****

**Escuela de Ingenierías**

**Industrial, Informática y Aeroespacial**

Sistema híbrido de recomendación de películas empleando grafos de conocimiento

Sistemas de Información de Gestión y Business Intelligence

**GRADO EN INGENIERÍA INFORMÁTICA**

(Diciembre, 2020)

Autor: D. Álvaro González Jiménez

Tutor: D. Enrique López González

**ÍNDICE**

[1. Descripción del problema 2](#_Toc57496313)

[2. Herramientas utilizadas 5](#_Toc57496314)

[1. Base de datos: 5](#_Toc57496315)

[2. Interfaz de usuario (Frontend): 6](#_Toc57496316)

[3. Servidor (Backend): 6](#_Toc57496317)

[3. Base de datos utilizada 7](#_Toc57496318)

[4. Descripción de la aplicación 8](#_Toc57496319)

[5. Explicación del algoritmo 9](#_Toc57496320)

[6. Análisis de resultados 10](#_Toc57496321)

[7. Dafo 11](#_Toc57496322)

[8. Líneas de futuro 12](#_Toc57496323)

[9. Lecciones aprendidas 13](#_Toc57496324)

[10. Bibliografía 14](#_Toc57496325)

1. Descripción del problema

Tal vez si pensamos en un sistema de recomendación, el primero que se nos viene a la mente es un sistema de recomendación de películas. Todas y cada una de las plataformas de cine en streaming emplean un complejo sistema de recomendación encargado de analizar gigantescos volúmenes de datos y realizar sugerencias a sus usuarios en tiempo record. Aunque para mucha gente lo desconozca, dichos sistemas no sólo tienen en cuenta los gustos del usuario para efectuar una recomendación, sino que analizan una gran cantidad de variables tales como el género, la edad, la raza, el dispositivo en el que sueles ver contenido, las horas del día en las que acostumbras a visualizar contenido o incluso el lugar de residencia con el fin de obtener recomendaciones lo más certeras posibles.

Estos sistemas, que se encuentran en constante desarrollo y evolución, son mucho más que simples recomendadores de películas. Si bien es posible que aunque utilices dichas plataformas con asiduidad no hayas reparado en ello, aunque estos sistemas recomienden la misma película o serie a dos personas diferentes, la portada mostrada no tiene por qué coincidir. Esto tiene una explicación muy sencilla, esos dos usuarios no son iguales, o al menos no lo son a ojos del sistema. Según la asunción que el sistema tenga del usuario, la portada que se le muestra difiere. Aunque desde luego este tipo de comportamientos no se pueden generalizar para toda la población, en la práctica ocurre que es infinitamente más probable que acabemos visualizando un contenido si la portada (que no deja de ser nuestra primera impresión sobre el contenido) es de nuestro agrado. Debido a esto, estas plataformas intentan que dicha primera impresión sea lo más fructífera posible, mostrando al usuario una portada que se asemeje al estilo de contenido que al usuario le guste visualizar, así como mostrando personajes que concuerden en raza y género con la asunción que el sistema tenga de esas dos características del usuario.

Tras esta breve descripción del estado del arte, supongo que lo más lógico es pensar ¿por qué iba a querer alguien desarrollar un sistema de recomendación de películas dado el extremo desarrollo y complejidad que caracteriza a los sistemas existentes en el mercado? Pues, aunque parezca increíble, tiene su respuesta. Vayamos por partes:

1. **Extensión del catálogo:** Como es evidente, los sistemas de recomendación de las grandes plataformas de streaming única y exclusivamente trabajan sobre el conjunto de películas que, en ese momento, están disponibles en la plataforma. Esto supone una gran diferencia con mi aplicación, que consta con una base de datos con información y valoraciones de algo más de 9.000 películas, mientras que por ejemplo Netflix apenas cuenta con unas 2.500, Prime Video con unas 4.400 y HBO con unas 900.
2. **Antigüedad de las películas:** En las plataformas de streaming, las películas que más abundan son aquellas más modernas, que por lo general tienden a gustarle mucho más al gran grueso de la población. Sin embargo, precisamente por la necesidad de tener en el catálogo películas que agraden a la mayoría, quedan en el olvido un sinfín de clásicos como *Con la muerte en los talones* de Alfred Hitchcock, auténticos clásicos del séptimo arte discriminados únicamente por su antigüedad. En mi sistema, esta discriminación no existe. Pueden encontrarse películas desde el año 1903 hasta la actualidad, recomendándose todas ellas por igual, todo en base a ciertos criterios y reglas matemáticas que serán perfectamente descritas en capítulos posteriores de este mismo documento.
3. **Ausencia de intereses externos:** Con el paso de los años, las plataformas de streaming han dejado de ser simples intermediarias entre productoras cinematográficas y usuarios, pasando a convertirse cada vez con más frecuencia en productoras del propio contenido que ofrecen. Aquí se plantea un dilema, ¿acaso alguien considera que los sistemas de recomendación no van a valorar especialmente bien las películas que sus propias compañías han producido? Esas películas no se pueden ver en una sala de cine, tampoco se pueden adquirir en un centro comercial. Que tengan éxito o no depende solamente de que una gran cantidad de usuarios las visualicen, y ahí, los sistemas de recomendación tienen la misma importancia que una buena campaña de marketing. Mi sistema, por el contrario, no se ve influenciado por ningún tipo de vicio ni interés externo. Es, si se quiere ver así, totalmente puro e imparcial.

Habiendo ya explicado en qué se diferencia de los sistemas de recomendación convencionales, queda por explicar exactamente cómo funciona el sistema, qué problemas resuelve y, en definitiva, cuál es su razón de ser.

En su función más elemental, es un sistema de recomendación híbrido capaz de recomendar películas en base a dos aspectos totalmente diferentes que son aunados finalmente para efectuar una recomendación final. Estos dos aspectos son los siguientes:

* **Algoritmo de recomendación basado en contenido:** En base a las características que definen a las películas valoradas por el usuario, se efectúan recomendaciones de películas similares a las valoradas.
* **Algoritmo de recomendación de filtrado colaborativo:** En base a las valoraciones asignadas a las películas, se recomiendan al usuario las películas mejor valoradas por otros usuarios, en este caso aquellos que más se parecen a él.

Además de como sistema de recomendación, la aplicación también puede funcionar como una agenda en la que apuntar aquellas películas que hemos visto y la puntuación que les hemos asignado, aunque claramente se trata de una función secundaria.

1. Herramientas utilizadas

En lo relativo a las tecnologías utilizadas, se pueden dividir en 3 grupos perfectamente diferenciados: base de datos, interfaz de usuario y servidor.

1. Base de datos: Entre las diversas tecnologías de bases de datos que pueden encontrarse en el mercado, hay dos que se llevan toda la atención: las bases de datos relacionales (MySQL, Oracle, MSSQL) y las bases de datos orientadas a documentos (MongoDB, CouchDB). Aunque sus usos son extremadamente variados (no en vano copan casi todo el mercado) para el problema que deseaba resolver no eran, ni en el mejor de los casos, lo más mínimamente viables.

El motivo es sencillo, no son capaces de computar cálculos sobre el más de millón de relaciones de las que dispone la base de datos del sistema en un periodo de tiempo aceptable. Por ello, me decidí a utilizar una base de datos orientada a grafos de conocimiento, Neo4j para ser más exactos. Su estructura y funcionamiento, en el que las relaciones son el aspecto más importante, las convierten en la base de datos más apropiada para resolver este tipo de problemas, así como otros problemas relacionados con detección de fraudes o incluso investigaciones genéticas.

Habiendo decidido el tipo de base de datos a utilizar, tan sólo quedaba una cosa, aprender a interactuar con ella. Neo4j implementa un lenguaje propio de consultas basado en SQL denominado Cypher, lo que hace que, si bien no sea necesario aprender la lógica que opera detrás de las consultas, sí que resulte extremadamente necesario aprender la sintaxis que emplea Cypher. Para ello, decidí completar el curso **Introduction to Neo4j 4.0** [1]**,** un curso de unas 16 horas de duración en el que pude aprender todas las funcionalidades de Cypher que posteriormente tuve que utilizar en el desarrollo del sistema, es decir, funcionamiento de una base de datos orientada a grafos de conocimiento, creación de nodos y relaciones, queries, indexación, importación de datos…

Sobre la base de datos hay muchos más aspectos que comentar, de modo que he decidido crear un capítulo que hable en exclusiva del origen del base de datos, procesamiento, importación, estructura, etc.

1. Interfaz de usuario (Frontend): A la hora de diseñar la interfaz de usuario, que generalmente se denomina frontend, decidí utilizar las tecnologías que ya conocía previamente. Así, al no tener que aprender desde 0 ninguna tecnología, puede realizar el desarrollo de una forma mucho más eficiente, obteniendo además un mejor resultado. Las tecnologías en cuestión son las siguientes:
2. **Javascript** [2]**:** Lenguaje de programación que aporta interactividad dinámica a la página web.
3. **Vue** [3]**:** Framework de Javascript que permite la construcción de interfaces de usuario y aplicaciones de una única página.
4. **Vuetify** [4]: Framework de Vue que permite la creación de interfaces de usuario basándose en componentes prediseñados.
5. **Vue router** [5]: Librería encargada de gestionar los movimientos entre vistas en una aplicación creada con Vue.
6. **Vuex** [6]: Librería encargada de gestionar los estados de una aplicación creada con Vue. Además, sirve como repositorio central de datos de la aplicación.
7. **Vuex persistedstate** [7]: Librería encargada de persistir la información de Vuex al refrescar una página.
8. **Axios** [8]: Cliente HTTP basado en promesas que permite efectuar peticiones API a un servidor.
9. Servidor (Backend): A la hora de diseñar el servidor de la aplicación, generalmente denominado Backend, se me presentaron dos alternativas entre las que tuve que decidir. Por un lado, tenía la opción de desarrollarlo en nodejs [9], un entorno en tiempo de ejecución basado en Javascript [2] que permite el desarrollo de servidores. Por otro lado, dado que es en el servidor donde se ejecutan los algoritmos de recomendación, estaba la opción de desarrollarlo en Flask [10], un framework de Python [11] que permite el desarrollo de servidores de forma sencilla.

Tras valorar pros y contras de ambas implementaciones me decidí por la segunda opción, especialmente porque Python iba a ser muchísimo más eficiente que Javascript a la hora de realizar operaciones de análisis y procesamiento de datos.

Como no había desarrollado jamás una aplicación con Flask, me vi en la necesidad de aprender desde 0 su funcionamiento, el cual aprendí mediante dos vías diferentes:

1. **Visualización de un vídeo en Youtube** [12]: En este vídeo se explica a la perfección el funcionamiento del framework. Peticiones, respuestas, gestión de rutas, argumentos, validación de entradas… Tan solo visualizando los 50 primeros minutos de este vídeo ya pude ponerme sin ningún problema a desarrollar el backend del proyecto.
2. **Observación del siguiente proyecto en Github** [13]: En este repositorio, dentro de la carpeta flask-api puede encontrarse un ejemplo de desarrollo de una aplicación Flask con acceso a base de datos neo4j que me sirvió de ejemplo para observar cómo se hacen las consultas a la base de datos desde Python.

Tras aprender el funcionamiento del framework, proceso que me llevó aproximadamente un día, pude empezar a desarrollar el backend del proyecto sin ningún tipo de problema. Las tecnologías que empleé, además de por supuesto Python y Flask, fueron las siguientes:

1. **Pandas** [14]**:** Módulo de Python que sirve para la manipulación y análisis de datos.
2. **Numpy** [15]**:** Módulo de Python que da soporte a la creación de vectores y matrices, así como a la realización de una gran cantidad de operaciones matemáticas complejas.
3. **Neo4j** [16]**:** Módulo de Python que permite la comunicación con bases de datos Neo4j.
4. **Sklearn** [17]**:** Módulo de Python que aporta una gran cantidad de funcionalidades relacionadas con Aprendizaje automático.
5. **Scipy** [18]**:** Módulo de Python que aporta una gran cantidad de herramientas y algoritmos matemáticos.

1. Base de datos utilizada
2. Descripción de la aplicación
3. Explicación del algoritmo
4. Análisis de resultados
5. Dafo
6. Líneas de futuro
7. Lecciones aprendidas
8. Bibliografía

|  |  |
| --- | --- |
| [1] | «Neo4j GraphAcademy,» [En línea]. Available: https://neo4j.com/graphacademy/online-training/v4/00-intro-neo4j-about/. |
| [2] | «API Javascript,» [En línea]. Available: https://developer.mozilla.org/es/docs/Web/JavaScript. |
| [3] | «API Vue,» [En línea]. Available: https://vuejs.org/. |
| [4] | «API Vuetify,» [En línea]. Available: https://vuetifyjs.com/en/. |
| [5] | «API Vue-router,» [En línea]. Available: https://router.vuejs.org/. |
| [6] | «API Vuex,» [En línea]. Available: https://vuex.vuejs.org/. |
| [7] | «API Vuex-persistedstate,» [En línea]. Available: https://www.npmjs.com/package/vuex-persistedstate. |
| [8] | «API axios,» [En línea]. Available: https://www.npmjs.com/package/axios. |
| [9] | «API nodejs,» [En línea]. Available: https://nodejs.org/es/. |
| [10] | «API Flask,» [En línea]. Available: https://palletsprojects.com/p/flask/. |
| [11] | «API Python,» [En línea]. Available: https://es.python.org/. |
| [12] | «Vídeo Flask,» [En línea]. Available: https://www.youtube.com/watch?v=GMppyAPbLYk. |
| [13] | «Código Flask,» [En línea]. Available: https://github.com/neo4j-examples/neo4j-movies-template/tree/master/flask-api. |
| [14] | «API Pandas,» [En línea]. Available: https://pandas.pydata.org/. |
| [15] | «API Numpy,» [En línea]. Available: https://numpy.org/. |
| [16] | «API Neo4j Python,» [En línea]. Available: https://neo4j.com/developer/python/. |
| [17] | «API Sklearn,» [En línea]. Available: https://scikit-learn.org/stable/. |
| [18] | «API Scipy,» [En línea]. Available: https://www.scipy.org/. |