Trabajo de Fin de Grado

Sistemas recomendadores en sistemas sociales

Recommender systems for social systems

Tinguaro Cubas Saiz

La Laguna, 7 de septiembre de 2015

D. **Carina Soledad González González**, con N.I.F. 54.064.251-Z profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

D. **Yeray del Cristo Barrios Fleitas**, con N.I.F. 54.106.627-R personal investigador contratado en el Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como cotutor

C E R T I F I C A (N)

Que la presente memoria titulada:

"Sistemas recomendadores en sistemas sociales"

Ha sido realizada bajo su dirección por D. **Tinguaro Cubas Saiz**, con N.I.F. 54.049.703-W.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 7 de septiembre de 2015.

Agradecimientos

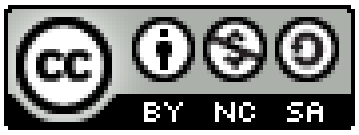
Primero dar gracias a todos los profesores e integrantes de la ULL, sobretodo a Carina Soledad y a Yeray del Cristo, por la educación ofrecida. He aprendido mucho de cada uno de ustedes y espero seguir aprendiendo.

Dar gracias a mis compañeros de la universidad por tantas horas que hemos pasado juntos. Tanto sufrimiento, estrés por entregar prácticas o por exámenes. Espero acabar con ustedes trabajando codo con codo para conseguir nuestro sueño.

Después tengo que agradecerles a mis amigos por estar siempre apoyándome tanto en ámbitos educacionales como personales. Gracias por aguantar mis conversaciones informáticas sin tener ni idea de lo que hablaba y gracias por pensar siempre que podía acabar mis estudios. Ya son muchos años que llevamos juntos y espero que sean muchos más.

Por último y no por eso menos importante agradecer a mi familia todo lo que han hecho por mí sin pedir nada a cambio. Gracias a ellos estoy donde estoy y siempre recordaré este momento de mi vida. Gracias por estar siempre pendiente de mí, por preguntar todos los días y por decirme siempre lo mucho que valgo. Sois los primeros en confiar en mí y espero no fallarles nunca. También darles las gracias por no obligarme hacer nada que no quería y gracias por apoyarme en todas las decisiones que he tomado. Siempre han estado ahí para todo lo que he necesitado y espero estar con ustedes toda mi vida. Por todo esto y mucho más, muchas gracias.

Licencia



Resumen

Las opiniones y experiencias compartidas en el uso de un producto forman una poderosa fuente de información sobre las preferencias de los usuarios. Estos datos son usados en distintos sistemas sociales con el objetivo de ofrecer una recomendación, mejorando así la experiencia de la persona. Este trabajo está destinado al proyecto EDHospi, proyecto colaborativo entre las agrupaciones escolares para mejorar el aprendizaje del alumnado hospitalizado. Para incrementar la eficacia didáctica se podría utilizar un sistema recomendador que indicase a los profesionales los mejores objetos de aprendizaje para el alumnado. En este proyecto se ha investigado, valorado e implementado un sistema recomendador utilizando para ello tecnologías webs con la intención de una posterior integración en el sistema EDHospi.

Palabras clave: Sistemas sociales, EDHospi, Sistema recomendor, objetos de aprendizaje.

Abstract

Opinions and shared experiences in the use of a product provide a powerful source of information regarding consumer preferences. The data are used in different social systems aiming to offer a recommendation, to better people's experience. This work is focused on the EDHospi project; a collaborative project between schools groups to improve hospitalized student's learning process. In order to increase teaching effectiveness, a recommender system could be used to guide professionals the best learning objects for students. In this project a recommender system has been researched, assessed and implement by using web technology with the intention of a subsequent integration into the EDHospi system.

Keywords: Social systems, EDHospi, recommender systems, learning objects.

Índice General

Índices de figuras

Capítulo 1. Introducción

Regularmente estamos expuestos a una cantidad de información que se extiende de tal forma que no somos capaces ni de comprenderla ni de asimilarla. Esta situación empeora cada día por el rápido crecimiento de los datos que se almacenan en internet. A raíz de este problema surgen los sistemas recomendadores para facilitar al usuario la comprensión en temas/dominios en los que la información es numerosa y variada. Un sistema recomendador ayuda a un usuario a elegir sin tener suficiente experiencia con las opciones a seleccionar, impulsándolo a la adquisición de la mejor opción.

La tarea principal de un sistema recomendador es proporcionar una lista de los posibles ítems que puedan interesarle al usuario, ahorrando al mismo un largo tiempo de búsqueda por la red. En este documente se puede encontrar toda la información relativa al diseño y la implementación del presente trabajo de final de grado.

1.1 Antecedentes

Se ha realizado un estudio de distintos proyectos sobre sistemas recomendadores con la intención de recopilar toda la información posible sobre la materia. Este estudio abarca desde los diferentes algoritmos y filtros de recomendación, hasta las estructuradas de datos y tecnologías con las que se han conseguido una implementación. Un filtro es la técnica o la forma con la que se consigue la lista de los mejores ítems para el usuario. Se suelen dividir en dos categorías las cuales se explican a continuación.

Los filtros colaborativos o basados en usuarios se rigen por la idea de que un ítem que ha sido bien valorado por una persona que comparte características con el usuario, es un buen ítem para recomendar. Normalmente se usa el algoritmo del vecino más cercano para encontrar relaciones entre los usuarios. Como ventaja se tiene que

permite recomendar contenido difícil de analizar y como desventaja se tiene el problema del usuario nuevo o la oveja gris. Ambos problemas se dan cuando no se consigue asignar al usuario en un grupo.

El segundo filtro es basado en contenido. Al contrario del anterior, se calculará la similitud entre los ítems del sistema y no entre usuarios. Como mejor ventaja tienen que no existe la dispersión ya que el modelado de la información está presente en el documento y no hace falta que la provea otro usuario. Como desventaja está el problema del usuario nuevo. Cuando un usuario es nuevo el sistema no tiene constancia de lo que le gusta ya que no ha valorado nada aún.

Actualmente es común mezclar los filtros, es decir filtros híbridos. Se intenta combinar ambas ideas usando lo mejor de cada para evitar los errores que individualmente se pueden tener. Con esto se consigue reducir el tiempo que se tarda en crear la recomendación y mejorar la calidad de la misma. Por ejemplo si tenemos el problema de la oveja gris y no se encuentra un grupo que asignarle al usuario se puede utilizar un algoritmo basado en el contenido. Si el usuario es nuevo seguramente no ha valorado ningún objeto pero se podría tener un perfil de usuario e intentar encontrarle un grupo.

Sobre los algoritmos de recomendación lo más común es usar el algoritmo del vecino más cercano tanto para asignarle al usuario en un grupo o para encontrar los objetos que más se parecen a los objetos que le gusta al usuario. Otro aspecto de suma importancia es el cálculo de la similitud y la elección del tipo de correlación donde tiene un papel importante el coeficiente de correlación de Pearson y el coeficiente de Jaccard.

Por último ciertos autores se han decantado por la programación lógica. Usando una base de conocimiento con restricciones y preferencias del usuario se consigue, tras un proceso de inferencia, la lista de recomendación.

En una primera instancia se diseñó un sistema recomendador utilizando una estructura de sistema multi-agente. Este sistema buscaría los objetos en un repositorio DSpace por ser uno de los repositorios de objetos más utilizados hoy en día.

Cuando se decidió integrar este trabajo en el proyecto EDHospi para aumentar su eficacia didáctica, se modificó el diseño del sistema para que dicha integración se realizase con los menores errores posibles. Utilizando la full-stack MEAN dejamos a un lado la estructura de un sistema multi-agente para diseñar un sistema utilizando el patrón de diseño modelo-vista-controlador. También descartamos DSpace como repositorio de objetos ya que EDHospi almacenará la información en una base de datos MongoDB.

1.2 Estado del arte

Para el desarrollo de un sistema recomendador se suelen usar estructuras de datos como árboles o algunas más complejas como las redes bayesianas o las redes de aprendizaje. Lo más común es estructurar todo el sistema recomendador como un sistema multi-agente (SMA) que es un sistema compuestos por varios agentes inteligentes donde cada uno tiene sus propios objetivos y colaboran entre sí para alcanzar el objetivo del sistema. Imaginar un sistema recomendador como varios agentes que interactúan entre sí para resolver la tarea principal, ofrecer al usuario una lista de recomendaciones, podría ser útil en el desarrollo del mismo. La estructura de un sistema multi-agentes podría seguir el siguiente esquema:

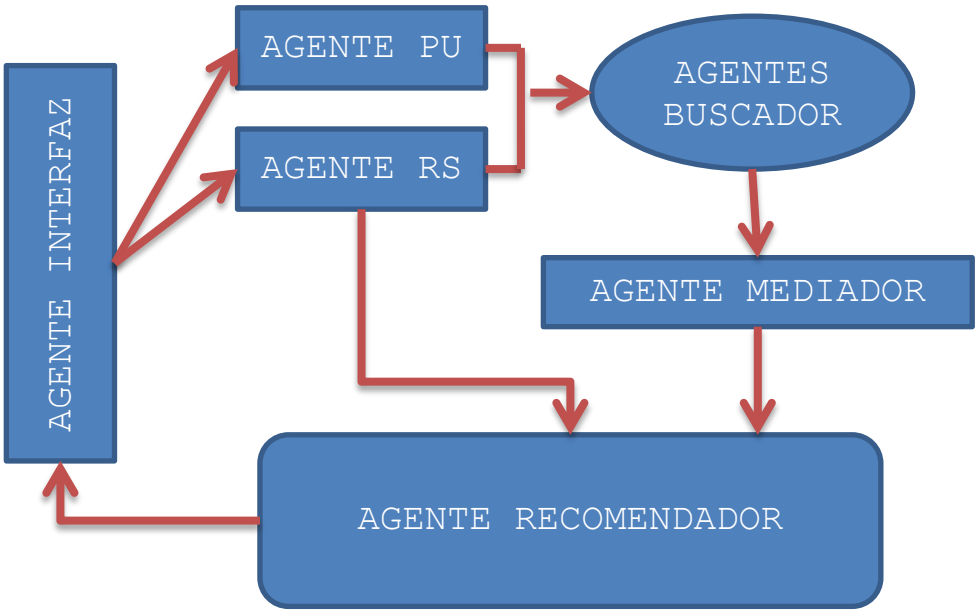


Figura 1 Esquema sistema multi-agente.

Ha nivel de usuario disponemos de tres agentes. El agente Interfaz que se encarga de recoger los datos del usuario y de mostrarle al mismo la lista de las mejores recomendaciones. El agente PU es el agente perfil de usuario. Recibe los datos, preferencias y restricciones necesarias del usuario para construir el perfil de búsqueda. El agente RS es el agente Refinador Semántico. Su función es añadir información a los términos que describen el interés del usuario. Para esto incluye sinónimos y conceptos relacionados.

Los agentes Buscadores son los encargados de buscar en los repositorios los objetos que coinciden con las preferencias y restricciones del usuario. En esta búsqueda entra en juego el perfil de usuario y las relaciones semánticas que se puedan añadir a los términos que describen el interés del usuario.

El agente Mediador recibe todos los objetos de los buscadores y se encarga de pasarle una única lista al agente recomendador. Para ello borra objetos duplicados, unifica la información de los objetos y elimina objetos no deseado.

El agente recomendador es el que contiene la lógica del sistema. Tras recibir el resultado de las búsquedas genera la lista de los posibles ítems que puedan interesarle al usuario. Después devuelve la lista al agente interfaz para que este la muestre por pantalla.

Usando esta estructura se consigue separar el desarrollo del sistema recomendador incluyendo las herramientas usadas. Por ejemplo, el agente interfaz (AI) y el agente recomendador (AR) pueden estar implementados en distintos lenguajes de programación. También si se necesita un nuevo agente buscador para un repositorio específico se podría añadir a posteriori y por último, en cualquier momento se puede modificar uno de los agentes sin que estas modificaciones puedan afectar a los demás agentes.

Se dispone de una gran gama de herramientas que se pueden usar para implementar un sistema recomendador. Por esta razón es difícil encontrar similitudes en los trabajos leídos sobre todo en los trabajos que se usa una estructura de sistema multi-agente. Esto ocurre porque los agentes pueden estar implementados en diferentes

lenguajes con la condición de que se puedan comunicar (interactuar) con los demás.

Lenguajes como java, ruby y phyton se suelen usar para la creación de la interfaz gráfica. También para la creación del perfil de usuario y para la búsqueda de los objetos en los repositorios. Por otro lado los lenguajes de programación lógicos como Prolog son los más usando en la tarea de recomendar sobre todo en aquellos proyectos donde se usen estructuras de base de conocimiento, árboles, redes bayesianas o redes de aprendizajes.

Resumiendo, usar un lenguaje que permita una interfaz de usuario y el tipo de comunicación de los repositorios. Como se dijo antes cada agente puede estar implementado en un lenguaje diferente si se poseen los mecanismos suficientes para conseguir la coordinación del sistema multi-agente.

1.3 Objetivos

El desarrollo de este proyecto estará dirigido por varios objetivos, normalmente, secuenciales. Empezando por el estudio del arte del proyecto y tecnologías. Estudio sobre la implementación de un sistema recomendador y de las herramientas que se usarán en el desarrollo del mismo.

Después se creará la estructura del proyecto siguiendo el estilo de una aplicación MEAN. Para cumplir este objetivo necesitamos crear la estructura de directorios, instalar todas las herramientas necesarias y, por último, crear un repositorio de github como forja del proyecto.

A continuación crearemos una aplicación MEAN durante el aprendizaje de dicha herramienta formada por tres partes. Nuestro sistema recomendador será una parte de esta aplicación.

Otro objetivo es usar librerías de recomendación con lo que dejaremos a un lado la implementación de algoritmos de recomendación y nos centraremos en el uso de librerías. Por esta razón se buscarán, estudiarán y probarán varias librerías de recomendación de Node.js.

Después de probar las librerías por separado las integraremos en la aplicación MEAN. Se diseñará una interfaz gráfica que nos permita definir distintas

variables que se usarán durante la recomendación y la visualización de los resultados.

Como último objetivo realizaremos comparaciones entre los distintos resultados obtenidos con los que determinaremos las conclusiones y líneas futuras del proyecto. Algunos ejemplos serían qué filtro, qué forma de calcular la similitud o qué tipo de algoritmo usar en este sistema.

Por tanto, los objetivos específicos del proyecto abarcan los siguientes aspectos:

1. Estudio del arte del proyecto, tecnologías y herramientas.
2. Creación de la estructura de directorios para el proyecto.
3. Creación de una aplicación MEAN.
4. Buscar, estudiar y probar varias librerías de recomendación.
5. Integración de las librerías en la aplicación MEAN.
6. Creación de la interfaz gráfica del sistema.
7. Obtener las conclusiones y líneas futuras del proyecto.

Capítulo 2. Repositorio de objetos de aprendizaje

Anteriormente se nombró que la tarea principal de un sistema recomendador es proporcionar una lista de los posibles ítems que puedan interesarle al usuario. Para cumplir esta tarea el sistema necesita datos con lo que calcular la recomendación. Ahora la pregunta es ¿Dónde están esos datos?

Normalmente se usa un repositorio de objetos para almacenar y devolver los datos al sistema recomendador. Por esta razón este capítulo está dedicado a dichos repositorios, sobre todo a los repositorios de objetos de aprendizajes, y a los diferentes componentes que los forman.

2.1 Introducción

Un objeto de aprendizaje es una entidad informativa digital desarrollada para la generación de conocimiento, habilidades y actitudes, que tiene sentido en función con una realidad concreta. Estos objetos tienen una cabecera, metadato, donde se almacenan sus características.

Los metadatos son datos que describen otros datos. Se usan para aumentar la descripción de los datos añadiendo información extra como nombre de autor, fecha, precio, etc. El sistema recomendador obviará el tipo de objeto y se enfocará en los metadatos para recomendar usando las características del objeto y no por el tipo. De esta forma da igual si el objeto es una imagen, video, sonido o texto que nuestro sistema podrá usarlo.

Los repositorios de objetos de aprendizaje permiten almacenar, buscar, recuperar, consultar y descargar objetos de aprendizaje de todas las áreas de conocimiento. Estos repositorios son las bases de datos para los objetos de aprendizaje. A diferencia de las bases de datos convencionales, es decir relacionales, los

objetos son guardados en colecciones. El concepto de colección es el mismo que se usa en bases de datos no relacionales como MongoDB.

Para que el objeto pueda ser almacenado y localizado para cualquier uso posterior, tiene que ser previamente etiquetado. El proceso de etiquetado sigue una serie de estándares internacionales que incluyen el identificador del objeto, su título, autor, resumen, descriptores, derechos de autor…

Un ejemplo de repositorio de objetos es Color (colección de objetos reusables). Color es un sistema que permite reunir y consultar objetos de aprendizaje pero que sobretodo permite generar e incorporar los metadatos y archivos específicos que describen a los recursos informáticos y la combinación de los mismos.

Otro ejemplo es LACLO (comunidad latinoamericana de objetos de aprendizaje). Esta es una comunidad abierta, integrada por personas e instituciones interesadas en la investigación, desarrollo y aplicación de las tecnologías relacionas con objetos de aprendizaje en el sector educativo latinoamericano.

2.2 Dspace

Durante el estudio del arte del proyecto se pensó en que repositorio de objetos de aprendizaje usar. Como primera opción se eligió Dspace por ser uno de los más famosos y usados.

Dspace provee herramientas para la administración de colecciones digitales. Comúnmente es usado como solución de repositorio institucional ya que soporta una gran variedad de datos. Los datos son organizados como ítems que pertenecen a una colección y cada colección pertenece a una comunidad.

Este software fue liberado en 2002, como producto de una alianza de HP y el MIT. Liberado bajo una licencia BSD permite a los usuarios personalizar o extender el software.

Muchas instituciones de investigación a nivel mundial utilizan Dspace para satisfacer una variedad de necesidades de archivo o archivaje digital como repositorios institucionales, administración de registros

electrónicos o preservación digital. Actualmente más de 800 instituciones utilizan Dspace.

Para utilizar Dspace optamos por descargar e instalar LibLiveCD que es una versión de Ubuntu con un entorno de escritorio GNOME. Esta versión de Ubuntu viene pre-configurada para la utilización de DSpace.

Para descargar este sistema operativo tenemos que acceder a www.sourceforge.net y buscar LibLiveCD. Después solo tenemos que darle al botón de descargar y esperar a que termine.

Para la instalación necesitamos crear una máquina virtual. Esta tarea se ha realizado con la ayuda de VirtualBox. En esta máquina instalamos la imagen iso de LibLiveCD. Cuando el sistema este corriendo deberíamos ver algo parecido a la figura 2, si no es así, ha tenido que haber un error durante la instalación.

Figura 2 Escritorio LibLiveCD.

A continuación tenemos que abrir el icono de Firefox con nombre ClickMe. Al abrirlo veremos una página que nos

da la bienvenida a DSpace, Koha y Drupal LiveDVD. En nuestro caso solo nos fijaremos en Dspace. Como el sistema ya lo tiene pre-configurado solo tenemos que acceder a la dirección “http://localhost/jspui”. Esta es la ruta de la interfaz gráfica de Dspace en la que podemos crear y consultar comunidades, colecciones e ítems.

Durante un breve estudio de Dspace y varias pruebas con LibLiveCD se ha decidido no usarlo. La interfaz gráfica jspui es liosa y costosa hasta tal punto de no poder subir un ítem al repositorio. Al igual que la documentación que supone un crecimiento de la curva de aprendizaje. Integrar Dspace en este proyecto supondría un coste de tiempo elevado sobre todo si valoramos que el repositorio de objetos de aprendizaje no es la parte más importante en este proyecto.

Capítulo 3. Aplicación MEAN

Actualmente existe una invasión de los frameworks en el desarrollo web. Estos frameworks se utilizan porque aceleran y automatizan el desarrollo y aumentan la eficiencia de la aplicación. En definitiva mejoran el producto final.

A lo largo de este proyecto nos decidimos por utilizar el fullstack MEAN por dos razones importantes. La primera es el desarrollo de una aplicación utilizando Angular por su desarrollo siguiendo el patrón modelo-vista-controlador. La segunda es que el repositorio donde se almacenará los objetos para recomendar será una base de datos MongoDB por ser flexible y fácil de escalar.

3.1 Introducción a una aplicación MEAN

MEAN es un fullstack dedicado a javascript que simplifica y acelera el desarrollo de aplicaciones web. Utilizando los cuatros pilares fundamentales de MEAN se consigue desarrollar el lado del cliente y del servidor de la aplicación al mismo tiempo.

MEAN utiliza express y nodejs para el lado del servidor, MongoDB como gestor de base de datos y Angular para el lado del cliente. Lo especial de estas cuatros herramientas es que todas funcionan con javascript con lo que se facilita la integración. Una aplicación MEAN debe tener una estructura de directorio similar al de la siguiente figura:

Figura 3 Estructura de directorios de una aplicación MEAN.

La estructura se divide en tres directorios principales. En el directorio app se almacena la lógica de nuestra aplicación. En el directorio config se mantiene los archivos de configuración de la aplicación. A medida que se vayan añadiendo nuevos módulos en nuestra aplicación serán configurados con un fichero javascript dedicado. En el directorio public se coloca todo el contenido que se cargará en el cliente.

3.2 MongoDB

MongoDB es la base de datos NoSQL líder, ayudando a las empresas a ser más agiles y escalables. MongoDB es un sistema de base de datos multiplataforma orientado a documentos, de esquema libre. Esto significa que los objetos de una colección pueden tener esquemas diferentes.

Para instalar MongoDB solo hace falta acceder a su página oficial y descargarlo. Una vez instalado tenemos que abrir una terminal (favorablemente como administrador), buscar la carpeta donde está instalado y ejecutar el comando de mongod. En la página también está disponible la documentación oficial y un apartado de enseñanza compuesto por cursos online.

3.3 Express

Express es un framework web mínimo y flexible para Node.js que proporciona un conjunto robusto y de

características para crear aplicaciones web. También dispone de una gran variedad de métodos de utilidad HTTP y middleware con lo que la creación de una API se convierte en rápida y fácil.

Para instalar express hace falta tener instalado el controlador de paquetes de node (npm), abrir una terminal (favorablemente como administrador) y ejecutar el comando "npm install express".

En la web oficial de express se encuentra la guía oficial y algunos ejemplos básicos para empezar con esta herramienta.

3.4 Angularjs

Angularjs es un framework de desarrollo de aplicaciones web que amplía las funcionalidades básicas del lenguaje HTML. El entorno de desarrollo resultante es expresivo, legible y consigue reducir el tiempo de implementación.

Para utilizar Angular tenemos que descargar su fichero ".js" desde su web oficial o desde su repositorio de github. Solamente es un archivo ya que es una librería escrita en javascript.

En la web oficial se encuentra la documentación y la guía de desarrollador. También dispone de un apartado de enseñanza con videos, cursos gratis y ejemplos.

3.5 Nodejs

Node.js es una plataforma para construir fácilmente aplicaciones rápidas de red escalables. Node.js es ideal para aplicaciones en tiempo real de datos intensivos que se ejecutan a través de dispositivos distribuidos. En definitiva nodejs consigue, con la ayuda de distintos modulos, que nuestro código javascript se ejecute como un servicio.

Para instalar nodejs lo hacemos desde su página oficial donde podremos descargarlo. Cuando la descarga termine solo tenemos que abrir el fichero y seguir los pasos de la instalación. Para saber si tenemos instalado nodejs solo tenemos que escribir en la terminar "node --version" y deberíamos tener un resultado.

En la web se dispone de un ejemplo base de nodejs con el que se puede empezar. En él el servicio responde con un “Hello Word” cuando se le hace una petición http al puerto indicado.

Capítulo 4. Librerías de recomendación

En este proyecto dejamos a un lado el desarrollo de un algoritmo de recomendación y optamos por utilizar algún framework o librería. Como los sistemas recomendadores están en auge y ya son casi obligatorios en muchas webs, existen una gran variedad de herramientas que proporcionan ayuda a la hora de recomendar. Puede ser desde una librería de recomendación hasta un sistema recomendador completo.

Existen herramientas de recomendación para lenguajes como php, ruby o java. Como vamos a desarrollar una aplicación MEAN utilizaremos librerías para javascript y como el sistema de recomendación se ejecutara como un servicio de Node.js utilizaremos librerías de Node.js para recomendar.

Durante las búsquedas de las librerías para Node.js se encontraron dos librerías. A continuación se describirá cada una de ellas especificando aspectos como algoritmos o filtros que usan.

4.1 Descripción de las librerías

En este apartado se describen las dos librerías usadas en este proyecto. Ambas son librerías de Node.js lo que significa que solo tenemos que usar el comando "npm install nombre-librería" para tenerlas en la máquina.

La primera librería tiene como nombre Raccoon que es un motor de recomendación con un filtro colaborativo de fácil uso. Esta librería está diseñada como un módulo de npm construida en una capa por encima de Node.js y Redis. Raccon utiliza el coeficiente de Jaccard para determinar la similitud entre los usuarios y el algoritmo k-vecinos más cercanos para crear la recomendación. Esta librería proporciona como resultado de los ítems que más han

gustado en el grupo de personas al que pertenece el usuario.

La librería Raccoon proporciona diferentes métodos que se pueden utilizar para aumentar o mejorar la recomendación. A parte de la lista de los ítems recomendados al usuario podemos pedir una lista de los mejores ítems del sistema, de los peores, de las personas que más se perezcan al usuario y las que menos.

Como requisito necesitamos un servidor de Redis activo. Esto ocurre porque cuando creamos una instancia de la librearía nos conectamos con un servidor de Redis. Esta herramienta es un servidor de estructura de datos diseñada para mapear los datos en pares clave-valor. Se suele definir como un servidor de tablas hash. Las claves pueden ser de diferentes tipos de datos desde strings hasta bitmaps. La librería Raccoon usa Redis para almacenar relaciones entre personas e ítems y consultar los valores, es decir la importancia de dichas relaciones, para la recomendación.

La segunda librería es node-recommendations. En este caso el filtro que se usa está basado en contenido y no en usuarios (filtros colaborativos) como lo hace Raccon. El cálculo de la similitud entre dos ítems se puede realizar de dos formas. La primera opción es una formula básica para calcular distancias y la segunda sería decantarnos por el coeficiente de correlación de Pearson. Se tiene que indicar el tipo de correlación en las opciones del recomendador. Posteriormente se mostrará un ejemplo donde se detallará con más detalles.

Al igual que la librería anterior necesitamos un servidor de Redis activo. Node-recommendations también necesita el módulo de Node.js Menwatch. Este paquete está diseñado para encontrar y reparar fugas de memorías en las aplicaciones Node.js. Durante la instalación de este módulo ocurre un error que no se ha conseguido solucionar. Por este motivo se marca como línea futura la modificación de la librería para que funcione sin necesidad del paquete Menwatch.

4.2 Ejemplos de uso

En el primer ejemplo veremos lo necesario para poder usar la librería Raccoon. Lo primero, como se muestra en

la figura 4, cargamos el módulo de Raccon para Node.js y llamamos al método connect. Este método nos conectará a al servidor Redis. Hay que recordar que es obligatorio el uso de Redis con Raccon, asi que debemos asegurarnos de tenerlo instalado. Déspues de la conexión añadimos los ratings, para esto, llamamos al método liked pasándole como parámetros el nombre del usuario y el nombre del ítem. El tercer y último paso es llamar al método recommendFor el que nos devolverá la lista de los ítems. Este método usa dos parámetros para indicar el nombre de la persona y el número de recomendaciones que deseamos saber.

```
var raccoon = require('raccoon');

raccoon.connect(port, url, auth);
// example of localhost:
// raccoon.connect(6379, '127.0.0.1');
// auth is optional, but required for remote redis instances

/*Add in ratings*/
raccoon.liked('garyId', 'movieId');
raccoon.liked('garyId', 'movie2Id');
raccoon.liked('chrisId', 'movieId');

/*Ask for recommendations*/
raccoon.recommendFor('chrisId', 10, function(results){
  // results will be an array of x ranked recommendations for chris
  // in this case it would contain movie2
});
```

Figura 4 Ejemplo de la librería Raccon.

La figura 5 muestra el ejemplo usando node-recommendations. Como se hiso anteriormente cargamos el módulo de Node.js y definimos las opciones del sistema recomendador. En las opciones se especifican el tipo de correlación que queremos usar como “distance” o “pearson” y el cliente de Redis. A continuación llamamos al método create pasándole un nombre y las opciones. Node-recommendations usa el nombre para crear un namespacing dentro de Redis lo que permite disponer de varias bases de datos usando la misma instancia. Igual que antes tendremos que tener instalado y ejecutándose el servidor de Redis.

```
/* Create a Recommendations instance */
var recommendations = require('node-recommendations');
var options = {
    correlation: 'distance' // distance and pearson are implemented
  , redisClient: null       // [optional] your redis client
};
var r = recommendations.create('Books',options);

/* Add people and critics */
var p = r.addPeople('Joe','optional-id'); // add a new person to the data
p.addItem('Batman',4,'optional-id');           // add a critic score to Joe
p.addItem('Superman',3.5);                          // another...
p.addItem('Titanic',1);

/* Calculate the items similitudes */
r.calculateItemsim(function(err,res) { ... });

/* Get a person by his name */
var person = r.getPeopleByName('Joe');

/* Get recommendations for a person */
person.getRecommendations();
```

Figura 5 Ejemplo de node-recommendations.

Con estos dos pasos tendremos creado un sistema recomendador usando la librería node-recommendations. Lo siguiente es añadir una persona y los ítems que ha valorado. Primero añadimos a la persona usando el método addPeople especificando un nombre y un id. Si no se especifica el id el sistema generará uno. Creada la persona ya podemos añadirle ítems usando el método addItem. Tenemos que indicar un nombre para el ítem, una puntuación y de forma opcional, un id. Si especificamos el id añade la puntación como si fuera de la persona si no, se le añade como de otra.

Después ejecutamos el método calculateItemsim para calcular las similitudes de los ítems del sistema. Este método tarda mucho por lo que se recomienda ejecutarlo en una máquina diferente o en horas muertas. Por último seleccionamos a la persona que queremos hacerle la recomendación usando getPeopleByName y ejecutamos el método getRecommendations que nos devolverá el resultado del sistema.

4.3 Integración en el sistema recomendador

Como se nombró en el capítulo anterior una aplicación MEAN dispone de tres directorios importantes de los cuales dos de ellos están dedicados para el servidor. Ambas librerías se han integrado en el lado del servidor, ya que son módulos de Node.js, lo que significa que solo trabajaremos en esas dos carpetas.

Lo primero es crear un fichero de configuración para cada librería que se guardarán en la carpeta "config" que se puede ver en la figura 3. En estos ficheros es donde se incluye las librerías y se crean funciones para utilizar sus métodos predefinidos.

El segundo paso es crear un fichero controlador para cada librería que se guardarán en la carpeta "app/controllers". Estos controladores incluyen los archivos de configuración y añaden lógica a las librerías. Es decir, si queremos que se verifiquen unos datos antes de recomendar es el controlador el que se debe encargar de ello.

El tercer y último paso es añadir los dos controladores, uno por cada librería, en nuestro controlador del sistema recomendador. Este controlador cargará los datos en las librerías usando para ello los métodos definidos en los controladores y elegirá que librería es la que recomiende, a que persona va dirigida la recomendación y el tamaño de la lista de los ítems. Estos datos son introducidos por el cliente a través de una interfaz gráfica que se explicará en el capítulo siguiente. No entraremos en más detalle con este archivo ya que se explicará con más detalles en el capítulo 6.

Capítulo 5.
Interfaz gráfica

Como se observa en la figura 6 la aplicación MEAN dispone de una interfaz gráfica divida en inicio, probar y leer más. Esta interfaz está diseñada para el lado del cliente lo que significa que para la implementación se ha utilizado Angularjs.

Durante el transcurso de este capítulo se introducirá cada parte de la interfaz y se mostrará un ejemplo donde se cree un banco de prueba y se pida la recomendación para una persona especifica. Después del ejemplo concretaremos ciertas limitaciones que afectan al sistema recomendador.

Figura 6 Interfaz para aplicación MEAN.

5.1 Introducción

Utilizando la barra de navegación disponible en la parte superior como se muestra en la figura 6 se puede navegar entre los diferentes contenidos de nuestra aplicación MEAN.

El primer apartado de la interfaz tiene como nombre "Inicio". Esta sección contiene información general del proyecto como un breve resumen y descripción de las tecnologías que se usan en una aplicación MEAN. También

se muestra una introducción a la teoría de los sistemas recomendadores y una breve descripción del alumno.

El segundo apartado, y el más importante, es la interfaz gráfica de nuestro sistema recomendador, "Probar". Como se puede observar en la figura 7 se disponen de dos formularios, una tabla y dos listas.

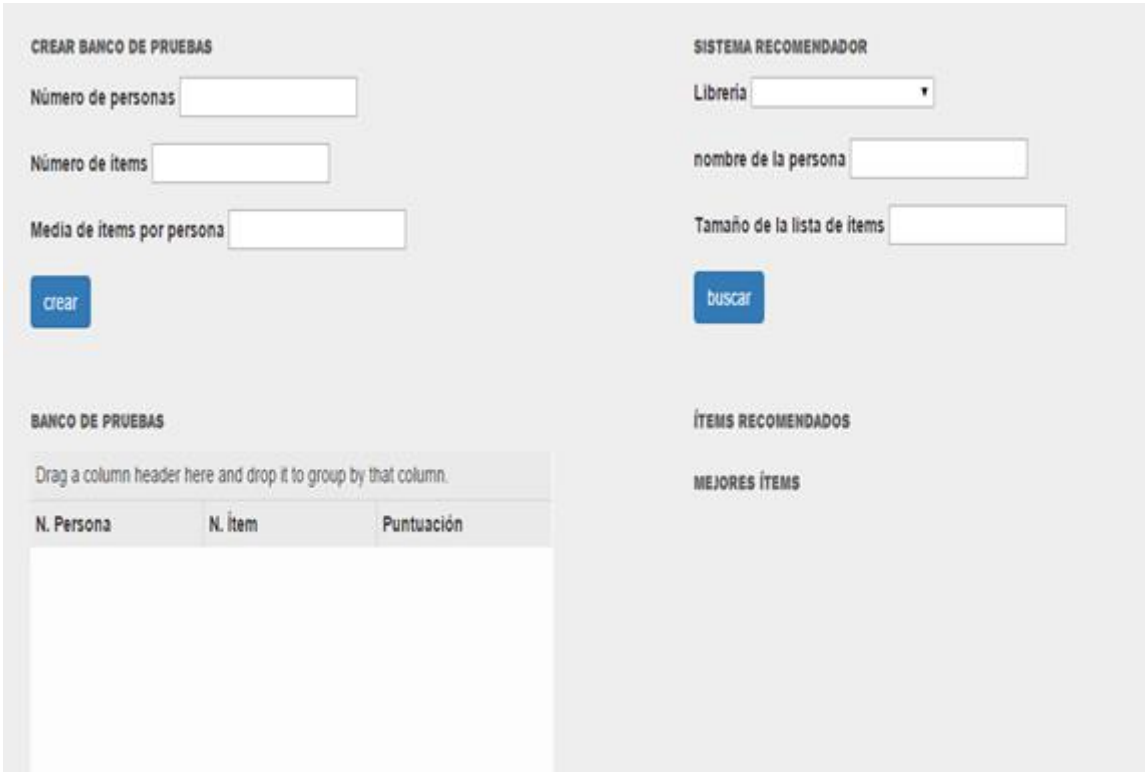


Figura 7 Interfaz gráfica del sistema recomendador.

El primer formulario y la tabla están destinados para la configuración, creación y visualización del banco de pruebas. Después de insertar los valores de número de personas, número de ítems y media de ítems por persona pulsamos el botón "crear". Esto realizará una petición al servicio pasándole los parámetros del formulario. Cuando el banco de pruebas este creado el servicio responderá con los datos. Estos datos son introducidos en la tabla para su visualización. Por último, gracias a Angularjs, podemos agrupar los datos de la tabla como se indica justo encima de ella.

El segundo formulario permite definir los parámetros de la recomendación. Actualmente se puede indicar la librerías que queremos usar (Raccoon o Node-recommendations), el nombre de la persona a la que va dirigida la recomendación y el tamaño de lista de los mejores ítems para el usuario. En el futuro se incluirá un cuarto parámetro para elegir el tipo de correlación

(Distance o Pearson) solo si hemos seleccionado la librería Node-recommendatios.

Después de este formulario se puede observar dos listas. En la figura 7 las listas aparecen vacías porque no se ha realizado ninguna recomendación. Cuando el formulario esté completo pulsamos el botón "buscar" que realizará una petición al servicio pasándole los datos de configuración de la recomendación. Cuando el servicio responda podemos observar la lista de los ítems que se recomienda para la persona. También se visualizará la lista de los mejores ítems del sistema para mejorar el resultado de la recomendación. En un futuro se propone añadir una tercera lista que muestre las personas más parecidas al usuario.

5.2 Ejemplo de uso

Antes de empezar hace falta aclarar los requisitos mínimos para que nuestra aplicación MEAN se pueda ejecutar. Necesitamos instalar las tecnologías npm, Node.js y MongoDB, también disponer de un servidor Redis. A continuación clonamos el repositorio de Github dedicado a este proyecto https://github.com/Tinguaro/TFG.

Cuando este proceso termine accedemos a través de la consola al directorio clonado y ejecutamos el comando "node server" para arrancar la aplicación MEAN. Para que este comando funcione hay que disponer de una segunda consola donde se esté ejecutando MongoDB e iniciar el servidor de Redis.

Si no hay errores, utilizando un navegador web, accedemos a la ruta http://localhost:3000/ como se puede observar en la figura 6 En este ejemplo nos centraremos en el sistema recomendador. Por esta razón avanzamos hasta el apartado "Probar" como se muestra en la figura 7.

Rellenamos los campos del primer formulario y pulsamos el botón crear. Después de este paso observaremos como se han añadido los valores del banco de pruebas en la tabla. En la figura 8 se puede ver como Angulajs nos permite agrupar los datos dentro de la tabla.

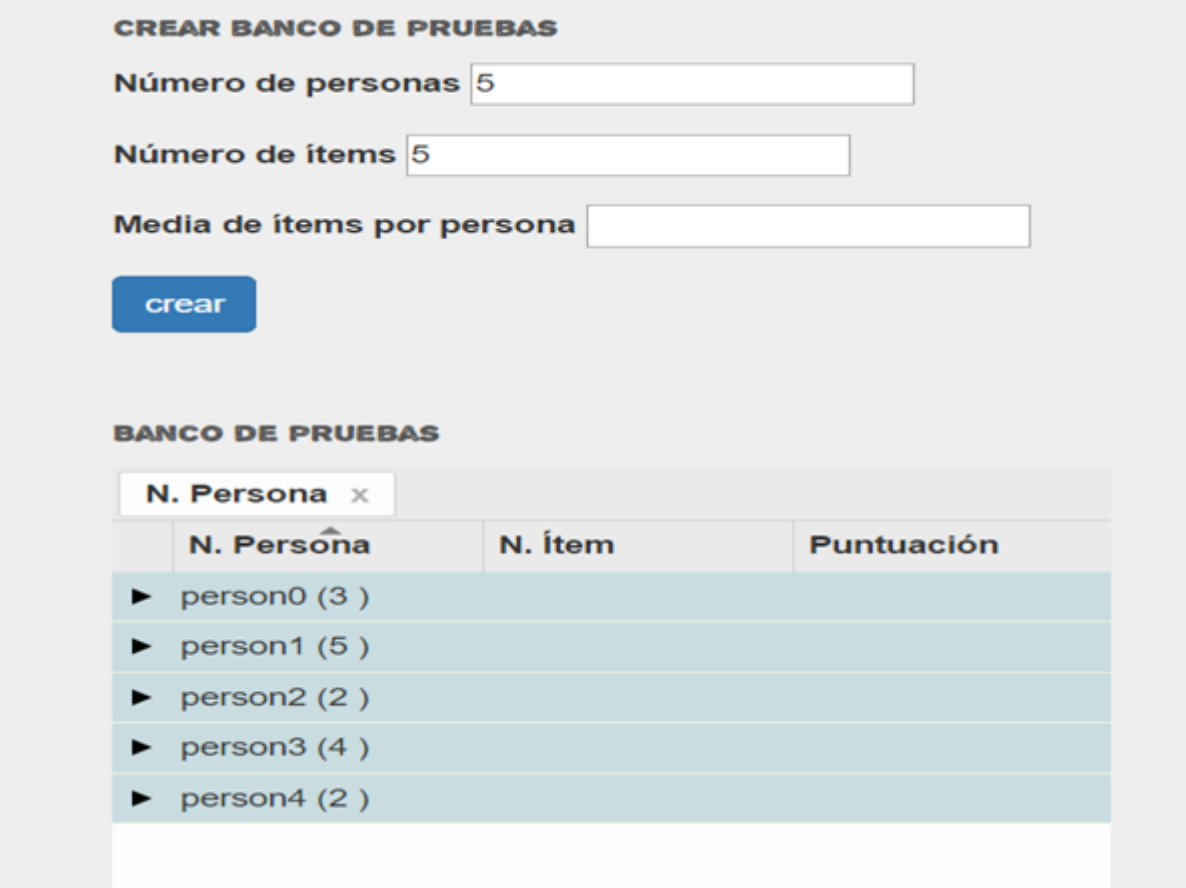


Figura 8 Datos agrupados por nombre de persona.

Para continuar utilizamos la segunda parte de la interfaz. Lo primero es rellenar los campos del formulario indicando el nombre de la persona y el tamaño de la lista. Actualmente solo está disponible la librería Raccoon por los problemas que han surgido en la instalación del módulo Menwatch que requiere Node-recommendations como se mencionó en el capítulo anterior. A continuación pulsamos el botón "buscar" y observaremos un resultado parecido al de la figura 9.

Gracias al agrupamiento de los datos en la tabla, resulta más fácil verificar el resultado de la recomendación. En este ejemplo sencillo se le recomienda los dos únicos ítems que el usuario no ha valorado. Lo normal es observar en el resultado los ítems que resulten los mejores para las personas que más se parezcan al usuario.

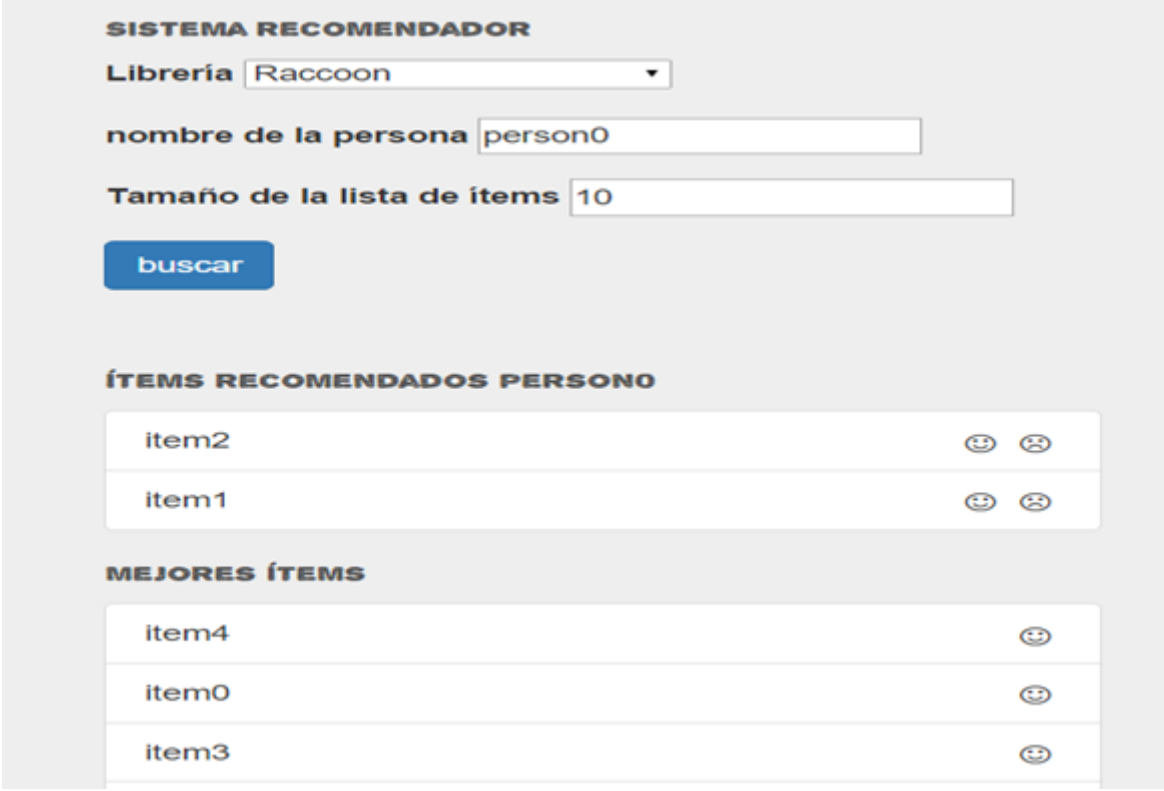


Figura 9 Resultado de una recomendación.

5.3 Limitaciones

En este proyecto, como en la mayoría, se ha puesto a prueba al sistema para verificar el buen funcionamiento del mismo. En este proceso se han encontrado limitaciones afectan al resultado de la recomendación.

La construcción del banco de pruebas, en el peor de los casos, creará una población donde a todas las personas le gustan todos los objetos. En este sentido, las iteraciones básicas producidas por este paso es el resultado de la multiplicación del número de personas por el número de ítems.

Se han probado diferentes tamaños donde se ha llegado a una dimensión límite de 2500 personas y 2500 ítems. Esta construcción tarda alrededor de nuevo o trece segundos en calcularse pero es incapaz de cargar los datos en la tabla. Esto ocurre porque el servicio de Node.js no tiene suficiente memoria para transmitir los datos. Como solución se propone crear un canal de stream para la transmisión de los datos. El resultado de la creación del banco se almacena en un string con formato JSON por esa razón, si superamos esta dimensión se producirá un error por superar el tamaño de la variable string.

Actualmente las librerías de recomendación elegidas para este trabajo están viéndose afectadas por el servidor de Redis. Cuando disponemos de un banco de pruebas con una dimensión media el servidor de Redis falla cuando se cargan los datos. El error indica que han había varios cambios en un tiempo determinado y que el servidor no ha sido capaz de almacenarlo.

Hay que entender que Redis almacena pares de clave-valor y que con una población de 500 personas y 500 ítems suele generar alrededor de 1000 claves. Se propone como línea futura arreglar este problema para mejorar el funcionamiento y rendimiento de nuestro sistema recomendador.

Capítulo 6.
Sistema recomendador

6.1 Introducción

El sistema recomendador está integrado dentro de una aplicación MEAN en el lado del servidor ya que esta implementado utilizando las tecnologías Node.js y Express. En el archivo server.js situado en la raíz del proyecto como se muestra en la figura 3 incluimos e iniciamos el controlador systemrecommend. Después de este paso tendremos nuestro sistema activo y la librería de Raccoon conectada a nuestro servidor de Redis.

Systemrecommend es el controlador del sistema recomendador formado por tres controladores y diversos métodos que nos permiten interactuar con el sistema. A través de peticiones http como una petición get o post el cliente consigue comunicarse con el servicio es decir, nuestro sistema. Así conseguimos, aunque estemos en el lado del cliente, modificar valores y ejecutar métodos que se encuentran en el servidor.

A continuación se especificará con más detalles cada parte o componente del sistema recomendador. En este punto hay que tener claro que cada componente del sistema es un controlador (controller) formado por variables y métodos que nos permiten realizar distintas acciones.

6.2 Componentes del sistema recomendador

El sistema está constituido por cuatro controladores donde el principal y el más importante es systemrecommend. En este controlador incluimos los tres ficheros restantes que forman parte del sistema como son el banco de pruebas, la librería Raccoon y la librería Node-recommendations. Todos los ficheros o controladores están situados en la carpeta “app/controllers” como se puede observar en la figura 10.

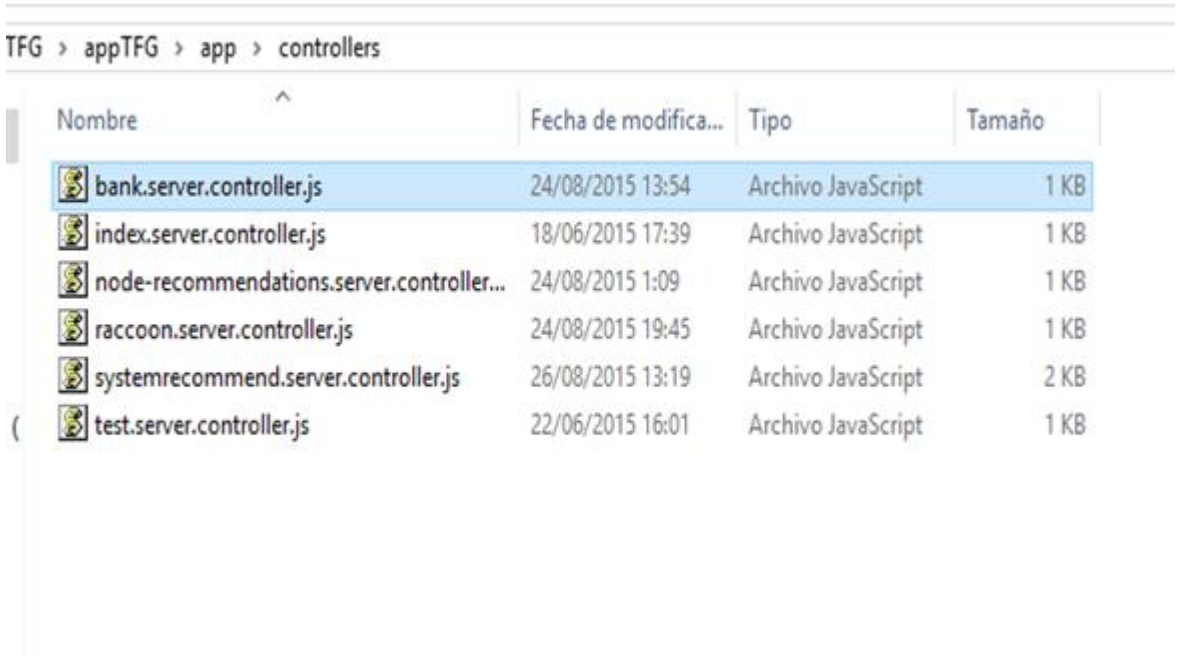


Figura 10 Archivos del sistema recomendador.

El controlador bank está defino para crear, almacenar y devolver los datos cuando systemrecommend se los pida. La construcción del banco se realiza de forma que cada persona dispone de una asignación de los ítems que le interesa. Esta asignación se hace de forma aleatoria con un 50% de probabilidad de que se elija el objeto.

Después disponemos de dos controladores, uno para cada librería. En estos dos ficheros se implementan métodos para manejar todas las funcionalidades que nos ofrecen Raccoon y Node-recommendations.

Systemrecommend es el único controlador con el que el cliente se podrá comunicar utilizando peticiones http. En este fichero se han implementado los métodos necesarios para la utilización del sistema. Cuando se inicializa el controlador se iniciaran también los controladores de las librerías. A parte, podemos crear bancos de pruebas, pedir esos bancos y pedir, a la librería indicada en la interfaz, la lista de los mejores ítems.

Capítulo 7. Conclusiones y líneas futuras

En este proyecto nos propusimos crear un sistema recomendador utilizando el framework MEAN para una posterior integración en el sistema EDHospi. Se ha llegado a la conclusión de que se puede implementar un sistema recomendador utilizando tecnologías webs.

El sistema está implementado como un servicio capaz de responder peticiones http. Con esto conseguimos separar la recomendación del cliente aumentando la velocidad del sistema. También de esta forma podriamos utilizar el mismo recomendador en distintos sitios webs.

Después de probar ambas librerías hemos descubierto que no es favorable utilizar un filtro basado en contenido ya que el cálculo de la similitud de los ítems retrasa la repuesta del sistema. Para solucionar este problema necesitaremos un servicio extra dedicado al cálculo o ejecutar dicho cálculo en las horas con menos intensidad de usuarios.

Como nombramos anteriormente la librería Raccoon utiliza un filtro basado en usuarios con lo que se incrementa la velocidad de respuesta del sistema. Como desventaja corremos el riesgo de disminuir la calidad de la recomendación.

Por último mencionar el ahorro de tiempo durante la implementación del sistema gracias al uso del framework MEAN y de las librerías de recomendación. Como se suele decir, “No reinventar la rueda”.

Como trabajo futuro se propone:

- Modificar la librería Node-recommendations para lograr su funcionamiento. Para esto necesitaremos eliminar la dependencia del módulo Menwatch y borrar las líneas de código donde se haga referencia a esta librería.

- Reparar los errores que se producen en el servidor redis cuando se cargan los datos y se realiza la recomendación.
- Probar el sistema utilizando un repositorio de objetos de aprendizaje. Actualmente el sistema utiliza una población creada aleatoriamente pero está diseñado para cargar datos en formato JSON.
- Probar el sistema en un caso real para verificar su correcto funcionamiento. Para cumplir esta línea futura utilizaremos el proyecto EDHospi.
- Mejorar la recomendación unificando ambas librerías con lo que se conseguiría un sistema híbrido y, posiblemente, una mejora en la calidad de la recomendación.

Capítulo 8.
Summary and Conclusions

In this project we aim to create a recommender system using the framework MEAN for a later integration into the EDHospi system. It has been concluded that a recommender system can be implemented using web technologies.

The system is implemented as a service able to respond to http requests. With this we are able to separate the customer from the recommendation would be able to use to increase the system speed. Furthermore, we could apply or use the same recommender in different websites.

Having tried both libraries, we have discovered that it is not favorable to use a content-based filter because the calculation of the similarity of the items delays the response of the system. To solve this problem we need an extra service dedicated to the calculation or execution of the calculation at times of less user traffic.

As named above, the Raccoon library uses a collaborative filter based on users with which the response speed system is increased. A disadvantage is that we risk diminishing the quality of recommendations.

Finally, the time saving during implementation of the system by using the framework MEAN and libraries recommendation must be mentioned. As they say, “Do not reinvent the wheel”.

As further research, we propose:

- Modify the Node-recommendations library to achieve its performance. For this we will need to eliminate dependence on Menwatch module and delete the lines of codes where reference to this library is made.
- Repair any errors that may occur in the Redis server when the data is loaded and the recommendation is made.
- Test de system using a repository of learning objects. Currently the system uses a randomly

created population but is designed to load data onto JSON format.

- Test the system in a real case to verify proper operation. To comply with this future line the EDHospi project will be used.
- Improve the system by unifying both libraries to create a hybrid system so as to better the quality of the recommendation.

Bibliografía

[1] Jesus Bobadilla, Antonio Hernando, Fernando Ortega, Jesus Bernal, *A framework for collaborative filtering recommender systems,* Expert Systems with Applications, 38:14609-14623, 2011.

[2] Robin Burke, *Hybrid Recommender Systems: Survey and Experiments,* California State University, Fullerton.

[3] Robin Burke, *Knowledge-based recommender systems,* Department of Information and Computer Science, University of California, Irvine.

[4] Silvana Aciar, Debbie Zhang, Simeon Simoff, John Debenham, *Recommender System Based on Consumer Product Reviews,* IEEE/WIC/ACM International conference on web intelligence, 2006.

[5] C. Porcel, A. Tejeda-Lorente, M. A. Martínez, E. Herrera-Viedma, *A hybrid recommender system for the selective dissemination of research resources in a Technology Transfer Office,* Information Sciences, 184:1-19, 2012.

[6] Luis M. de Campos, Juan M. Fernández-Luna, Juan F. Huete, Miguel A. Rueda-Morales, *Combining content-based and collaborative recommendations: A hybrid approach based on Bayesian networks,* International Journal of Approximate Reasoning, 51:785-799, 2010.

[7] Gediminas Adomavicius, Alexander Tuzhilin, *Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions*, IEEE Transactions on Knowledge and Data Engineering Vol. 17, No. 6, 2005.

[8] Rayid Ghani, Andrew Fano, *Building Recommender Systems using a Knowledge Base of Product Semantics,* Accenture Technology Labs, Chicago.