la aplicación de modelos de matrices dispersas para representar los datos ya que por su naturaleza los datos de ratings son muy poco densos.

De forma general se lograron los objetivos planeados en este trabajo con buenos resultados.

6.2. Trabajo Futuro

En este trabajo se presentó un enfoque para abordar los problemas de eficiencia temporal y espacial a partir del método de entrenamiento con SGD y la aplicación de las estructuras de datos de matrices dispersas. Y el cold-start con el uso de contenidos que describen a los artículos. Al tratarse de datos de negocios y las preferencias de usuarios se limitaron de forma espacial a una ciudad o un territorio para mantener la utilidad en las recomendaciones con respecto a la ubicación del usuario activo. Este trabajo

puede complementarse añadiendo una restricción temporal que modele de una forma más confiable los cambios en las preferencias de los usuarios con respecto al tiempo.

A su vez pueden integrarse los módulos desarrollados en este trabajo, que incluyen la extracción de contenido con su respectivo preprocesamiento de datos, los módulos de entrenamiento de cada modelo y predicciones sobre un sistema integrado con servicios en la nube, que permitan ejecutar el proceso completo desde la captura de ratings, hasta las recomendaciones sugeridas al usuario activo, en forma de interfaz gráfica, sin que exista la necesidad de conocer la implementación de cada módulo.

Sería interesante evaluar el SR propuesto en este trabajo en datos de distinta naturaleza a los utilizados de negocios, estos nuevos datos podrían ser películas, canciones, o productos de comercio por internet. Esta evaluación serviría para comprobar la robustez del sistema y su generalización de forma independiente a los datos seleccionados.

Otro trabajo futuro sería incorporar un análisis de grafos utilizando relaciones de amistad entre usuarios que permitan ofrecer recomendaciones de artículos a partir de los gustos de los nodos conectados al usuario activo. También podrían recomendar amigos a partir de un cálculo de similitud entre usuarios de acuerdo a sus preferencias sobre artículos en común. El set de datos utilizado en este trabajo contiene información de amistad entre usuarios por lo que podrían utilizarse esos valores para un análisis de ese tipo.

BIBLIOGRAFÍA

[1] J. B. Schafer, J. Konstan, y J. Riedl, "Recommender systems in e-commerce", en *Proceedings of the 1st ACM conference on Electronic commerce*, 1999, pp. 158–166.

[2] D. Goldberg, D. Nichols, B. M. Oki, y D. Terry, "Using collaborative filtering to weave an information tapestry", *Commun. ACM*, vol. 35, núm. 12, pp. 61–70, 1992.

[3] C. C. Aggarwal, *Recommender Systems: The Textbook*. Springer, 2016.

[4] A. Poriya, T. Bhagat, N. Patel, y R. Sharma, "Non-personalized recommender systems and user-based collaborative recommender systems", *Int J Appl Inf Syst*, vol. 6, núm. 9, pp. 22–27, 2014.

[5] Z.-K. Zhang, C. Liu, Y.-C. Zhang, y T. Zhou, "Solving the cold-start problem in recommender systems with social tags", *EPL Europhys. Lett.*, vol. 92, núm. 2, p. 28002, 2010.

[6] R. Burke, "Hybrid recommender systems: Survey and experiments", *User Model. User-Adapt. Interact.*, vol. 12, núm. 4, pp. 331–370, 2002.

[7] Y. Koren, R. Bell, y C. Volinsky, "Matrix factorization techniques for recommender systems", *Computer*, vol. 42, núm. 8, 2009.

[8] N. J. Belkin y W. B. Croft, "Information filtering and information retrieval: Two sides of the same coin?", *Commun. ACM*, vol. 35, núm. 12, pp. 29–38, 1992.

[9] G. Salton, E. A. Fox, y H. Wu, "Extended Boolean information retrieval", *Commun. ACM*, vol. 26, núm. 11, pp. 1022–1036, 1983.

[10] A. Huang, "Similarity measures for text document clustering", en *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand*, 2008, pp. 49–56.

[11] S. E. Robertson, "The probability ranking principle in IR", *J. Doc.*, vol. 33, núm. 4, pp. 294–304, 1977.

[12] G. Linden, B. Smith, y J. York, "Amazon. com recommendations: Item-to-item collaborative filtering", *IEEE Internet Comput.*, vol. 7, núm. 1, pp. 76–80, 2003.

[13] C. A. Gomez-Uribe y N. Hunt, “The Netflix recommender system: Algorithms, business value, and innovation”, *ACM Trans. Manag. Inf. Syst. TMIS*, vol. 6, núm. 4, p. 13, 2016.
[14] Sander Dieleman, “Recommending music on Spotify with deep learning”, *Sander Dieleman*. [En línea]. Disponible en: http://benanne.github.io/2014/08/05/spotify-cnns.html. [Consultado: 07-oct-2016].
[15] M. D. Ekstrand, J. T. Riedl, y J. A. Konstan, “Collaborative filtering recommender systems”, *Found. Trends Hum.-Comput. Interact.*, vol. 4, núm. 2, pp. 81–173, 2011.
[16] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, y J. Riedl, “GroupLens: an open architecture for collaborative filtering of netnews”, en *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, 1994, pp. 175–186.
[17] J. B. Schafer, J. A. Konstan, y J. Riedl, “E-commerce recommendation applications”, en *Applications of Data Mining to Electronic Commerce*, Springer, 2001, pp. 115–153.
[18] W. Hill, L. Stead, M. Rosenstein, y G. Furnas, “Recommending and evaluating choices in a virtual community of use”, en *Proceedings of the SIGCHI conference on Human factors in computing systems*, 1995, pp. 194–201.
[19] Y. Koren, “Factorization meets the neighborhood: a multifaceted collaborative filtering model”, en *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2008, pp. 426–434.
[20] R. Kohavi, “A study of cross-validation and bootstrap for accuracy estimation and model selection”, en *Ijcai*, 1995, vol. 14, pp. 1137–1145.
[21] Y. Koren y J. Sill, “Collaborative Filtering on Ordinal User Feedback”, p. 5.
[22] “Yelp Dataset”. [En línea]. Disponible en: https://www.yelp.com/dataset/documentation/json. [Consultado: 31-oct-2017].
[23] “Python Data Analysis Library — pandas: Python Data Analysis Library”. [En línea]. Disponible en: https://pandas.pydata.org/. [Consultado: 18-jun-2018].
[24] “reverse_geocode”, *PyPI*. [En línea]. Disponible en: https://pypi.org/project/reverse_geocode/. [Consultado: 19-jun-2018].
[25] “SciPy — SciPy v1.1.0 Reference Guide”. [En línea]. Disponible en: https://docs.scipy.org/doc/scipy/reference/index.html. [Consultado: 25-jun-2018].
[26] “Matrix Factorization-based algorithms — Surprise 1 documentation”. [En línea]. Disponible en: http://surprise.readthedocs.io/en/stable/matrix_factorization.html#surprise.prediction_algorithms.matrix_factorization.SVD. [Consultado: 10-jul-2018].
[27] “sklearn.metrics.pairwise.cosine_similarity — scikit-learn 0.18.1 documentation”. [En línea]. Disponible en: http://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html. [Consultado: 12-may-2017].

APÉNDICE A. Estructura del dataset original


```
{
    "business_id": "YDf95gJZaq05wvo7hTQbbQ",
    "name": "Richmond Town Square",
    "neighborhood": "",
    "address": "691 Richmond Rd",
    "city": "Richmond Heights",
    "state": "OH",
    "postal_code": "44143",
    "latitude": 41.5417162,
    "longitude": -81.4931165,
    "stars": 2.0,
    "review_count": 17,
    "is_open": 1,
    "attributes": {
        "RestaurantsPriceRange2": 2,
        "BusinessParking": {
            "garage": false,
            "street": false,
            "validated": false,
            "lot": true,
            "valet": false
        },
        "BikeParking": true,
        "WheelchairAccessible": true
    },
    "categories": ["Shopping", "Shopping Centers"],
    "hours": {
        "Monday": "10:00-21:00",
        "…": "…",
    }
}
```


Figura 33. Estructura de los elementos del archivo business.json

```
{
    "review_id": "VfBHSwC5Vz_pbFluy07i9Q",
    "user_id": "cjpdDjZyprfyDG3RlkVG3w",
    "business_id": "uYHaNptLzDLoV_JZ_MuzUA",
    "stars": 5,
    "date": "2016-07-12",
    "text": "My girlfriend and I stayed here for 3 nights and loved it. The location
of this hotel and very decent price makes this an amazing deal...",
    "useful": 0,
    "funny": 0,
    "cool": 0
}
```


Figura 35. Estructura de reviews en el archivo review.sjon


```
{
  "user_id": "IsSiIjAKVl-QRxKjRErBeg",
  "name": "Cin",
  "review_count": 272,
  "yelping_since": "2010-07-13",
  "friends": ["M19NwFwAXKRZzt8koF11hQ", "…"],
  "useful": 17019,
  "funny": 16605,
  "cool": 16856,
  "fans": 209,
  "elite": [2014, 2016, 2013, 2011, 2012, 2015, 2010, 2017],
```


Figura 34. Estructura de usuarios en el archivo users.json

APÉNDICE B. Listado completo de variables

Tabla 13. Listado completo de atributos en el dataset business

attributes.Corkage	attributes.Music.no_music
attributes.DietaryRestrictions.dairy-free	address
attributes.DietaryRestrictions.gluten-free	attributes.AcceptsInsurance
attributes.DietaryRestrictions.halal	attributes.AgesAllowed
attributes.DietaryRestrictions.kosher	attributes.Alcohol
attributes.DietaryRestrictions.soy-free	attributes.Ambience.casual
attributes.DietaryRestrictions.vegan	attributes.Ambience.classy
attributes.DietaryRestrictions.vegetarian	attributes.Ambience.divey
attributes.DogsAllowed	attributes.Ambience.hipster
attributes.DriveThru	attributes.Ambience.intimate
attributes.GoodForDancing	attributes.Ambience.romantic
attributes.GoodForKids	attributes.Ambience.touristy
attributes.GoodForMeal.breakfast	attributes.Ambience.trendy
attributes.GoodForMeal.brunch	attributes.Ambience.upscale
attributes.GoodForMeal.dessert	attributes.BYOB
attributes.GoodForMeal.dinner	attributes.BYOBCorkage
attributes.GoodForMeal.latenight	attributes.BestNights.friday
attributes.GoodForMeal.lunch	attributes.BestNights.monday
attributes.HairSpecializesIn.africanamerican	attributes.BestNights.saturday
attributes.HairSpecializesIn.asian	attributes.BestNights.sunday
attributes.HairSpecializesIn.coloring	attributes.BestNights.thursday
attributes.HairSpecializesIn.curly	attributes.BestNights.tuesday
attributes.HairSpecializesIn.extensions	attributes.BestNights.wednesday
attributes.HairSpecializesIn.kids	attributes.BikeParking
attributes.HairSpecializesIn.perms	attributes.BusinessAcceptsBitcoin
attributes.HairSpecializesIn.straightperms	attributes.BusinessAcceptsCreditCards
attributes.HappyHour	attributes.BusinessParking.garage
attributes.HasTV	attributes.BusinessParking.lot
attributes.Music.background_music	attributes.BusinessParking.street
attributes.Music.dj	attributes.BusinessParking.valet
attributes.Music.jukebox	attributes.BusinessParking.validated
attributes.Music.karaoke	attributes.ByAppointmentOnly
attributes.Music.live	attributes.Caters
attributes.Music.video	attributes.CoatCheck
attributes.NoiseLevel	attributes.RestaurantsGoodForGroups
attributes.Open24Hours	attributes.RestaurantsPriceRange2
attributes.OutdoorSeating	attributes.RestaurantsReservations
attributes.RestaurantsAttire	attributes.RestaurantsTableService
attributes.RestaurantsCounterService	attributes.RestaurantsTakeOut
attributes.RestaurantsDelivery	attributes.Smoking
attributes.WheelchairAccessible	attributes.WiFi

Tabla 14. Listado completo de variables en user y review

Variables user	Variables review
average_stars	business_id
compliment_cool	cool
compliment_cute	date
compliment_funny	funny
compliment_hot	stars
compliment_list	text
compliment_more	useful
compliment_note	user_id
compliment_photos	
compliment_plain	
compliment_profile	
compliment_writer	
cool	
elite	
fans	
friends	
funny	
name	
review_count	
useful	
yelping_since	

APÉNDICE C. Pasos de procesamiento de datos de entrada

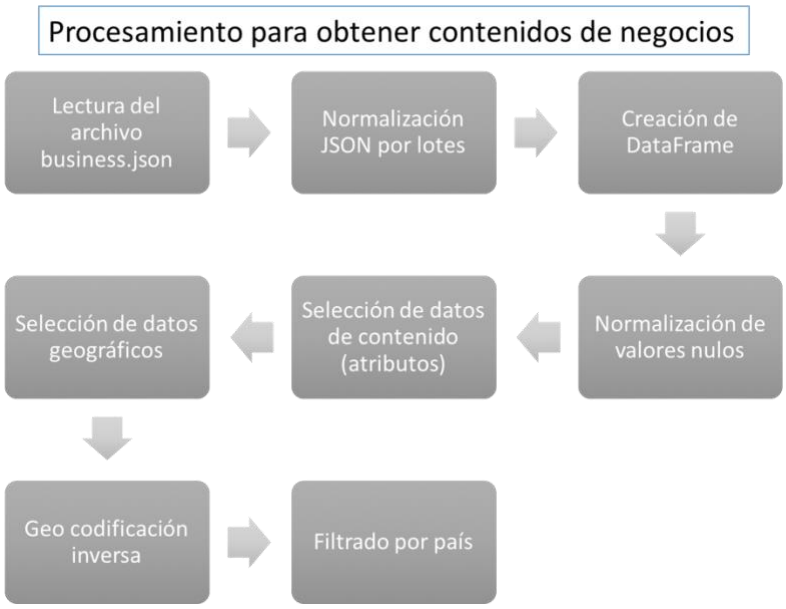

Figura 36. Procesamiento de datos para extraer contenidos de negocios

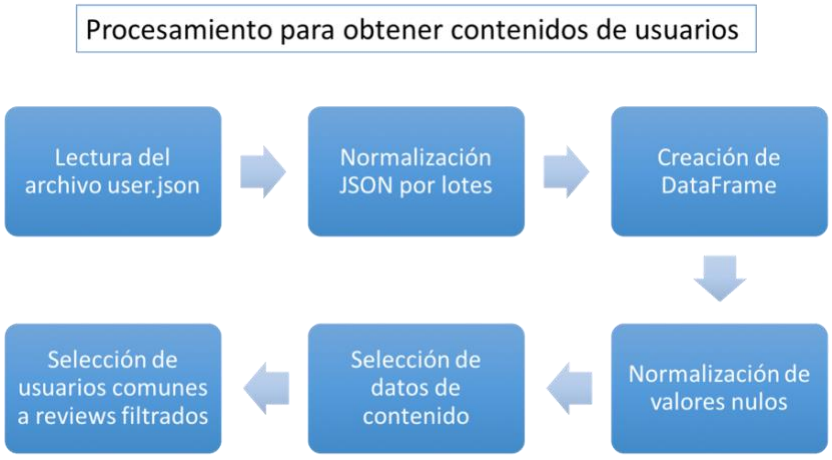

Figura 37. Procesamiento de datos para obtener contenido de los usuarios

Figura 38. Procesamiento de datos para obtener ratings de reviews

APÉNDICE D. Elementos del SR

Tabla 15. Ventajas y desventajas de las representaciones de matrices dispersas [25]

Representación	Ventajas	Desventajas	Intención de uso
COO	• Facilita la conversión entre formatos sparse. • Permite entradas duplicadas • Rápida conversión a CSR/CSC	• No soporta directamente operaciones aritméticas	Es una forma rápida de construir matrices dispersas, una vez construida puede convertirse a CSR o CSC para operaciones aritméticas y vectoriales.
CSC	• Eficiente para operaciones aritméticas • Eficiente para slicing por columnas • Rápido producto de vectores.	• Lento slicing por filas • Cambios en la estructura son procesos pesados.	Son usadas para operaciones aritméticas o slicing por columnas eficientemente.

CSR	• Eficiente para operaciones aritméticas • Eficiente para slicing por filas • Rápido producto de vectores	• Lento slicing por columnas. • Cambios en la estructura son procesos pesados.	Son usadas para operaciones aritméticas o slicing por filas eficientemente.
BSR	• Eficientes para operaciones aritméticas • Son más apropiadas cuando la matriz dispersa tiene submatrices más densas.	• Su proceso de conversión es más tardado por el cálculo del tamaño de bloque.	Son recomendadas para los casos con submatrices densas.