UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 3

GRUPO DE WEB SEMÁNTICA

ALGORITMO PARA LA RECOMENDACIÓN DE TEXTOS CIENTÍFICOS BASADOS EN SU CONTENIDO

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE
INGENIERO EN CIENCIAS INFORMÁTICAS

AUTOR: MARCOS RIVAS GAGO
TUTOR: MSC. YUSNIEL HIDALGO DELGADO

La Habana, septiembre de 2020

Declaración Jurada de Autoría

Yo, **Marcos Rivas Gago**, con carné de identidad 96011409962, declaro que esta tesis titulada ***'Algoritmo para la recomendación de textos científicos basados en su contenido'*** y la investigación presentada en ella, son de mi autoría. A su vez reconozco a la **Universidad de las Ciencias Informáticas** los derechos patrimoniales con carácter exclusivo sobre la misma para usarla a su beneficio. Por lo tanto confirmo que:

- Este trabajo ha sido realizado completamente para optar por el título de Ingeniero en Ciencias Informáticas en esta Universidad.
- Donde he consultado y citado el trabajo publicado por otros, siempre se ha referenciado claramente y se ha especificado la fuente.

Y para que así conste se firma la presente a los _____ días del mes de _____ del año 2020.

Marcos Rivas Gago

MSc. Yusniel Hidalgo Delgado

“Tu tiempo es limitado, así que no lo malgastes viviendo la vida de otra persona [...] No dejes que el ruido de las opiniones de otros apague tu propia voz interior”

Steve Jobs

Dedicatoria

Texto.

Marcos Rivas Gago

Agradecimientos

Texto.

Marcos Rivas Gago

Resumen

Un algoritmo de recomendación establece un conjunto de criterios y valoraciones sobre los datos de los usuarios para realizar predicciones sobre recomendaciones de elementos que puedan ser de utilidad o valor para el usuario. Existen varios tipos y dependen de variables que son las que condicionan su funcionamiento, variando en cuanto a la forma de predecir los elementos. En la investigación se hace uso del tipo de algoritmo de recomendación conocido como filtrado por contenido, el cual es el que más se ajusta al objetivo general, que no es más que desarrollar un algoritmo que recomiende textos científicos a partir de su contenido. Este tipo de algoritmo está teniendo un gran auge y presencia en muchos sistemas y plataformas en la actualidad. Por ello el algoritmo resultante y la visualización de los elementos recomendados que devuelve el mismo, formarán parte de la biblioteca digital desarrollada por la Universidad de Ciencias Informáticas (UCI), denominada SIGEDIC, y más directamente formará parte del componente de creación de páginas del sistema.

Palabras clave: algoritmo de recomendación, biblioteca digital, filtrado por contenido, textos científicos.

Abstract

A recommendation algorithm establishes a set of criteria and ratings on user data to make predictions about recommendations of items that may be useful or valuable to the user. There are several types and they depend on variables that condition their operation, varying in terms of how to predict the elements. The research makes use of the type of recommendation algorithm known as content filtering, which is the one that best fits the general objective, which is simply to develop an algorithm that recommends scientific texts based on their content. This type of algorithm is having a great boom and presence in many systems and platforms today. For this reason, the resulting algorithm and the display of the recommended elements that it returns, will be part of the digital library developed by the University of Computer Science (UCI), called SIGEDIC, and more directly will be part of the system's page creation component.

Keywords: recommendation algorithm, digital library, content filtering, scientific texts

Índice

Índice de tablas

Índice de figuras

Índice de ecuaciones

Introducción

Con el actual desarrollo de las Tecnologías de la Información y las Comunicaciones (*TIC*) y el creciente aumento de la Internet en el mundo, los investigadores y personas en busca de conocimiento científico recaen en el gran cúmulo de **textos científicos** que se encuentran en la red. Estos surgen como resultado de una investigación anterior que aborda teorías, conceptos y temas con una base clara del conocimiento científico, todo ello a través de un lenguaje técnico especializado, producto de un trabajo metódico y sistemático en el cual se presenta un proceso de investigación, datos, pruebas, resultados y conclusiones. (Zita, 2019)

Existen diversos sitios web en los cuales los textos científicos están recopilados y agrupados; estas son las comúnmente llamadas **bibliotecas digitales** las cuales son un sistema de tratamiento técnico, acceso y transferencia de información digital, estructurado alrededor de una colección de documentos digitales, sobre los cuales se ofrecen servicios interactivos para el usuario (Tramullas, 2002). Estas bibliotecas constan de un conjunto de componentes, entre los se encuentran: una colección (o colecciones), un sistema informático el cual ofrece diversos servicios sobre la colección (una infraestructura técnica), personas, y un entorno para el cual se ha construido el sistema informático (Salas et al., 2019).

Entre las disímiles **bibliotecas digitales** que existen actualmente en la red encontramos:

- La Biblioteca Digital Hispánica (*BDH*), la cual es un recurso en línea de la Biblioteca Nacional de España (*BNE*) que proporciona acceso libre y gratuito a miles de documentos digitalizados por la Biblioteca.
- La Biblioteca Digital Mundial (*BDM*)[1], una biblioteca digital internacional creada por la Biblioteca del Congreso de Estados Unidos y la UNESCO[2], en cooperación con bibliotecas, archivos, museos, instituciones educativas y organizaciones internacionales de todo el mundo.

[1] World Digital Library (*WDL*) por sus siglas en inglés
[2] Organización de las Naciones Unidas para la Educación, la Ciencia y la Cultura

- Google Libros[3], un servicio de Google que busca el texto completo de los libros que Google digitaliza, convierte el texto por medio de reconocimiento óptico de caracteres y los almacena en su base de datos en línea.
- The European Library (*TEL*)[4] la cual ofrece un acceso único a los recursos de 47 Bibliotecas Nacionales de Europa. Es un proyecto de la Conferencia de Directores de Bibliotecas Nacionales de Europa (*CENL*).
- Wikisource, un proyecto hermano de Wikipedia cuyo objetivo es crear una biblioteca de textos originales libres, que hayan sido publicados con una licencia GFDL, Creative Commons o que sean de dominio público.
- SciELO[5], iniciativa de la Fundación para el Apoyo a la Investigación del Estado de São Paulo, Brasil[6] y del Centro Latinoamericano y del Caribe de Información en Ciencias de la Salud (*BIREME*), la cual permite la publicación electrónica de ediciones completas de las revistas científicas mediante una plataforma de software que posibilita el acceso a través de distintos mecanismos, incluyendo listas de títulos y por materia, índices de autores y materias y un motor de búsqueda.
- ScienceDirect, que proporciona acceso por suscripción a una gran base de datos de investigación científica y médica. Alberga más de 12 millones de contenidos de 3.500 revistas académicas y 34.000 libros electrónicos.
- Sunshine, propuesta de la comunidad de la Universidad de la Ciencias Informáticas, en el cual se puede compartir y aprender; en donde los usuarios pueden subir y descargar los documentos o libros de su interés.

En muchos de estos sitios web se puede apreciar que la información se agrupa por temáticas o elementos en común, haciendo uso de algoritmos que ayudan a encontrar la similitud entre sus contendidos, apoyándose principalmente en el procesamiento del lenguaje natural (PLN)[7]; entendido como la habilidad de la computadora para procesar la información comunicada, no simplemente las letras o los sonidos del lenguaje (Gelbukh, 2010). Esta información para procesarla la computadora y poder comunicarnos las personas con ella, esta primera debe entender las oraciones que le sean proporcionadas usando los lenguajes naturales; en otras palabras, entender ese medio que utilizamos de manera cotidiana para establecer nuestra comunicación con las demás personas (Vásquez et al., 2009).

[3] Conocido antes como Google Book Search y Google Print
[4] La Biblioteca Europea
[5] Scientific Electronic Library Online o Biblioteca Científica Electrónica en Línea
[6] Fundação de Amparo à Pesquisa do Estado de São Paulo — FAPESP
[7] Natural Language Processing (*NLP*) por sus siglas en inglés

Las agrupaciones de información se suelen hacer creando páginas nuevas y procesando el contenido de los documentos existentes en el sitio web, para luego dar una respuesta al usuario llenando dicha página. Cuando estas no son automáticas, suelen tener que hacerla los usuarios de forma manual y obligándolos a buscar entre cientos de textos y documentos que pueden estar o no acorde con la temática que se desea agrupar. Todo esto conlleva a que se vaya disminuyendo la experiencia de usuario. Sin embargo, existe una técnica que se utiliza actualmente que ayuda a mejorar dicha experiencia a la hora de crear estas páginas: **los algoritmos de recomendación**.

Estos tienen el objetivo de brindar a los usuarios resultados de búsqueda cercanos o adaptados a sus necesidades, realizando predicciones de sus preferencias y entregando aquellos elementos (*items*) que podrían acercarse más a lo esperado (Vera et al., 2015).

Estos no son más que es un sistema inteligente que proporciona a los usuarios una serie de sugerencias personalizadas (recomendaciones) sobre un determinado tipo de *items*. Los que estudian las características de cada usuario o elemento a relacionar, para que luego mediante un procesamiento de los datos obtenidos, se encuentre un subconjunto de *items* que puede resultar similar a lo que se busca. En los últimos años, se han propuesto diversos algoritmos y herramientas para el desarrollo de sistemas de recomendación, convirtiéndose en un área activa de investigación que hace grandes aportes a la experiencia del usuario. Esto se debe, en su mayoría, a que un usuario al ver que es elemento de su interés podrá seguir ese contenido e ir indagando entre otros recomendados para así llegar a encontrar lo que realmente estaba buscando, o simplemente conocer mucho más contenido que le puede ser de utilidad.

En la Universidad de las Ciencias Informáticas (*UCI*), se está desarrollando la biblioteca digital nombrada SIGEDIC[8]. La misma gestiona una colección de textos científicos, generalmente artículos de revistas y congresos. En esta biblioteca se quiere generar páginas de forma automática, atendiendo a un conjunto de palabras clave o temáticas específicas definidas por un administrador, el cual será un usuario que desee crear la página de acuerdo a una temática que le sea de su interés. Luego de creada, otros usuarios pueden seguir a la página, o de igual forma el administrador puede compartirla con otros usuarios para que estos la sigan. Sin embargo, este componente carece de un algoritmo de recomendación de textos científicos que ayuden al administrador a decidir qué tipo de documentos se relacionan con su página, en dependencia de las características propias del contenido del documento, para que posteriormente el administrador pueda escoger de entre estos documentos cuáles desea añadir.

Atendiendo a la situación descrita anteriormente se plantea el siguiente **problema a re-**

[8] Sistema de Gestión de Datos e Información Científica

solver: ¿Cómo aumentar la experiencia de usuario al crear páginas en el sistema SIGEDIC si se recomiendan textos científicos atendiendo a su contenido?

A partir de lo anterior se puede definir como **objeto de estudio** los algoritmos de recomendación, especificándose como **campo de acción** los algoritmos de recomendación de textos científicos basados en su contenido.

Con la presente investigación se tiene como **objetivo general** el desarrollar un algoritmo que recomiende textos científicos a partir de su contenido, para ello se definen los siguientes **objetivos específicos**:

1. Elaborar el marco teórico y el estado del arte del objeto de estudio de la investigación mediante el análisis bibliográfico documental para identificar tendencias y adoptar posiciones al respecto.
2. Diseñar un algoritmo que recomiende textos científicos a partir de su contenido.
3. Implementar un algoritmo que recomiende textos científicos a partir de su contenido.
4. Validar los resultados obtenidos con la utilización del algoritmo desarrollado mediante la realización de un diseño experimental.

Para apoyar lo anterior se tienen las siguientes tareas a cumplir durante la investigación:

- Estudio de las principales aproximaciones existentes para la recomendación de textos científicos.
- Diseño del algoritmo deseado.
- Implementación del algoritmo.
- Análisis de los resultados obtenidos con la utilización del algoritmo en un escenario real.
- Validación de los resultados obtenidos por el algoritmo.

Apoyándose en lo todo expuesto se puede plantear como **idea a defender** que si se desarrolla un algoritmo para recomendar textos científicos atendiendo a su contenido aumentará la experiencia de usuario en las páginas creadas en el sistema SIGEDIC.

La investigación realizada está sustentada sobre la base de emplear **métodos científicos**. Entre los **métodos teóricos** ha usar se encuentran:

- Analítico-sintético: utilizado para analizar y detallar todas las partes que componen un sistema de recomendación de textos científicos basándose en su contenido, así como las funciones de similitud que este necesita para su correcto funcionamiento.

- Histórico-lógico: para estudiar a fondo la trayectoria por la que han pasado las diferentes propuestas de sistemas de recomendación, comprender cada uno de sus elementos y conocer la evolución que han tenidos los mismos a lo largo del desarrollo del Internet y las TIC.
- Modelación: para describir la propuesta de solución juntos a las reglas y métricas que exige un algoritmo de recomendación basado en el contenido de textos científicos.

Y entre los **métodos empíricos** se encuentran los siguientes:

- Análisis documental: para seleccionar y escoger toda la bibliografía referente a los sistemas de recomendación, sus usos, implementación, y particularidades; permitiendo posteriormente estudiarla a fondo y elaborar el marco teórico y código fuente de la investigación.
- Observación científica: para examinar y analizar los resultados obtenidos luego de ejecutar los experimentos para validar el algoritmo de recomendación de textos científicos.
- Experimento: para la realización de los experimentos diseñados para la validación del algoritmo de recomendación de textos científicos.
- Medición: utilizado con el objetivo de medir y calcular la eficacia del algoritmo de recomendación para obtener resultados que mejoren la experiencia del usuario.

Por la validación de la investigación y la propuesta de solución se dispondrá de un conjunto de datos que contengan los metadatos necesarios para que el algoritmo tenga una correcta ejecución o no, en dependencia de los escenarios previstos previamente. Por ello se selecciona como **población** los metadatos obtenidos de revistas científicas. Mientras que por otra parte la **muestra** seleccionada serán los metadatos obtenidos de 10 revistas científicas cubanas.

Al finalizar se tendrá como **posibles resultados de la investigación** el código fuente de un componente de software que recomiende textos científicos a partir de su contenido, que será incluido en el sistema SIGEDIC.

La investigación consta de un resumen plasmado en idioma español e inglés, un índice general, uno de tablas, uno de figuras, y otro de ecuaciones. Además cuenta con una introducción, tres capítulos, conclusiones, recomendaciones, glosario de términos, símbolos, abreviaturas y referencias bibliográficas. Cada capítulo contendrá una introducción y una conclusión parcial del mismo; por ello quedan estructurados de la siguiente forma:

Capítulo 1: ”Fundamentación teórica”: se realiza un análisis de la bibliografía consultada, se define el marco teórico de la investigación, junto a los conceptos que tiene asociados para

su correcto entendimiento. Se define además el estado de arte de la investigación, el cual contiene herramientas y algoritmos de recomendación existentes actualmente, las herramientas utilizadas y los lenguajes que sirven de soporte a la solución propuesta.

Capítulo 2: ”Fundamentación de la propuesta de solución”: se describe todo lo relacionado al algoritmo a implementar. En él se plasman todos los componentes, elementos y artefactos que componen el algoritmo de recomendación de textos científicos basado en su contenido.

Capítulo 3: ”Validación y análisis de la solución”: se centra en el diseño y descripción de los experimentos para validar el algoritmo de recomendación propuesto como solución. En el capítulo juegan un importante papel las variables definidas y su operacionalización, para ello se aplica un diseño experimental que determinará si realmente se mejora la experiencia del usuario con la aplicación del algoritmo.

Capítulo 1

Fundamentación teórica

En el presente capítulo se presentarán los principales conceptos, definiciones y características del procesamiento de lenguaje natural, los sistemas de recomendación y los algoritmos de similitud que son de utilidad para entender la investigación y la posterior propuesta de solución; se plasman las bases en los métodos histórico-lógico y analítico-sintético para hacer mención a investigaciones afines, algoritmos, posibles aproximaciones y herramientas utilizadas en los sistemas de recomendación que sean semejantes a la solución que se desea proponer; se definen los elementos utilizados en la posterior propuesta de solución; y se analiza además la bibliografía consultada y el impacto que esta tiene para la investigación, haciendo uso del análisis bibliométrico y documental.

1.1. Análisis bibliométrico y documental

Para la realización de la presente investigación se llevó a cabo un estudio documental que abarca principalmente la literatura publicada en los últimos 5 años. Se consultaron numerosas fuentes de datos bibliográficos, entre ellos se encuentran: IEEE[9], Google Scholar[10] y Scielo[11]. Se consultaron sitios web oficiales en las cuales se ha desarrollado aproximaciones de sistemas de recomendación de artículos científicos para estudiar su funcionamiento. A continuación se muestra una tabla resumen que contiene el tipo de fuente bibliográfica, la cantidad consultada de esta y cuantas de ellas pertenecen a los últimos 5 años (2015 - 2020).

[9] IEEE Xplore: https://ieeexplore.ieee.org
[10] Google Scholar: https://scholar.google.com
[11] SciELO.org: https://scielo.org

Tipo de fuente bibliográfica	Cantidad consultada	Perteneciente a los últimos 5 años (2015 - 2020)
Artículos en revistas científicas	17	7
Artículos en congresos	9	2
Libros	6	2
Tesis de pregrado	3	1
Reportes de investigación	1	0
Páginas webs	9	8
Total	**45**	**20**

Tabla 1.1: Resumen de la revisión bibliográfica realizada

La anterior arroja un total de 45 bibliografías consultadas, de las cuales 18 pertenecieron a los últimos 5 años, representando un porcentaje de 44 % del total de consultada.

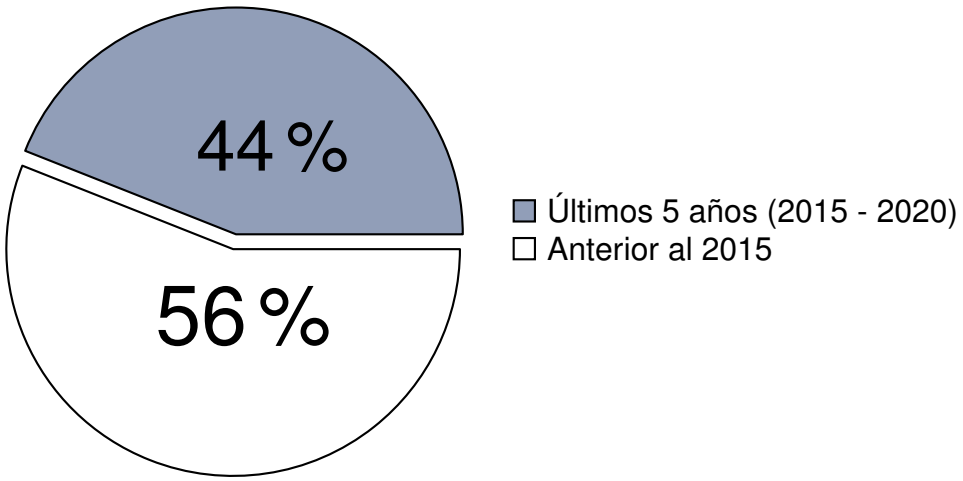

Figura 1.1: Relación entre la bibliografía perteneciente a los últimos años y la bibliografía consultada

1.2. Marco Teórico. Conceptos y definiciones

En esta sección se describen los principales conceptos asociados al dominio del problema con el objetivo de avalar la investigación, para posteriormente crear la base de conocimientos necesaria para el desarrollo de la propuesta de solución. Con el objetivo de lograr un mejor

entendimiento se comienza desde los conceptos más básicos que son necesarios para la solución, entrelazando y vinculando cada uno de ellos hasta llegar a los conceptos más amplios y específicos, los cuales serán de utilidad en el completo entendimiento del funcionamiento de los sistemas de recomendación basados en el contenido de textos científicos.

1.2.1. Lenguaje

Un **lenguaje** se puede definir de diferentes formas: desde el punto de vista funcional lingüístico se define como una función que expresa pensamientos y comunicaciones entre las personas; esta puede realizarse mediante signos escritos o mediante señales y vocales. Mientras que desde un punto de vista formal se define como un conjunto de frases, que se forman con combinaciones de elementos tomados de un conjunto llamado alfabeto, respetando un conjunto de reglas de formación (*sintácticas o gramaticales*) y de sentido (*semánticas*). Además una de las características fundamentales del **lenguaje** es que debe permitirnos expresar nuestras ideas de forma clara. (Vásquez et al., 2009)

Esta se pueden diferenciar en dos clases de lenguajes: los lenguajes naturales (*inglés, alemán, español, etc.*) y lenguajes formales (*matemático, lógico, programable, etc.*).

1.2.2. Lenguaje Natural

Aunque la definición exacta varía entre los estudiosos, el **lenguaje natural** puede definirse en términos generales en contraste con los lenguajes artificiales o construidos (*como los lenguajes de programación informática y los lenguajes auxiliares internacionales*) y con otros sistemas de comunicación en la naturaleza.

Por ello en neuropsicología, lingüística y filosofía del lenguaje, definen que un **lenguaje natural** es cualquier lenguaje que ha evolucionado naturalmente en los humanos a través del uso sin planificación consciente o premeditación. Los **lenguajes naturales** pueden tomar diferentes formas, como el habla o las señas. Se distinguen de los lenguajes construidos y formales como los que se utilizan para programar ordenadores o para estudiar la lógica. (John, 1991)

Los **lenguajes naturales** tienen un gran poder expresivo y una función como herramienta para razonamiento donde se caracterizan por definirse a partir de una gramática G, sin embargo, este se enriquece progresivamente modificando así la gramática que la define. Con esto dificulta la formalización de lo definición de G.

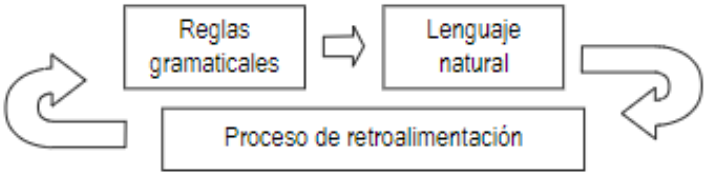

Figura 1.2: Proceso del lenguaje natural

1.2.3. Texto Científico

Un **texto científico** se puede definir como una producción escrita cuya función es referencial o informativa, la cual aborda teorías, conceptos o cualquier otro tema con base en el conocimiento científico a través de un lenguaje técnico especializado que se caracteriza por la exactitud, la precisión y la sistematicidad de los medios expresivos (Domínguez García, 2010).

El objetivo del **texto científico** es transmitir, de manera apropiada, clara y concisa, los resultados de una investigación realizada sobre un tema específico a la comunidad científica, así como al público interesado en general.

Estos tienen a su vez varias tipologías, las cuales son: informe de investigación, artículo científico, monografía, ponencia, tesina, tesis, proyecto de investigación, y póster. Todas ellas son diferentes y tienen características propias, sin embargo poseen puntos en común como pueden ser:

- Ser objetivos y claros.
- Basarse en datos verificables.
- Utilizar una terminología o un léxico propio de cada área científica que aborda (*matemática, informática, biología, física, química, etc.*).
- Buscar siempre transmitir al lector los aspectos de la realidad abordados con un rigor metódico y una formalidad definida.

1.2.4. Biblioteca Digital

Una de las definiciones más extendida de una **biblioteca digital** es la definida por las instituciones asociadas en la Federación de Bibliotecas Digitales (*DLF*) en donde atendiendo a sus funciones se definen como:

"*Las **bibliotecas digitales** son organizaciones que proporcionan los recursos, incluido el personal especializado, para seleccionar, estructurar, ofrecer acceso intelectual, interpretar,*

distribuir, preservar la integridad y garantizar la persistencia a lo largo del tiempo de las colecciones de obras digitales, de modo que estén disponibles de forma fácil y económica para su uso por una comunidad o conjunto de comunidades definidas." (Waters, 1998)

También se puede definir atendiendo a sus características, como lo es la propuesta de la *Association of Research Libraries*, que en 1995 planteaba una integración de definiciones atendiendo a los siguientes rasgos: (Tramullas, 2002)

- La **biblioteca digital** no es una entidad única.
- La **biblioteca digital** requiere tecnología para vincular los recursos de muchos.
- Los vínculos entre las numerosas **bibliotecas digitales** y los servicios de información son transparentes para los usuarios finales.
- El acceso universal a las **bibliotecas digitales** y a los servicios de información es un objetivo.
- Las colecciones de la **biblioteca digital** no se limitan a los sustitutos de documentos, se extienden a los artefactos digitales que no pueden ser representados o distribuidos en formatos impresos.

En el *IEEE CAIA'94 Workshop on Intelligent Access to On-Line Digital Libraries* (Gladney et al., 1994) se discutieron los requisitos y la arquitectura de los sistemas de **bibliotecas digitales** y se definió que una **biblioteca digital** es un conjunto de maquinaria de computación, almacenamiento y comunicaciones digitales junto con el contenido y el *software* necesarios para reproducir, emular y ampliar los servicios prestados por las bibliotecas convencionales basadas en papel y otros medios materiales de recopilación, catalogación, búsqueda y difusión de información. Una biblioteca digital de servicio completo debe cumplir todos los servicios esenciales de las bibliotecas tradicionales y también explotar las conocidas ventajas del almacenamiento, la búsqueda y la comunicación digitales.

Por todo lo anterior podemos decir que la **biblioteca digital** se convierte en una pasarela al conocimiento (*gateway to knowledge*); la cual podríamos definir como un sistema de tratamiento técnico, acceso y transferencia de información digital, estructurado alrededor una colección de documentos digitales (*entre ellos* ***textos científicos***), sobre los cuales se ofrecen servicios interactivos de valor añadido para el usuario final que accede a ellas. (Tramullas, 2002)

1.2.5. Inteligencia Artificial

Alan Turing definió la **Inteligencia Artificial** (*IA*) de la siguiente manera:

"*Si hay una máquina detrás de una cortina y un humano está interactuando con ella (por medios de odio, por ejemplo, audio o a través de la escritura, etc.) y si el humano siente que está interactuando con otro humano, entonces la máquina es artificialmente inteligente*" (Joshi, 2020).

Esta es una forma bastante única de definir la **IA**. No apunta directamente a la noción de inteligencia, sino que se centra en el comportamiento humano; en donde apuesta por no construir una máquina extraordinariamente inteligente que pueda resolver cualquier problema en poco tiempo, sino que en construir una máquina que sea capaz de un comportamiento similar al de un humano.

Desde una perspectiva más moderna la **IA** se refiere a un campo de la informática dedicado a la creación de sistemas que realizan tareas que normalmente requieren inteligencia humana. En la **IA**, las máquinas completan la tarea basándose en las reglas y algoritmos estipulados. La **IA** es un término general para cualquier programa informático que tenga el toque de inteligencia humana y abarca el aprendizaje de las máquinas (*machine learning*) y el aprendizaje profundo (*deep learning*). (Jakhar y Kaur, 2020)

La **IA** es una ciencia interdisciplinaria con múltiples enfoques, pero los avances en el *machine learning* y el *deep learning* están creando un cambio de paradigma en prácticamente todos los sectores de la industria tecnológica.

1.2.6. Procesamiento del Lenguaje Natural

El **procesamiento del lenguaje natural** (*PLN o NLP*[12]) tiene sus raíces en la década de 1950. En el año 1950, Alan Turing publicó un artículo titulado "*Computing Machinery and Intelligence*" en el que proponía lo que ahora se denomina el **test de Turing**[13] como un criterio de inteligencia, una tarea que implica la interpretación y generación automatizada del lenguaje natural, pero que en aquel momento no se articulaba como un problema separado de la inteligencia artificial.

Durante el transcurso del tiempo el **PLN** pasó por varias etapas: PLN Simbólico (*1950 – 1990*), PLN Estadístico (*1990 – 2010*) y PLN Neural (*2010 – al presente*). En esta última etapa los métodos de aprendizaje automático de redes neuronales profundas se generalizaron en el **procesamiento del lenguaje natural**, debido en parte a una avalancha de resultados que muestran que esas técnicas pueden lograr resultados de vanguardia en muchas tareas

[12] *Natural Language Processing* por sus siglas en inglés

[13] Es un examen de la capacidad de una máquina para exhibir un comportamiento inteligente similar al de un ser humano o indistinguible de este.

del lenguaje natural (Goldberg, 2016), por ejemplo, en el modelado del lenguaje, el análisis sintáctico, y muchos otros.

El **procesamiento de lenguaje natural** tiene habilidades de procesar la información comunicadora, por lo cual es una disciplina que se sitúa en la intersección de otras disciplinas científicas como las ciencias de la computación, la **IA**, la lingüística y la psicología cognitiva. En especial relación con la **IA** se usa en áreas como la representación del conocimiento, la planificación, la percepción y el aprendizaje (Lastra, 2019).

Las aplicaciones del **PLN** son variadas, ya que su alcance es muy grande, entre ellas se puede encontrar la traducción automática, la recuperación de la información, la extracción de resúmenes, la resolución cooperativa de problemas, el reconocimiento de voz, el análisis del sentimiento de los textos, la clasificación de documentos por categorías, los sistemas conversacionales, entre otros.

1.2.7. Algoritmo

Un **algoritmo** es una serie ordenada de instrucciones, pasos o procesos que llevan a la solución de un determinado problema. Permiten describir claramente una serie de instrucciones que debe realizar la maquina para lograr un resultado previsible. Vale la pena recordar que un procedimiento de maquina consiste de una serie de instrucciones muy precisas y escritas en un lenguaje de programación que la maquina entienda.

Otra de las definiciones según Ricardo Peña Marí, autor de (Marí, 2006) es que un **algoritmo** es un "*conjunto de reglas que, aplicada sistemáticamente a unos datos de entrada apropiados, resuelven un problema en un número finito de pasos elementales"*. Recalca además, que "*es importante notar que el **algoritmo** tiene que ser finito y que ejecuta las instrucciones de manera sistemática, es decir, que es ciego ante lo que está haciendo, y que los pasos con los que opera son elementales"*.

1.2.8. Recomendación

La palabra **recomendación** hace referencia tanto a la acción como al efecto de recomendar, vocablo formado por los siguientes términos de origen latino: el prefijo de reiteración, "re" y el verbo *"commendare"*, a su vez integrado por el prefijo de unión, *"con"* y por *"mandare"*, de *"manus"* con el significado de *"mano"* y *"dare"* en el sentido de *"dar"*. Podemos decir que es un consejo o sugerencia por el cual se le brinda a otro una idea de lo que debe hacer en determinada situación.

Se puede definir que **recomendación** es la acción y la consecuencia de recomendar, sugerir algo o brindar un consejo. Sin embargo, el concepto de **recomendación** tiene un significado más complejo, pues es utilizado en algunos contextos muy específicos. En sentido general, hacer una recomendación implica que una persona sugiere a otra una idea o cuestión. Es una propuesta que se hace para que el otro se beneficie en algún sentido.

1.2.9. Algoritmo de Similitud

Los **algoritmos de similaridad** o **algoritmos de similitud**, son aquellos que ejecutan cálculos matemáticos para determinar los niveles de semejanzas o compatibilidades que existen entre conjuntos de datos. En las bibliotecas de algoritmos de Neo4j[14] existen diversos algoritmos capaces de realizar con facilidad los cálculos de coeficientes de similaridad o similitud con facilidad. (Gra, 2019a)

Existen dos tipos de similitud que son utilizadas por los **algoritmos de similitud**: la similitud sintáctica y la semántica. La similitud sintáctica puede definirse como funciones sobre términos. Mientras que la similitud semántica puede definirse como una variable continua que describe el grado de sinonimia entre dos palabras. (Vargas-Vera y Motta, 2004)

1.2.10. Algoritmo de Recomendación

Un **algoritmo recomendación** es una herramienta que establece un conjunto de criterios y valoraciones sobre los datos de los usuarios para realizar predicciones sobre recomendaciones de elementos que puedan ser de utilidad o valor para el usuario. Estos sistemas seleccionan datos proporcionados por el usuario de forma directa o indirecta, y proceden a analizar y procesar información del historial del usuario para transformar estos datos en conocimiento de recomendación. (Gra, 2019b)

Los **algoritmos de recomendación** en la actualidad tienen un nivel de eficiencia alto, ya que pueden asociar elementos de nuestros perfiles para realizar las recomendaciones, entre ellos, el historial de compras, la selección de contenidos e inclusive nuestras horas de actividad.

Estos recogen toda la actividad de los usuarios en forma de *ratings* (*un valor para cada par usuario – ítem*) o frecuencias (*número de interacciones del usuario con el ítem*) y utilizan estos datos para establecer una posible afinidad entre cada usuario y los *items* que éste desconoce, asignándole una puntuación. De esta forma, se genera una recomendación en forma de *ranking*

[14] Neo4j Graph Platform: https://neo4j.com

donde los primeros *items* son aquellos con mayor puntuación y los últimos, los que tienen una menor. (Marina Pepa et al., 2014)

Existen varios tipos de **algoritmos de recomendación**. Estos dependen de variables principales que son las que condicionan su funcionamiento y varían en cuanto a la forma de predecir los *items* que le puedan interesar a cada usuario. A continuación se describirán los principales algoritmos estudiados y algunas de sus características: (Gra, 2019b)

- Filtrado colaborativo: es muy novedoso ya que genera recomendaciones analizando datos, identificando perfiles y haciendo contraste entre la información del perfil del usuario y la de un colectivo de usuarios. Esto permite al modelo aprender a agrupar perfiles similares y aprender de los datos que recibe de forma general, para desarrollar recomendaciones individuales.
- Filtrado por contenido: son aquellos que tomando en cuenta algunos datos del historial del usuario intentan predecir qué busca el usuario y qué sugerencias similares puede mostrar. Este tipo de sistemas es uno de los que tiene mayor presencia en la actualidad. Con ellos podemos descubrir opciones que se ajusten a las características de los productos o contenidos que hemos disfrutado con anterioridad y elegir elementos similares nuevos.
- Filtrado por popularidad: son implementados principalmente en las ventas de productos o sugerencias concretas. Estos toman como referencia la popularidad del objeto de estudio por una variable principal que puede ser el número de ventas, una característica especial o inclusive una oferta y se muestra de forma general a todos los usuarios que investiguen el área a la que pertenece el objeto. Estos sistemas suelen ser fáciles de implementar y gozan de cierto nivel de efectividad. Su desventaja principal es la imposibilidad de personalizar los criterios de sugerencia para el usuario.

Los **algoritmos de recomendación** según su tipo de funcionamiento, funcionan gracias a algunos algoritmos que optimizan el análisis de los datos para construir las recomendaciones. Uno de los algoritmos más utilizados para regir un sistema de recomendaciones es el conocido como algoritmo de vecinos próximos (kNN). Este determina según los datos proporcionados un patrón de gustos y preferencias y utiliza los datos de un vecino próximo con características similares al inicial y partiendo de estos datos genera las recomendaciones. (Gra, 2019b)

Los datos de las recomendaciones también pueden ser detectados o calculados con otro algoritmo reconocido como es el de correlación de Pearson. Este es un **algoritmo de similaridad** que recolecta los datos de preferencias de los usuarios y determina un peso de similitud

para estimar la relación que existe entre dos usuarios y crear recomendaciones de contenido en base a dicha similaridad.

1.3. Algoritmos y filtros de recomendación

Existe un amplio número de **algoritmos de recomendación**, la mayoría de ellos con uno o varios parámetros y componentes configurables. Por ello, se ha hecho una selección de aquellos más representativos con el objetivo de realizar una comparación entre ellos para identificar sus características y cual se ajusta más a lo necesitado para la propuesta de solución. A continuación se definen con detalle los algoritmos que se han incluido en el estudio realizado.

1.3.1. Filtrado colaborativo

Este tipo de algoritmo se caracteriza porque en la recomendación a un usuario se utiliza conocimiento sobre otros usuarios, de tal forma que los usuarios aprovechan la experiencia unos de otros. Dentro de este filtrado se suelen diferenciar dos subgrupos: los basados en memoria y los basados en modelo. (Adomavicius y Tuzhilin, 2005)

La diferencia entre estos subgrupos radica en que en estos últimos el algoritmo genera una representación propia de los datos, mientras que en los basados en memoria el algoritmo utiliza los datos en crudo en tiempo de recomendación.

Existe una amplia diversidad de algoritmos desarrollados en este tipo de **filtrado colaborativo**, las dos estrategias más extendidas y actualmente exitosas son, en cuanto a los métodos basados en memoria, los algoritmos de vecinos próximos, y entre los basados en modelo, la factorización de matrices. (Marina Pepa et al., 2014) Estos se detallarán y profundizarán en la sección a continuación.

Las ventajas del **filtrado colaborativo** son principalmente dos: buenos niveles de acierto (*en términos de error de predicción de recomendación*), así como de novedad y de diversidad; y puede recomendar *items* de los que no se tenga ninguna descripción, sino sólo *ratings* de usuarios. Sin embargo, tiene la limitación del arranque en frío, es decir, tiene problemas para recomendar a usuarios poco activos o *items* poco conocidos.

1.3.1.1. Vecinos próximos (kNN)

Este método selecciona los ***k*** vecinos más similares al usuario o al *items* objetivo, de forma que mediante la combinación lineal del *rating* de los vecinos se realiza una predicción de *rating*. A partir de esta predicción, se puede obtener el *ranking* de *items* a recomendar, simplemente ordenándolos por orden descendente del *rating* predicho. (Cremonesi et al., 2010)

Los algoritmos de **vecinos próximos** se han desarrollado en dos perspectivas posibles:

- **Basado en usuario (*User kNN*)**

  Se recomiendan al usuario los *items* que han gustado a usuarios similares a éste. Su fórmula es representada de la siguiente manera:

$$f(u,i) = \sum_{\substack{v \in N_k(u) \\ r(v,i) \neq \emptyset}} \text{sim}(u,v) * r(v,i) \tag{1.1}$$

- **Basado en ítem (*Item kNN*)**

  Se recomiendan al usuario los *items* que se parecen a *items* que le han gustado. Su función de *ranking* es muy parecida a la de User kNN:

$$f(u,i) = \sum_{\substack{j \in N_k(i) \\ r(u,j) \neq \emptyset}} \text{sim}(i,j) * r(u,j) \tag{1.2}$$

  Este algoritmo se suele utilizar con el número de vecinos igual al número total de *items*, es decir, que el vecindario de cada *ítem* son todos los demás existentes en el conjunto de datos en cuestión.

En cuanto a la función de similitud entre los usuarios o *items*, se puede realizar mediante varios métodos existentes. A continuación se detalla la similitud por coseno, Jaccard y Pearson.

- **Similitud por coseno**

  Este mide el ángulo entre los vectores (*ratings*) de cada par de usuarios o *items*, de forma que son más similares aquellos cuyos vectores tienen la misma orientación.

$$\text{sim}(u,v) = \frac{\sum_{i:r(u,i)\neq\emptyset, r(v,i)\neq\emptyset} r(u,i) * r(v,i)}{\sqrt{\sum_{i:r(u,i)\neq\emptyset} r(u,i)^2} \sqrt{\sum_{r(v,i)\neq\emptyset} r(v,i)^2}} \tag{1.3}$$

- **Similitud por Jaccard**

  Dos usuarios o *items* son más similares cuanto más parecida sea la intersección a la unión de ambos, es decir, cuantos más *ratings* en común tengan, independientemente del valor de *rating*.

$$\mathrm{sim}(u,v) = \frac{|u \cap v|}{|u \cup v|} = \frac{|u \cap v|}{|u| + |v| - |u \cap v|} \tag{1.4}$$

- **Similitud por Pearson**

  Equivalente a la similitud por coseno, pero cada *rating* se centra en la puntuación promedio del usuario (*o ítem*) correspondiente. Para este caso existen dos versiones dependiendo de la forma de calcular el módulo de v: por intersección, donde se suman los *ratings* de los *items* que tiene en común con u; y total, donde también se incluyen los *ratings* de aquellos que desconoce.

$$\mathrm{sim}(u,v) = \frac{\sum_{i:r(u,i)\neq\emptyset, r(v,i)\neq\emptyset} \left(r(u,i) - \bar{r}_u\right)\left(r(v,i) - \bar{r}_v\right)}{\sqrt{\sum_{i:r(u,i)\neq\emptyset} \left(r(u,i) - \bar{r}_u\right)^2} \sqrt{\sum_{i:r(v,i)\neq\emptyset} \left(r(v,i) - \bar{r}_v\right)^2}} \tag{1.5}$$

  Esta fórmula representa la versión total de Pearson, pero la versión por intersección se diferencia de ésta en que, en el módulo de v que se calcula en el denominador, la condición del sumatorio no es $i : r(v,i) \neq \emptyset$ (*todos los items de* v) sino $i : r(v,i) \neq \emptyset, r(u,i) \neq \emptyset$ (*los items en común de* u *y* v).

Estos tres tipos de similitud tienen a su vez dos versiones, la normalizada y la no normalizada. En donde la normalización se realiza para cada similitud entre usuario e ítem, y consiste en el cociente de la similitud concreta para ese *ítem* y la similitud total del usuario con todos sus *items* conocidos. De esta forma, la similitud obtenida para cada par *usuario – ítem* se encuentra en el rango de valores de preferencia esperado.

1.3.1.2. Factorización de matrices

En los gustos de los usuarios y las características de los *items* se esconde un "*espacio no visible"* que realmente determina por qué les gustan los *items*. Se encuentran formalismos matemáticos que permiten mostrar este tipo de espacio latente como unas cuantas dimensiones reducidas, sin tener que concretar qué son realmente esas dimensiones, y pudiendo sin embargo generar recomendaciones operando con ellas. La idea es representar tanto los *items* como

los usuarios como vectores en un espacio de factores latentes, con una coordenada por factor que representa el grado de afinidad del usuario (*o del ítem*) hacia dicho factor. (Marina Pepa et al., 2014)

Este efecto se puede realizar mediante la **factorización de matrices**, descomponiendo la matriz original de *ratings* en un producto de varias matrices, dos o tres dependiendo del algoritmo, obteniendo siempre una primera matriz de usuarios por factores y una última de factores por *items*.

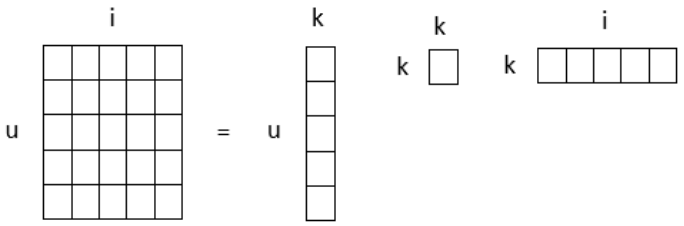

Figura 1.3: Descomposición de una matriz en el producto de otras tres

De esta forma, se obtienen k factores latentes que establecen un espacio de características común, tanto para los usuarios como para los *items*, permitiendo la comparación directa entre ellos. Así, un usuario define un *rating* de acuerdo a sus factores latentes y a los factores latentes del *ítem* en cuestión.

Existen diversas cantidades de algoritmos para obtener la **factorización de matrices**, entre todos ellos en la investigación se trabajó con los siguientes:

- **Probabilistic Latent Semantic Analysis (*pLSA*)**

  Divide la matriz de *ratings* en dos, usuarios por factores y factores por *items*, de forma que su función de *ranking* equivale a la probabilidad de que el usuario puntúe al *ítem* según el espacio de factores latentes. (Hofmann, 1999)

$$f(u,i) = p(i \mid u;\theta) = \sum_{f} p(i \mid f) * p(f \mid u) \tag{1.6}$$

- **Singular Value Decomposition (*SVD*)**

  La factorización de la matriz de *ratings* genera tres matrices, para lo que se tiene en cuenta todos los *ratings*, asignándoles un cero a aquellos no conocidos. Se obtienen los vectores de usuario e *ítem* mediante el producto de las matrices obtenidas según la

fórmula que se indica a continuación, de forma que su función de *ranking* consiste en la multiplicación de los vectores de usuario e *ítem*, centrado en la media del usuario. (Sarwar et al., 2000)

$$\begin{aligned} r &= W * S * F^T \\ f(u,i) &= \bar{r}_u + W\sqrt{S}^T(u) * \sqrt{S}F^T(i) \end{aligned} \tag{1.7}$$

- **SVD No-empty Entries (*SVDN*)**

  Esta es una variante del anterior algoritmo en la que se obtienen dos matrices en lugar de tres, se tiene en cuenta únicamente los *ratings* conocidos, y su función de *ranking* no se centra en la media del usuario. El cálculo de dicha función es más inmediato, ya que los vectores de usuario e *ítem* se corresponden con la fila de la primera matriz y la columna de la segunda respectivamente, por lo que únicamente se multiplican la fila por la columna correspondiente. (Koren, 2008)

$$f(u,i) = U_F(u) * I_F(i) \tag{1.8}$$

- **SVD with Hypergraph transformation (*HSVD*)**

  Este algoritmo está dirigido a la recomendación de *items* a usuarios nuevos, es decir, con poca actividad en el entrenamiento. Por ello, el primer paso del algoritmo consiste en dividir la matriz de *ratings* en tres: *ratings* de los usuarios que no están en entrenamiento, *ratings* conocidos de los usuarios que están en entrenamiento, y *ratings* desconocidos de los usuarios que están en entrenamiento (*lo que se recomendará*). (Pu y Faltings, 2013)

  Se realiza la **factorización de matrices** mediante los algoritmos anteriormente de **SVDR** o **SVDN**, binarizando la matriz de *ratings* y normalizando cada columna de la matriz.

$$\begin{aligned} X_m &= (R_m > 0) \\ \bar{X}_m &= D_e^{-\frac{1}{2}} * X_m * D_v^{-\frac{1}{2}} \\ SVD\left(\bar{X}_m\right) &= W * S * F^T \end{aligned} \tag{1.9}$$

  Utilizando únicamente la matriz de la derecha (*tercera o segunda dependiendo el método usado*), se resuelve mediante *least-squares*[15] la matriz de *ratings* estimados completa, a partir de la cual se realizan las recomendaciones.

[15] Es una técnica de análisis numérico enmarcada dentro de la optimización matemática

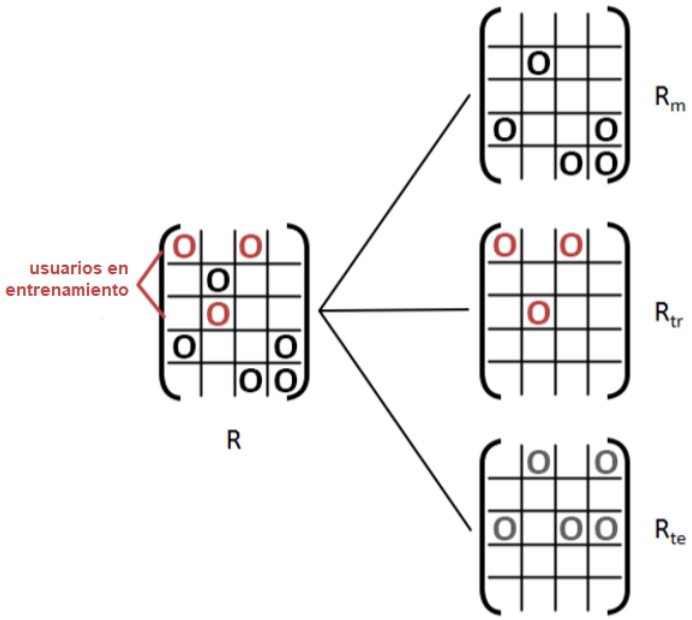

Figura 1.4: Descomposición de una matriz en otras tres según plantea el algoritmo HSVD

Resolver $F\theta = R_{\text{tr}}^T$ mediante mínimos cuadrados para obtener θ.

$$\begin{aligned} \hat{R}_{te} &= \theta^T F^T \\ f(u,i) &= \hat{R}_{te}(u,i) \end{aligned} \tag{1.10}$$

- **Assymetric SVD (*ASVD*)**

  Se trata de la versión asimétrica de **SVD**, que solo usa la tercera matriz de la descomposición para realizar las recomendaciones. Sigue exactamente los mismos pasos que **HSVD**, con la diferencia de que no se binariza la matriz de *ratings* de los usuarios viejos, sino que se utiliza la original con los *ratings* numéricos. En otras palabras, el algoritmo es idéntico excepto que, apoyándose en la figura 1.4, en lugar de usar Xm se usa Rm.

1.3.2. Filtrado por contenido

Con esta denominación se suele hacer referencia a algoritmos que generan recomendaciones mediante la comparación del contenido que describe cada *ítem* y el contenido que le interesa al usuario objetivo. (Adomavicius y Tuzhilin, 2005)

Este grupo de algoritmos se basa en la utilización de la descripción de cada *ítem* para recomendar, sin utilizar información de otros usuarios para generar la recomendación al usuario

objetivo. Así, estos algoritmos intentan recomendar *items* que son parecidos a los que le han gustado al usuario anteriormente. (Marina Pepa et al., 2014)

Frente al **filtrado colaborativo**, los **algoritmos basados en contenido** pueden recomendar *items* nuevos o poco conocidos, aunque tienen en general la misma dificultad que el filtrado colaborativo para usuarios con poca actividad, ya que no se conoce lo suficiente sobre su perfil. La principal limitación de la recomendación basada en contenido es que, recomendando *items* parecidos al perfil del usuario, se puede producir un cierto efecto de encasillamiento para éste usuario, existiendo poca novedad y diversidad en la recomendación propuesta.

De entre los algoritmos de este tipo de filtrado fueron seleccionados para realizar la comparación el Rocchio y kNN, este último similar pero no idéntico al del **filtro colaborativo** (*sólo los diferencia la función de similitud, se explicará en el punto correspondiente*). Estos usan *ratings* como pesos en el cálculo de los centroides y en la suma de las similitudes, respectivamente.

1.3.2.1. Rocchio

Se basa, como ya se ha anticipado, en el cálculo de centroides para cada usuario, de forma que se obtenga un vector "*representante"* para cada uno. Estas clases se corresponderán con las características (*features*) de los *items*, por ejemplo, en *Twitter*, las palabras clave del contenido de los *tweets*. De esta forma, se obtiene para cada usuario un centroide que representa su relación con cada característica (*término*). (Adomavicius y Tuzhilin, 2005)

$$u[f] = \frac{1}{|\mathbf{u}|} \sum_{i:r(u,i)\neq\emptyset} tfidf(f,i) * r(u,i), \text{ donde } \mathbf{u} = \{r(u,i) \neq \emptyset \mid i \in \mathcal{J}\} \tag{1.11}$$

Donde $u[f]$ denota el valor del centroide de usuario para una característica f, y $tfidf(f,i)$ es la importancia que tiene la característica en el *item* y en general, ya que también tiene en cuenta la "*rareza"* de la característica (*una muy común tendrá menos importancia que otra más rara en el item*).

Una vez que se dispone de los centroides, el cálculo de la similitud de los usuarios con cada uno de los *items* se realiza mediante cualquiera de los métodos anteriormente descritos. En este caso se ha seguido la fórmula de similitud mediante coseno:

$$f(u,i) = \text{sim}(u,i) = \frac{\sum_f u[f] * tfidf(f,i)}{\sqrt{\sum_f u[f]^2}\sqrt{\sum_f tfidf(f,i)^2}} \tag{1.12}$$

1.3.2.2. Vecinos próximos (Item kNN)

La estructura de este algoritmo es idéntica a la del **filtro colaborativo** del mismo nombre, pero se diferencian en la forma de calcular la similitud entre los *items*. Mientras el del **filtro colaborativo** utiliza los *ratings* de otros usuarios, este utiliza la descripción de los *items*. Por ejemplo, mientras el primero recomendaría películas siguiendo las puntuaciones de los usuarios, el segundo se basaría en, por ejemplo, el género, la sinopsis, el director, y/o el reparto de cada una de ellas. (Adomavicius y Tuzhilin, 2005)

Las fórmulas entonces se representan prácticamente de la misma forma, pero teniendo en cuenta la diferencia anteriormente descrita.

$$\mathrm{sim}(i,j) = \frac{\sum_f tfidf(f,i) * tfidf(f,j)}{\sqrt{\sum_f tfidf(f,i)^2}\sqrt{\sum_f tfidf(f,j)^2}} \tag{1.13}$$

1.3.3. Filtrado por popularidad

Este algoritmo recomienda *items* sin conocer ningún dato del usuario en donde los recomienda por orden de popularidad y, por tanto, no interesa el *ranking* a cada uno de los usuarios. Por "*popularidad de un ítem"* podemos entenderlo como el número de usuarios que han interactuado con el *ítem*.

$$f(u,i) = |\mathbf{i}|, \text{ donde } \mathbf{i} = \{r(v,i) \neq \emptyset \mid v \in U\} \tag{1.14}$$

Dicho método, que puede parecer trivial a simple vista, por su sencillez, es sin embargo uno de los más extendidos en escenarios reales. Es el que se nos muestra en las listas de "*más vendidos"*, vídeos con más visitas en Youtube, *tweets* más retweeteados, programas de TV con más audiencia, *ranking* de taquilla en cine, *best-sellers*, entre otros.

1.3.4. Filtrado aleatorio

Recomienda *items* de forma aleatoria a cada usuario y su precisión está relacionada con la densidad del conjunto de datos. De igual manera que el anterior recomienda estos *items* sin conocer ningún dato del usuario.

$$\mathrm{sim}(u, i) = \mathrm{random}() \tag{1.15}$$

La efectividad de la **recomendación aleatoria**, medida con una cierta métrica, se puede interpretar como la esperanza del valor de la métrica sobre el conjunto de datos en el que se aplica, y depende generalmente de la densidad de los datos. Es decir, a mayor tasa de pares usuario–*ítem* con *rating* observado, mayor es el valor de la métrica de la recomendación aleatoria.

1.4. Enfoques existentes en la literatura

Durante el estudio de la literatura y las herramientas consultadas se encontraron disímiles aproximaciones a algoritmos de recomendación que podrían ser usados en la propuesta de la solución. A continuación se presentan una síntesis de algunas de ellas, para posteriormente realizar un análisis de ellas:

- **MyMediaLite** [16]

  Es una librería *open source* en C# distribuida bajo los términos de GNU General Public License (GPL). Implementa varios algoritmos de filtrado colaborativo, como kNN (*ítem y usuario*), algoritmos no personalizados (*random, popularidad*) o de factorización de matrices (*WRMF y BPRMF, entre otros*).

- **Apache Mahout** [17]

  Es un proyecto de Apache Software Foundation para producir implementaciones gratuitas de algoritmos de aprendizaje automático distribuidos o escalables enfocados principalmente en álgebra lineal.

- **Graphlab** [18]

[16] MyMediaLite Recommender System Library: http://www.mymedialite.net.
[17] Apache Mahout: https://mahout.apache.org.
[18] Graphlab Create™. Fast, Scalable Machine Learning Modeling in Python: https://turi.com.

El proyecto GraphLab fue iniciado por el Prof. Carlos Guestrin de la Universidad Carnegie Mellon en 2009. Es un proyecto de *open source* que utiliza una licencia Apache. Aunque GraphLab fue originalmente desarrollado para tareas de *machine learning*, ha encontrado un gran éxito en una amplia gama de otras tareas de minería de datos.

- **LibRec** [19]

  Es una biblioteca de Java para sistemas de recomendación (*se requiere la versión 1.7 o superior de Java*). Implementa un conjunto de algoritmos de recomendación de última generación, con el objetivo de resolver dos tareas de recomendación clásicas: la predicción de la calificación y la clasificación de los artículos.

- **LensKit** [20]

  Es un conjunto de herramientas de *open source* para construir, investigar y estudiar sistemas de recomendación. Esta implementada en Java, pero posee ya una nueva versión implementada en Python [21].

- **MLlib** [22]

  Es la biblioteca de *machine learning* escalable de Apache Spark, con APIs en Java, Scala, Python y R. Se desarrolla como parte del proyecto Apache Spark, por lo tanto, se prueba y actualiza con cada lanzamiento de Spark.

- **Recommender Extension** [23]

  Recommender Extension es un código fuente desarrollado en el Instituto Rudjer Boskovic como parte del proyecto e-LICO FP7. Es una extensión de RapidMiner que integra varios operadores de recomendación en RapidMiner. La extensión se basa en la biblioteca del sistema de recomendación MyMediaLite, la cual fue convertida manualmente de C# a Java plugin de RapidMiner.

Luego del estudio de literatura y herramientas se puede descartar Recommender Extension, MyMediaLite y Graphlab, debido a que los dos primeros están implementados en C# y el último en Python; mientras que el sistema objetivo final (*SIGEDIC*) al que va destinado el algoritmo de recomendación esta desarrollado utilizando lenguajes (*Groovy*[24]) y tecnologías (*Grails*[25]) que no permiten la incorporación de estos. Por otro lado el MLlib trae consigo una

[19] LibRec - A Java Library for Recommender Systems: https://guoguibing.github.io/librec/index.html.
[20] LensKit Recommender Toolkit: https://java.lenskit.org.
[21] LensKit for Python: hhttps://lenskit.org.
[22] MLlib | Apache Spark: https://spark.apache.org/mllib.
[23] Recommender Extension: http://www.e-lico.eu/recommender-extension.html.
[24] Groovy es un lenguaje de programación orientado a objetos implementado sobre la plataforma Java. Tiene características similares a Python, Ruby, Perl y Smalltalk.
[25] Grails es un framework para aplicaciones web libre desarrollado sobre el lenguaje de programación Groovy.

gran variedad de algoritmos y utilidades, pero sin embargo, algoritmos de recomendación solo posee uno, el de alternar los mínimos cuadrados (*ALS*[26]), el cual no es el más apropiado para la propuesta de solución.

Con lo anterior expuesto quedarían LibRec y LensKit como posibles propuestas a utilizar para resolver el problema planteado por la investigación. Estos resultantes están implementados en Java, por lo que es sencillo su integración al sistema; además entre sus algoritmos se encuentran los de filtrado por contenido, que son los mayores candidatos a la solución final. Cabe destacar que los dos son de código abierto, lo cual permite modificar sus métodos para adaptarlos de acuerdo a lo necesitado, o incluso llegar a juntar las dos propuestas para crear una mejor recomendación de los *items* a los usuario.

1.5. Conclusiones parciales

Apoyándose en la información contenida en este capítulo se concluye:

- que el análisis bibliométrico y documental demuestra que la investigación está sustentada un 44 % en bibliografías recientes;
- que la descripción de los principales conceptos y definiciones asociados al dominio del problema logran avalar la investigación y permiten profundizar en la comprensión de la posterior propuesta de solución;
- que el análisis de los algoritmos de recomendación existentes brindan varios tipos de filtrado: colaborativo, por contenido, por popularidad, y aleatorio; en donde luego del estudio se define que filtro a utilizar será por contenido, debido a las características del mismo, las cuales se asemejan a la que se plantea como problema a resolver;
- además, el estudio de las herramientas y librerías con una aproximación al problema planteado concluyó que existen disímiles propuestas pero no todas pueden ser implementadas o incorporadas al sistema objetivo final, quedando como resultantes las librerías LibRec y LensKit.

[26] Alternating Least Squares por sus siglas en inglés.

Capítulo 2

Fundamentación de la propuesta de solución

En el presente capítulo se describen los aspectos técnicos principales, los procedimientos seguidos y la implementación de la propuesta de solución desarrollada para recomendar textos científicos a partir de su contenido. La misma consta de subsecciones o etapas por las que el algoritmo debe pasar para obtener una serie de *items* a recomendar al usuario, entre ellos, la carga y procesamiento de datos, obtención de temáticas introducidas por el usuario, el filtrado por contenido de los datos obtenidos en el dataset, y la visualización de los *items* a recomendar.

2.1. Descripción general de la propuesta

El algoritmo propuesto para recomendar textos científicos basado en su contenido permite cumplir el objetivo de la investigación y visualizar, al finalizar, una serie de *items*, en este caso textos científicos, que el usuario utilizará para la creación de su página en la biblioteca digital nombrada SIGEDIC. Para lograr el resultado final, el algoritmo sigue el siguiente procedimiento que se describe en las subsecciones a continuación:

1. Carga de datos.
2. Preprocesamiento de datos.
3. Obtención de temáticas.
4. Filtrado de datos.

5. Visualización de recomendaciones.

Para una mayor comprensión del procedimiento a seguir se muestra en la figura siguiente el proceso general para el algoritmo de recomendación de textos científicos basado en su contenido.

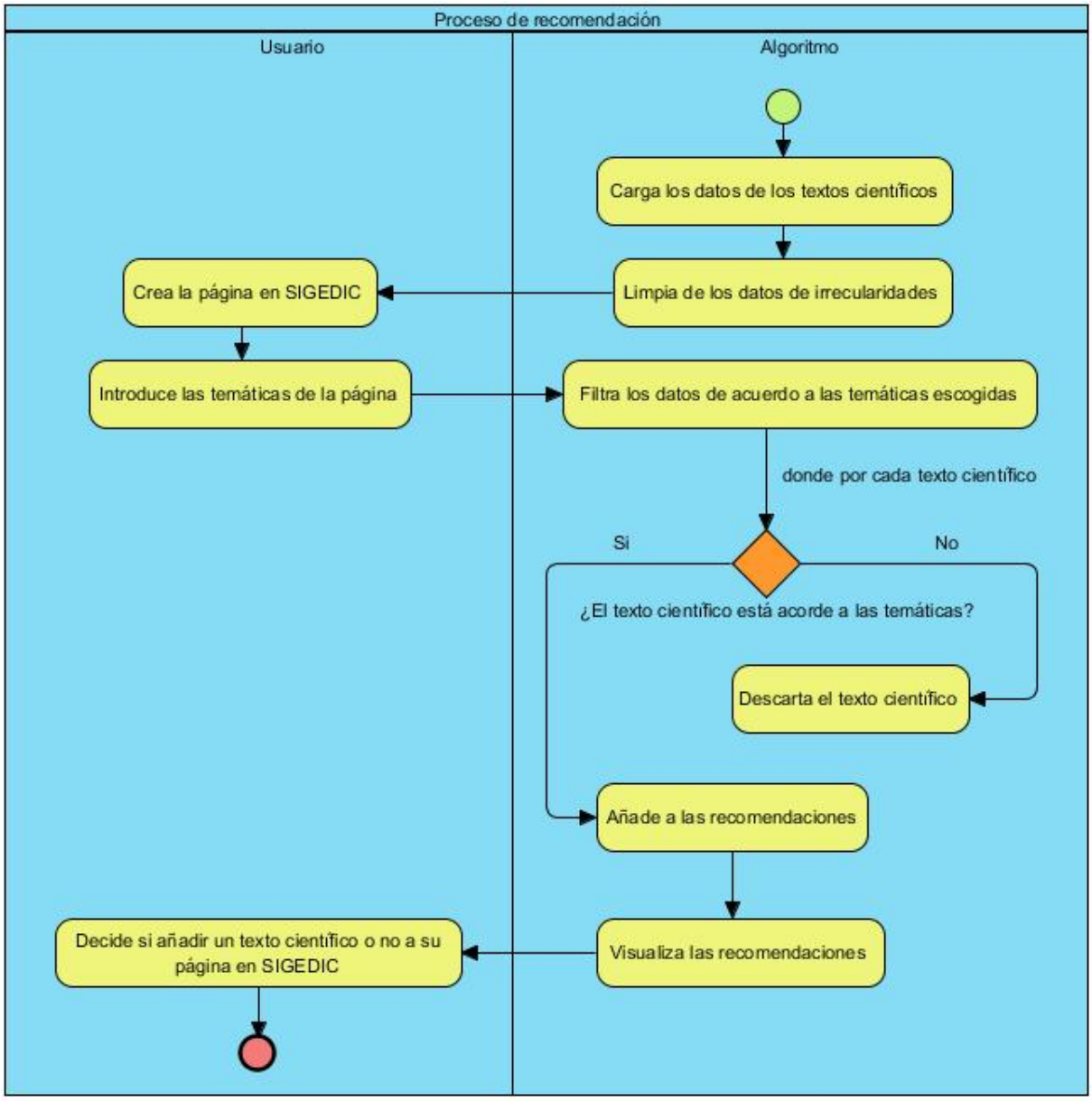

Figura 2.1: Proceso general para el algoritmo de recomendación

2.1.1. Carga de datos

Los datos de los textos científicos, en específico sus principales metadatos[27], en la investigación se encuentran almacenados en una base de datos externa al sistema SIGEDIC. Esta es una base de datos relacional orientada a objetos, manejada por PostgreSQL, un poderoso sistema de base de datos relacional de código abierto con más de 30 años de desarrollo activo que ha ganado una sólida reputación por su confiabilidad, solidez de funciones y rendimiento. La base es nombrada "*metharto"* y se conforma de 7 tablas: *author*, *record*, *journal*, *set*, *metadata_format*, *author_record*, *journal_set*. A continuación se describen las columnas y datos que almacenan estas tablas.

- ***author***: Almacena información sobre los autores: nombre y afiliación.
- ***record***: Almacena la información sobre los registros de textos científicos: título, resumen, año de publicación, fuente, entre otros.
- ***journal***: Almacena la información acerca de las revistas científicas y académicas de donde provienen los metadatos.
- ***set***: Almacena la información relacionada con las colecciones de registros bibliográficos.
- ***metadata_format***: Almacena el formato que deben tener los metadatos.
- ***author_record***: Almacena la relación entre las tablas *author* y *record*.
- ***journal_set***: Almacena la relación entre las tablas *journal* y *set*.

[27] Los metadatos son datos que describen otros datos.

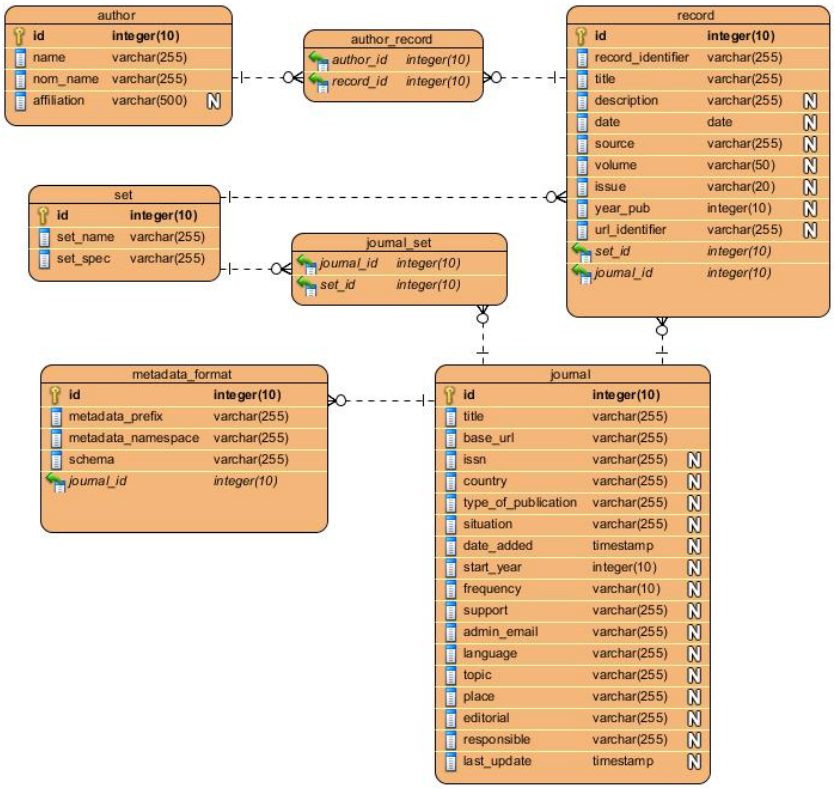

Figura 2.2: Modelo de la base de datos "*metharto*"

Luego del estudio de la base de datos se determina que las columnas necesarias para el algoritmo son: *title* y *description*, las dos pertenecientes a la tabla *record* que almacena los principales datos de los textos científicos. Se decide entonces guardar esta información de dichas columnas de la tabla en un archivo CSV[28]. Estos son un tipo de documento en formato abierto sencillo para representar datos en forma de tabla, en las que las columnas se separan por comas (,) y las filas por saltos de línea.

Para ello se utiliza SQL[29], un lenguaje de dominio específico utilizado en programación, diseñado para administrar, y recuperar información de sistemas de gestión de bases de datos relacionales. Con el mismo haciendo uso de la herramienta para ejecutar sentencias SQL en el PostgreSQL, se utiliza el la siguiente sentencia que permite guardar en un archivo CSV la información requerida ordenada de forma ascendente por el *title*.

[28] Comma-Separated Values por sus siglas en inglés.
[29] Structured Query Language por sus siglas en inglés.

```
SQL >> COPY (SELECT r.id, title, description FROM record r INNER JOIN author_record ar ON r.id = ar
    .record_id INNER JOIN author a ON a.id = ar.author_id ORDER BY title) TO 'D:\\file.csv'
    DELIMITER ',' CSV HEADER;
```

Figura 2.3: Sentencia SQL para obtener el dataset en CSV

Como resultado de la sentencia te obtiene el siguiente **dataset**[30] almacenado en un formato CSV. En él, *id* representa el identificador en el sistema del texto científico; *title*, el título del texto científico; y *abstract*, el resumen correspondiente al texto científico.

```
id,title,abstract
3265,Alteraciones de la superficie ocular en pacientes con VIH/SIDA,"Describir las
principales manifestaciones clínicas del VIH/SIDA..."
```

Figura 2.4: Dataset en formato CSV

2.1.2. Preprocesamiento de datos

El preprocesamiento de datos es una etapa esencial del procedimiento, debido a que se encarga de la limpieza de datos, su integración, transformación y reducción para los siguientes pasos, los cuales necesitan de un conjunto de datos que sea de calidad y utilidad para el algoritmo (Herrera, 2016).

La preparación de datos está formada por una serie de técnicas que tienen el objetivo de inicializar correctamente los datos que servirán de entrada para el algoritmo. Este tipo de técnicas pueden clasificarse como de uso obligatorio, ya que sin ellas el algoritmo no podría ejecutarse u ofrecería resultados erróneos (Herrera, 2016).

Entre estas técnicas se encuentran la transformación de datos y normalización, integración, limpieza de ruido e imputación de valores perdidos; estas en su mayoría para la preparación de los datos.

[30] Un conjunto de datos o dataset corresponde a los contenidos de una única tabla de base de datos.

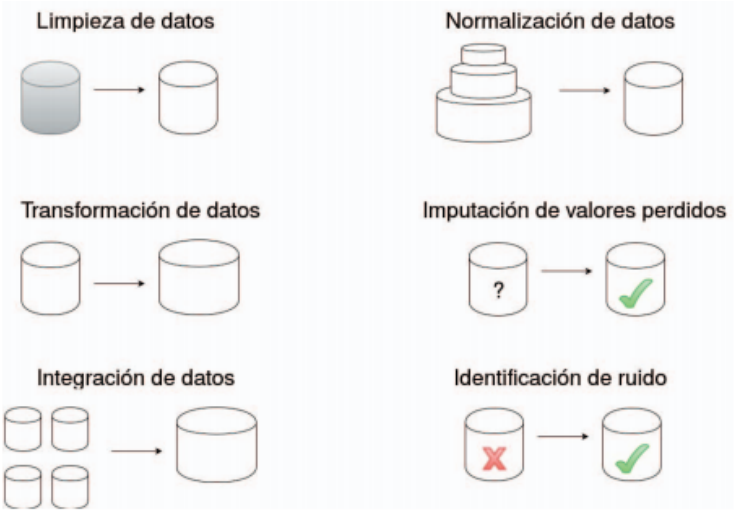

Figura 2.5: Técnicas de preparación de datos

Mientras que para las técnicas de reducción las más relevantes son: la selección de atributos (*Feature Selection (FS) en inglés*), la selección de instancias (*Instance Selection (IS) en inglés*) o la discretización.

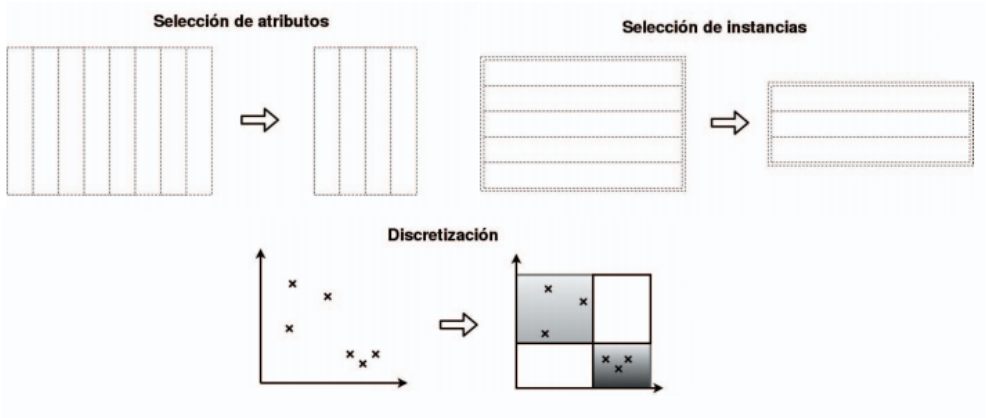

Figura 2.6: Técnicas de reducción de datos

En el estudio realizado al dataset se encuentra que existen filas donde *title* y *abstract* se repiten, para resolver esto se añade a la sentencia SQL original el comando **GROUP BY r.id, title, description** para agrupar las filas con id, títulos y resumen con igual valor.

Se aprecian también algunas filas con *abstract* con valor nulo o con un punto (.). Con el objetivo de resolver esto se añade a la sentencia SQL el comando **WHERE r.description != ‘ ’ AND r.description != ‘ . ’ AND LENGTH(r.description)** $>$ **3**, con lo cual no se obtendrán filas en las que ocurra dicho problema.

Como resultado de lo anterior se obtiene la siguiente sentencia SQL:

```
SQL >> COPY (SELECT r.id, title, description FROM record r INNER JOIN author_record ar ON r.id = ar.record_id INNER JOIN author a ON a.id = ar.author_id WHERE r.description != '' AND r.description != '.' AND LENGTH(r.description) > 3 GROUP BY r.id, title, description ORDER BY title) TO 'D:\\file.csv' DELIMITER ',' CSV HEADER;
```

Figura 2.7: Sentencia SQL para obtener el dataset en CSV más limpio

Luego de tener un dataset limpio se hace necesario que el mismo contenga, además de las columnas ya existentes, otra denominada *keyword*, que no son más que las palabras clave de los textos científicos. Esto se debe a que el algoritmo como entrada debe recibir estas palabras clave junto al título y el resumen. Debido a que la base de datos utilizada no contiene dicha columna se procede a aplicar sobre la columna *abstract* un algoritmo de extracción de palabras clave.

Para ello se creará un script[31] en el lenguaje Python, un lenguaje de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Dicho lenguaje tiene gran cantidad de paquetes y/o módulos de terceros hospedados en Python Package Index (PyPI), un repositorio bastante amplio.

Entre sus paquetes se encuentra el que se utilizará, llamado **rake-nltk**, que es una implementación del algoritmo de extracción automática rápida de palabras clave (Rapid Automatic Keyword Extraction en inglés, siendo Rake el acrónimo) utilizando NLTK[32]. Según su página oficial en PyPI, es un algoritmo de extracción de palabras clave independiente del dominio que intenta determinar frases clave en un cuerpo de texto mediante el análisis de la frecuencia de aparición de palabras y su coexistencia con otras palabras en el texto (rak, 2020). Se utiliza además, otro paquete llamado **pandas**, que es una herramienta de análisis y manipulación de datos de código abierto rápida, potente, flexible y fácil de usar, construido sobre el lenguaje de programación Python (pan, 2020).

El script tendrá como entrada el dataset en formato CSV y al concluir devolverá como salida otro dataset pero incluyendo la columna *keyword*, en este caso en dos formatos: CSV y

[31] Un script, secuencia de comando o guion es un término informal que se usa para designar a un programa relativamente simple.

[32] NLTK es un conjunto de bibliotecas y programas para el procesamiento del lenguaje natural simbólico y estadísticos para el lenguaje de programación Python.

JSON[33] [34].

Figura 2.8: Flujo de datos del script

Primeramente se carga el dataset en CSV y con él se crea un nuevo dataframe, la estructura de datos con la que trabaja **pandas**, este nuevo dataframe será la salida del script. En el fragmento de código **pd** representa la instancia de **pandas**.

```
# Load all data from CSV
data_frame_from_query = pd.read_csv('query.csv')

# Create the new DataFrame
data_frame_to_export = pd.DataFrame(columns=['id', 'title', 'abstract', 'keywords'])
```

Figura 2.9: Carga y creación de dataframe

Luego se inicializa el *Rake*, el lenguaje y las *stopword*, que estas últimas no son más que palabras vacías que son poco frecuentes y no representan información importante para la extracción de las palabras clave, ejemplos de estas se encuentran: pronombres, preposiciones, conjunciones, artículos, entre otros.

```
# Download the language
nltk.download('europarl_raw')

my_stop_word = get_stop_words('spanish')

rake = Rake(
    min_length=1,
    max_length=2,
    stopwords=my_stop_word,
    language=spanish,
    ranking_metric=Metric.DEGREE_TO_FREQUENCY_RATIO
)
```

Figura 2.10: Inicialización de Rake y stopword

[33] JavaScript Object Notation por sus siglas en inglés.
[34] JSON es un formato de texto sencillo para el intercambio de datos.

Luego de estar preparado todo, se prosigue a iterar sobre el dataframe $data\_frame\_from\_query$ para aplicar el algoritmo de extracción de palabras clave (*Rake*) al resumen de cada elemento del dataframe. A su vez, se crean y añaden estos elementos junto a sus palabras clave al nuevo dataframe creado anteriormente, $data\_frame\_to\_export$.

```
for index, row in data_frame_from_query.iterrows():
    id_index = row['id']
    title_index = row['title']
    abstract_index = row['abstract']

    # Extraction given the text.
    rake.extract_keywords_from_text(row['abstract'])

    # To get keyword phrases ranked highest to lowest.
    keywords_index = rake.get_ranked_phrases()
        # Create the keywords string
        keywords_index_text = ""
        for i in range(5):
            if i < len(keywords_index):
                if i == 0:
                    keywords_index_text += keywords_index[i]
                else:
                    keywords_index_text += (", " + keywords_index[i])
            else:
                break

        # Add to DataFrame
        data_frame_to_export.loc[index] = [id_index, title_index, abstract_index, ↘
            keywords_index_text]

    # Show progress
    print("{:.0%}".format((index + 1) / len(data_frame_from_query)) + " -> " + str(index + 1) + " ↘
        of " + str(
        len(data_frame_from_query)))
```

Figura 2.11: Iteración sobre el dataframe origen y aplicación del algoritmo Rake

Para la salida del script se ejecutan los métodos $to\_csv()$ y $to\_json()$, pertenecientes a **pandas**, con el cual se exportan los datos a formato CSV el primero y a JSON el segundo.

```
# Export to CSV
data_frame_to_export.to_csv('dataset.csv', index=True, header=True)

# Export to JSON
data_frame_to_export.to_json('dataset.json', orient='records')
```

Figura 2.12: Salida del script, creación del dataset con las palabras clave

Como resultado de esto se obtienen los siguientes archivos:

```
id,title,abstract,keywords
3265,Alteraciones de la superficie ocular en pacientes con VIH/SIDA,"Describir las principales manifestaciones clínicas del VIH/SIDA...","variantes atípicas, superficie ocular, periodo comprendido, VIH/SIDA"
```

Figura 2.13: Nuevo dataset en formato CSV

```
[
    ...
    {
        "id":3265,
        "title":"Alteraciones de la superficie ocular en pacientes con VIH\/SIDA",
        "abstract":"Objetivo: describir las principales manifestaciones cl\u00ednicas del VIH\/SIDA...",
        "keywords":"variantes at\u00edpicas, superficie ocular, periodo comprendido, VIH\/SIDA"
    }
    ...
]
```

Figura 2.14: Nuevo dataset en formato JSON

El posterior estudio y análisis del nuevo dataset con las palabras clave, muestra que existen filas en las que el algoritmo *Rake* no extrajo las palabras clave del resumen o solo extrajo una sola palabra clave. Por lo tanto se procede a realizar otra limpieza de los datos para eliminar las filas del dataset que no posean la columna *keywords* con elementos válidos.

Para ello se añade en la iteración una condicional luego de que el algoritmo Rake extraiga las palabras clave. En este paso se verifica si $keywords\_index\_text$, es la lista de palabras clave que devuelve el método $get\_ranked\_phrases()$ es mayor o igual que 2.

```
# Check if keywords is null
if len(keywords_index) >= 2:
    # Create the keywords string
    keywords_index_text = ""
    for i in range(5):
        if i < len(keywords_index):
            if i == 0:
                keywords_index_text += keywords_index[i]
            else:
                keywords_index_text += (", " + keywords_index[i])
        else:
            break

    # Add to DataFrame
    data_frame_to_export.loc[index] = [id_index, title_index, abstract_index, keywords_index_text]
else:
    # Print a error message
    print('Week keywords')
```

Figura 2.15: Verificación de palabras clave débiles o nulas en la iteración

Como resultado del preprocesamiento se obtiene un dataset limpio y transformado para los próximos pasos del procedimiento. De esta forma se logra mejorar la calidad del dataset para tener una mayor y mejor información con el algoritmo de recomendación basado en contenido.

2.1.3. Obtención de temáticas

En esta etapa del procedimiento, se diseña una interfaz la cual será la encargada de obtener los datos las temáticas y el nombre de la página que el usuario desea crear. Para ello se utiliza el lenguaje Groovy y el framework de desarrollo de aplicaciones web, Grails; estos son los utilizados por el sistema SIGEDIC y de ahi que sean los escogidos para la lógica de la interfaz de usuario.

Para mostrar y diseñar las vistas en Grails, se utiliza *Groovy Server Pages* o GSP por sus siglas en inglés, una tecnología de renderizado de vistas del lado del servidor basada en Groovy. Esta tiene las típicas etiquetas del lenguaje HTML (HyperText Markup Language por sus siglas en inglés.), pero incorpora otras propias para la comunicación entre la vista y el controlador, como por ejemplo: $< g : form >< /g : form >$, $< g : if >< /g : if >$, $< g : each >< /g : each >$, entre otras.

Para la introducción del nombre de la página se añade un campo de texto junto a un botón: 'Definir nombre'. Al presionar el botón el valor del campo de texto se envía a la clase controladora denominada $PagesController.groovy$ y se almacena temporalmente hasta que haya sido creada la página.

```
1   <g:form controller="pages" action="name" class="form-inline">
2       <div class="col-7">
3           <div class="input-group">
4               <input type="text" name="name" id="name" class="${classesName}" value="↘
                    ${name}"
5                       placeholder="Nombre de la pagina"/>
6               <g:if test="${message}">
7                   <div class="invalid-feedback">
8                       ${message}
9                   </div>
10              </g:if>
11          </div>
12      </div>
13
14      <div class="col-5">
15          <button class="btn btn-primary" type="submit">
16              <i class="fa fa-check"></i> Definir nombre
```

```
                        </button>
                    </div>
                </g:form>
```

Figura 2.16: Código en GSP del botón para definir el nombre de la página

```
class PagesController {
    String name

    def name(){
        if (params.name == null || params.name.length() == 0) {
            def thematicCount = thematicList.size()
            render(view: '/pages/index', model: [thematicList: thematicList, thematicCount: thematicCount, message: 'No puede ser vacio', classes: 'form-control', classesName: 'form-control is-invalid', name: name])
            return
        }
        name = params.name
        def thematicCount = thematicList.size()
        render(view: '/pages/index', model: [thematicList: thematicList, thematicCount: thematicCount, message: null, classes: 'form-control', classesName: 'form-control', name: name])
    }

}
```

Figura 2.17: Función en Groovy del botón para almacenar el nombre de la página en la controladora

De igual forma para introducir las temáticas, se añade un campo de texto junto a un botón con un icono de **+**. Al presionar en el botón de **+** se envía el valor del campo de texto hacia la controladora, la cual lo almacena en una lista de *string*: $List < String > thematicList$ y posteriormente lo muestra en un listado.

```
                <ul class="list-group">
                    <g:if test="${thematicList.size() == 0}">
                        <i>No hay tematicas aun</i>
                    </g:if>

                    <g:if test="${thematicList.size() > 0}">
                        <g:each in="${thematicList}" var="thematic">
                            <li class="list-group-item">
                                <div class="row">
                                    <div class="col-10">
                                        <b>#</b>${thematic.encodeAsHTML()}
                                    </div>

```

```
                                    <div class="col-2">
                                        <g:form controller="pages" action="delete">
                                            <button type="submit" class="close" aria-label="
                                                Close">
                                            <span aria-hidden="true">&times;</span>
                                            </button>
                                            <input type="hidden" name="thematic" id="delete" 
                                                value="${thematic}"/>
                                        </g:form>
                                    </div>
                                </div>
                            </li>
                        </g:each>
                    </g:if>
                </ul>
```

Figura 2.18: Código en GSP de la lista de temáticas introducidas

```
class PagesController {
    List<String> thematicList = new ArrayList<>()

    def add() {
        if (params.thematic == null || params.thematic.length() == 0) {
            def thematicCount = thematicList.size()
            render(view: '/pages/index', model: [thematicList: thematicList, thematicCount: 
                thematicCount, message: 'No puede ser vacio', classes: 'form-control is-invalid', 
                classesName: 'form-control is-invalid', name: name])
            return
        }
        thematicList.add(params.thematic)
        def thematicCount = thematicList.size()
        render(view: '/pages/index', model: [thematicList: thematicList, thematicCount: 
            thematicCount, message: null, classes: 'form-control', classesName: 'form-control', 
            name: name])
    }
}
```

Figura 2.19: Función en Groovy del botón para almacenar y listar las temáticas en la controladora

Como se observa, en el código GSP se provee la posibilidad de poder eliminar alguna temática ya añadida, por ello se añade a cada elemento una **x** que la elimina de la lista. A continuación se muestra el método que permite eliminarlas en la clase controladora.

```
class PagesController {
    List<String> thematicList = new ArrayList<>()

    def delete() {
```

```
5           thematicList.remove(params.thematic)
6           def thematicCount = thematicList.size()
7           render(view: '/pages/index', model: [thematicList: thematicList, thematicCount: ↘
                thematicCount, message: null, classes: 'form-control', classesName: 'form-control', ↘
                name: name])
8       }
9   }
```

Figura 2.20: Función en Groovy de eliminar una temática en la controladora

Todo lo anterior da como resultado la siguiente vista funcional, con la cual el usuario será capaz de introducir los datos que serán utilizados en la posterior etapa del procedimiento:

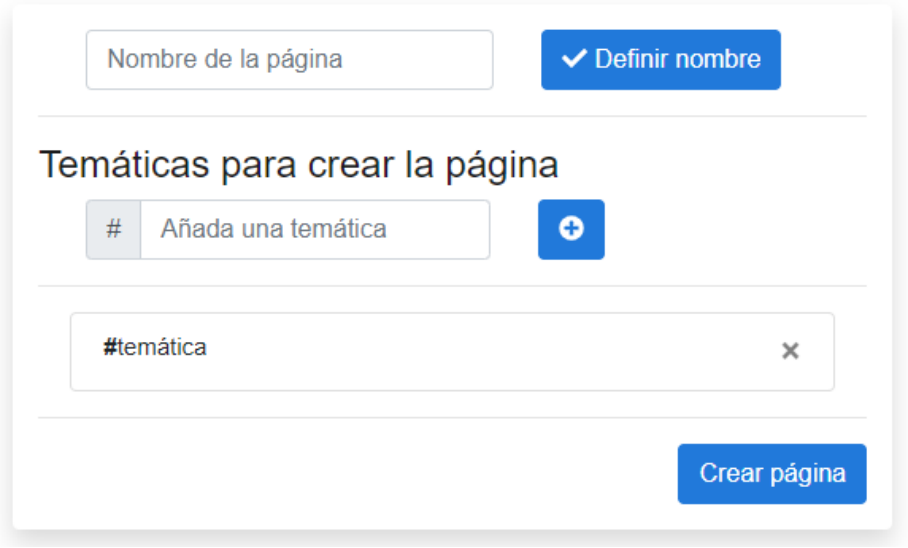

Figura 2.21: Vista de nombre y temáticas de la página a crear por el usuario

2.1.4. Filtrado de datos

Texto

2.1.5. Visualización de recomendaciones

Texto

2.2. Conclusiones parciales

Capítulo 3

Validación y análisis de la solución

Se debe dejar bien claro el objetivo del capítulo así como los métodos científicos empleados.

3.1. Diseño experimental

Se debe diseñar experimentos para validar que se ha dado solución al problema de investigación. Aquí juegan un rol importante las variables y su operacionalización, elementos definidos en la introducción de la tesis.

3.2. Conclusiones parciales

Conclusiones

Al finalizar la presente investigación, se arriba a las siguientes conclusiones:

- ...

Recomendaciones

A partir del estudio realizado se propone las siguientes recomendaciones ...

Glosario de términos

Deep Learning Es un conjunto de algoritmos de aprendizaje automático que intenta modelar abstracciones de alto nivel en datos usando arquitecturas computacionales que admiten transformaciones no lineales múltiples e iterativas de datos expresados en forma matricial o tensorial.

Filosofía del Lenguaje Es la rama de la filosofía que estudia el lenguaje en sus aspectos más generales y fundamentales, como la naturaleza del significado y de la referencia, la relación entre el lenguaje, el pensamiento y el mundo, el uso del lenguaje, la interpretación, la traducción y los límites del lenguaje.

Internet Es un conjunto descentralizado de redes de comunicación interconectadas que utilizan la familia de protocolos TCP/IP, lo cual garantiza que las redes físicas heterogéneas que la componen, constituyan una red lógica única de alcance mundial.

Java Es el nombre de un entorno o plataforma de computación originaria de Sun Microsystems, capaz de ejecutar aplicaciones desarrolladas usando el lenguaje de programación Java u otros lenguajes que compilen a bytecode y un conjunto de herramientas de desarrollo.

Lingüística Es el estudio científico del origen, la evolución y la estructura del lenguaje, a fin de deducir las leyes que rigen las lenguas.

Machine Learning Es el subcampo de las ciencias de la computación y una rama de la inteligencia artificial, cuyo objetivo es desarrollar técnicas que permitan que las computadoras aprendan.

Monografía Estudio detallado sobre un aspecto concreto y particular de una materia acotada.

Neuropsicología Es una disciplina y especialidad clínica, que converge entre la neurología y la psicología.

Sitio Web Es una o varias páginas web relacionadas y comunes a un dominio de internet o subdominio en la World Wide Web dentro de Internet.

Software Conjunto de programas y rutinas que permiten a la computadora realizar determinadas tareas.

Glosario de símbolos

U	Conjunto de usuarios existentes en el conjunto de datos con el que se está trabajando.
I	Conjunto de ítems que se disponen para recomendar a los diferentes usuarios.
R	Ránking de los ítems a recomendar a un usuario en particular.
r	Matriz de ratings donde se encuentran las puntuaciones que los usuarios han dado a ciertos ítems.
$r(u, i)$	Puntuación del usuario u al ítem i.
$f(u, i)$	Función de ránking, calcula la puntuación del ítem i para el usuario u.
$\mathrm{sim}(u, v)$	Similitud entre dos usuario u y v calculada mediante la funciń de similitud correspondiente.
$\mathrm{sim}(i, j)$	Similitud entre dos ítems i y j calculada mediante la funciń de similitud correspondiente.
$N_k(i)$	Conjunto de los k vecinos más similares a un ítem objetivo.
$N_k(u)$	Conjunto de los k vecinos más similares a un usuario objetivo.
$u[f]$	Valor del centroide de usuario para una característica f.

Abreviaturas

SIGEDIC	Sistema de Gestión de Datos e Información Científica
UCI	Universidad de Ciencias Informáticas
TIC	Tecnologías de la Información y las Comunicaciones
IEEE	Institute of Electrical and Electronics Engineers
NLP	Natural Language Processing
IA	Inteligencia Artificial
kNN	k-Nearest Neighbors

Referencias Bibliográficas

Algoritmos de Similiridad | Conoce los algoritmos de grafos. 2019a. URL https://www.grapheverywhere.com/algoritmos-de-similaridad.

Qué son los algoritmos de recomendación - ideas frescas. 2019. URL http://ideasfrescas.com.mx/que-son-los-algoritmos-de-recomendacion.

Sistemas de recomendación - qué son, tipos y ejemplos. 2019b. URL https://www.grapheverywhere.com/sistemas-de-recomendacion-que-son-tipos-y-ejemplos.

Concepto de recomendación - definición en deconceptos.com. 2020. URL https://deconceptos.com/ciencias-sociales/recomendacion.

Definición de recomendación - qué es y concepto. 2020. URL https://definicion.mx/recomendacion.

pandas - Python Data Analysis Library. 2020. URL https://pandas.pydata.org.

rake-nltk. 2020. URL https://pypi.org/project/rake-nltk.

What is artificial intelligence? how does ai work? | built in. 2020. URL https://builtin.com/artificial-intelligence.

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

William Y Arms, Christophe Blanchi, y Edward A Overly. An architecture for information in digital libraries. *D-lib magazine*, 3(2), 1997. URL http://mirror.dlib.org/dlib/february97/cnri/02arms1.html.

Paul Brassard, Gilles; Bratley. *Fundamentos de Algoritmia*. PRENTICE HALL, 1997. ISBN 848966000X.

Augusto Cortez Vásquez, Hugo Vega huerta, Jaime Pariona Quispe, y Ana Maria Huayna. Procesamiento de lenguaje natural. *Revista de investigación de Sistemas e Informática*, 6(2):45 – 54, 2014. URL https://revistasinvestigacion.unmsm.edu.pe/index.php/sistem/article/view/5923.

Paolo Cremonesi, Yehuda Koren, y Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. En *Proceedings of the fourth ACM conference on Recommender systems*, págs. 39–46. 2010.

I. Domínguez García. *Comunicación y texto*. Editorial Pueblo y Educación, 2010.

Alexander Gelbukh. Procesamiento de lenguaje natural y sus aplicaciones. *Komputer Sapiens*, 1:6–11, 2010.

HENRY Gladney, Edward A Fox, Zahid Ahmed, Ron Ashany, Nicholas J Belkin, y Maria Zemankova. Digital library: Gross structure and requirements: Report from a march 1994 workshop. tomo 94, págs. 101–107. 1994.

Yoav Goldberg. A primer on neural network models for natural language processing. *J. Artif. Intell. Res.*, 57:345–420, 2016. ISSN 1076-9757. doi:10.1613/jair.4992.

Andrés González. Sistemas de recomendación de contenido con machine learning – cleverdata. https://cleverdata.io/sistemas-recomendacion-machine-learning/.

Francisco Herrera. Big data: Preprocesamiento y calidad de datos. *novática*, (237):17, 2016.

Pedro Hípola, Benjamín Vargas-Quesada, y José A Senso. Bibliotecas digitales: situación actual y problemas. *El profesional de la información*, 9(4):4–13, 2000.

Thomas Hofmann. Probabilistic latent semantic indexing. En *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, págs. 50–57. 1999.

Ernesto Ortiz Muñoz Ing. Yusniel Hidalgo Delgado. *Detección de comunidades a partir de redes de coautoría en grafos RDF*. Tesis de pregrado, Universidad de las Ciencias Informáticas, 2015.

Luis Enrique Alonso Sierra Ing. Yusniel Hidalgo Delgado. *Desambiguación del nombre de los autores en metadatos bibliográficos usando técnicas de agrupamiento*. Tesis de pregrado, Universidad de las Ciencias Informáticas, 2014.

Deepack Jakhar y Ishmeet Kaur. Artificial intelligence, machine learning and deep learning: definitions and differences. *Clinical and experimental dermatology*, 45(1):131–132, 2020.

Jaywrkr. Guía para construir un sistema de recomendación (parte 1 - 2 - 3 - 4). *Medium*, 2019. ISSN 2914-7810. URL https://medium.com/@jaywrkr/gu%C3%ADa-para-construir-un-sistema-de-recomendaci%C3%B3n-parte-1-2b1a65d6eac3.

Lyons John. *Essays in linguistic theory. 1, Natural language and universal grammar*. Cambridge, Cambridge University Press, 1991. ISBN 978-0521246965.

Ameet V Joshi. Recommendations systems. En *Machine Learning and Artificial Intelligence*, págs. 199–204. Springer, 2020.

Mounir A& Khalil y Raja Jayatilleke. Digital libraries: their usage from the end user point of view. En *NATIONAL ONLINE MEETING*, tomo 21, págs. 179–188. Information Today, Inc; 1999, 2000.

Yehuda 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*, págs. 426–434. 2008.

Mohamed Zakaria Kurdi. *Natural language processing and computational linguistics: speech, morphology and syntax*, tomo 1. John Wiley & Sons, 2016. ISBN 978-1848218482.

Mohamed Zakaria Kurdi. *Natural language processing and computational linguistics 2: semantics, discourse and applications*, tomo 2. John Wiley & Sons, 2017. ISBN 978-1848219212.

Rodrigo Lastra. Inducción del sentido de las palabras para el idioma español. 2019.

Ricardo Peña Marí. *De Euclides a Java: historia de los algoritmos y de los lenguajes de programación*. Nivola, 2006.

Sofía Marina Pepa et al. *Suite de algoritmos de recomendación en aplicaciones reales*. B.S. thesis, 2014.

Li Pu y Boi Faltings. Understanding and improving relational matrix factorization in recommender systems. En *Proceedings of the 7th ACM conference on Recommender systems*, págs. 41–48. 2013.

Chris Rusbridge. Towards the hybrid library. 1998. URL https://era.ed.ac.uk/handle/1842/1736.

Josefina Moreno Salas et al. Biblioteca digital. *Vida Científica Boletín Científico de la Escuela Preparatoria No. 4*, 7(14):1–5, 2019.

Badrul Sarwar, George Karypis, Joseph Konstan, y John Riedl. Application of dimensionality reduction in recommender system-a case study. Inf. téc., Minnesota Univ Minneapolis Dept of Computer Science, 2000.

Taly Sharon y Ariel J Frank. Digital libraries on the internet. 2000.

Jesús Tramullas. Propuestas de concepto y definición de la biblioteca digital. pág. 10. Univ. Politécnica de Madrid, 2002. URL http://eprints.rclis.org/15118.

Maria Vargas-Vera y Enrico Motta. An ontology-driven similarity algorithm. *Knowl. Media Inst., Open Univ.*, 2004.

Augusto Cortez Vásquez, Jaime Pariona Quispe, Ana Maria Huayna, et al. Procesamiento de lenguaje natural. *Revista de investigación de Sistemas e Informática*, 6(2):45–54, 2009.

Julio Vera, Alexander Ocsa Mamani, y Klinge Villalba. Modelo de sistema de recomendación de objetos de aprendizaje en dispositivos móviles, caso: Desarrollo del pensamiento computacional. En *XX Congreso Internacional de Informática Educativa, TISE 2015. Nuevas Ideas en Informática Educativa (Santiago, Chile, 1-3 de diciembre de 2015)*, págs. 730–734. 2015.

Donald J Waters. What are digital libraries. *CLIR issues*, 4(1):5–6, 1998.

Ana Zita. Significado de Texto científico. *Significados*, 2019. URL https://www.significados.com/texto-cientifico.