Universidad de Alcalá de Henares

Sistema de recomendación de grupos de música con Discogs y Surprise Framework

Trabajo Fin de Máster

Máster en Data Science

AUTOR: Asier Aguayo Velasco

TUTOR/ES: Miguel Ángel Sicilia

Resumen

En este proyecto se estudian dos de las formas de crear un sistema de recomendación como son los sistemas de recomendación basados en filtrado colaborativo y los sistemas de recomendación basados en contenido. El objetivo final es crear un sistema de recomendación híbrido que combine ambos modelos.

Para este propósito, se estudiará la librería Surprise Framework, librería diseñada para poder crear un sistema de recomendación colaborativo.

También se utilizará Spark como herramienta para gestionar la gran cantidad de datos recogidos desde Discogs. Mediante estos datos se creará el sistema de recomendación basado en contenido.

Finalmente se estudiarán las diferentes maneras que hay de combinar ambos sistemas para conseguir un sistema de recomendación híbrido.

En esta memoria, se estudiarán las técnicas mencionadas y se explicará en detalle el proceso de desarrollo del proyecto hasta conseguir el objetivo final.

Índice

Índice de ilustraciones

Índice de ecuaciones

1. Introducción y contexto

En el año 2009 se hizo viral una noticia en la que Netflix, el popular servicio de transmisión de contenidos de vídeo, ofrecía 1 millón de dólares a la persona o grupo que consiguiera mejorar el sistema de recomendación de la plataforma.[1] Esta noticia ocupó decenas de portadas en la prensa digital. Para poder optar al premio, Netflix compartió una parte de su bien más preciado con los concursantes, los votos de los usuarios. El ganador del concurso mejoró el algoritmo de recomendación, reduciendo el error en más de un 10% con respecto al sistema que utilizaba Netflix hasta la fecha. [2]

La mayoría de las grandes compañías que ofrecen sus servicios a través de la web, utilizan también este tipo de sistemas para ofrecer a los usuarios alguno de sus servicios. El clicar en "Acepto las Cookies", activa un mecanismo de recogida de datos masivo que son explotados para ofrecer a cada usuario los productos que más puedan interesarle. De esta manera Amazon ofrece el artículo que encaja con alguna de las compras realizadas anteriormente, Spotify recomienda el grupo de música que más se adapta a las reproducciones que se han hecho durante el tiempo o Google ofrece noticias de actualidad teniendo en cuenta lo que los usuarios han leído.

De esta forma, los sistemas de recomendación se han consolidado como una parte indispensable en todo sitio web que ofrece cualquier servicio. La razón es que de estos sistemas depende, en una parte nada despreciable, lo que es capaz de generar económicamente, o por contrario, por todo lo que no se pierde.

Y es que, continuando con el ejemplo mencionado anteriormente de Netflix, desde la propia compañía se estima que el sistema de recomendación que se utiliza en la plataforma actualmente, ha evitado pérdidas potenciales de mil millones de dólares a la compañía.[3] En el artículo se menciona que los usuarios pierden la atención tras permanecer entre 60 y 90 segundos escogiendo el contenido para ver a continuación. Esto genera el riesgo de perder a los usuarios indecisos que no deciden antes de ese tiempo, lo que conlleva a cancelar la suscripción de la plataforma. Mencionan que, mediante un buen sistema de recomendación, el tiempo de decisión se puede reducir, orientando al usuario al contenido potencialmente interesante y manteniendo así su fidelidad.

La forma de conseguir acertar con la recomendación que se hace a un usuario en concreto se puede lograr de maneras distintas. A lo largo de este trabajo se explicarán y se aplicarán los dos tipos de sistemas de recomendación que se pueden desarrollar:

[1] Enlace al concurso: https://www.netflixprize.com/

[2] En este artículo se describe en qué consistía el concurso y la influencia que tuvo el mismo a la hora de manejar los datos recogidos para crear sistemas de recomendación: https://www.thrillist.com/entertainment/nation/the-netflix-prize

[3] Entrevista con Carlos Gómez-Uribe, exdirector de innovación en Netflix hablando sobre el tema: https://www.businessinsider.com/netflix-recommendation-engine-worth-1-billion-per-year-2016-6?IR=T

- Los basados en los datos generados por los usuarios, llamados sistemas de **filtrado colaborativo.**
- Los basados en el contenido propio de del objeto/artículo/producto que se ofrece, llamados sistemas de **filtrado por contenido**.

Para poder implementar un sistema de recomendación es imprescindible la explotación de una cantidad de datos suficiente. En este caso, se utilizarán datos sobre una fuente inagotable de datos, la música.

La música ha acompañado al ser humano durante toda su existencia, ya sea de manera más elaborada o de la manera más rudimentaria. Ha habido y habrá millones y millones de artistas musicales en el mundo. La fuente de datos utilizada no recoge todos y cada uno de los grupos de música que ha habido, pero tiene una gran muestra de las bandas que han cambiado la historia de la música. Esta fuente de datos se llama **Discogs**.

Discogs es una plataforma web que almacena información sobre discografía oficial, lanzamientos promocionales, canciones, etc. de cientos de miles de grupos de música de todo el mundo. Es una web colaborativa sin ánimo de lucro en la que cualquier usuario puede aportar información de manera altruista.

El sitio web proporciona el acceso a las bases de datos y permite utilizarlas de manera libre. Se permite el acceso tanto a la información pública de los usuarios que puede encontrarse en la web, como a la información, pública también, de los productos musicales. Con estos dos tipos de datos, se han creado los dos sistemas de recomendación mencionados anteriormente.

Debido a la gran cantidad de datos recogidos, la estrategia más adecuada para explotarlos es mediante un ecosistema Big Data. Se le llama Big Data a una cantidad masiva de datos que no se puede procesar mediante los métodos ordinarios del procesamiento de datos.

Para poner en marcha los sistemas de recomendación, se combinan diferentes algoritmos de Machine Learning con Big Data.

En este proyecto se explican los pasos llevados a cabo para unir las diferentes tecnologías para crear un sistema de recomendación basado en los datos proporcionados por Discogs.

Se estudia el funcionamiento de los dos tipos de sistemas de recomendación comentados y los resultados que se pueden obtener de ellos para lograr los objetivos que se mencionan a continuación.

1.1.Objetivos del proyecto

En esta parte de la memoria se introducen los diferentes objetivos a conseguir utilizando las técnicas y herramientas mencionadas anteriormente.

El objetivo final es crear dos sistemas de recomendación.

- El primero de ellos basado en **filtrado colaborativo**, utilizando para ello las puntuaciones que algunos usuarios han dado a las diferentes publicaciones en Discogs.
- El segundo, basado en **filtrado por contenido**, se utilizará como complemento del primero de los sistemas de recomendación desarrollado para comprobar y verificar que el sistema de recomendación funciona de manera correcta.

Además de los dos objetivos mencionados, se estudiarán diferentes maneras de crear un sistema de recomendación híbrido para mejorar el sistema de recomendación final.

2. Objetivos y aportaciones del trabajo

2.1.Descripción

A lo largo de este proyecto se estudia la librería Surprise Framework que sirve para crear sistemas de recomendación basado el filtrado colaborativo. Además, también se estudia el funcionamiento de los sistemas de recomendación basado en filtrado por contenido. Para la explotación de los datos se utilizan herramientas de Big Data, en este caso, Spark. Utilizando estas y otras tecnologías se consigue lograr el objetivo de crear los sistemas de recomendación mediante los datos recogidos de Discogs.

Para ello, se estudia la base teórica y los algoritmos sobre los que se sustenta la librería Surprise. Para el sistema de recomendación basado en contenido, se estudian los diferentes algoritmos para encontrar la similitud en las diferentes publicaciones.

2.2.Objetivos y motivación

Para poder alcanzar el objetivo final, antes hay que realizar una serie de subobjetivos. La primera parte del trabajo consiste en estudiar qué ofrece la librería seleccionada Surprise Framework.

Una vez adquiridos los conocimientos básicos y saber qué tipo de sistema de recomendación se puede generar mediante el framework, el siguiente paso es estudiar las diferentes opciones que hay para recoger los datos y limpiarlos de manera que la librería pueda hacer uso de ellos.

Terminada la tarea anterior, se estudian los diferentes algoritmos que proporciona la librería y se ponen en práctica.

Además, como se ha mencionado, se generará también un sistema de recomendación basado en contenido para poder verificar el funcionamiento del sistema generado mediante Surprise. Esto implica recoger los datos que más interesan de las publicaciones, filtrarlos, limpiarlos y dejarlos de manera que se puedan utilizar algoritmos de similitud para crear un sistema de recomendación.

El alumno, mediante este proyecto, pone en práctica los conocimientos adquiridos durante el máster en asignaturas donde se trabaja el procesamiento de grandes cantidades de datos con Spark. Profundiza en los conceptos de creación de sistemas de recomendación mencionada en algunas asignaturas durante el curso. Además, estudia diferentes algoritmos matemáticos para hallar la similitud entre elementos.

2.3.Planificación

El proyecto se divide en varias fases compuesto por subobjetivos para conseguir los objetivos finales.

- Primera fase:

    - **Estudiar los diferentes tipos de sistemas de recomendación**. Al no haber profundizado en los conceptos de los sistemas de recomendación durante el curso, estudiar las diferentes formas que hay para crear sistemas de recomendación y ver las diferencias entre ellas.

    - **Estudiar las posibilidades que ofrece la librería Surprise Framework**. Una vez estudiadas los diferentes tipos de sistemas de recomendación, se estudia Surprise Framework para ver qué puede ofrecer y qué tipo de sistema de recomendación se puede crear.

    - **Preparar el entorno de desarrollo**. Al utilizar diferentes librerías y un framework de computación en clúster como es Apache Spark, en la primera fase se instalan las diferentes herramientas y se comprueba que funcionan de manera correcta.

    - **Gestión del proyecto.** Dedicar un tiempo a organizar las tareas y poner límites en caso de excederse en el tiempo planificado. Esta tarea se llevará a cabo durante todo el proyecto.

- Segunda fase:

    - **Recoger las votaciones de los usuarios desde Discogs.** Para el sistema de recomendación colaborativo se recogen las votaciones de algunos usuarios de la plataforma web. Discogs tiene una API para poder recoger las votaciones de los usuarios introduciendo su nombre de usuario.

    - **Recogida de usuarios desde el foro de Discogs para acceder a sus votaciones.** Para poder recoger las votaciones mediante la API, primero hay que crear una lista con nombres de una muestra de usuarios al azar. Para ello se utiliza Scrapy, un framework para poder recoger información de páginas web de manera automática. Esta tarea implica estudiar el funcionamiento del framework e implementarlo en el foro de Discogs.

    - **Filtrado y limpieza de los datos para el sistema de recomendación por contenido.** Discogs proporciona volcados de la base de datos mensuales que contienen todas las publicaciones de la web. La cantidad de información es muy grande por lo que en esta fase el objetivo es quitar la información no relevante

y filtrar y limpiar los datos de manera que se puedan usar para buscar la similitud entre publicaciones.

- Tercera fase:

  - **Estudiar y aplicar los algoritmos de Surprise para generar el sistema de recomendación colaborativo**. Estudiar las diferentes opciones que permite Surprise, así como los diferentes algoritmos. Una vez entendidas las diferencias, aplicar las que se consideren mejores para el objetivo.

  - **Aplicar ingeniería de atributos en los datos de las publicaciones para el sistema de recomendación por contenido.** Estudiar las diferentes técnicas de preparar los datos para poder aplicar algoritmos de similitud. Para ello, utilizar la librería propia de Spark MLlib.

  - **Generar el sistema de recomendación aplicando algoritmos de similitud.** Estudiar las posibilidades que hay y utilizar la librería de MLlib.

- Cuarta fase:

  - **Analizar los resultados de ambos sistemas de recomendación**. Surprise proporciona un módulo para estudiar la precisión del modelo entrenado en el sistema de recomendación colaborativo. En esta primera subtarea, analizar los resultados y estudiar diferentes maneras de mejorar el modelo.

  - **Estudiar la forma de juntar ambos sistemas de recomendación**. Estudiar las formas de hacerlo híbrido y hacer un sistema de recomendación más consistente.

  - **Conclusiones y futuras aplicaciones.** Para terminar el trabajo, se mencionarán las posibles aplicaciones reales que pueden tener las técnicas aplicadas.

2.3.1. Planificación temporal inicial

Mediante un diagrama de Gantt (Tabla 1) se ilustra la planificación del proyecto. Las tareas del diagrama son las mencionadas en el apartado anterior. Ya que se compagina el trabajo personal con el trabajo de fin de máster, los plazos son orientativos y se tomarán las semanas de dos en dos.

En la siguiente tabla se mostrarán las horas estimadas a cada tarea y subtarea durante el proyecto.

Ilustración 1. Diagrama de Gantt con las horas estimadas del trabajo de fin de máster.

A continuación, se muestran las horas estimadas a cada tarea dentro del proyecto.

Fase	Tarea	Horas estimadas
Fase 0:	Presentación del anteproyecto	20
Primera Fase:	Estudio tipos de sistemas de recomendación	8
	Estudiar Surprise Framework	12
	Preparar entorno de desarrollo	30
	Gestión del proyecto	60
Segunda Fase:	Recogida de votaciones de usuarios	20
	Recogida de usuarios desde el foro	20
	Filtrado y limpieza de datos	40
Tercera Fase:	Estudio y aplicación de algoritmos en Surprise	30
	Ingeniería de atributos	30
	Generar el sistema de recomendación	20
Cuarta Fase:	Análisis de los resultados	15
	Estudiar la manera de juntar ambos sistemas	20
	Suma total:	325

Ilustración 2: Horas estimadas a las tareas del proyecto

Como se puede apreciar en ambas tablas, la tarea que más tiempo va a llevar es la de la gestión del proyecto, en esta tarea se incluye la documentación y redacción de la memoria.

La siguiente tarea que más horas va a requerir es la de filtrar y limpiar los datos de la base de datos que proporciona Discogs. Al manejar una cantidad de datos tan grande seguro que surgirán problemas y hasta dejar los datos de la manera idónea para poder aplicar los algoritmos precisos, requerirá de una gran cantidad de pruebas.

Se ha estimado que se utilizarán 30 horas para preparar el entorno de desarrollo. Esta tarea implica crear un clúster y se estima que con los problemas que pueda haber de incompatibilidades debido a las versiones, etc. se necesite esa cantidad de tiempo.

La misma cantidad de horas se ha estimado que se utilizarán para la ingeniería de atributos en el caso del sistema de recomendación por contenido y en el estudio e implementación de algoritmos de Surprise para el sistema de recomendación colaborativo. Estas tareas implican estudiar con detenimiento la documentación para adoptar las mejores medidas para tratar los datos, por lo tanto, la cantidad de horas estimada es bastante alta.

Es complicado hacer una estimación de horas cuando se trata de temas que no se han estudiado anteriormente. Seguramente la dedicación final variará en muchas de las tareas desarrolladas.

2.3.2. Planificación temporal real

Una vez terminado el proyecto y recogidos los tiempos invertidos en cada tarea, se ha elaborado una tabla (Ilustración 3) de forma análoga a la del anterior punto con las horas aproximadas invertidas a cada tarea.

Fase	Tarea	Horas reales
Fase 0:	Presentación del anteproyecto	15
Primera Fase:	Estudio tipos de sistemas de recomendación	10
	Estudiar Surprise Framework	10
	Preparar entorno de desarrollo	50
	Gestión del proyecto	60
Segunda Fase:	Recogida de votaciones de usuarios	15
	Recogida de usuarios desde el foro	10
	Filtrado y limpieza de datos	55
Tercera Fase:	Estudio y aplicación de algoritmos en Surprise	20
	Ingeniería de atributos	40
	Generar el sistema de recomendación	15
Cuarta Fase:	Análisis de los resultados	15
	Estudiar la manera de juntar ambos sistemas	20
	Suma total:	335

Ilustración 3: Horas totales invertidas en cada tarea del proyecto

Como se puede observar si se comparan las horas invertidas reales con las estimadas, la mayor diferencia se puede apreciar en la tarea de preparación del entorno de desarrollo. Hubo bastantes problemas a la hora de lanzar el servidor de Spark en la máquina donde se desarrollaron los sistemas de recomendación.

También se puede apreciar que las tareas que más tiempo han llevado son las que se preveían que así iba a ser. La tarea de filtrado y limpieza de datos se estimaba costosa y así ha sido. El peso total de la base de datos con la que se ha trabajado es de 35 GB, para evitar trabajar continuamente con la base de datos, se han ido generando ficheros intermedios hasta conseguir los datos de manera apropiada para trabajar con ellos.

Se han invertido 40 horas aproximadamente a la ingeniería de atributos. Debido a las diferentes formas de tratar los datos, se han probado diferentes algoritmos hasta conseguir unos valores coherentes y entendibles para buscar la similitud entre los ítems.

Por el lado contrario, se preveía invertir más tiempo en las tareas relacionadas con la elaboración del sistema de recomendación colaborativo, pero al utilizar librerías expresamente desarrolladas para ello y documentadas de manera muy clara, han hecho la tarea mucho más sencilla.

El tiempo invertido finalmente ha sido de 335 horas aproximadamente, 10 horas más de las previstas inicialmente. Si bien es cierto que uno de los objetivos iniciales era implementar un sistema de colaboración híbrido, debido a que las horas invertidas se estaban disparando, se ha decidido dejarlo como futuro trabajo y hacer un pequeño análisis de las posibilidades que se presentan para ello.

A continuación, se mostrarán las fechas en las que se han implementado las tareas mediante un diagrama de Gantt (Ilustración 4).

Fase	Tarea	Fecha Inicio	Fecha Fin	Junio	Julio	Agosto	Septiembre	Octubre	Noviembre	Diciembre
Fase 0:	Presentación del anteproyecto	14/06/2019	30/06/2019							
Primera Fase:	Estudio tipos de sistemas de recomendación	01/07/2019	08/07/2019							
	Estudiar Surprise Framework	08/07/2019	15/07/2019							
	Preparar entorno de desarrollo	16/07/2019	04/08/2019							
	Gestión del proyecto	30/06/2019	20/12/2019							
Segunda Fase:	Recogida de votaciones de usuarios	05/08/2019	12/08/2019							
	Recogida de usuarios desde el foro	05/08/2019	12/08/2019							
	Filtrado y limpieza de datos	13/08/2019	30/09/2019							
Tercera Fase:	Estudio y aplicación de algoritmos en Surprise	01/10/2019	13/10/2019							
	Ingeniería de atributos	14/09/2019	17/11/2019							
	Generar el sistema de recomendación	18/11/2019	24/12/2019							
Cuarta Fase:	Análisis de los resultados	25/12/2019	08/12/2019							
	Estudiar la manera de juntar ambos sistemas	09/12/2019	20/12/2019							

Ilustración 4: Diagrama de Gantt con las horas reales invertidas en el proyecto.

Cabe destacar que debido a los problemas que hubo a la hora de preparar el entorno de desarrollo las fechas reales se retrasaron respecto a las horas estimadas, sin embargo, se recuperó cuando las tareas de recogida de datos de usuarios se llevaron a cabo de manera más rápida.

También comentar que la mayoría del trabajo se realizó durante las vacaciones de verano debido a que se disponía de más tiempo para realizarlo.

2.4. Análisis de riesgos

Antes de comenzar a desarrollar el proyecto es interesante analizar cuáles son los posibles problemas a los que se puede enfrentar el proyecto. Para subsanar los problemas que pudiesen ocurrir, se realizará un plan de contingencia. Los posibles problemas que pueden surgir son los siguientes.

- **Problemas técnicos**. Debido a que se desarrollará un proyecto en un ordenador, no se pueden obviar los problemas que puedan surgir en el equipo. Ya sean de software o de hardware, un problema de este tipo puede suponer un severo retraso en el desarrollo. Para evitar este tipo de problemas, se subirán a la nube los Notebooks que se vayan desarrollando. También se guardarán las bases de datos intermedias que se vayan creando. Además, para las tareas que no requieran de un gran procesamiento, se instalarán las librerías básicas en otro ordenador para poder trabajar de forma simultánea en ambos equipos.

- **Falta de documentación**. Al utilizar una librería Open Source, a pesar de tratarse de una librería bastante utilizada en lo que a creación de sistemas de recomendación se refiere, puede que durante el proyecto no se encuentre la información precisada y haya que buscar en foros o en otros sitios. Esto puede suponer un cierto retraso en el desarrollo del proyecto. Aunque no se pueda evitar este tipo de problemáticas, en caso de que ocurra algún caso parecido a este, se utilizará el foro como herramienta para resolver las dudas que puedan ir surgiendo.

- **Sobreexceso de trabajo**. Al compaginar el trabajo personal y el máster, habrá ocasiones que no haya demasiado tiempo para llevar a cabo el proyecto. Para evitar este tipo de problemas se valorarán las diferentes tareas a realizar y se realizará primero la más prioritaria. Más adelante se llevarán a cabo las demás tareas. Teniendo en cuenta además que durante el desarrollo del trabajo de fin de máster aún habrá asignaturas por terminar, es muy probable que los plazos se alarguen o que se simplifiquen algunas de las tareas que se quieran hacer.

3. Conceptos básicos

En este apartado de la memoria se explican las bases teóricas que se utilizan durante el desarrollo del proyecto. Se hace especial hincapié en los sistemas de recomendación, se explicará la base de estos sistemas, los diferentes tipos que se pueden encontrar, los problemas más típicos que pueden encontrarse, etc.

3.1.Sistemas de recomendación

Es complicado colocar el sistema de recomendación en un solo ámbito de las ciencias de la computación, ya que para crear un sistema de recomendación se utilizan técnicas de Machine Learning, inteligencia artificial, estadística, matemáticas, etc.

Se podría decir que los sistemas de recomendación derivan de los sistemas de filtrado de información. Estos sistemas utilizados desde antes de la llegada de internet, son métodos que se utilizaban en diferentes ámbitos, no solo en computación, para filtrar la información irrelevante y quedarse con la realmente importante. Esa información se podía utilizar para sacar valor en cualquiera que fuese el ámbito en el que se aplicase.

A medida que iba pasando el tiempo, debido a que cada vez se generan más y más datos, se hacían necesarias nuevas técnicas para filtrar los datos y poder procesarlos de manera más rápida. La primera vez que se mencionó el término de sistema de recomendación fue por Jussi Karlgren en la Universidad de Columbia en 1990. 4 años más tarde publicaría su artículo llamado *Newsgroup Clustering Based On User Behavior - A Recommendation Algebra* lo que se considera como el inicio de los sistemas de recomendación tal y como se conocen hoy en día. En su trabajo, Jussi Karlgren utilizó por primera vez la información propia de los usuarios y mediante algoritmos matemáticos los agrupó (clustering).

Un sistema de recomendación se puede definir como un algoritmo que sugiere artículos (películas, música, series, artículos de venta…) a un usuario.

Hay dos grandes paradigmas para lograr esto:

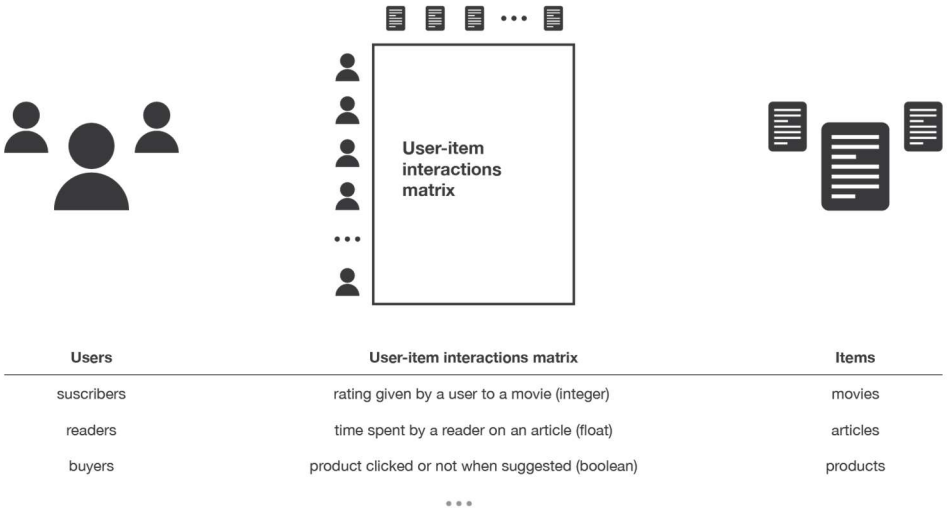
- Sistemas de filtrado colaborativo.
- Sistemas de filtrado por contenido.

3.1.1. Sistemas de filtrado colaborativo

El primero de los sistemas de recomendación, se basa solamente en interacciones pasadas entre usuarios y artículos para crear nuevas recomendaciones.

La filosofía de este sistema de recomendación se basa en que, si un usuario tiene unos gustos X y otro usuario, tiene unos gustos muy parecidos a X, es probable que al primer usuario le gusten los artículos que le han gustado al segundo.

Para manejar esta información, se utiliza una matriz llamada "matriz usuario-artículo" (user-item matrix). En esta matriz se almacena el usuario, el artículo y la acción entre el usuario y el artículo. Esta acción puede ser una votación de 1 a 5, indicar si le gusta o no le gusta mediante un sistema binario, etc. En la siguiente imagen se puede apreciar el funcionamiento de esta matriz.



Users	User-item interactions matrix	Items
suscribers	rating given by a user to a movie (integer)	movies
readers	time spent by a reader on an article (float)	articles
buyers	product clicked or not when suggested (boolean)	products
...		

Ilustración 5: Matriz usuario-artículo

Dentro de los sistemas por filtrado colaborativo se pueden diferenciar dos subcategorías.

3.1.1.1. Basados en memoria:

Se basa directamente en los datos recogidos y utiliza los algoritmos de vecinos más cercanos para hallar los artículos más parecidos.

La manera de hallar los artículos más parecidos se puede hacer por el método llamado user-user o ítem-ítem. La diferencia entre ambos es que, en el primer modo, las diferencias calculadas por los algoritmos de vecinos más cercanos se calculan entre usuarios. De modo opuesto, el método ítem-ítem calcula las distancias entre ítems.

La diferencia a modo matricial radica en que en el modo user-user, las diferencias se calculan entre las filas (usuarios), mientras que en el modo ítem-ítem se calculan las diferencias entre las columnas (artículos). En las siguientes imágenes se puede apreciar la diferencia claramente.

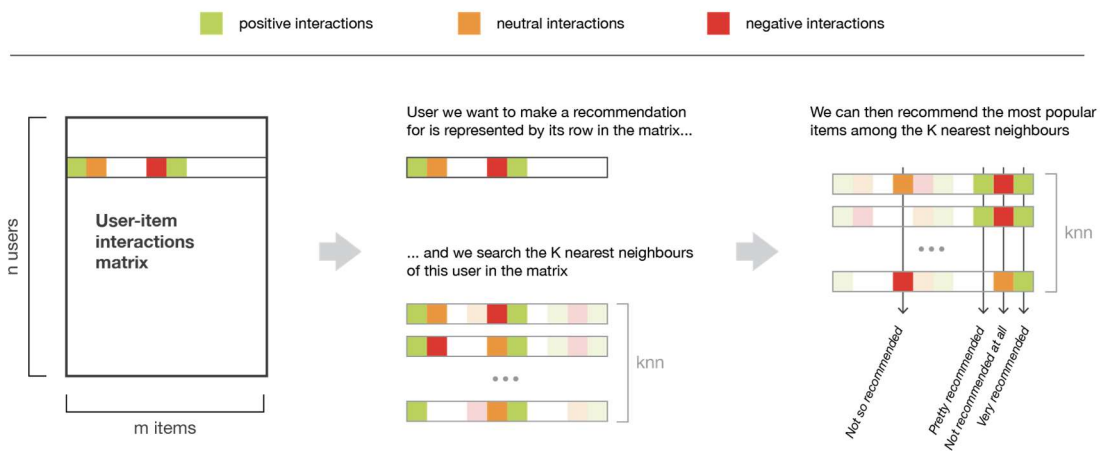


Ilustración 6: ejemplo método user-user

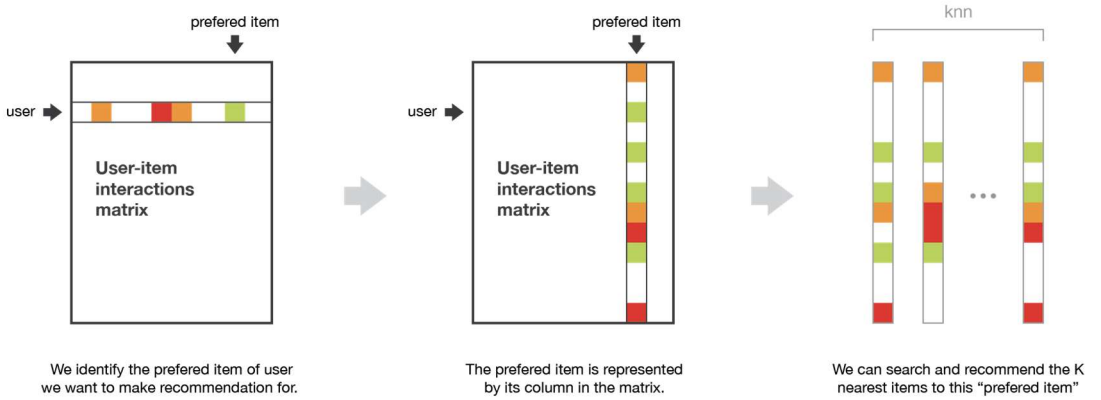


Ilustración 7: ejemplo método ítem-ítem

La forma de calcular las distancias y seleccionar a los vecinos más cercanos depende de la librería que se utilice para ello, en este caso, más adelante se explicará las diferentes opciones que propone Surprise Framework para ello.

Dependiendo de los datos, puede ser mejor utilizar uno de los métodos u otro. Al utilizar el método user-user aumenta la varianza debido a que, de manera general, los usuarios han interaccionado con pocos artículos, haciendo que las interacciones tengan excesiva importancia. Sin embargo, al tratarse de información basada en usuarios, se obtienen resultados más personalizados.

Si se utiliza el método ítem-ítem, ocurre lo contrario, se reduce la varianza debido a que los artículos tienen muchas interacciones, pero los resultados que se obtienen son mucho menos personalizados ya que se tiene en cuenta los artículos y no a los usuarios y sus gustos.

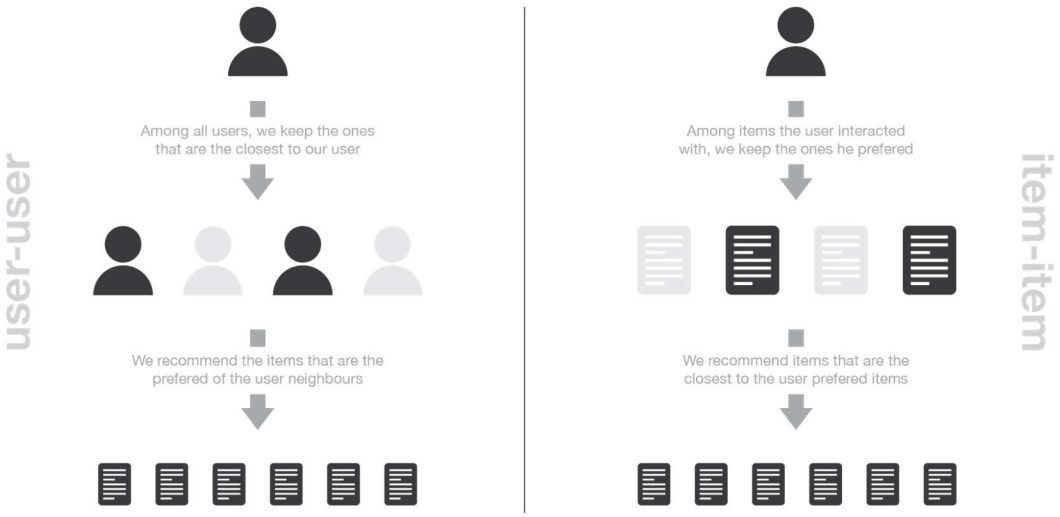


Ilustración 8: diferencia entre los métodos user-user e ítem-ítem

3.1.1.2. Basados en modelos:

Este tipo de método se basa en generar un modelo que va a ir modificándose a medida que vayan introduciéndose nuevos casos. Este tipo de método se aproxima más a las vistas en Machine Learning. Utiliza técnicas como las *redes bayesianas*, *clustering* o s*istemas basados en reglas* para generar dicho modelo.

La base sobre la que se sustentan este tipo de métodos es la de reducir el tamaño de la matriz user-ítem. Esta matriz generalmente es una matriz dispersa, es decir, es una matriz en la que la mayoría de los elementos es cero. El objetivo de esta técnica es reducir la matriz original al resultado de la multiplicación de dos submatrices, una que contiene la representación de los usuarios y la otra que contiene la representación de los artículos valorados. A este tipo de técnica se le llama **Factorización de matrices.**

En la ilustración 9 se puede ver de manera clara y sencilla el funcionamiento de esta técnica.

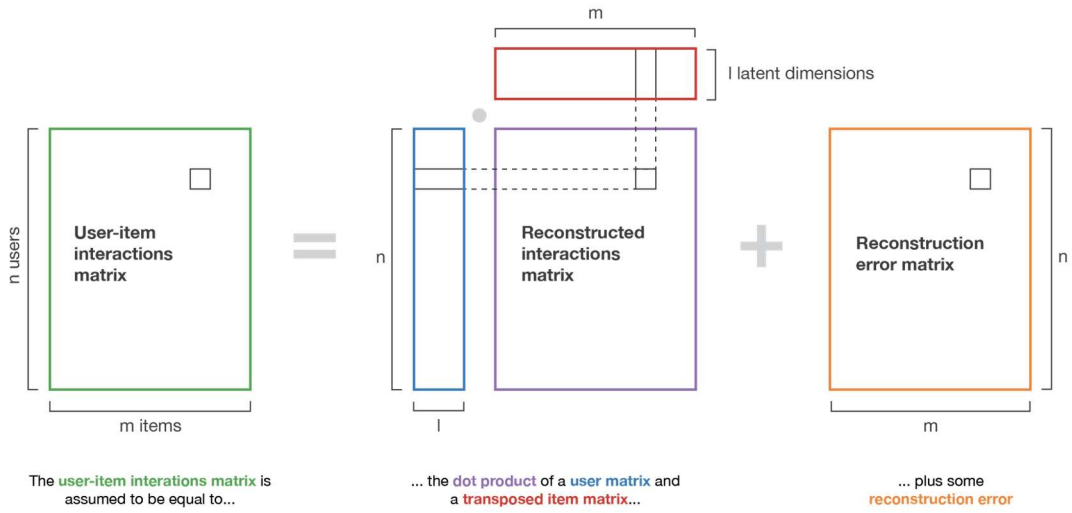


Ilustración 9: Ejemplo de la factorización de matrices

Una vez explicadas las diferentes técnicas que existen para el filtrado colaborativo, hay que destacar que, para este tipo de técnicas, no se necesita información alguna ni de los usuarios ni de los artículos valorados, por lo tanto, se puede extrapolar este tipo de técnicas a una gran cantidad de ámbitos.

Cuanto mayor sea la cantidad de información disponible, se podrá elaborar un sistema de recomendación más fiable y robusto. Además, a medida que se van introduciendo datos, el sistema de recomendación irá incrementando su fiabilidad.

El problema en este tipo de sistemas de recomendación se da al inicio de un proyecto, cuando la cantidad de datos recogida es pequeña. Al haber pocas interacciones entre usuarios y artículos, las relaciones de similitud en la matriz user-item no es lo suficientemente importante como para recomendar uno u otro artículo. A este tipo de problema se le llama “cold start problema”, cuya traducción podría ser algo como “el problema del inicio frío”.

Para evitar este tipo de problemas se suelen utilizar diferentes estrategias:

- Recomendación aleatoria: Hasta que no haya una cantidad consistente de datos, las operaciones realizadas para hallar el artículo recomendado, podrán recomendar artículos totalmente aleatorios entre los que se han valorado.

- Recomendación de artículos populares: Mediante esta técnica, primero se recomendarán los artículos más valorados debido a que es probable que al tratarse de los más populares, gusten también a los usuarios que buscan recomendaciones.

- Estrategia de exploración: Otro método que se utiliza es el de recomendar nuevos artículos que aún no tienen interacciones. De esta manera se conseguirá que se empiecen a generar interacciones con este tipo de artículos.

Sin embargo, la estrategia más utilizada y aceptada para evitar este tipo de problemas, suele ser la de utilizar métodos de recomendación no colaborativos como pueden ser los métodos de filtrado por contenido.

3.1.2. Sistemas de filtrado por contenido

Los sistemas de filtrado por contenido utilizan las características propias de los usuarios o de los artículos para crear un modelo que recomiende un artículo u otro. De manera contraria a los sistemas de filtrado colaborativo, las valoraciones de los usuarios no se tienen en cuenta para este tipo de filtrado.

Mediante este tipo de técnicas se pueden sacar conclusiones estadísticas en base a las recomendaciones. Al disponer de las características de los artículos y también de los usuarios, se pueden agrupar los artículos de manera que se concluya que al grupo X le gustan los artículos X1, X2, etc.

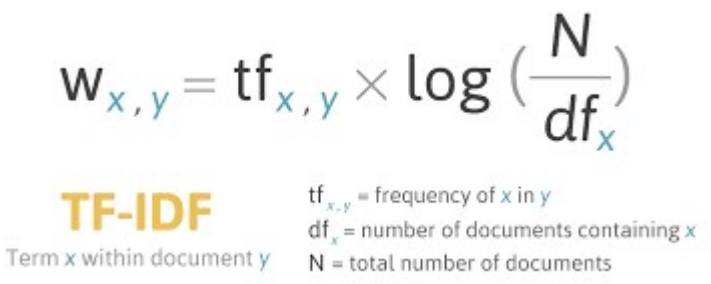
El sistema de recomendación de filtrado por contenido evita de esta manera el problema "cold start" comentado anteriormente con los sistemas de filtrado colaborativo, ya que, puede recomendar artículos parecidos entre sí sin necesidad de información de los usuarios.

Debido a que la mayoría de los datos recogidos en los artículos son textos descriptivos, hay que realizar un tratamiento de datos para poder operar con ellos, ya que, para hallar la similitud se utilizan operaciones matemáticas que se explicarán más adelante.

Existen diferentes técnicas para tratar los datos de los diferentes artículos. Una de las técnicas más utilizadas es el algoritmo TF-IDF.

3.1.2.1. TF-IDF

El algoritmo TF-IDF es una técnica que mide la frecuencia de un término (TF) y su frecuencia inversa en el documento (IDF). Cada palabra tiene su respectivo valor TF e IDF en el documento. El documento hace referencia, en este caso, a todas las palabras que se puedan recoger en una columna de la base de datos, de esta manera, se puede hallar la frecuencia de la palabra dentro de un conjunto de palabras. Por lo tanto, para conseguir el valor TF-IDF de una palabra, se multiplica su TF * su IDF, de manera que el que tenga un valor más alto, será el término menos repetido.



Ecuación 1: Algoritmo TF-IDF

De esta manera, se pueden hallar similitudes entre las palabras encontradas en la descripción de los artículos. Los valores que tengan un valor TF-IDF cercano, serán más similares que los que tengan un valor alejado.

Existen otras técnicas para conseguir asignar un valor numérico a los elementos de una base de datos. En este trabajo, se utilizan el TF-IDF y otras técnicas más sencillas que se explicarán en el capítulo del desarrollo del proyecto.

Una vez calculada la representación numérica de los artículos basándose en su descripción, se puede calcular la relevancia de similitud entre los artículos. Uno de los métodos para calcular esta similitud es la **Similitud coseno**.

3.1.2.2. Similitud coseno

La similitud coseno se utiliza para hallar la similitud entre dos vectores en un espacio vectorial de n dimensiones. En este caso, se utiliza para calcular la similitud entre los artículos de la base de datos.

La similitud se calcula evaluando el valor del coseno del ángulo que se forma entre los vectores en el espacio vectorial.

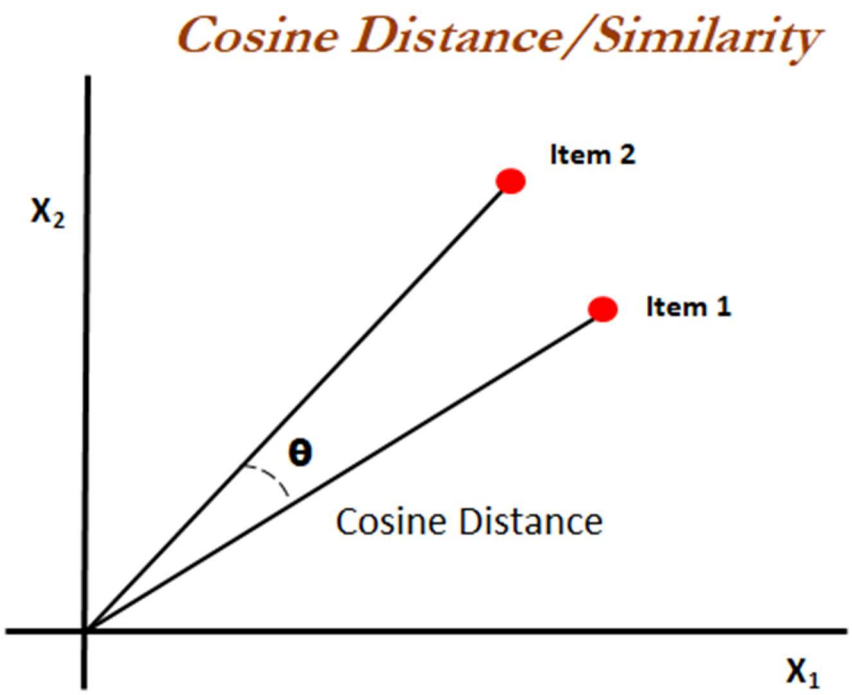


Ilustración 10: Ejemplo de la similitud coseno

Para ello se utiliza la fórmula del cálculo del coseno entre vectores, mostrada a continuación.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

Ecuación 2: Cálculo del coseno

El problema de las recomendaciones de filtrado por contenido es que carecen de personalización, sólo se recomendarán aquellos artículos que tienen relación por el motivo que sea. Por ejemplo, siguiendo el ejemplo de este trabajo, a una persona que le gusta un disco de Rock, se le recomendarán en mayor medida discos Rock, aunque a ese usuario sólo le guste ese disco de Rock.

Para evitar estos problemas, la técnica más adecuada suele ser la implementación de sistemas de recomendación híbridos.

3.1.3. Sistemas de recomendación híbridos

Los sistemas de recomendación híbridos combinan dos o más estrategias de recomendación para beneficiarse de las ventajas de cada tipo de sistema.

Los sistemas que se utilicen para construir el sistema de recomendación híbrido, no tienen por qué ser de distinto tipo expresamente, se pueden combinar dos sistemas de filtrado colaborativo utilizando dos técnicas diferentes, por ejemplo, user-user e ítem-item.

No existe una manera única de combinar los sistemas de recomendación. A continuación, se enumeran 3 de las formas más comunes para combinar sistemas de recomendación.

- **Unificación**: Combinar los diferentes en uno solo. De esta manera se realizarán las predicciones desde el último modelo.[4]

- **Votación**: Mediante este sistema, se recogen las votaciones finales de cada sistema de recomendación y se opera con ellos de manera que se genera un nuevo valor. La manera más simple para combinar los sistemas mediante este método es sumar los valores de cada sistema de recomendación y guardar el nuevo valor para el artículo valorado correspondiente.[5]

- **Cascada**: Se le asigna una prioridad a cada sistema de recomendación. En caso de que el sistema escogido en primer lugar sea capaz de proporcionar una recomendación, se escoge esa recomendación, en caso contrario, se utiliza el siguiente sistema de recomendación, etc.

- **"Business rules"**: El objetivo de este método es realizar un análisis exploratorio del sistema que mejor resultados obtiene en base a la elección del usuario. Al usuario se le ofrecen las diferentes recomendaciones emitidas por los sistemas de recomendación y escoge el que más se acerca a lo que desea. De esta manera se recomendará el sistema que más interacciones positivas tenga.

[4] Ejemplo de unificación de sistemas de recomendación:
https://ieeexplore.ieee.org/abstract/document/4664342

[5] Ejemplo de votación para combinar sistemas de recomendación (aplicado en el concurso de Netflix comentado al inicio de la memoria):
https://www.netflixprize.com/assets/GrandPrize2009_BPC_BellKor.pdf

4. Spark

En este capítulo se explicará Apache Spark, un framework de computación en clúster que permite procesar grandes cantidades de datos sobre un conjunto de máquinas de manera simultánea.

4.1. Historia y arquitectura

Apache Spark se lanza por primera vez el 30 de mayo de 2014 para paliar el problema de computo en un conjunto masivo de datos, Big Data. Debido a la gran cantidad de datos generados se hacía insostenible utilizar los servicios de alojamiento de datos antiguos. Empezó a desarrollarse en 2009 en la Universidad de Berkeley. En 2013 se donó a Apache Foundation, empresa sin ánimo de lucro creada para dar soporte a proyectos Open Source de software.

La filosofía de Spark radica en la distribución del código en diferentes máquinas, llamadas trabajadores (workers), desde un nodo central, llamado maestro. Cada tarea asignada a los nodos *workers* forma parte de una tarea global. La distribución se utiliza para agilizar el procesado de los datos trabajando en subtareas en paralelo.

La arquitectura se basa en el paradigma MapReduce. Este paradigma se basa en las funciones map y reduce.

- Map: La función map() se encarga de mapear cada ítem en la entrada de datos. Esto produce una lista de ítems en formato dupla con una clave y un valor. Esta función divide la tarea en subtareas y lo distribuye a los workers.

- Reduce: Esta función se aplica a cada grupo generado por la función Map y devuelve el resultado generado en forma de lista.

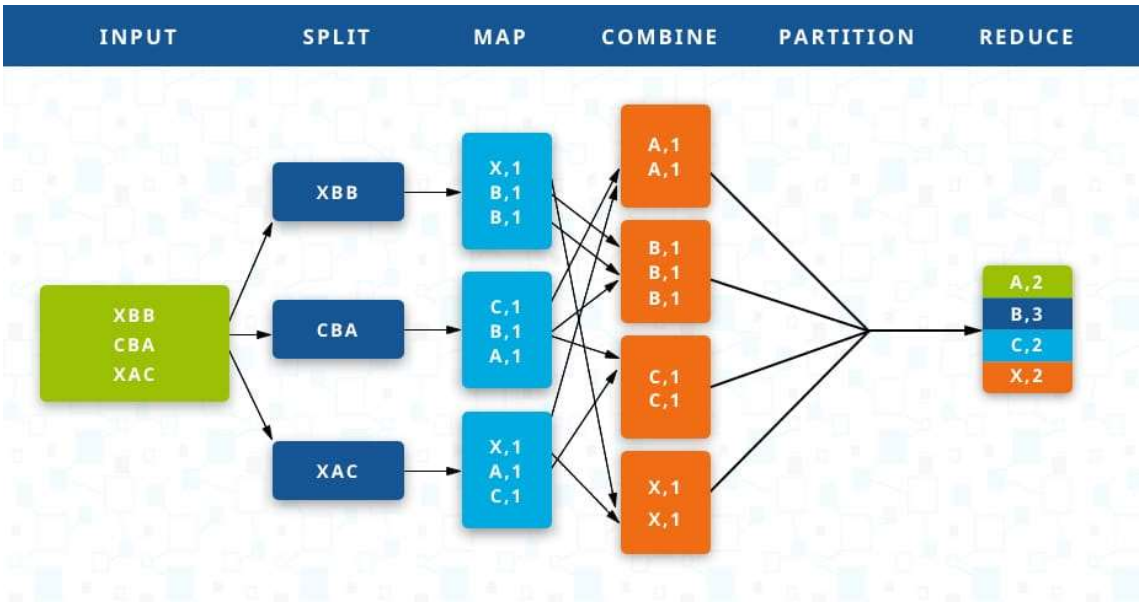


Ilustración 11: Funcionamiento MapReduce

En Spark se utiliza una arquitectura llamada RDD (Resilient Data Distributed). Esta estructura está compuesta de datos sólo de lectura distribuidos por todo el clúster de máquinas. Están distribuidas de manera que se asegura la tolerancia a fallos utilizando el paradigma MapReduce.

Inicialmente se trabajaba directamente sobre la estructura RDD, sin embargo, a lo largo del tiempo han implementado tres APIs llamadas Spark SQL, Datasets y Dataframe. Sirven para trabajar de manera más similar a los gestores de bases de datos.

Debido a la filosofía de distribución de los datos en clúster, permite la opción de utilizar sistemas de ficheros distribuido como son HDFS, Cassandra o Kudu.

Para gestionar los recursos y las máquinas conectadas en el clúster, se pueden utilizar Standalone, YARN o Apache Mesos. Para el desarrollo de este proyecto se ha optado por utilizar Standalone.

Al utilizar el gestor de recursos en modo Standalone, permite utilizar cualquier gestor de recursos siempre y cuando se configure. Si se utiliza Standalone es el administrador el que tiene que iniciar los nodos que forman parte del clúster, tanto el nodo maestro como los nodos worker. En el capítulo del desarrollo del proyecto se explicará cómo se han lanzado estos nodos y con qué configuración se han configurado.

4.2. Características

Spark se diferencia de otros gestores de clústeres por la compatibilidad con una gran cantidad de lenguajes gracias a las diferentes APIs que dispone para ello. Spark soporta Java, Scala, Python, R y SQL.

Spark está diseñado para aumentar la velocidad de procesamiento. Se calcula que es 100 veces más rápido que Hadoop cuando se ejecutan operaciones en memoria y 10 veces más rápido cuando se escribe en disco.

Spark proporciona una serie de librerías que hacen mucho más potente el uso del gestor. Los principales componentes de Spark, además del propio núcleo, son los siguientes:

- **Spark SQL**: Permite la utilización de lenguaje SQL utilizando un concepto de abstracción de datos llamado SchemaRD. Este esquema proporciona soporte para datos estructurados y semi-estructurados de igual manera.

- **Spark Streaming**: Librería diseñada para recoger, analizar y operar con datos en tiempo real. Recoge los datos en pequeños lotes de datos para agilizar el proceso, los distribuye y luego los procesa en diferentes nodos.

- **MLlib**: Es la librería para implementar técnicas de aprendizaje automático de Spark. Proporciona una gran cantidad de algoritmos de regresión, clasificación, clustering, así como técnicas de ingeniería de atributos, TF-IDF, String Indexer, etc.

- **Graphx**: Es la librería gráfica de Spark. Con esta librería se pueden generar gráficos sobre los datos de manera distribuida.

5. Surprise Framework

Surprise es un framework especialmente creado para crear y analizar sistemas de recomendación utilizando votaciones como entrada. Está creado como un SciKit. Un SciKit es un paquete para SciPy, desarrollado y distribuido de forma separada del paquete global. Los SciKits son librerías científicas open source desarrolladas para Python pensadas para un fin concreto no contemplado en la librería principal SciPy.

Surprise se lanzó en el año 2015 y hoy en día es mantenido por muchos colaboradores.

Surprise se diseñó para proporcionar a los desarrolladores una forma sencilla de crear sistemas de recomendación colaborativos. Para ello se han implementado algoritmos de predicción como los que se han ido mencionando a lo largo de la memoria:

- Algoritmos de vecinos cercanos.
    - KNN Basic
    - KNN With Means
    - KNN With Z Score
    - KNN Baseline

- Basados en Factorización de matrices.
    - SVD
    - PMF
    - SVD++
    - NMF

- Algoritmo SlopeOne

- Algoritmos basados en Co-Clustering

Además, cuenta con algoritmos para medir la similitud en la matriz user-item:

- Similitud coseno.
- MSD.
- Similitud Pearson.

Permite construir algoritmos de predicción propios. Proporciona herramientas para evaluar y analizar los resultados de los modelos generados, así como la técnica de validación cruzada o búsqueda de hiperparámetros.

5.1. Algoritmos para generar modelos predictivos en Surprise

Más adelante, en el capítulo 7.1, se puede ver la manera en la que se han implementado los algoritmos mencionados que dispone la librería Surprise. Para entender el funcionamiento

de los mismos, a continuación, se explicarán las nociones básicas de cada uno, aunque se dedicará mayor atención a los algoritmos de vecinos cercanos ya que son los que mejor resultado han dado.

5.1.1. Algoritmos de vecinos cercanos

Esta familia de algoritmos se caracteriza en encontrar los vecinos (ítems) más cercanos a un nuevo ítem dado. El algoritmo requiere de un parámetro k que indica el número de vecinos que valorará para tomar la decisión de a qué grupo pertenece. Es un algoritmo de aprendizaje supervisado, es decir, el objetivo de esta clase de algoritmos es clasificar o predecir correctamente al nuevo ítem.

El funcionamiento del algoritmo se resume en 3 partes:

1. Calcular las distancias entre los distintos ítems que componen el dataset y el nuevo ítem a evaluar.

2. En base a la distancia calculada, calcular los K vecinos más cercanos del ítem. Siendo K el parámetro introducido en la función.

3. Contar los K vecinos de manera que la clase más representada será la que se asigne al nuevo ítem.

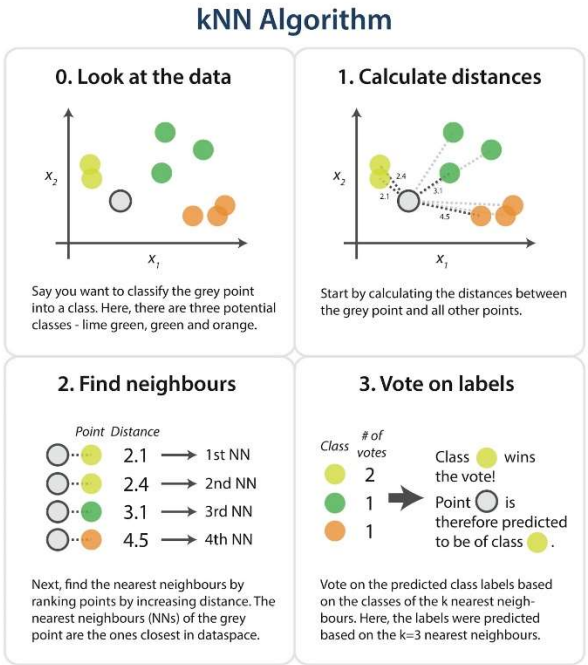


Ilustración 12: Explicación algoritmo KNN

En el caso de la librería de Surprise, el algoritmo trata de predecir el rating que se le va a dar al ítem dependiendo de los ratings de los K ítems más cercanos.

Surprise implementa 4 variaciones del algoritmo:

- KNN Basic:

  El funcionamiento del algoritmo es el descrito en este capítulo. La predicción 'r' se logra mediante la siguiente función:

$$\hat{r}_{ui} = \frac{\sum_{v \in N_i^k(u)} \text{sim}(u,v) \cdot r_{vi}}{\sum_{v \in N_i^k(u)} \text{sim}(u,v)}$$

Ecuación 3: Algoritmo KNN Básico

- KNN With Means:

  Este algoritmo difiere con el básico en que utiliza además la media de cada usuario para calcular el rating:

$$\hat{r}_{ui} = \mu_u + \frac{\sum_{v \in N_i^k(u)} \text{sim}(u,v) \cdot (r_{vi} - \mu_v)}{\sum_{v \in N_i^k(u)} \text{sim}(u,v)}$$

Ecuación 4: KNN con las medias de los usuarios

- KNN With Z Score:

  Además de utilizar la media de cada usuario para calcular el rating, en este algoritmo previamente normaliza los resultados por usuario.

$$\hat{r}_{ui} = \mu_u + \sigma_u \frac{\sum_{v \in N_i^k(u)} \text{sim}(u,v) \cdot (r_{vi} - \mu_v)/\sigma_v}{\sum_{v \in N_i^k(u)} \text{sim}(u,v)}$$

Ecuación 5:KNN con normalización

- KNN Baseline:

  Es el algoritmo básico teniendo en cuenta además un rating de base. El rating de base se calcula mediante las tendencias calculadas para cada usuario. [6]

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{v \in N_i^k(u)} \text{sim}(u,v) \cdot (r_{vi} - b_{vi})}{\sum_{v \in N_i^k(u)} \text{sim}(u,v)}$$

Ecuación 6:KNN Baseline

5.2. Algoritmos de similitud

Para el cálculo de las recomendaciones, hace falta calcular la similitud entre los ítems, ya que, como se ha explicado con los algoritmos de vecinos cercanos, es una parte fundamental para hallar el rating entre los ítems.

Hay diferentes formas de calcular la similitud entre un conjunto de ítems. En el capítulo anterior se ha mencionado la función coseno. Sin embargo, para los algoritmos KNN explicados, se ha podido observar que se dan mejores resultados mediante la similitud de Pearson.

5.2.1. Correlación de Pearson

Es una función que mide la relación estadística entre dos variables continuas. Puede tomar un rango de valores entre -1 y 1. Donde -1 indica que no hay una asociación positiva, es decir, a medida que un valor aumenta, el otro decrece. 0 indica que no hay asociación alguna entre las dos variables. 1 indica que la asociación es total, es decir, a medida que un valor aumenta, el otro valor aumenta de igual manera.

[6] Surprise referencia al paper "Factor in the neighbors: Scalable and accurate collaborative filtering", apartado 2.1 Baseline Estimates para el cálculo de este valor.

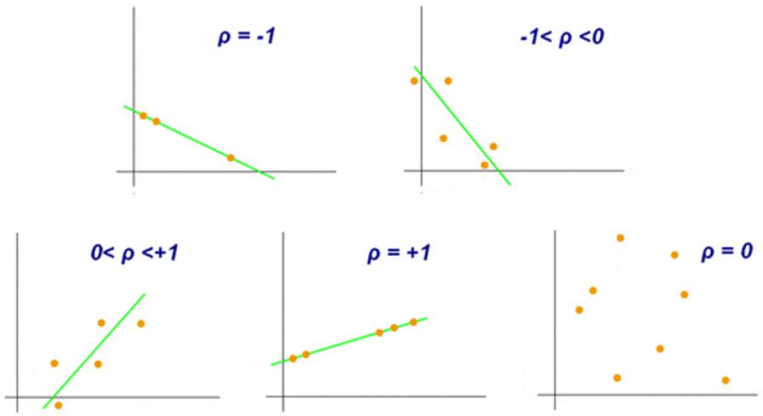


Ilustración 13: Correlación de Pearson

La forma de calcular la relación de Pearson es la siguiente.

$$\text{pearson\_sim}(u,v) = \frac{\sum_{i \in I_{uv}} (r_{ui} - \mu_u) \cdot (r_{vi} - \mu_v)}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - \mu_u)^2} \cdot \sqrt{\sum_{i \in I_{uv}} (r_{vi} - \mu_v)^2}}$$

Ecuación 7: Fórmula correlación de Pearson

6. Discogs

6.1.Historia

Discogs es una página que surgió en el año 2000 por Kevin Lewandoski, con el ánimo de convertirse en la biblioteca de publicaciones musicales más grande de internet. Es una página web de catalogación social, es decir, son los propios usuarios de la plataforma los que añaden las publicaciones, así como los datos de las mismas. A día de hoy más de 472.000 usuarios componen la mayor página de información musical del mundo.

Originalmente se diseñó para recoger solamente música electrónica, pero debido al éxito que estaba cosechando, se decidió ampliar en 2004 el abanico de géneros musicales a todos los existentes.

Además de almacenar toda la información de las publicaciones, tiene una sección en la web de compraventa de música en formato físico, donde los usuarios pueden intercambiarse, comprar y vender música de manera totalmente privada.

Como dato interesante, Discogs presume de ser el sitio web con mayor catálogo de vinilos de todo el mundo. Hoy en día, la colección de Discogs es de más de 260.000 publicaciones.

6.2.Información de las publicaciones

La información que se puede recoger de las publicaciones de Discogs es muy amplia:

- Se puede encontrar información técnica del disco. Como puede ser el género, los estilos, el número de pistas que compone la publicación, la duración de cada pista, etc.

- También se puede encontrar información comercial de la publicación. La compañía discográfica, la distribuidora, la productora, la diseñadora, etc.

- Información sobre el artista que ha compuesto la pieza musical. País, nacimiento, años en activo, géneros, estilos, etc.

Como se ha mencionado anteriormente, esta información es publicada por los usuarios que componen la plataforma. Existe un sistema de verificación para que la publicación pueda considerarse como válida o no válida. Este sistema se basa en la cantidad de votos positivos que recibe del resto de la comunidad, de manera que, al superar un umbral establecido, se considera validada y se coloca la etiqueta de “información Completa y correcta”.

La información se muestra a través de la página web de Discogs en formato HTML, aun así, Discogs ha desarrollado una API para poder acceder a los datos de manera programática mediante el uso de tecnologías REST (HTTP GET y HTTP POST).

6.3.API de Discogs

La API de Discogs está pensada para poder acceder a toda la información de la plataforma web disponible de manera pública y, en la mayoría de los casos, sin restricciones de uso.

Utiliza una interfaz basada en tecnologías REST. De esta manera es muy sencillo acceder a la información de la web y recogerla en formato JSON.

Discogs realiza cada mes un volcado de todos los datos relativos a publicaciones musicales, accesible de manera totalmente libre. Estas bases de datos están en formato XML. Se hacen los siguientes 4 volcados mensualmente:

- **Artists**: En este archivo se encuentra toda la información relativa a los artistas que tienen alguna publicación en Discogs:

- **Labels**: recoge la información acerca del sello discográfico del artista:

- **Masters**: se puede encontrar la información relativa a un conjunto de lanzamientos parecidos. Es decir, en caso de haberse lanzado un *EP* (Extended Play) y un videoclip de una misma canción, se agruparía en un único *master_release*. De esta forma, solo se mantendría la fecha del primer lanzamiento y se evitaría la redundancia de los datos. Se podría decir que en este archivo se agrupan las diferentes versiones de un mismo lanzamiento. Entre los diferentes tipos de lanzamiento, se pueden encontrar un EP, un videoclip, un vinilo, etc.

- **Releases**: En este archivo se encuentran todos los trabajos publicados por todos los artistas en la base de datos. Es una base de datos muy rica en información.

Además de los datos propios de las publicaciones, otra de las funcionalidades de la API de Discogs es la de acceder a la información relativa a los usuarios, siempre y cuando tengan la información de manera pública. Algunos de los datos de los usuarios a los que se puede acceder son:

- **Identity**: Datos personales del usuario, imágenes, colaboraciones, etc.

- **Collection**: Colecciones de cada usuario. Las colecciones en Discogs sirven para que cada usuario almacene publicaciones o grupos de música de manera que pueda acceder a la información de los mismos de forma más rápida. Además, permite la opción de recibir información sobre nuevos lanzamientos, novedades, etc. acerca de los elementos en la colección. Desde la colección de cada usuario se puede acceder a las votaciones que ha realizado sobre cada elemento.

- Wantlist: Es la lista donde los usuarios almacenan las publicaciones que no tienen en formato físico. Se utiliza para recibir información del mercado online. Al introducir una publicación en la lista de deseados, cuando el sistema detecta que otro usuario ha

puesto a la venta ese artículo, le llega un mensaje al usuario con el precio e información de venta del producto.

- **Lists**: Se permite al usuario crear listas para que almacene en ellas las publicaciones que le interesen. Por ejemplo, una lista podría ser “Discos de la década de los 80”.

Para acceder a estos datos, Discogs proporciona librerías propias en lenguajes diferentes como Python, Ruby, PHP y Node.js.

Además, para poder acceder con los mayores permisos posibles, la API permite identificarse en el sistema mediante el protocolo OAuth.

6.4. Protocolo OAuth

OAuth es un protocolo estándar normalmente utilizado para acceder a información de páginas web en las que se necesite unos permisos especiales. Creado en el año 2006 por Blaine Cook, para solucionar el problema de acceder a Twitter desde otros sistemas para poder manejar el flujo de Tweets.

El funcionamiento de este protocolo consiste en permitir el acceso a los recursos del servidor mediante una clave única (Token) propia para cada usuario, aprobada por el servidor. Para obtener la clave, generalmente el cliente deberá pedirla al servidor al que se quiere conectar.

Una vez obtenida la clave, el cliente podrá, mediante una API, acceder a los recursos del servidor. A continuación, se ilustra el funcionamiento del protocolo de manera simple y clara.

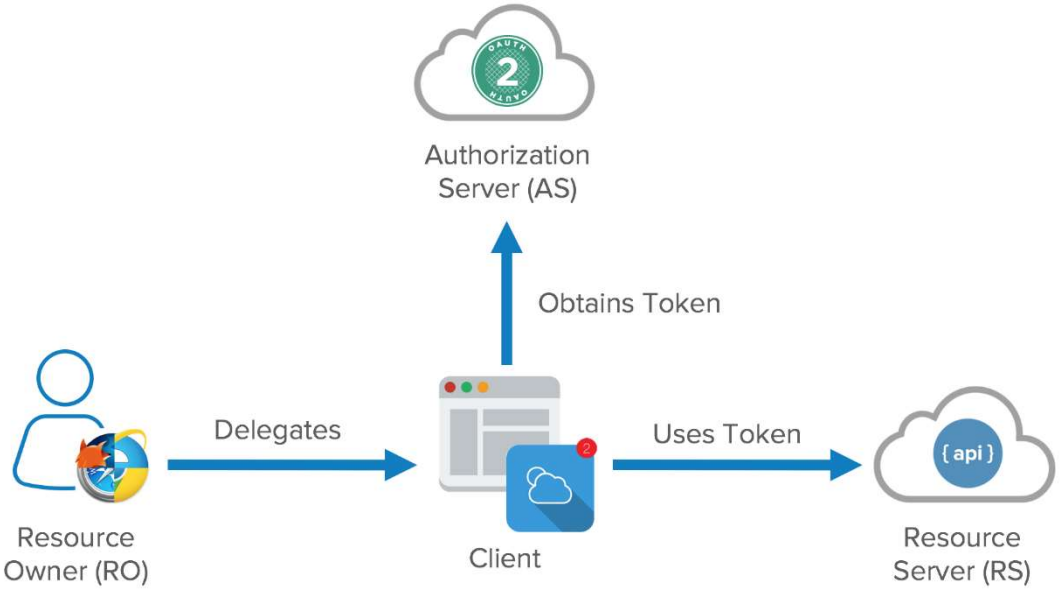


Ilustración 14: Funcionamiento del protocolo OAuth

7. Desarrollo del proyecto

En esta sección de la memoria se explicará el desarrollo del proyecto. Por un lado, se detallará el proceso de recogida de datos, transformación y aplicación de la librería Surprise para el sistema de recomendación basado en filtrado colaborativo. Por otro lado, se explicará la implantación del sistema de recomendación basado en el filtrado por contenido, comenzando por la implementación del clúster Spark, el tratamiento de los datos y la aplicación de la función de recomendación.

El proyecto, al estar dividido en dos partes diferenciadas, se explicará primero uno de los sistemas de filtrado y a continuación se explicará el segundo. Además, se detallará la estructura de archivos para que, durante el transcurso de la lectura de la memoria, se pueda acceder a los archivos que se están explicando en cada momento.

Los notebooks utilizados para desarrollar el sistema de recomendación colaborativo, son los que tienen el prefijo CLB, mientras que los notebooks propios del sistema de recomendación por contenido, son los del prefijo CNT. Estos archivos se pueden encontrar en el directorio Notebooks.

De igual manera, los ficheros resultantes del tratamiento de datos, filtrado, limpieza, etc. se pueden encontrar en el directorio TFM Files. Al igual que los Notebooks, están divididos con los prefijos CLB y CNT. El número hace referencia al Notebook del cual se obtienen los datos.

7.1. Sistema de recomendación basado en filtrado por contenido

En primer lugar, se detallarán los pasos para generar el sistema de recomendación basado en contenido. Como se ha explicado anteriormente, este tipo de sistemas de recomendación se basan en los propios datos de los elementos a recomendar. Es muy importante seleccionar los datos que pueden aportar valor al sistema de recomendación.

Este sistema de recomendación se consigue mediante la aplicación de la función del coseno para medir distancias entre vectores. Con los resultados de la función, se recomiendan los ítems que más próximos se encuentran al elemento del que se pide la recomendación.

A continuación, se detalla el proceso llevado a cabo para elaborar un sistema de recomendación, partiendo de la configuración de Spark, siguiendo por la recogida de los datos más interesantes y la ingeniería de atributos para poder utilizar, finalmente, el algoritmo que recomiende los ítems.

7.1.1. Configuración de Spark

Para la elaboración del sistema de recomendación por contenido se necesitaba en primer lugar descargar la parte más importante, los datos de las publicaciones. Como se ha explicado en el apartado de Discogs, están accesibles para descargar los volcados de los datos de los lanzamientos de la plataforma web.

El problema que se encontró es que los ficheros tenían un tamaño considerable, sin ir más lejos, el tamaño del fichero menor, es de 244MB, mientras que el tamaño del fichero más grande es de 36GB. Esto hizo indispensable la implantación de un sistema Big Data para el tratado de los datos.

Después de barajar varias opciones, se optó por la implantación de Spark. Como entorno de desarrollo se han utilizado los Notebook de Jupyter, ya que se pueden implementar las librerias de Spark de manera sencilla. El equipo donde se ha desarrollado el proyecto es un sistema operativo Windows 10 con un procesador Intel Core i5 de novena generación, 6 núcleos a 3,70GHz y una memoria de 16GB. Para la instalación y la configuración se utilizaron varios recursos web.[7]

Se decidió utilizar Apache Standalone como administrador del clúster. La idea original no contemplaba el uso de más de un equipo para el desarrollo del proyecto. La puesta en marcha en local de Apache Standalone era realmente sencilla ya que el propio modo Standalone está incorporado con Spark.

Por defecto, el modo Standalone utiliza todos los núcleos disponibles del sistema. La utilización de los núcleos hace posible que se puedan procesar los datos en paralelo. Durante el proyecto, se ha utilizado la configuración por defecto, excepto en algunas ocasiones que se ha reducido el número de núcleos en uso para evitar un sobrecalentamiento de un equipo que no está preparado para ello.

La forma de unir la configuración de Spark con el entorno de desarrollo es mediante SparkContext. Esta funcionalidad está accesible a través de la API de Spark, en este caso, en Python. En este objeto se define el clúster al que se tiene que conectar para poner en marcha las funcionalidades de Spark.

7.1.2. Recogida de datos desde los volcados de Discogs – CNT1

Una vez configurado Spark, el siguiente paso es cargar los datos y generar un dataset con los datos que más relevancia tienen a la hora de generar un sistema de recomendación por contenido.

[7] Guide to install Spark and use PySpark from Jupyter in Windows,

Como se ha comentado anteriormente, hay 4 archivos donde se han volcado los datos de las publicaciones, son los siguientes:

- **Artists**: En este archivo se encuentra toda la información relativa a los artistas que tienen alguna publicación en Discogs:
    - images: Imágenes del artista.
    - id: número de identificación del artista.
    - name: nombre del artista.
    - realname: nombre real del artista.
    - profile: breve descripción de la carrera del artista.
    - data_quality: indica si la información del artista está verificada.
    - namevariations: variaciones del nombre del artista por el que es conocido.
    - aliases: diferentes motes que recibe el artista.

- **Labels**: recoge la información acerca del sello discográfico del artista:
    - images: Imágenes del sello discográfico.
    - id: identificación del sello.
    - name: nombre del sello discográfico.
    - contactinfo: Dirección, teléfono e información de relevancia del sello.
    - profile: Breve descripción del tipo de música que lleva la productora.
    - data_quality: indica si la información del artista está verificada.
    - urls: páginas web del sello discográfico.
    - sublabels: Variaciones del nombre del sello discográfico.

- **Masters**: se puede encontrar la información relativa a un conjunto de lanzamientos parecidos. Es decir, en caso de haberse lanzado un EP (Extended Play) y un videoclip de una misma canción, se agruparía en un único master_release. De esta forma, solo se mantendría la fecha del primer lanzamiento y se evitaría la redundancia de los datos. Se podría decir que en este archivo se agrupan las diferentes versiones de un mismo lanzamiento. Entre los diferentes tipos de lanzamiento, se pueden encontrar un EP, un videoclip, un vinilo, etc. Los diferentes atributos que contiene el archivo masters son los siguientes:

    - main_release: Hace referencia al lanzamiento principal.
    - images: imágenes del lanzamiento principal.
    - artist: Información sobre el artista del lanzamiento principal.
    - genre: Género del lanzamiento principal.
    - style: Estilo del lanzamiento principal.
    - year: año del lanzamiento principal.
    - title: título del lanzamiento principal.
    - data_quality: Indica si la información del lanzamiento principal es correcta.
    - video: Diferentes videos sobre el mismo lanzamiento principal.

- **Releases**: En este archivo se encuentran todos los trabajos publicados por todos los artistas en la base de datos. Es una base de datos muy rica en información. A continuación, se seleccionan las características más relevantes para el proyecto:

    - format: De este atributo se obtiene el formato en el que fue grabado el lanzamiento. Es interesante disponer de este dato para poder filtrar por formato a la hora de realizar una recomendación.
    - country: País donde se grabó el disco. Las influencias de los productores a la hora de decidirse por un grupo u otro puede influir a la hora de recomendar.
    - tracklist: Lista de canciones incluidas en el trabajo. Mediante estos atributos se calcula el número de canciones y la duración del disco, EP o vinilo. Puede ser un factor a la hora de recomendar publicaciones.
    - company: De company interesa el atributo name. Name hace referencia al nombre de la productora que se puede obtener desde el archivo labels comentado en el apartado anterior.

Partiendo de los cuatro archivos descargados, se quiere contar con un único archivo final que englobe los atributos más importantes de cada publicación de cada artista.

Los atributos recogidos de cada uno de los archivos son los siguientes:

- **masters:**
    - artist
    - genre
    - style
    - year
    - title

- **releases**:
    - format
    - country
    - tracklist
        - number_tracks: generado de los atributos de tracklist.
        - duration: generado de los atributos de tracklist.
    - company
        - name

Como se puede comprobar, finalmente se genera el archivo resultante que engloba toda la información desde sólo dos archivos, masters.xml y releases.xml, ya que la información que aportan los otros dos archivos carece de valor para el propósito del sistema de recomendación.

En el Notebook *CNT1 – Cleaning and Merging Data* se puede encontrar el proceso para recoger los datos y generar un único dataset final llamado *MastersReleasesJoined.json*. Se ha escogido utilizar el formato JSON ya que algunas de las columnas contienen más de un valor. En

el siguiente punto se explica como se ha trabajado con las columnas que contienen más de un valor.

Ilustración 15: Muestra de una publicación recogida de los volcados de Discogs

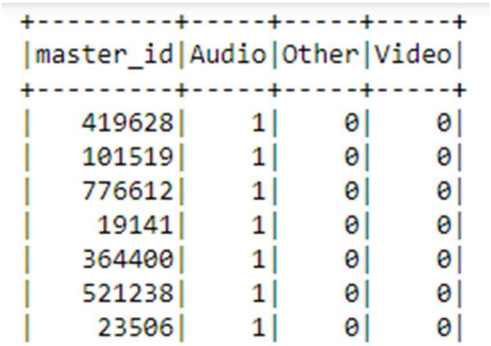
7.1.3. Ingeniería de atributos – CNT2

Una vez generado el dataset con los datos seleccionados para el sistema de recomendación, el siguiente paso ha sido el de transformar los datos de manera que se pueda calcular la similitud entre ellos y así hallar las recomendaciones de los elementos. En el Notebook *CNT2 – Feature Engineering* se pueden ver detalladamente las diferentes estrategias que se han utilizado para cada dato.

Para implementar las transformaciones sobre las columnas se ha utilizado la librería propia de Spark MLlib.

La columna **formats** se ha decidido optar por reducir el número de opciones a tres. Se han englobado en Audio las publicaciones cuyo formato es un reproductor de audio, a video los formatos de vídeo y a otros, los demás formatos. De esta forma, se puede utilizar la lógica del algoritmo **One Hot Encoder**. Este algoritmo se utiliza cuando se quiere representar variables categóricas en una manera numérica. De esta manera, pueden utilizarse como atributos para algoritmos que no aceptan variables de tipo categórica.

El algoritmo transforma las categorías en columnas y asigna un 1 en la columna de la categoría a la que pertenece el elemento. En caso contrario, si no pertenece a alguna de las columnas, se representa con un 0.



master_id	Audio	Other	Video
419628	1	0	0
101519	1	0	0
776612	1	0	0
19141	1	0	0
364400	1	0	0
521238	1	0	0
23506	1	0	0

Ilustración 16: Ejemplo de One Hot Encoding

La misma técnica se ha utilizado para representar de manera numérica la variable **genres.** Se han generado para la representación de los géneros 15 columnas.

El problema de utilizar esta técnica es que las operaciones matriciales ocupan mucha memoria en el sistema, por lo que la utilización de demasiadas columnas podía derivar en problemas de memoria.

Por lo tanto, aunque no sea la mejor estrategia, para las demás variables categóricas (styles, artists, companies y country), se ha utilizado la estrategia llamada String Indexer.

Este algoritmo asigna un índice a cada variable de tipo texto. Utiliza una estrategia donde el orden del índice lo asigna dependiendo del número de veces repetido el texto que se evalúa. Es decir, a la categoría más representada se le asigna el 0. Mientras que, a la categoría menos representada, se le asigna el índice mayor que pueda adquirir dependiendo del número de categorías.

Se han tratado también los problemas que han ocasionado las variables nulas. En el caso de la duración de las publicaciones, alguna de las entradas estaba vacía. Se ha optado por rellenar esos valores con la media de todas las demás publicaciones para que no sea relevante a la hora de sacar las recomendaciones.

Finalmente se han normalizado las columnas para que los distintos valores se expresen en una misma escala común. De esta forma, se pueden aplicar los datos a la función para que calcule la similitud entre los ítems. pueda calcularse la similitud entre elementos con una misma base. Para la normalización se ha utilizado la función StandardScaler de MLlib. El algoritmo transforma las variables de tal manera que la media de todas ellas sea 0 y la escala sea de -1 a 1.

Se ha generado un dataset final llamado *dataScaled.json* compuesto por tres columnas:

- **master**_id: identificador de la publicación.
- **title**: Título de la publicación
- **scaledFeatures**: Vector con las variables normalizadas.

```
{
    "master_id":81137,
    "title":"Greenbank Drive",
    "scaledFeatures":{"type":1,
                      "values":[0.13340853593140198,-0.7877696971513337,-0.6833990119234601,
```


Ilustración 17: Muestra del dataset con el vector de datos normalizado

7.1.4. Aplicación de la función del coseno – CNT3

En el último Notebook llamado *CNT3 - Calculing Similarity (Content Based Recommendation),* se implementa la función coseno para calcular los elementos más relacionados con el elemento escogido. Se ha escogido la publicación *Per Un Amico* porque en el sistema de recomendación colaborativo es la publicación que más interacciones tiene.

Para el calculo de la similitud, se recogen los datos del elemento seleccionado (la fila) y se calcula con todas las demás publicaciones (filas) la similitud coseno. Se utiliza una función lambda que va generando un objeto rdd, de esta manera se garantiza el cálculo en paralelo.

Finalmente, del objeto rdd se genera un dataframe que se ordena de manera descendente y se muestran las 10 publicaciones recomendadas para el elemento.

```
Las 10 publicaciones recomendadas son:
+------+-------+-----------------------------------------------+
|id    |sim    |title                                          |
+------+-------+-----------------------------------------------+
|313950|0.99697|Snuffy / Wells Fargo / Rhodomagnetics / Count Down|
|23151 |0.99418|Marrakech                                      |
|53118 |0.99382|Ars Longa Vita Brevis                          |
|22420 |0.99291|Station To Station                             |
|61688 |0.99159|Ornette On Tenor                               |
|248430|0.99045|Soprano Sax                                    |
|155249|0.9904 |Bottom Heavy 2008                              |
|47579 |0.99009|Rhapsody And Blues                             |
|14612 |0.98976|Saturday Night (Remix '94)                     |
|47901 |0.98944|Sunlight                                       |
+------+-------+-----------------------------------------------+
only showing top 10 rows
```

Ilustración 18: Recomendaciones para Per Un Amico (Sistema de recomendación por contenido)

7.2. Sistema de recomendación basado en filtrado colaborativo

Para esta primera parte de creación de un sistema de recomendación, lo más importante era conocer los límites de la librería Surprise, para recoger y tratar los datos de la manera más adecuada.

Como se ha mencionado en el apartado dedicado a Surprise, Surprise es un Framework basado exclusivamente en el filtrado colaborativo y precisa de ratings (votaciones) para calcular las recomendaciones.

Surprise requiere que los datos que aparezcan en el dataset sean:

- El identificador del usuario.
- El identificador del ítem.
- El rating del usuario al ítem.

Discogs es la plataforma de la que se han recogido estos datos y se ha hecho de la siguiente manera.

7.2.1. Recogida de datos desde Discogs

Como se ha mencionado en el apartado dirigido a Discogs, Discogs proporciona una API para acceder a los datos tanto de los usuarios como de los lanzamientos. En este caso, interesa recoger las votaciones de los usuarios.

Los usuarios disponen de un apartado en la web donde pueden crear colecciones con las publicaciones que les interesen y posteriormente compartirlas, filtrarlas o valorarlas.

La API de Discogs requiere del nombre de usuario para poder acceder a sus datos, sin embargo, no hay manera alguna de acceder a los nombres de los usuarios desde la API. Por eso, se ha tenido que utilizar un Marco de rastreo web (Web-Crawler) para recoger esos datos.

Para ello se ha utilizado Scrapy.

7.2.1.1. Scrapy – CLB0

Scrapy es un pequeño framework open source que permite a los desarrolladores automatizar la recogida de datos desde cualquier página web desde el propio código HTML. Este framework, escrito en Python y lanzado en 2008, utiliza “arañas” (spiders) para la recogida de información.

El flujo que sigue Scrapy para la recogida de información es el siguiente:

- En primer lugar, se definen las páginas de las que se quiere sacar información.

- A continuación, utilizando las etiquetas propias de HTML se indica la información que se quiere extraer.

- Finalmente, se indica si se quiere realizar alguna acción en la página actual para seguir recogiendo datos. Por ejemplo, una acción podría ser la de clicar en un botón que lleve a otra página para poder continuar con la extracción de información.

Además, Scrapy permite configurar el modo en el que se extraerá la información. Se puede configurar el tiempo de espera entre petición y petición, el número de hilos en paralelo que recogerá información, etc.

Para este proyecto, se han definido unos hilos del foro propio de Discogs como páginas webs donde buscar[8]. Se han seleccionado algunos de los hilos con más comentarios. Debido a las limitaciones de peticiones de la página de Discogs, se ha configurado el *Spider* para que recoja datos en paralelo con dos hilos y con un tiempo de espera entre petición y petición de 3 segundos. De esta manera se ha conseguido burlar el bloqueo de Discogs. La configuración del Spider se puede encontrar en el fichero *settings.py* del directorio *discogsScrapy*.

Utilizando las etiquetas HTML del código de los posts, se ha seleccionado el nombre de usuario del autor de cada post en el hilo.

Al tratarse de un hilo compuesto por más de una página, se ha definido como acción, que una vez ha recogido los nombres de los usuarios, automáticamente se redirija a la próxima página.

La definición de las etiquetas en las que buscar el nombre de usuario y el botón de siguiente página están, así como la definición de las páginas web, están especificadas en el archivo *usernameSpider.py* dentro del directorio *spiders.*

Scrapy permite exportar los datos que va recogiendo en formato json. En este caso, el archivo exportado se llama *usernames.json*.

[8] Please ping other users here regarding an issue within the database., HELP Wanted in Merge requests!!, The last item you sold thread y Requests for votes on your submissions - post them here only please

Ilustración 19: Muestra de algunos usuarios recogidos mediante Scrapy

Una vez terminado el Spider, el número total de usuarios recogidos ha sido 41.352. Sin embargo, no es el número real de usuarios, ya que hay muchos usuarios que han escrito más de un post en los hilos que se han recorrido y su nombre se encuentra repetido en la lista.

7.2.2. Limpieza de usuarios repetidos y recogida de valoraciones – CLB1

En el Notebook CLB1 inicialmente se eliminan los usuarios repetidos del fichero generado por Scrapy y a continuación se recogen las votaciones de los usuarios de la lista.

Después de identificarse correctamente en el sistema de Discogs mediante OAuth, se ejecuta un script que utiliza la API de Discogs para recoger las valoraciones de los usuarios. Debido a las limitaciones propias de la API, se ha tenido que recoger los datos en diferentes momentos. Discogs bloquea la IP del equipo que realiza más peticiones que una cantidad fijada en un determinado tiempo. Se han ido generando archivos temporales con los datos de usuario y su valoración y finalmente se ha juntado toda la información en único archivo llamado *usernameRatingsMerged.csv*

La estructura del fichero es la siguiente:

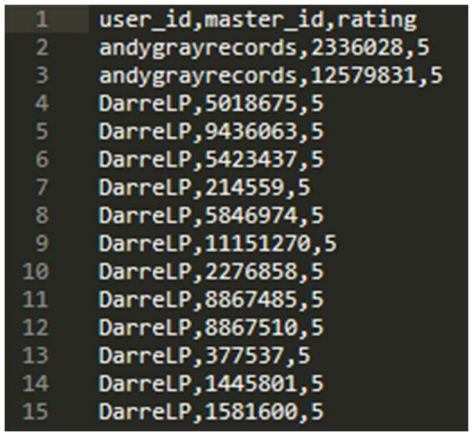


Ilustración 20: Muestra de las valoraciones de los usuarios

Como se puede observar, el archivo CSV contiene el formato que pide Surprise para la implementación de los algoritmos de predicción.

- **user_id**: Nombre del usuario. (se ha decidido utilizar el propio nombre de usuario en vez del identificador).
- **master_id**: El identificador de la publicación (un número único por publicación).
- **rating**: La valoración que ha otorgado el usuario a la publicación. (Del 1 al 5)

7.2.3. Análisis de los datos obtenidos – CLB2

Mediante este Notebook se ha analizado la cantidad de usuarios distintos que contiene la lista de votaciones, así como el número total de elementos distintos valorados.

El fichero con las votaciones tiene un total de 535 usuarios que han emitido en total 379.952 valoraciones. De esa cantidad de valoraciones, se calcula además el número de elementos distintos, 311.570.

Como se puede observar, la cantidad de elementos distintos es extremadamente grande. Este hecho ha generado problemas a la hora de procesar los datos en el equipo ya que, uno de los pasos para calcular las recomendaciones con Surprise, es generar una matriz de similitud entre elementos y la memoria del equipo no era lo suficientemente potente para tal propósito.

Las votaciones obtenidas desde Discogs, recogen todo tipo de publicaciones, tanto publicaciones menores como pueden ser videoclips, singles, EPs, como publicaciones “master”, como, por ejemplo, discos, vinilos, conciertos, etc. Por lo tanto, para solucionar el problema se ha optado por utilizar solo las publicaciones master que componen los datos del sistema de recomendación por contenido. De esta manera se pierde información que puede ser relevante

a la hora de predecir los ratings y recomendar publicaciones, aun así, se igualan las publicaciones que aparecen en ambos modelos.

Finalmente, el número de votaciones con el que se van a calcular las recomendaciones es de 15.850. El número de usuarios se reduce también hasta 319 y el número de elementos únicos es de 11.406.

7.2.4. Implementación de Surprise – CLB3

Una vez filtrados los datos y transformados en la manera que Surprise puede interpretar, se implementan los algoritmos de Surprise para crear el sistema de recomendación por filtrado colaborativo.

Como se ha mencionado en el apartado de Surprise, proporciona 4 familias de algoritmos distintas. En este Notebook se han entrenado las 4 familias para evaluar cuál de ellas es la que mejores resultados proporciona.

Antes del entrenamiento de los algoritmos, se ha utilizado el algoritmo GridSearchCV() para probar diferentes hiperparámetros y seleccionar el que mejores resultados ofrecía. Para ello, se ha seleccionado el 75% de los datos como entrenamiento dejando un 25% restante al margen para probar los modelos. Como parámetro del algoritmo, se ha definido que divida en 5 los datos de entrenamiento y utilice la valoración cruzada.

Una vez establecidos los mejores hiperparámetros para cada algoritmo, se han entrenado sobre todo el conjunto de datos utilizando para ello la validación cruzada con 5 particiones. Según se puede observar en la siguiente imagen, el algoritmo que mejores resultados ofrece es el algoritmo KNNBaseline.

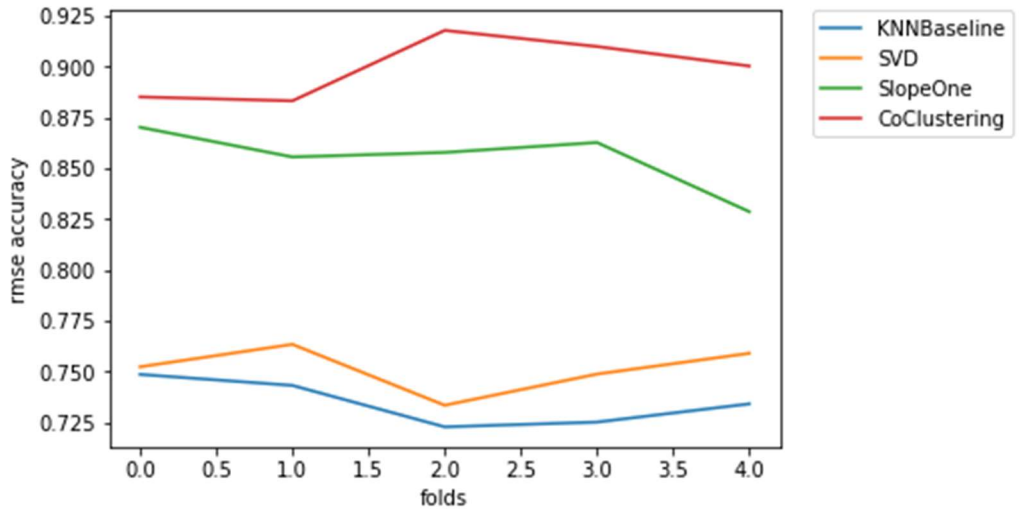


Ilustración 21: RMSE de los algoritmos de Surprise

Los algoritmos KNN proporcionan la funcionalidad de obtener los K ítems más relacionados, es decir, obtener las K recomendaciones en este caso. Por ello, se analizan las diferentes variantes que ofrece Surprise del algoritmo KNN para ver cuál de ellos ofrece mejores resultados.

Se ha seguido la misma estrategia que para hallar el mejor algoritmo entre las diferentes familias. También se han utilizado los mismos parámetros en las 4 variantes para buscar mediante gridSearchCV() los mejores parámetros. Los resultados son los siguientes.

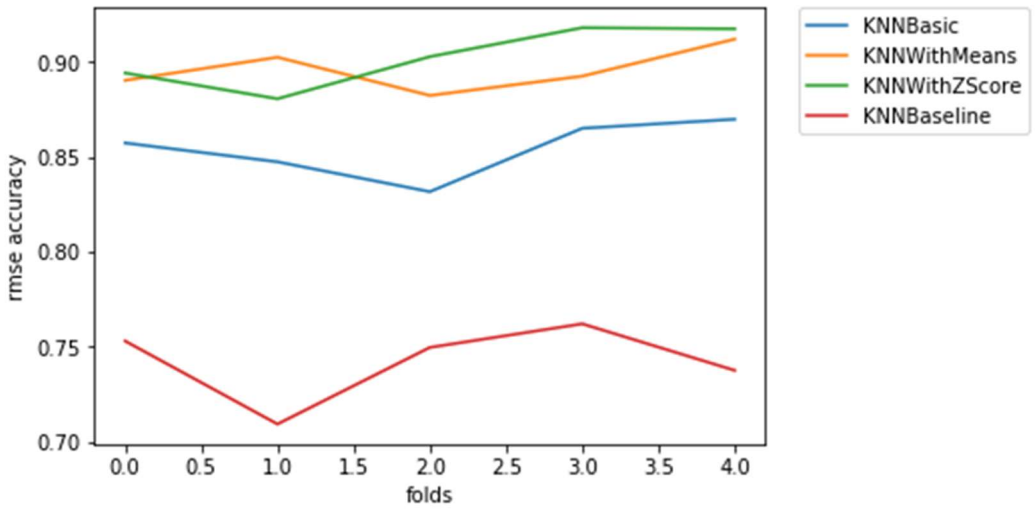


Ilustración 22:RMSE de los algoritmos KNN

Como se puede observar, el algoritmo KNNBaseline es el que mejores resultados obtiene y es el que se utilizará a continuación para hallar las K publicaciones recomendadas.

Para poder tener los mejores resultados, se ha utilizado como publicación de la que obtener las recomendaciones, la publicación con más ratings. Se trata de *Per Un Amico*, publicación que ha sido valorada 17 veces.

Surprise proporciona la función get_neighbors() como método propio de los algoritmos KNN, recoge como parámetros el identificador del elemento del que se quiere recoger los vecinos más cercanos y el número de elementos que se quieren mostrar, en este caso, 10.

```
Las 10 recomendaciones para Per Un Amigo:
Cameosis
Even If You Don't
Die Barke Mit Der GlÃ□Â¤sernen Fracht
( )
QuotÃ□Â¤reglÃ□Â¤r
Tunnel Trance Force Vol.57
Get On Up
Adagio Furioso
Instinctual
Eternal (New Remix Edition)
```

Ilustración 23: Recomendaciones para Per Un Amico

8. Conclusiones y trabajo futuro

Una vez terminado el proyecto e implementado los dos sistemas de recomendación, hay que destacar una serie de limitaciones y problemas que se han encontrado a lo largo del proyecto que han dificultado su desarrollo.

Uno de los objetivos que se proponía al inicio del proyecto era el de crear un sistema de recomendación híbrido juntando los sistemas generados durante el proyecto. No se ha podido llevar a cabo este objetivo ya que Surprise, a pesar de ser una librería muy potente, no considera la funcionalidad de combinar sistemas de recomendación colaborativos con sistemas de recomendación por contenido.

Sí permite la posibilidad de generar un sistema de recomendación híbrido juntando dos o más modelos generados con Surprise, creando un algoritmo propio de generación de modelos. Aun así, el desarrollo de un algoritmo propio no entraba dentro del alcance del proyecto y requería de un estudio mucho más exhaustivo del tema. Podría ser una línea de futuro el desarrollo de un algoritmo que utilice la matriz de similitud del sistema de recomendación basado en contenidos, junto con los modelos propuestos por Surprise.

Seguro que el error producido por el modelo de recomendación disminuiría utilizando un sistema híbrido, ya que, tal y como se demostró en el concurso de Netflix, el grupo ganador empleó 107 modelos diferentes reduciendo un 10% el error.[9]

Hay que destacar la facilidad para implementar los algoritmos de Surprise, debido en parte, a la documentación tan clara que se facilita.

El uso de un solo equipo ha limitado la capacidad de computo de la librería de Surprise. Se ha tenido que reducir el volumen de datos para poder generar el sistema de recomendación. El archivo original de votos recogidos era de más de 300.000 filas. Incluso se podrían haber obtenido más mediante la API de Discogs, pero vistas las limitaciones, no tenía mucho sentido seguir recogiendo información.

Una manera de mejorar el recomendador, sería crear un clúster de varios equipos con los que mejorar la capacidad de computación y entrenar un modelo con más datos y más distribuidos. Esa es otra de las limitaciones a destacar, el hecho de disponer de tantas publicaciones, ha hecho que la gran mayoría de publicaciones solo tuviesen 1 valoración, por lo que calcular las recomendaciones se hacía muy complicado.

También se podría mejorar el recomendador dedicando más tiempo a la búsqueda de hiperparámetros. Tal y como se comenta en el paper de M. Claese y B. de Moor, la mejora de los parámetros puede mejorar significativamente el rendimiento de un modelo de predicción o de recomendación en este caso.

Respecto al sistema de recomendación por contenido, se podría mejorar la técnica de tratar las variables categóricas. Una de las técnicas que mejores resultados da es la que se ha

[9] Enlace a la solución utilizada https://www.netflixprize.com/assets/GrandPrize2009_BPC_BellKor.pdf

implementado con los valores con pocas categorías (formato y el género). La técnica de One Hot Encoding. Sin embargo, debido a las limitaciones técnicas, el uso de tantas columnas hubiese derivado en problemas de memoria.

Además, si bien es cierto que los atributos seleccionados para generar el sistema de recomendación por contenido son valiosos, haber ponderado el peso, habría hecho un sistema mejor balanceado. Es decir, probablemente la duración del disco no es tan relevante como puede ser el género, el estilo o el país del artista.

En este proyecto, se han tratado muchos de los temas aprendidos durante el curso, aquellos que tienen que ver con el procesamiento de grandes volúmenes de datos con Spark, aplicaciones de técnicas de Machine Learning y Scrappeo de datos desde la web.

El objetivo final se ha cumplido ya que se han generado dos sistemas de recomendación básicos que pueden dar mucho de sí, debido a la gran cantidad de información que se puede obtener de Discogs.

9. Bibliografía

1. Hanani, U., Shapira, B., & Shoval, P. (2001). Information filtering: Overview of issues, research and systems. *User modeling and user-adapted interaction*, *11*(3), 203-259.

2. Karlgren, J. (1994). Newsgroup Clustering Based On User Behavior-A Recommendation Algebra. *SICS Research Report*.

3. Shi, Y., Larson, M., & Hanjalic, A. (2014). Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Computing Surveys (CSUR)*, *47*(1), 3.

4. Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE transactions on information theory*, *13*(1), 21-27.

5. Do, Minh-Phung & Nguyen, Dung & of Loc Nguyen, Academic Network. (2010). Model-based approach for Collaborative Filtering.

6. Pazzani, M. J., & Billsus, D. (2007). Content-based recommendation systems. In *The adaptive web* (pp. 325-341). Springer, Berlin, Heidelberg.

7. Ramos, J. (2003, December). Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning* (Vol. 242, pp. 133-142).

8. Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern information retrieval* (Vol. 463). New York: ACM press.

9. Çano, E., & Morisio, M. (2017). Hybrid recommender systems: A systematic literature review. *Intelligent Data Analysis*, *21*(6), 1487-1524.

10. Takács, G., Pilászy, I., Németh, B., & Tikk, D. (2008, August). A unified approach of factor models and neighbor-based methods for large recommender systems. In *2008 First International Conference on the Applications of Digital Information and Web Technologies (ICADIWT)* (pp. 186-191). IEEE.

11. Koren, Y. (2009). The bellkor solution to the netflix grand prize. *Netflix prize documentation*, *81*(2009), 1-10.

12. Baptiste Rocca. (2019). Introduction to recommender systems. EU.: Towards Data Science. Recuperado de https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada

13. Nikita Sharma. (2019). Recommender Systems with Python. HeartBeat. Recuperado de https://heartbeat.fritz.ai/recommender-systems-with-python-part-i-content-based-filtering-5df4940bd831

14. Bartosz Góralewicz. (2018). The TF*IDF Algorithm Explained. Onely. Recuperado de https://www.onely.com/blog/what-is-tf-idf/

15. Domonkos Tikk. (2017). How can I combine recommender system outputs (user item matrix) via an ensemble? Budapest, Hungary. Quora. Recuperado de https://www.quora.com/How-can-I-combine-recommender-system-outputs-user-item-matrix-via-an-ensemble

16. Dean, J., & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. *Communications of the ACM*, *51*(1), 107-113.

17. Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., ... & Ghodsi, A. (2016). Apache spark: a unified engine for big data processing. *Communications of the ACM*, *59*(11), 56-65.

18. Biswas, B., Chaudhury, T. H., & Mohammad, S. N. A Fast and Scalable Recommender System using Collaborative Filtering Technique in Python Scikit-learn.

19. Nicolas Hug. (2016). Surprise. Columbia, United States. Recuperado de http://surpriselib.com/

20. Luke Saunders. (2019). Discogs: what is it, where it came from and how to use it. happymag. Recuperado de https://happymag.tv/discogs-what-is-it-where-it-came-from-and-how-to-use-it/

21. Discogs. (2018). Normas de la Base de Datos. Discogs. Recuperado de https://support.discogs.com/hc/es/articles/360005055593-Normas-de-la-Base-de-Datos-20-Normas-de-votaci%C3%B3n#Voting

22. Discogs. (2018). Condiciones del uso del API. Discogs. Recuperado de https://support.discogs.com/hc/es/articles/360009334593-API-Terms-of-Use

23. Hardt, D. (2012). The OAuth 2.0 authorization framework.

24. Matt Raible. (2017). What the Heck is OAuth?. okta. Recuperado de https://developer.okta.com/blog/2017/06/21/what-the-heck-is-oauth

25. Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, *46*(3), 175-185.

26. Koren, Y. (2010). Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, *4*(1), 1.

27. Benesty, J., Chen, J., Huang, Y., & Cohen, I. (2009). Pearson correlation coefficient. In *Noise reduction in speech processing* (pp. 1-4). Springer, Berlin, Heidelberg.

28. Dataflair team. (2018). Apache Spark Cluster Managers – YARN, Mesos & Standalone. DataFlair. Recuperado de https://data-flair.training/blogs/apache-spark-cluster-managers-tutorial/

29. Dataflair team. (2018). Learn SparkContext – Introduction and Functions. Recuperado de https://data-flair.training/blogs/learn-apache-spark-sparkcontext/

30. Rodríguez, P., Bautista, M. A., Gonzalez, J., & Escalera, S. (2018). Beyond one-hot encoding: Lower dimensional target embedding. *Image and Vision Computing*, *75*, 21-31.

31. Spark. StringIndexer. Recuperado de https://spark.apache.org/docs/latest/ml-features#stringindexer

32. Donders, A. R. T., Van Der Heijden, G. J., Stijnen, T., & Moons, K. G. (2006). A gentle introduction to imputation of missing values. *Journal of clinical epidemiology*, *59*(10), 1087-1091.

33. Claesen, M., & De Moor, B. (2015). Hyperparameter search in machine learning. *arXiv preprint arXiv:1502.02127*.