Grado en Ingeniería Informática, Universidad de León.

Recomendador de libros

Realizado por David González García

Diciembre de 2019

Índice:

Introducción:

La idea principal de este proyecto es el de la creación de un programa capaz de recomendar libros a un usuario. Para realizar la recomendación, el programa navegará la base de datos teniendo en cuenta los gustos y preferencias del usuario, como que libros ha leído, que géneros le gustan más, que libros ha puntuado mejor etc. Para ello se utilizará Neo4J como base de datos de grafos, y para realizar la recomendación se utilizarán algoritmos de detección de la comunidad y filtrado colaborativo

Este proyecto ha supuesto un gran esfuerzo personal debido a que la mayoría de las tectologías y técnicas que se han utilizado en él han sido aprendidas desde cero, es decir, no tenía experiencia previa en ellas. Sin embargo y desde un punto de vista personal, estoy muy satisfecho por el trabajo realizado y por los conocimientos adquiridos. Este trabajo me ha servido para descubrir el mundo de la ciencia de datos, un mundo totalmente desconocido para mí, y que ha resultado ser infinitamente interesante, con un futuro brillante y millones de posibilidades por aprovechar.

Considero que este trabajo es un trabajo bastante innovador ya que busca algo que de momento no existe que es el de realizar una recomendación de libros automática y personalizada para cada usuario en función de la información que te tiene del mismo. Si bien planteamientos similares existen en otros sectores como puede ser el de la música con Spotify, o de venta de productos como en Amazon. No existe nada similar en el mundo de los libros. Otros proyectos intentan algo parecido, pero desde un enfoque muy distinto, que es el de la creación de “redes sociales” de lectores, donde usuarios pueden opinar sobre los libros leídos y ver las opiniones de otros, a la vez que seguir a ciertos usuarios, pero no ofrecen ningún medio de recomendación automática.

Objetivos:

El objetivo de este proyecto es el de resolver el problema que se plantea una persona cuando ha acabado de leer un libro y no sabe con que continuar. Normalmente cuando esto ocurre, la persona suele buscar en libros del mismo autor o temática similar y rara vez se aventura con libros distintos al perfil que suele leer. Incluso cuando pregunta a un librero, este no puede saber con seguridad los gustos de esa persona y únicamente puede limitarse a recomendar libros similares al que el lector acaba de leer o suele comprar.

Sin embargo, este proyecto busca explorar otras maneras de recomendar un libro a un lector, y es por ello por lo que, en vez de basar su recomendación en las características de los libros, basa la recomendación en las características de la persona, abriendo la puerta a libros de estilos totalmente distintos que nunca pasarían por la cabeza del lector al no ser parecidos al perfil de libros que suele leer.

Para realizar esta recomendación, se utilizan algoritmos de filtrado colaborativo basados en el usuario, así como algoritmos de detección de comunidad. Además, todos estos trabajan sobre una base de datos de grafos implementada en Neo4J y una interfaz gráfica web creada mediante HTML y PHP.

Lo que se busca con todo esto es ofrecer un método automático de recomendación que le diga a un usuario, en función de los libros que ya ha leído y puntuado, que otros libros le podrían interesar.

Herramientas:

Para la realización de este proyecto se han utilizado una gran variedad de herramientas, además de explorado muchas otras que por distintos motivos han sido rechazadas. Las herramientas que han sido utilizadas son las siguientes:

Neo4J: Neo4j es un software libre de base de datos orientado a grafos, implementado en java y lanzado en 2007. Actualmente ocupa el primer puesto en bases de datos de grafos, y el puesto numero 22 en cuanto al total de las bases de datos. Utiliza Cypher para realizar las consultas, que es un lenguaje muy intuitivo fácil de encadenar, lo que hace que trabajar con el sea una gran ventaja. La principal ventaja que tiene una base de datos de grafos frente a una relacional es la facilidad de navegación por el mismo, la flexibilidad a la hora de definir atributos, posibilidad de recorrer la base de datos de forma jerárquica, etc.

Este tipo de bases de datos es el más popular en el campo de la ciencia de datos pues permite la construcción de los conocidos como "grafos de conocimiento" y además muestra de una manera muy visual las relaciones entre las distintas entidades. Este tipo de bases de datos se componen de Nodos (Entidades) y Aristas (Relaciones). A diferencia de las bases de datos relacionales, no existen tablas, sino etiquetas. De esta manera resalta la importancia de la entidad (nodo) frente a la etiqueta que pertenece. En nuestro caso particular es muy útil ya que lo que nos interesa es navegar entre los distintos nodos para así poder realizar una recomendación.

HTML: es el lenguaje standard para el desarrollo de páginas web. Es un lenguaje muy sencillo e intuitivo, pensado para el diseño de interfaces gráficas. Además, permite la utilización de estilos CSS.

Brackets: Brackets es un framework orientado al desarrollo de sitios web. Soporta HTML, PHP y JavaScript y además permite la instalación de plugins que hacen el trabajo mucho mas sencillo. Una de sus características estrella es que permite la visualización en directo de cualquier cambio realizado en el código, por lo que no es necesario refrescar la página para poder comprobar los cambios.

PHP: Utilizaremos PHP para interactuar con la base de datos mediante la librería de Graphaware*(https://github.com/graphaware/neo4j-php-client), y para desarrollar y aplicar el algoritmo propio de recomendación. PHP es un lenguaje de programación de propósito general, y que ejecuta código del lado del servidor (esto es, puede intercambiar información entre el cliente y el servidor) que además se complementa a la perfección con HTML.

Graphaware: Es una librería desarrollada por la comunidad de Neo4j que permite la conexión de una base de datos hospedada en Neo4J con el sitio web hospedado en un servidor. Es una librería de implementación sencilla, con una guía muy detallada y en constante actualización. Su facilidad de integración, conexión e interacción la hacen el candidato idóneo para cubrir esta parte del proyecto.

XAMPP: es un paquete de software libre que permite la implementación de servidores web apache, además de gestión de bases de datos MySQL e interpretes de lenguajes script como PHP y Perl. Es de fácil instalación e interacción y se ajusta a la perfección a lo que estamos buscando.

Apache: servidor web implementado por XAMPP en el cual hospedaremos nuestra página web que hará las veces de interfaz gráfica de la aplicación.

Trabajo:

Como ya hemos mencionado anteriormente, el objetivo de este trabajo es el de proporcionar una recomendación fiable de un libro a un usuario en función de las notas otorgadas por otros usuarios de características similares. Hay que destacar que este tipo de algoritmos funciona mejor cuanto mayor sea el tamaño de la base de datos. Debido al reducido tamaño de la base de datos que estamos utilizando actualmente, y al ser las puntuaciones asignadas totalmente aleatorias, no se puede medir con exactitud el grado de acierto del algoritmo. Si bien este algoritmo es capad de puntuar un libro para un usuario en función de las notas de otros usuarios, podría mejorarse la predicción con la implementación de más algoritmos (como ejemplo, Spotify utiliza hasta 3 algoritmos a la hora de proporcionar la recomendación de una canción). Algo que si se puede ver a simple vista es que cuantos mas usuarios participan en el algoritmo, mas acertada y lógica es la recomendación.

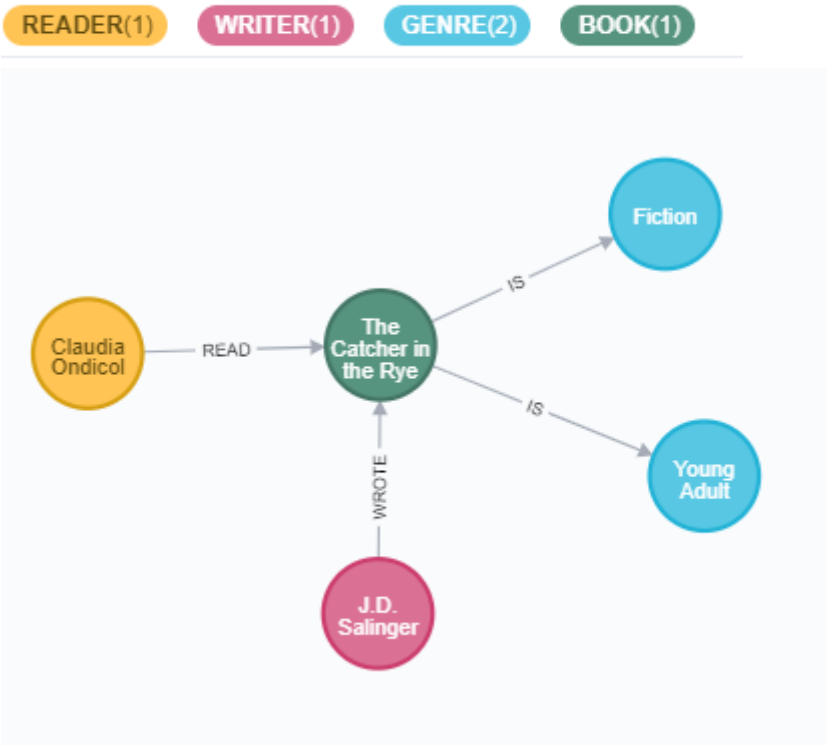
La estructura de este trabajo se compone de 3 partes principales, la base de datos, que contiene toda la información que vamos a utilizar, el sitio web, que sirve como interfaz gráfica para interactuar con la propia base de datos y para lanzar el algoritmo, y el algoritmo propio, que es la parte que más tiempo y dedicación se ha llevado en este proyecto.

Base de datos

La base de datos que se decide implementar es una base de datos basada en nodos, mediante el software proporcionado por Neo4j.

La base de datos estará compuesta por nodos de hasta 4 tipos diferentes: Lectores, Escritores, Libros y Géneros, conectados entre si mediante relaciones. Hay que añadir que, si bien los nodos con los que trabaja el algoritmo por el momento son únicamente los de Libro y Lector, ya que el algoritmo que utilizamos es un algoritmo de filtrado colaborativo basado en el usuario, los otros dos nodos (Escritor y Genero) pueden ser utilizados para realizar recomendaciones basadas en el objeto (en este caso el nodo Libro). Además, a esta estructura básica se le pueden añadir más nodos que proporcionen información adicional, como puede ser el país de los usuarios para realizar recomendaciones por región.

Por lo que, de manera gráfica, la estructura de la base de datos seguiría el siguiente esquema:



Como se puede observar, el nodo central sobre el que gira todo el trabajo y el algoritmo es el nodo de tipo libro. Cada libro tendrá uno o varios lectores, pertenecerá a uno o varios géneros y tendrá un solo escritor. En base a esto se realizará la recomendación.

Para poblar la base de datos con libros, se recurrió a kaggle. Kaggle es una comunidad online que permite a los usuarios publicar y compartir datasets, además de trabajos relacionados la ciencia de datos, por lo que es el lugar idóneo para buscar una base de datos ya creada que contenga libros y sus autores, además de otra información al respecto. Para este trabajo se escogió la base de datos de goodreads-best-books (https://www.kaggle.com/meetnaren/goodreads-best-books). Se eligió este data set debido a que es uno de los más completos que se encontró, que contenía información suficiente (únicamente se utilizaron los 200 primeros libros) y que no contenía celdas en blanco.

Pero antes de introducir la información en la base de datos de Neo4J fue necesario realizar una inspección y selección manual. De esta manera se pudieron encontrar, por ejemplo, libros duplicados, dobles autores para un mismo libro (Autor y seudónimo) o colecciones de libros. Todos los ejemplos mencionados anteriormente son datos que NO queremos dentro de nuestra página web ya que podrían producir inconsistencias.

Por ello, se realizó una depuración y se introdujeron todos los datos que sí son de interés en un archivo CSV, nombrado como BOOKS_AND_WRITERS.csv

Además, aparte de autores y libros, necesitamos poblar la base de datos con usuarios. Para ello se crean 50 usuarios, a los cuales se asignan de manera totalmente aleatoria 20 libros a cada uno con sus respectivas notas, también aleatorias. Todo esto se introduce en otro archivo CSV nombrado como USERS.csv.

Para importar los datos de ambos archivos csv en nuestra base de datos deberemos hacer lo siguiente:

1º Crear una nueva base de datos

2º Añadir un nuevo grafo

3º Iniciar el grafo y abrir el browser

4º Escribir los comandos:

```
LOAD CSV WITH HEADERS FROM 'file:///BOOKS_AND_WRITERS.csv' AS line
FIELDTERMINATOR ';'

UNWIND split(line.Genre, '|') AS genres

MERGE(w:WRITER {name:line.Writer})

MERGE(b:BOOK {title:line.Title, pages:line.Pages, rating:line.Rating,
rating_count:line.Rating_Count, review_count:line.Review_Count} )

MERGE(g:GENRE {genre: genres})

MERGE(w)-[:WROTE]->(b)

MERGE (b)-[:IS]->(g)
```

Para los libros, géneros y escritores

```
LOAD CSV WITH HEADERS FROM 'file:///USERS.csv' AS line FIELDTERMINATOR ';'
MERGE(n:READER {name:line.USERS1})
MERGE(m:READER {name:line.USERS2})
MERGE(b1:BOOK {title:line.BOOKS1})
MERGE(b2:BOOK {title:line.BOOKS2})
MERGE(n)-[r1:READ]->(b1)
ON CREATE SET r1.grade=line.GRADE1
ON MATCH SET r1.grade=line.GRADE1
MERGE(m)-[r2:READ]->(b2)
ON CREATE SET r2.grade=line.GRADE2
ON MATCH SET r2.grade=line.GRADE2
```

Para el conjunto de los usuarios.

Una vez poblada nuestra base de datos, obtenemos la obtenemos los siguientes datos:

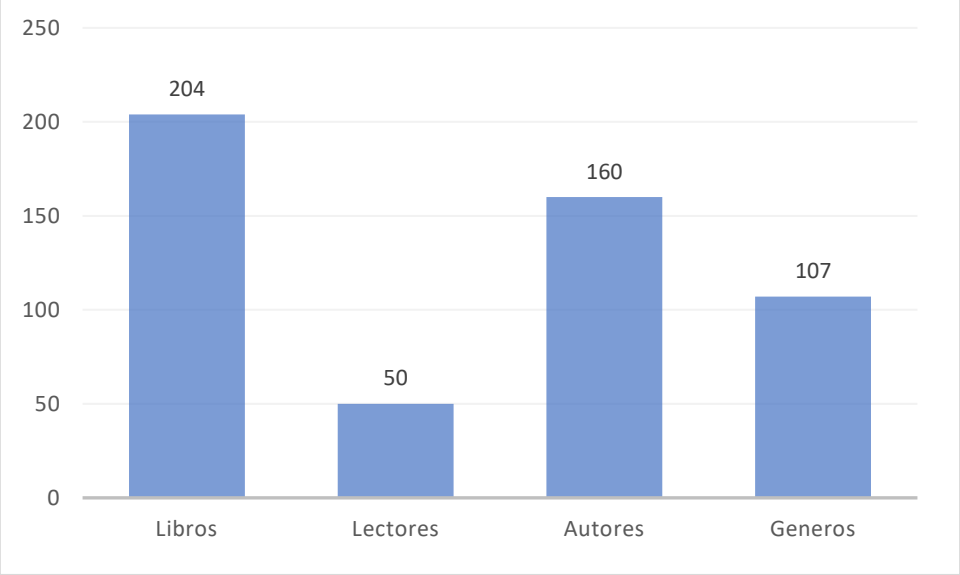


Tabla 1 con en numero de nodos perteneciente a cada una de las 4 clases

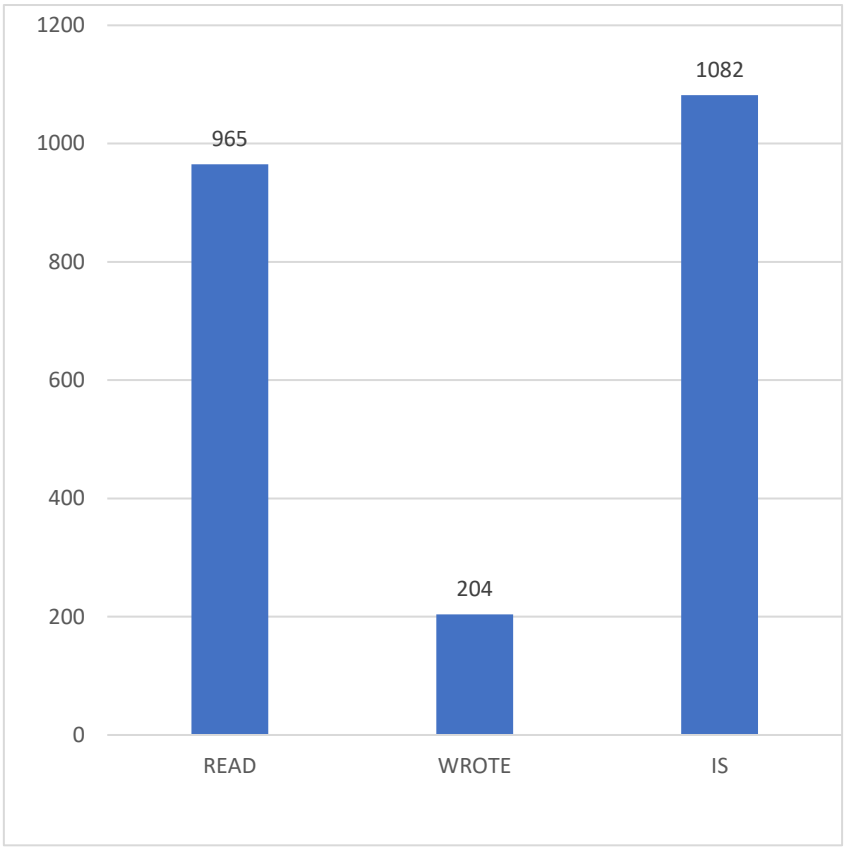
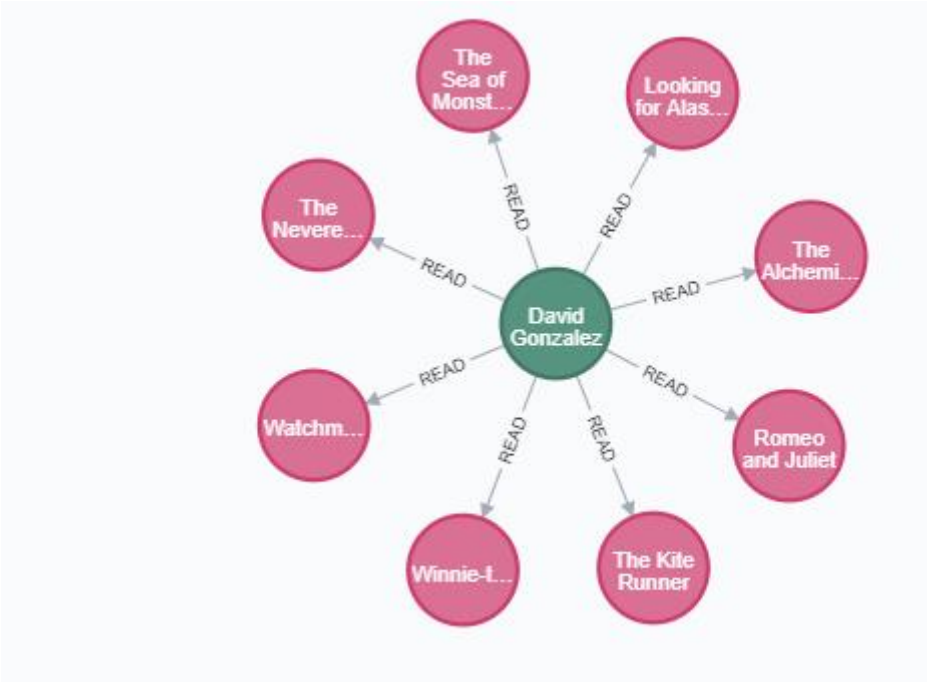


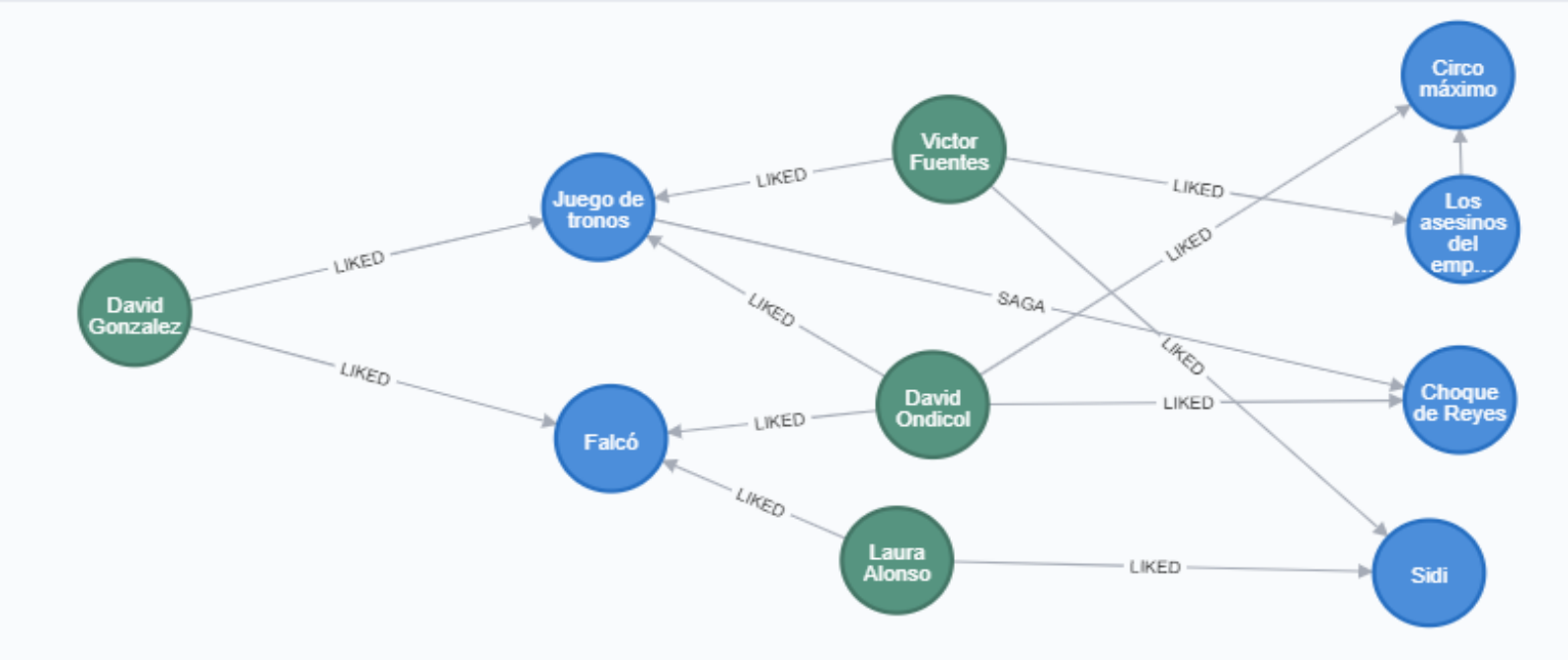
Tabla 2 con el numero de relaciones de cada tipo

Las conclusiones que podemos sacar con estas dos graficas sobre la base de datos es que las principales relaciones por las que podemos navegar en la base de datos van a ser la de READ (conectando Libros y Lectores) y la de IS (conectando Libros y Géneros). En nuestro caso, vamos a utilizar la relación READ para realizar la recomendación. Cabe destacar que, a pesa de ser una base de datos bastante pequeña, cumple bien sus funciones a modo de proyecto de prueba, ya que los algoritmos que se utilizan son muy pesados y su tiempo de ejecución aumenta a gran velocidad cuanto mayor es la base de datos. Con la base de datos actual, y limitando la comparación a 5 usuarios el algoritmo tarda entre 3 y 5s en realizar la tabla de puntuaciones.

En las siguiente imágenes podemos ver ejemplos de como los usuarios se relacionan con los libros e incluso con otros usuarios.



En concreto, en la siguiente imagen podemos ver como un usuario que ha leído determinados libros puede “navegar” la base de datos y llegar a otros libros que podrían ser de su interés.



Sitio web:

Introducción

El sitio web hace las veces de interfaz gráfica, permitiendo la interacción del usuario con la base de datos, así como la ejecución del algoritmo de recomendación. Se decidió utilizar una pagina web como interfaz grafica del proyecto debido a la facilidad de programación de esta respecto a otros medios como puede ser Java o Python. Además, también se ajusta al modelo actual de negocio de este tipo de proyectos: paginas web que ofrecen soporte y proporcionan servicios a una comunidad. De esta manera se intenta simular como debería de trabajar el proyecto si en un futuro se quisiera lanzar al mercado.

El sitio web esta desarrollado desde un punto de vista académico y de facilitar la interacción del usuario directamente con la base de datos y el algoritmo, proporcionándole información que en principio no debería de ser visible de cara al público. De esta manera no tenemos que preocuparnos de pantallas de logueo ni registro, ni de controlar los privilegios del usuario; pasamos directamente a controlar el programa desde un punto de vista del administrador del sitio.

La decisión de diseñar la interfaz de esta manera se debe principal mente ahorro de tiempo invertido en diseño, al no tener que realizar una interfaz de cara al público, la cual tiene un diseño totalmente destino y requiere una gran inversión de tiempo en el diseño de los sitios, mejoras de accesibilidad y desarrollo de métodos y opciones que, si bien son útiles desde el punto de vista del usuario, no suponen ninguna ventaja o beneficio. De esta manera, el tiempo ahorrado en cada uno de los puntos anteriores ha sido reinvertido en mejorar el propio algoritmo de recomendación.

El sitio web está diseñado de tal forma que permite interactuar plenamente con la base de datos. Esto es, permite a la persona que este utilizando la página administrar directamente la base de datos, creando o borrando nodos de cada una de las 4 categorías prediseñadas. Además, permite al usuario establecer relaciones entre los nodos y acceder a la información interna de los mismo. La propia programación del sitio web evita que puedan crearse relaciones o nodos duplicados que puedan afectar al funcionamiento de la base de datos.

Estructura:

El sitio web está dividido en 5 partes:

Home: es la página principal, en ella se explica brevemente el propósito del sitio web y se da una pequeña guía de cómo utilizarlo

Lectores: Esta es la parte principal del sitio web. Está compuesta por 3 páginas:

- La lista de lectores: Muestra en una tabla cada uno de los lectores que existen en la base de datos. Haciendo click en el nombre de cualquier lector, se dirigirá al usuario al perfil de dicho lector. Además, permite tanto añadir nuevos lectores a la base de datos, que aparecerán automáticamente en la tabla, como borrar lectores existentes.
- El perfil del lector: En el perfil del lector se mostrará una tabla con todos los libros que dicho lector ha leído, y la puntuación correspondiente, la cual puede variar entre 1 (la más baja) y 5 (la más alta). Existe la opción de añadir nuevos libros como leídos para dicho lector o de eliminar la relación entre el lector y el libro. A la hora de añadir un libro, si dicho libro ya existiera en la tabla, únicamente se actualizaría la tabla; de esta manera se evitan relaciones duplicadas en la base de datos. Desde la lista de libros leídos se puede acceder al perfil de los libros mediante un click en el nombre del libro.
  Además se de la lista de libros, se incluye una lista con TODOS los lectores que están conectados con el lector que estamos consultando. Esto es, todos aquellos lectores que tengan al menos un libro en común con el lector que consultamos. Estos lectores serán catalogados como lectores vecinos. Algunos de estos lectores serán utilizados por el algoritmo de recomendación en el futuro.
  Botón de volver: devuelve al usuario a la lista de lectores
  Botón de recomendar: dirige al usuario a la página de recomendación del lector que se está consultando.

- Página de recomendación: En esta página es donde se ejecuta el algoritmo de recomendación, el cual se explicará al detalle más adelante. Se compone principalmente de una tabla donde se muestran los usuarios y libros utilizados para realizar la recomendación, así como el resultado final de la misma. Al lado aparecerá una lista con los hasta 5 libros que mejor resultado hayan obtenido tras la aplicación del algoritmo. Estos libros serán los que se recomienden para el usuario.

Libros: Esta sección esta compuesta por la lista de libros, desde la cual se pueden añadir o borrar libros en la base de datos y los perfiles de cada libro, que muestran el título, autor, una tabla con los géneros a los que pertenece y otra con los lectores y las notas aportadas por cada lector.

Desde el "perfil" de cada libro se pueden añadir lectores y géneros al propio libro

Escritores: Similar a la sección de libros. Muestra una lista con todos los escritores desde la cual se puede interactuar con la base de datos, añadiendo escritores o borrando los ya existentes. Al hacer click en cualquiera de los escritores se dirigirá al usuario al perfil del escritor desde el cual podrá borrar al escritor, añadir o borrar libros escritos.

Géneros: Al igual que en las dos secciones anteriores, existe una lista con todos los géneros introducidos en la base de datos. Se permitirá al usuario añadir o eliminar géneros. Al hacer click en cualquiera de los géneros, el usuario será dirigido al perfil de dicho género, donde se mostrará una tabla con todos los libros que pertenecen a ese genero y se dará la posibilidad al usuario tanto de añadir como de eliminar relaciones entre género y libro.

Sistema de recomendación:

Los sistemas de recomendación son sistemas de filtrado de información, los cuales presentan temas, objetos, etc., que pueden ser de interés para el usuario. Los sistemas de recomendación generalmente comparan el perfil del usuario interesado con características de referencia de los que esta buscando para predecir de alguna manera si el usuario puede estar interesado en cualquiera de los ítems que el sistema tiene almacenado.

Una de las técnicas más populares es la de filtrado colaborativo, la cual permite, basándose en experiencias de otros usuarios, predecir la opinión de otro usuario sobre un producto. Al tratarse de un sistema que basa su resultado en experiencias de usuarios anteriores, suelen incluir conjuntos de datos muy grandes. A mayor numero de usuarios, mayor número de interacciones entre los usuarios y los ítems, lo cual implica más información que el sistema puede utilizar a la hora de realizar una recomendación, lo que suele proporcionar una mejor aproximación a la valoración real de un usuario.

El filtrado colaborativo es un método que permite realizar predicciones automáticas de los intereses de un usuario, recopilando sus intereses, gustos, interacciones con otros usuarios o productos etc. Se basa en que, si una persona A tiene la misma opinión que una persona B en un tema, es muy probable que A tenga la misma opinión que B en un tema distinto frente a la opinión de otra persona escogida al azar. El filtrado colaborativo tiene un sinfín de aplicaciones y se ha popularizado con el aumento de trafico de usuarios de Internet. Algunas empresas que usan filtrado colaborativo en sus productos son: Spotify, Twitter y Amazon, entre muchos otros.

Si bien existe infinidad de manera de tratar los datos, los sistemas colaborativos pueden reducirse a dos pasos:

- Búsqueda de usuarios que comparten los mismos patrones con el usuario que se está realizando la predicción
- Utilización de la información de estos usuarios afines para realizar una predicción.

Los sistemas de filtrado colaborativo que siguen el esquema anterior son los denominados sistemas de filtrado basado en usuario.

Una alternativa a los sistemas basados en usuario son los sistemas de filtrado basado en elemento, inventado por Amazon, que cual se basan en que a si unos usuarios A y B compraros los ítems 1 y 2, entonces un usuario C que compre el ítem 1, será recomendado a comprar el ítem 2 también, ya que el algoritmo reconoce que 1 y 2 están relacionados. Si bien el algoritmo por si solo no puede saber que es lo que les relaciona realmente mas que varios usuarios han comprado ambos, este tipo de sistemas suelen apoyarse en el uso de inteligencia artificial (IA) capaz de crear comunidades de productos generales y sus características en común.

Los sistemas de filtrado basados en elementos siguen el siguiente esquema:

- Se construye una matriz de elemento-elemento que determina la relaciones entre pares de elementos (ha comprado x y ha comprado y = 1)
- Usando la matriz y los datos sobre el usuario, se filtran los elementos en común en base a los gustos.

Para este proyecto vamos a implementar un sistema de filtrado colaborativo basado en el usuario, si bien también es viable incluso probable la implementación de un sistema de filtrado basado en elementos (en este caso, libros) en el futuro.

Estos sistemas, a pesar de ser muy útiles y no muy difíciles de implementar, presentan una serie de problemas, numerados a continuación:

- **Escasez de datos:** Al utilizar datos del usuario en cuestión y de otros usuarios, el sistema requiere de una gran base de datos de la cual poder extraer la información. Cuando la base de datos es pequeña, o el usuario es nuevo (no tiene datos suficientes datos del mismo) las predicciones pueden ser muy inexactas o incluso imposibles. Cuanto mayor sea la base de datos, mejor será la predicción.
  En nuestro caso, este es un problema que tenemos que afrontar desde el principio, ya que carecemos de una base de datos que contenga libros, usuarios y su relación. Además, los equipos de los que disponemos para realizar el proyecto no serían lo suficientemente potentes como para soportar una cantidad de datos lo suficientemente grande como para realizar una predicción exacta. Sin embargo, con la cantidad de datos actuales podemos comprobar que el algoritmo funciona perfectamente, aunque afrontamos muchos casos de polarización al no existir compañeros suficientes o libros con los que comparar.

- **Escalabilidad:** Si bien cuando mayor sea la base de datos, mejor es la predicción, una base de datos demasiado grande puede producir serios problemas de escalabilidad. Al tener que comparar el usuario con todos lo usuarios de la base de datos y a su vez con todos los ítems de los usuarios, una gran cantidad de datos puede suponer una grave ralentización del proceso de recomendación. Es importante encontrar un equilibrio entre tamaño y fiabilidad. Generalmente, cuando el conjunto de datos a manejar es demasiado grande, se tiende a descartar aquellos que se sabe que tienen un peso menor y que apenas van a influir en el resultado final. Si bien esto puede proporcionar un resultado más inexacto, ahorra considerablemente el tiempo de ejecución. Por ello es muy importante implementar un buen algoritmo de detección de comunidad que la cantidad de datos a tratar es muy grande.

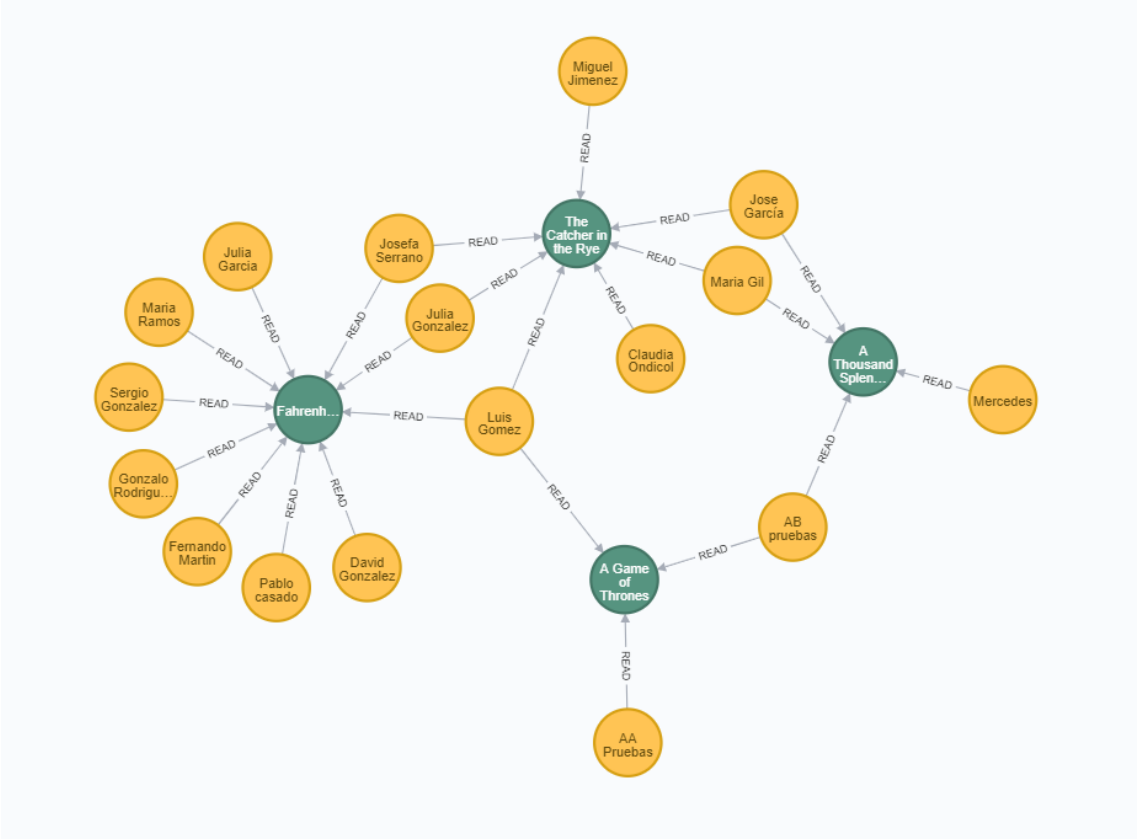
- **Grey Sheep:** Oveja gris, hace referencia a aquellos usuarios que son difícilmente encajables en ninguna comunidad, ya que tienen gustos muy dispares o simplemente acaba de ingresar en la base de datos, por lo que es muy difícil de realizar una recomendación para este tipo de usuarios, o incluso imposible al carecer de relaciones suficientes con otros usuarios. Normalmente los resultados mostrados para este tipo de usuarios se muestran muy polarizados y carecen de mucho sentido cuando se analizan en profundidad.

Si bien existen algunos problemas más para los sistemas de filtrado colaborativo, únicamente los expuestos anteriormente son los que influyen en mayor o menor grado en el desarrollo de este proyecto.

El Algoritmo

En nuestro caso, al tratarse de un sistema de filtrado basado en el usuario, lo primero que tendremos que hacer es encontrar lo que se denomina la “comunidad” del usuario. Para ello vamos a seguir los siguientes pasos:

1.- Búsqueda de vecinos:



Ejemplo de relaciones entre usuarios a través de libros

Lo primero que debemos hacer para poder realizar una recomendación en encontrar todos aquellos lectores que sean vecinos del usuario. Esto es, que tengan al menos un libro en común. Para encontrar este conjunto de lectores, desde la página web se mandará una query (consulta) a la base de datos, con la siguiente estructura

MATCH (n:READER {name: $name}) – [r] – >(b:BOOK)<-[r2]-(m:READER)

WHERE r2.grade= r.grade+1 OR r2.grade=r.grade OR r2.grade= r.grade-1

RETURN DISTINCT m

Lo que estamos pidiendo a la base de datos con esta consulta es que encuentre (MATCH) a todos aquellos usuarios lectores (r:READER) que tengan al menos un libro en común con el usuario al que estamos haciendo la recomendación (n:READER {name:$name}) siendo $name el nombre de dicho usuario, que estamos proporcionando desde la página web.

Además, añadimos como filtro que únicamente queremos aquellos usuarios cuya nota para el libro sea igual o diferente en un punto con la nota proporcionada por el usuario. Si bien para una base de datos del tamaño que estamos utilizando no es necesario este filtrado inicial, si es útil en el futuro, ya que rechaza todas aquellas relaciones cuya puntuación r (Usuario) y r2 (Otro) sea muy diferente, lo que implicaría que dichos usuarios, a pesar de haber leído un mismo libro, no tienen los mismos gustos ya que los han puntuado de manera totalmente distinta.

De esta manera vamos adelantando trabajo futuro, ya que podemos eliminar usuarios que tuvieran gustos distintos de los cuales como mucho podemos obtener recomendaciones contaminadas por las notas de usuarios que no se ajustan al perfil del cual estamos recomendando, por lo que, a menos que exista alguna otra relación que si cumpla las condiciones establecidas, no se procederá a añadir al usuario, lo cual ahorra tiempo de ejecución. Al final de la consulta se pide RETURN DISTINCT m, evitando que, si un usuario ha sido alcanzado mediante 2 caminos distintos, no aparezca duplicado.

Una vez hemos obtenido todos los usuarios potenciales para ser utilizados en la predicción, debemos de realizar un filtrado a mayores. Para realizar este filtrado se decidió asignar a cada lector un coeficiente de relación (CR) con el usuario.

Este coeficiente es calculado en base al numero de libros que tienen en común y la diferencia entre las notas otorgadas. De esta manera, si un usuario B respecto al usuario principal A ha puntuado un libro igual que A, entonces sumará 2 pts. Si B ha puntuado con +1 o -1 respecto a A, sumará únicamente 1 punto. Al final, se dividen los puntos obtenidos entre el numero de libros leídos por A.

De esta manera podemos obtener de manera aproximada que usuarios son los que más se parecen a A, dado que han leído y puntuado de manera similar los mismos libros. Los pesos asignados en base a la relación de la notas (2 si coincide, 1 si la diferencia es de 1, 0 si la diferencia es mayor de 1) podrían ser modificados.

Para ejecutar este ejercicio, se crea una matriz con los libros del usuario a recomendar y todos los libros de los usuarios vecinos que hemos recogido. Posteriormente se comparan las notas de cada usuario con las notas otorgadas por el usuario al recomendar. Al final tenemos un array de nombre de usuarios con sus CR, el cual recorreremos para encontrar los 5 con mejores CR.

Así, al final de este proceso cada usuario tendrá un CR distinto, siendo 2 es máximo (un usuario B que ha leído TODOS los libros que ha leído A, y además los ha puntuado de igual manera). Con este método además nos aseguramos de excluir todos aquellos usuarios que a pesar de haber leído los mismos libros que nuestro usuario a recomendar, los ha puntuado de manera totalmente distinta.

Para este proyecto hemos decidido que únicamente se compare con los 5 usuarios con mayor CR. Este numero está asignado a la variable $MAXUSERS, que limita el tamaño del array de usuarios más parecidos. La razón de hacer esto es que, por cada usuario por el que tiene comparar para hacer una recomendación, el algoritmo tiene que recorrer en cada vez toda la matriz, para comprobar que libros han sido puntuados por cada usuario. Esto resulta en que, incluso con únicamente 5 usuarios, el algoritmo ya tarde entre 3 y 5s en acabar de ejecutarse.

2.- La recomendación

Una vez hemos determinado los lectores más similares podemos pasar a la realización de la recomendación. Para realizar la recomendación final se ha desarrollado un algoritmo inspirado en el algoritmo de Slope One

El algoritmo Slope One usa una forma simple de regresión con un solo parámetro, que es la diferencia del promedio entre la valoración de dos elementos.

	Item 1	Item 2	Item 3
Usuario A	5	3	2
Usuario B	3	4	0
Usuario C	x	2	5

Dada la tabla anterior, si quisiéramos predecir la nota que el usuario C otorgaría al Ítem 1, haríamos lo siguiente:

Calcularíamos la media de las diferencias ente el ítem a recomendar y los otros ítems

MD12= [(1A)-(2A) + (1B)-(2B)] / 2 = (5-3) + (3 – 4) / 2 = 1 / 2 = 0’5

Esto significa que, de media, los usuarios califican el objeto 1 con 0,5pts más que el objeto 2.

Ahora hacemos lo mismo, pero comparando 1 y 3:

MD13= (5-2) / 1 = 3

Como en este caso, el usuario B no ha puntuado 3 (marcándolo como 0) no se añade a la ecuación. Como ultimo paso, predecimos la nota de C1:

[(C2+MD12) * N2 + (C3+MD13) * N3] / (N2+N3)

Siendo Cx la nota otorgada por el usuario C al elemento X, MDXY la media de las diferencias entre 2 elementos X e Y, y Nx el número de usuarios que participaron en la operación de calcular las medias. De esta manera el resultado sería el siguiente:

[(2+0’5)*2 + (5+3)*1] / (2+1) = (5+8) / 3 = 4’33

Por lo que, podemos decir que el usuario C va a otorgar al ítem 1 una nota de 4’33.

Si bien este algoritmo otorga una predicción que simple vista puede parecer lógica, rápidamente cuando ejecutamos las recomendaciones en nuestro programa reparamos en que los resultados no parecen del todo correctos. Aunque el resultado esté bien, las notas otorgadas no parecen lógicas y rápidamente encontramos la razón.

En el caso anterior estamos suponiendo que el usuario C va a puntuar con un 4'33 al objeto 1 dadas las notas que los usuarios A y B han dado a ese mismo objeto. Sin embargo, observamos que las notas dadas por ambos a los objetos 2 y 3 son muy distintas a las otorgadas por el Usuario C. Por ello no podemos fiarnos de que el usuario C vaya a valorar con el mismo criterio el objeto numero 1.

Es por esto que es importante antes de realizar la recomendación mediante este algoritmo, realizar una selección de los usuarios que vamos a utilizar mediante nuestro algoritmo de creación de comunidad. De esta manera, podemos decir al algoritmo que escoja a los X individuos más parecidos al usuario y establecer un mínimo de similaridad (recordemos que 2 es el máximo y 0 el mínimo) . Lamentablemente para este caso de estudio, no podemos establecer dicho límite mínimo, ya que el conjunto de usuarios no es lo suficientemente grande como para conseguir usuarios lo suficientemente parecidos unos con otros. Estaríamos limitando tanto que las recomendaciones se harían únicamente con 1 o 2 usuarios.

Además, podemos integrar el coeficiente de correlación CR obtenido en el algoritmo de creación de comunidad en este mismo algoritmo para así variar los pesos de cada usuario. De esta manera, las notas otorgadas por un usuario con un CR de 2 influirán más en la nota final que aquellos que tengan un CR menor.

Para conseguir esto, tomando el ejemplo anterior, haríamos lo siguiente:

MD12 = { [(1A)-(2A)] * CR_A + [(1B)-(2B)] * CR_B } / 2

Asi pues, supongamos que CR_A = 2 y CR_B = 1. Mientras en el ejemplo anterior obtenemos MD12=0,5 (esto es, el usuario C otorgará 0,5pts más al ítem 1 de lo que haya puntuado al ítem 2), en este caso obtenemos que

MD12= (5-3)*2 + (3-4)*1 / 2 = (4-1)/2 = 1,5

De esta manera la nota otorgada por C se acerca más a la que ha otorgado A, ya que es más parecido.

En el siguiente caso de ejemplo vamos a analizar que libros son los más recomendados para un usuario de pruebas (AA Pruebas).

Una vez iniciado el algoritmo, este creará una tabla para mostrar todas las puntuaciones otorgadas. Los libros leídos por el usuario serán marcados en azul, para que sean fácilmente distinguidos de los libros que estamos recomendando.

Además, cabe destacar que la razón de que existan valores superiores al 5 (nota máxima que un usuario puede otorgar a un libro) es debido que el CR máximo otorgado a los usuarios es de 2.

Una vez el algoritmo ha acabado de trabajar, recorrerá la tabla de resultados recogiendo los 5 libros con mayor puntuación, que será los que devolverá al usuario como posibles libros que más le van a gustar.

```
Los libros mejor puntuados para este usuario son:
Anne of Green Gables
A Thousand Splendid Suns
Jane Eyre
The Call of the Wild
The Handmaid's Tale
```

En la tabla siguiente, que es la correspondiente al output del algoritmo podemos apreciar perfectamente como se realiza la recomendación, y como el CR influye en la misma. Podemos observar que en el caso del libro 4, aunque ha recibido 5 por parte de dos usuarios, la nota final es de 3,5 debido a que estos usuarios tienen poco parecido con el usuario inicial (David Gonzalez tiene un CR de 0,5 y Carmen Vázquez tiene un CR de 0,5). Sin embargo, en el libro 24, obtiene una predicción de 4,5 a pesar de que la nota del otro usuario es únicamente de 4. Esto es debido a que el usuario que esta recomendando tiene un CR de 2, por lo que su nota tiene mucho más peso.

Otro caso que muestra este es el libro 30, que recibe una predicción de 2,85 con notas de 3 y 2, mientras que el libro 44 recibe 2,75 con notas también de 3 y 2, pero de distintos usuarios.

		AA Pruebas	David Gonzalez	AB pruebas	Luis Gomez	Carmen Vazquez	Irene Lopez
0	1984	5	3	5	0	0	0
1	A Prayer for Owen Meany	5	0	5	0	0	0
2	A Game of Thrones	3	0	3	2	3	3
3	A Clockwork Orange	1	1	1	0	0	0
4	Looking for Alaska	3.5	5	0	0	5	0
5	Romeo and Juliet	3	5	0	0	0	0
6	The Alchemist	3	5	0	0	0	0
7	The Kite Runner	3	5	0	0	0	0
8	The Godfather	2.5	4	0	0	0	0
9	The Neverending Story	2.5	4	0	0	0	0
10	The Sea of Monsters	3	4	0	0	0	4
11	Watchmen	2.5	4	0	0	0	0
12	Winnie-the-Pooh	2.5	4	0	0	0	0
13	Divergent	2	3	0	0	0	0
14	Goodnight Moon	2	3	0	0	0	0
15	The Adventures of Tom Sawyer	2.875	3	0	5	0	0
16	Fahrenheit 451	2.625	2	0	5	0	0
17	Lord of the Flies	2.5	2	0	0	4	0
18	Flowers for Algernon	2.25	1	0	0	0	4
19	The Complete Stories and Poems	1	1	0	0	0	0
20	The Maze Runner	1	1	0	0	0	0
21	The Perks of Being a Wallflower	2	1	0	0	0	3
22	The Pillars of the Earth	1	1	0	0	0	0
23	The Road	1	1	0	0	0	0
24	A Thousand Splendid Suns	4.5	0	4	0	0	0
25	The Outsiders	3.75	0	0	5	0	0
26	The Secret Life of Bees	3.75	0	0	5	0	0
27	Thirteen Reasons Why	3.75	0	0	5	0	0
28	Watership Down	3.75	0	0	5	0	0
29	The Lightning Thief	3.5	0	0	4	0	4
30	Anna Karenina	2.875	0	0	3	2	0
31	The Catcher in the Rye	3.25	0	0	3	0	0
32	Ulysses	3.625	0	0	3	5	0
33	Heart of Darkness	3	0	0	2	0	0
34	Tess of the D'Urbervilles	3	0	0	2	0	0
35	The Bell Jar	3	0	0	2	0	0
36	The Girl with the Dragon Tattoo	3	0	0	2	0	0
37	The Last Olympian	3	0	0	2	0	0
38	Where the Red Fern Grows	3	0	0	2	0	0
39	Jonathan Livingston Seagull	2.75	0	0	1	0	0
40	Middlesex	2.75	0	0	1	0	0
41	The Secret Garden	2.75	0	0	1	0	0
42	Anne of Green Gables	4	0	0	0	5	0
43	Their Eyes Were Watching God	3.5	0	0	0	4	0
44	Harry Potter and the Chamber of Secrets	2.75	0	0	0	3	2
45	The Other Boleyn Girl	3	0	0	0	3	0
46	A Tale of Two Cities	2.5	0	0	0	2	0
47	Charlotte's Web	2.5	0	0	0	2	0
48	Dracula	2.5	0	0	0	2	0
49	How the Grinch Stole Christmas!	2.5	0	0	0	2	0
50	The Count of Monte Cristo	2.5	0	0	0	2	0
51	Treasure Island	2.5	0	0	0	2	0
52	Where the Wild Things Are	3	0	0	0	2	4
53	Oliver Twist	2	0	0	0	1	0
54	The Battle of the Labyrinth	2	0	0	0	1	0
55	The Lovely Bones	2	0	0	0	1	0
56	Jane Eyre	4	0	0	0	0	5
57	The Call of the Wild	4	0	0	0	0	5
58	The Handmaid's Tale	4	0	0	0	0	5
59	The Metamorphosis	4	0	0	0	0	5
60	The Scarlet Letter	4	0	0	0	0	5
61	The Phantom Tollbooth	3.5	0	0	0	0	4
62	The Stand	3.5	0	0	0	0	4
63	Brave New World	3	0	0	0	0	3
64	Interview with the Vampire	3	0	0	0	0	3
65	The Princess Bride	3	0	0	0	0	3
66	Atonement	2.5	0	0	0	0	2
67	The Joy Luck Club	2.5	0	0	0	0	2
68	The Very Hungry Caterpillar	2.5	0	0	0	0	2

3.- Conclusión:

Según lo observado en los casos hechos, ha quedado demostrado que el uso de CR a la hora de calcular la recomendación tiene un impacto positivo sobre la misma, dando lo que a simple vista parecen resultados más lógicos.

Sobre esta parte del algoritmo hay que destacar que es la parte más complicada de todo el trabajo y que ha llevado aproximadamente entre el 50% y el 60% de toda la carga de trabajo total, debido a la necesidad de buscar, informarse, y estudiar los distintos algoritmos que podían ser utilizado. Además, gran parte del tiempo fue invertido en la programación del mismo algoritmo, ya que aparentemente sencillo en el papel, es increíblemente complicado de programar, al estar accediendo continuamente a los datos de una matriz y de necesitar mantener información guardada de cada usuario (como su CR) disponible en todo momento para acceder a ella.

La complejidad de implementación del mismo se puede apreciar perfectamente en el número de líneas de código empleadas. Mientras que de media cada pagina del sitio web lleva unas 100 líneas de código, la parte del algoritmo por si sola ocupa aproximadamente 300 líneas. Además, a esto hay que añadir que la complejidad de cómputo del algoritmo es bastante grande, lo que produce que el tiempo de ejecución aumente considerablemente cuanto mayor sea el numero de datos que debamos de analizar. Es por eso que se decide no hacer crecer la base de datos ni utilizar más de 5 usuarios en la comparación. Ente problema es solucionado en las grandes empresas que manejan cantidades gigantes de datos mediante algoritmos muy buenos de detección de comunidad que descartan de ante mano aquellos usuarios que saben que no deben ser comparados con el usuario a recomendar, haciendo que el numero de comparaciones se reduzca de millones a miles. Aun así, utilizan equipos con gran capacidad de cómputo y normalmente combinan varios algoritmos de recomendación.

Análisis

Como en todo proyecto, es recomendable realizar un análisis SWOT del mismo, con el fin de identificar las posibles mejoras que se puedan implementar y las amenazas que el proyecto presenta, de manera que puedan ser afrontadas correctamente.

Fortalezas:

El usuario puede interactuar fácilmente con la base de datos y modificarla a su gusto, pudiendo crear situaciones concretas.

Gracias a la utilización del coeficiente de relación a la hora de efectuar la predicción, devuelve resultados mucho más lógicos que los dados por el algoritmo Slope One.

Los resultados obtenidos son visibles en todo momento gracias a la tabla de output, por lo que se puede interpretar los datos y además detectar posibles fallos en el algoritmo.

Debilidades:

La base de datos que se utiliza no es lo suficientemente grande para realizar una predicción realmente precisa; sin embargo, es adecuada como primera toma de contacto con el mundo de la ciencia de datos.

Debido al tamaño de la base de datos, en algunos casos las predicciones están muy polarizadas, al no tener usuarios suficientes con los que comparar.

La base de datos carece de opiniones de usuarios reales.

Debido a esto último, no se ha procedido a utilizar otros datos como, géneros más leídos por usuario o escritores mas populares, ya que al haber realizado estas conexiones usuario-libro al azar, aparecían resultados inconcluyentes que contaminaban la recomendación.

Por el mismo motivo, existe una gran cantidad de usuarios “grey sheep”.

Los algoritmos de filtrado colaborativo tienen una complejidad muy alta debido a la necesidad de iteración, por lo que se vuelven lentos y pesados cuando deben procesar una gran cantidad de datos. Esto puede solucionarse con equipos mas potentes o con la combinación con otros algoritmos.

Diseño del sitio web: el sitio web esta diseñado de una manera muy simple, pensado únicamente como un entorno de pruebas y no para ser lanzado al publico general.

Oportunidades:

Debido a la estructura de la base de datos es muy fácil hacerla crecer y por lo tanto conseguir recomendaciones más precisas.

Una vez poblada la base de datos con usuarios reales, las recomendaciones pueden afinarse teniendo en cuenta otros datos que actualmente están dejados al azar.

Puede combinarse perfectamente con métodos en IA que mejoren los pesos utilizados a la hora de realizar recomendaciones o encontrar la comunidad del usuario.

No existe nada parecido en el sector de los libros. Otras páginas que tienen un objetivo parecido tienen estructura de red social, donde los usuarios comentan y puntúan libros, pero la página no ofrece ninguna recomendación basada en el usuario, sino en los libros mejor valorados o por las personas a las que sigue.

Existen muchas tecnologías y algoritmos que pueden utilizarse para mejorar la recomendación.

Hay una gran cantidad de ideas que no fueron implementadas por falta de tiempo:

- Recomendación basada en los libros
- Recomendaciones por regiones mediante el nodo de País
- Recomendaciones por géneros
- Filtros de búsqueda

Mejorar la interfaz grafica para proporcionar un diseño mas enfocado en el usuario final que en su uso para fines académicos.

Futuro:

De cara al futuro, me gustaría seguir trabajando en este proyecto. Lo primero que me gustaría hacer es dedicarle más tiempo a pulir el algoritmo de recomendación para dar unos resultados más precisos.

El algoritmo de recomendación ha sido lo que más tiempo me ha llevado y mas quebraderos de cabeza me ha producido. Sin embargo, me ha servido para aprender mucho sobre el mundo de la ciencia de datos y de las predicciones / recomendaciones. Es por ello por lo que a medida que he ido avanzando con el proyecto, durante el proceso de búsqueda de información he dado con numerosas tecnologías que me gustaría poder implementar en este proyecto. Me ha servido para aprender las distintas técnicas que existen para realizar recomendaciones y hay muchas que me han parecido muy interesantes y que me gustaría poder llegar a aplicar en el trabajo.

Una de esas tecnologías es el de desarrollo de una inteligencia artificial capaz de obtener las características de los usuarios para poder así crear una comunidad de usuarios con perfiles muchos más parecidos. Mientras que ahora mismo dicha comunidad es generada en función únicamente de las notas otorgadas a los libros que tienen en común varios usuarios, mediante el uso de IA, podemos añadir otros factores como, por ejemplo, que géneros son los que mas han leído, que autores son los mejores puntuados, o el tamaño medio de los libros, y otorgarles un peso a cada uno de estos factores que influyan en mayor o menor medida en el CR final del usuario.

Además, me gustaría poder llegar a implementar ideas que si bien estaban en la propuesta inicial, no pudieron ser implementadas por falta de tiempo, como por ejemplo la implementación de un algoritmo de filtrado basado en libros. Esto es, seleccionando un libro, recomendar otros que tengan unas características similares. Otra idea que me gustaría poder implementar es las recomendaciones por regiones mediante los nodos correspondientes a países.

Conclusión:

Este proyecto me ha abierto la puerta a un mundo que desconocía completamente, que es el de la ciencia de datos. La verdad es que me parece un mundo apasionante y con mucho futuro, y que las grandes compañías del momento tienen muy en cuenta de cara a como afrontar el futuro. Me parece una temática muy interesante y con la cual se puede hacer un sinfín de cosas muy útiles.

También me ha servido para aprender como funciona el mundo de las bases de datos de grafos, muy relacionadas con el mundo de la ciencia de datos mencionado anteriormente. Considero que este tipo de bases de datos tienen un futuro muy brillante y son perfectas para los casos donde lo importante es navegar por las relaciones entre las distintas entidades y no simplemente almacenar la información. Ofrecen una visualización maravillosa de lo expuesto anteriormente que te permite a simple vista ver como pueden estar relacionadas dos personas y lo lejos o cerca que están unas de otras, así como detectar comunidades.

Me ha ayudado a interesarme por temas desconocidos completamente antes de comenzar este trabajo, y a descubrir nuevas tecnologías como Python o cypher, las cuales, si bien había escuchado hablar sobre ellas, nunca había tenido interés en aprender como funcionan. Aunque no he tenido la posibilidad de utilizar Python para el desarrollo de este proyecto si que he realizado un pequeñ0 trabajo de investigación de como funciona y como poder llegar a implementarlo en un futuro, sobre todo orientada a la parte de utilización de Inteligencia Artificial para mejorar el resultado de las recomendaciones.

Este proyecto a exigido mucho de mi parte ya que casi todo lo que hacia lo tenia que aprender de cero. Para empezar, nunca antes había utilizado una base de datos de grafos. Es más, desconocía de su existencia por lo que las primeras semanas las tuve que dedicar a aprender su funcionamiento e interacción a la vez que iba pensando en el proyecto que quería realizar y como implementarlo.

Si bien estoy satisfecho con el trabajo realizado, me quedo con muchas ganas de continuar con el proyecto y sobre todo poder aplicar de alguna manera muchas de las tecnologías sobre las que he leído en la fase de investigación pero que no he tenido tiempo de aplicar debido a la falta de tiempo.

Bibliografía:

Neo4J:

https://neo4j.com/developer/cypher-basics-i/

https://neo4j.com/graphacademy/

https://neo4j.com/developer/guide-import-csv/

https://neo4j.com/developer/php/

Graphaware:

https://github.com/graphaware/neo4j-php-client

XAPMM:

https://www.apachefriends.org/es/index.html

Composer:

https://getcomposer.org/

Brackets:

http://brackets.io/

HTML/CSS:

https://www.w3schools.com/

Grafos de conocimiento:

https://thenextweb.com/podium/2019/06/11/what-is-a-knowledge-graph-and-how-does-one-work/

https://searchengineland.com/can-retailers-benefit-from-the-knowledge-graph-schema-you-bet-122676

Inteligencia Artificial:

https://www.gnoss.com/inteligencia-artificial

https://cleverdata.io/sistemas-recomendacion-machine-learning/

Sistemas de recomendación:

https://promocionmusical.es/como-funcionan-algoritmos-recomendacion-spotify

https://www.analiticaweb.es/primeros-pasos-sistemas-recomendacion/

https://www.genbeta.com/web/asi-funcionan-las-recomendaciones-de-amazon

https://www.antevenio.com/blog/2018/12/algoritmo-seo-de-amazon/

https://www.netflixprize.com/assets/GrandPrize2009_BPC_BigChaos.pdf

Algoritmos:

https://es.wikipedia.org/wiki/Coeficiente_de_correlaci%C3%B3n_de_Pearson

https://es.wikipedia.org/wiki/Sistema_de_recomendaci%C3%B3n

https://eprints.ucm.es/43975/1/Sistemas%20de%20Recomendaci%C3%B3n%20basados%20en%20t%C3%A9cnicas%20de%20predicci%C3%B3n%20de%20enlaces%20para%20jueces%20en%20l%C3%ADnea%20-%20Marta%20Caro%20Mart%C3%ADnez.pdf

https://medium.com/@eng.saavedra/sistemas-de-recomendaci%C3%B3n-parte-2-b8a5dc9dc730

https://es.wikipedia.org/wiki/Filtrado_colaborativo

Otros

https://arxiv.org/pdf/1905.02840.pdf

GitHub

https://github.com/DavidGG98/Book_Recomendation-NEO4J-