Collaborative Filtering Recommender System for Bodas.net

Víctor Gil Rodríguez

Resumen–
Dado el crecimiento que han experimentado los portales web en los últimos años y el aumento del contenido a mostrar, para muchos es fácil perderse o no seguir navegando por falta de interés. En este artículo se cubre todo el proceso de creación, desarrollo y puesta en marcha de un sistema de recomendación para la empresa bodas.net. Sistema capaz de mostrar contenido afín a los intereses de los usuarios, mejorando así su experiencia en la web.
Palabras clave–
web, sistema de recomendación, filtrado colaborativo, minería de datos

Abstract–
The growth without any control of webs and their content in recent years can make users look away from them, users who can't find what they want or don't see anything appealing enough. And this is a huge problem for companies, because they lose money. In this paper it is described the whole process of design, development and tunning of a recommender system for bodas.net company. A solution able to show users what they want to see, improving their experience in the website.
Keywords–
web, recommender system, collaborative filtering, datamining

1 Introducción

La enorme cantidad de información que las páginas web guardan en sus bases de datos y *data warehouses* muchas veces es mostrada a los usuarios sin ninguna especie de filtro ni contención, demostrando así que la web tiene una gran variedad de información donde elegir. Pero a su vez, esta gran exposición de datos crea un arma de doble filo, ya que dificulta la búsqueda del contenido atractivo o deseado por el usuario. Lo oculta entre masas de información [1]. Además, esta situacion puede llegar al extremo de ser abrumadora para individuos sin suficiente experiencia, rechazando la navegación por dichas webs. Para solucionar estos problemas, surgieron los Sistemas de Recomendación, o RSs, de las siglas en inglés de *Recommender Systems*. Un conjunto de *software* y técnicas capaces de realizar sugerencias de contenido afines a los intereses de los usuarios, facilitándoles la navegación y mejorando su experiencia. Su efectividad se basa en la premisa de que, en la rutina del día a día, las personas se dejan guíar por recomendaciones de otros [2].
En este artículo se detalla el proceso de planificación, diseño, desarrollo, implementación, puesta en marcha, muestra y análisis de resultados sobre el proyecto de creación de un RS para la empresa Bodas.net [3], portal de bodas líder en el mundo.

• E-mail de contacto: victor.gilr@e-campus.uab.cat
• Mención realizada: Tecnologías de la Información
• Trabajo tutorizado por: Juan Carlos Sebastián Pérez (DEIC)
• Curs 2016/17

2 Objetivos del Proyecto

2.0.1 Principales

- Realización de un sistema capaz de predecir los intereses de los usuarios y recomendarles contenido de los foros de debates afín y personalizado.
- Analizar los datos que se disponen de los usuarios y los ítems para definir el conjunto de estructuras de datos y ponderaciones asignadas a cada una de las acciones y valoraciones disponibles.
- Mostrar el contenido relacionado de manera intuitiva, atractiva y simple al usuario. Utilizando herramientas de sugerencias en la navegación de los usuarios por el portal.
- El sistema no debe impactar negativamente en el rendimiento de la web.
- El sistema debe ser escalable con el crecimiento y mejora de la infraestructura de la empresa.

2.0.2 Secundarios

– Posibilidad de addicionar al sistema la recommendación de otros ámbitos del portal.

3 METODOLOGÍA

El proyecto se ha realizado con la metodología ágil de desarrollo de software SCRUM [**?**]. Se realizaron 10 *sprints* en total, y la longitud de cada uno fue de dos semanas. Exceptuando el décimo *sprint*, que constó de 10 días.

El resumen de las tareas realizadas en cada uno ha sido:

TABLA 1: SPRINTS RESUME

SPRINT 1	20 FEBRERO- 5 MARZO
Búsqueda información y estado del arte	
SPRINT 2	6 MARZO - 19 MARZO
Análisis de la información disponible	
SPRINT 3	20 MARZO - 2 ABRIL
Diseño Algoritmos	
SPRINT 4	3 ABRIL - 16 ABRIL
Implementación Algoritmos	
SPRINT 5	17 ABRIL- 30 ABRIL
Implementación Algoritmos y primeros tests	
SPRINT 6	1 MAYO- 14 MAYO
Diseño interfaces de usuario	
SPRINT 7	15 MAYO - 28 MAYO
Finalización Implementación algorítmica	
SPRINT 8	29 MAYO- 11 JUNIO
Tests finales	
SPRINT 9	12 JUNIO - 25 JUNIO
Propuesta de Presentación, dosier TFG e informe	
SPRINT 10	26 JUNIO- 6 JULIO
Preparación para la defensa y presentación del trabajo	

4 ESTADO DEL ARTE

Los RS aparecieron como un área independiente de estudio en los años 50, y, en la actualidad, son imprescindibles para grandes compañías como Amazon, YouTube, Netflix o Yahoo. Esto es debido a que las funciones de un RS, tal y como se indican en [4] son: *Aumentar el número de ítems vendidos.* Donde *ítem* es la palabra utilizada en este área para referirse al objeto o servicio a recomendar. Es obvio que en una web de comercio electrónico, el interés reside en vender más ítems (productos), mientras que en otro tipo de web, el interés podría encontrarse en aumentar el número de páginas visitadas. Ésta es la función principal de un RS, y lo consigue porque se recomiendan ítems a los usuarios con alta probabilidad de que sean de su agrado. Además, estos sistemas no sólo permiten aumentar las ventas, sino *mostrar ítems más diversos.* Permiten a los usuarios encontrar ítems que de otra manera serían difíciles de ver. De esta manera, también se aumenta la *satisfacción y fidelidad de los usuarios.* Por un lado, los RS mejoran la experiencia de navegación por la web, los usuarios encuentran interesantes las recomendaciones. Y por otro lado, si un usuario tiene cierta historia en la web, los RS lo reconocen, y son capaces de dar sugerencias mucho más precisas.

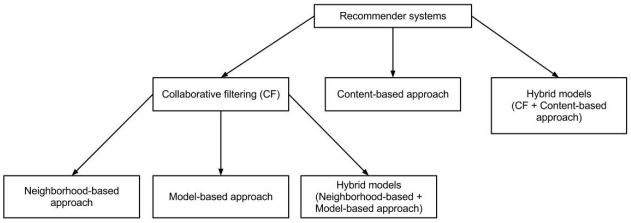

Fig. 1: "Taxonomia de los RS", por Aditi Gupta

Los RS son sistemas de procesamiento de información, recopilan grandes conjuntos de datos para crear sus recomendaciones. Estos datos son principalmente información de los ítems a sugerir y de los usuarios destino. Para poder realizar buenas sugerencias, el sistema debe predecir qué ítems son más atractivos e interesantes para el usuario. Existen diversos tipos de técnicas para conseguir ese resultado. En este proyecto, se ha decantado por la realización de un *Collaborative filtering* [5], CF, técnica que recomienda a un usuario los ítems que gustaron a otros usuarios con intereses parecidos en el pasado. Este tipo de soluciones a menudo reciben el termino de filtrado social, ya que son los usuarios que de forma directa o indirecta evalúan el contenido y calidad de los ítems. Los CF pueden ser clasificados en dos grandes grupos, los basados en vecinos, o *neighborhood-based methods*, y los basados en modelos, o *model-based methods*. Los métodos basados en vecinos, utilizan las relaciones usuario-ítem guardadas en el sistema para calcular las predicciones de futuros elementos. Esto se puede realizar de dos maneras diferentes, conocidas como *user-based* o *ítem-based recommendations*. Las basadas en usuario, calculan el interés de un usuario por un ítem usando los ratings de ese mismo ítem por otros usuarios con gustos parecidos, o lo que es lo mismo, con patrones de puntuación similares. En cambio, los basados en ítems, predicen el atractivo de un usuario por un ítem basado en las puntuaciones de ese usuario con ítems parecidos. La similitud entre ítems és más alta cuando múltiples usuarios los han puntuado de manera parecida. De aquí el término*neighbors*. Por el contrario, las soluciones que utilizan un modelo, no utilizan directamente las relaciones usuario-ítem del sistema, sino que a partir de ellas aprenden un modelo predictivo. Estos sistemas deben ser entrenados con factores que representen las características de los usuarios e ítems, para así poder predecir.

Una simple pero ilustrativa taxonomía de los sistemas de recomendación puede apreciarse en la Figura 1, dónde se muestran además, las diversas combinaciones de las clasificaciones explicadas, que dan lugar a nuevos acercamientos.

En la realización de este proyecto, el método escogido ha sido el de un CF utilizando la técnica de vecinos cercanos con métodos basados en ítems. Los motivos de estas selecciones residen en que las técnicas de CF son las más populares y extendidas dentro de los RS, y el método de vecinos cercanos también presenta mucha popularidad gracias a su eficiencia, simplicidad y precisos resultados.

5 DEFINICIÓN DEL PROBLEMA

Bodas.net ofrece a los usuarios la oportunidad de planificar al detalle el día de su boda con todo tipo de herramientas de organización. Para ello, se cuenta con centenares de miles de empresas clientes que abarcan todos los servicios imaginables: banquetes, catering, alquiler de coches, vestidos, trajes y complementos, fotografía, vídeo, animación, flores, joyería... que son recomendadas y contratadas mediante el portal. La web no solo está disponible en versión de escritorio, también está adaptada para dispositivos móviles, e incluso cuenta con varias app para Android y IOs.
La web cuenta con una de las comunidades de usuarios más grandes en el sector. Dispone de foros donde miles de novias y novios comparten sus experiencias y consultan todas sus dudas con otras parejas. Éstos están clasificados en distintos grupos, como moda nupcial, juegos, fiesta... Estos grupos contienen muchos debates creados por los usuarios, y la base de este proyecto consistirá en analizar estos debates y los usuarios que participan en cada uno de ellos para crear un RS que sugiera otros debates de interés mientras se navega por ellos. Es decir, que al visitar uno, el usuario continúe mirando otros debates, ya que los sugeridos le están pareciendo interesantes. Por eso, estos debates, se definirán como los ítems a recomendar por el sistema. Nótese que a diferencia de otro tipo de portales, como podría ser un comercio electrónico donde los usuarios compran productos y los ponderan explícitamente según lo mucho que les ha gustado, como el caso de las estrellas en el rating de Amazon, en ellos se dispone directamente de esta informacíon, pero en este proyecto no ha sido así. En los foros de esta comunidad no se ofrece la posibilidad de que los usuarios cuantifiquen lo mucho que les atrae cierto tema de conversación. Por eso, en el siguiente apartado se describe detalladamente el proceso de análisis del conjunto de datos con el que se ha trabajado para realizar los algoritmos que calculan los ratings, que también se explicarán en la siguiente sección.

6 ANÁLISIS DE DATOS Y CÁLCULO DE RATINGS

Como se ha comentado, las puntuaciones de estas relaciones usuario-debate son vitales para la realización de un filtrado colaborativo basado en vecinos, ya que indican la semejanza que existe entre los elementos, en este caso, los debates. Además, influyen directamente en los resultados finales y la precisión del RS. Existen varias formas de medir el interés de un usuario por un ítem, pero siempre se depende de los datos disponibles. Para este proyecto, se ha contado con los datos de la comunidad de Brasil. Esta comunidad dispone de millones de usuarios activos, que crean miles de debates cada mes. Los datos que dispone la empresa sobre cada debate son limitados, para el RS se han tenido en cuenta:

- *Listado de participantes y mensajes.* Usuarios registrados que han comentado en el debate.
- *NMENSAJES.* Número de mensajes totales de usuarios registrados dentro del debate
- *VISITAS.* Número de peticiones al servidor por un determinado debate y el usuario está registrado.

TABLA 2: DATOS BRASIL, CASAMENTOS.COM.BR

AVG(NMENSAJES)	$STDDEV(NMENSAJES)$	$AVG(VISITAS)$	$STDDEV(VISITAS)$
32.886	157.636	173.582	638.594

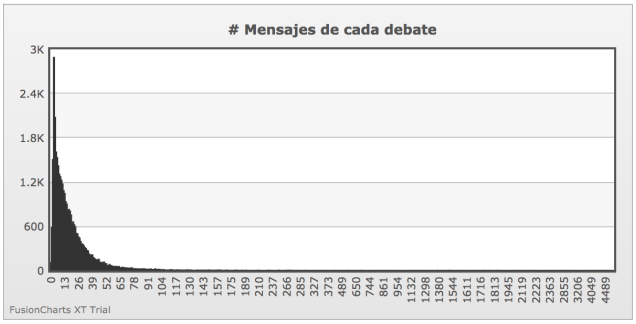

Fig. 2: Mensajes en debates este 2017

- *FECHA CREACION.* Fecha en la cual el usuario dió origen al tema.

Por cada usuarió, también se ha usado:

- *NMENSAJES DEBATE.* Número de comentarios del usuario en un mismo debate.
- *NMENSAJES TOTAL.* Número de debates totales en los que el usuario ha participado.

Para comprender como están estos datos, la tabla 2 refleja un primer análisis para este último año 2017. Como se puede observar, existen desviaciónes muy elevadas, esto es debido a que no existe una distribución normal en los debates, como indican los gráficos del número de mensajes, figura 2, y número de visitas, figura 3. En ellos se agrupan el número de coincidencias iguales de cada tipo. El eje de abcisas indica el número de comentarios o visitas en el debate, mientras que el de ordenadas dice el número de debates totales que tienen esa cantidad.

A partir de este análisis y de los intereses de la empresa se ha creado el algoritmo de cálculo de ratings de cada pareja usuario-debate. Los objetivos marcados son, priorizar los debates con más comentarios y visitas, ya que tienen un gran factor de éxito antes de comenzar. Además de dar más importáncia a los debates más novedosos. Por eso, el proceso de cálculo se ha dividido en dos partes. La primera, aplicar estas restricciones a cada debate, para así en la segunda fase, encontrar el interés de cada usuario con cada tema partiendo de esa base precalculada.

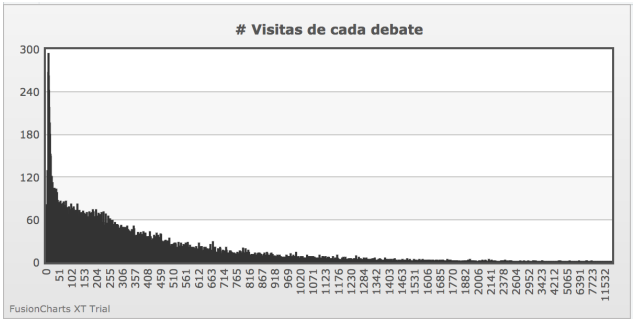

Fig. 3: Visitas en debates este 2017

La fórmula 1 muestra el proceso de cómputo del rating para cada ítem, a partir de sus propias características.

$$R(I_k) = max((\frac{NMENSAJES_k}{VISITAS_k}) * (min(\sqrt{(}NMENSAJES_k), 10) + min(\sqrt{(}VISITAS_k/10), 10)) + calcTimeRate(FECHACREACION_k)), 0) \quad (1)$$

Donde para calcular el rating de un ítem *k* del conjunto de ítems *I*, se obtiene el número de mensajes realizados en ese debate, $NMENSAJES_k$, el número de visitas, $VISITAS_k$, se les aplica una raíz cuadrada y se suman. Esta raíz es aplicada debido a la gran desviación que hay presente entre debates, de esta manera, y aplicando un mínimo, *min()* sobre 10, se normalizan los datos. Si un debate tiene más de 100 comentarios o 1000 visitas, dispondrán de la mayor puntuación, si no es así, y tienen menos cantidad, se verá reflejado. Esta raíz y coeficientes se han escogido porqué comprenden el 97,9% y 98,7% de los datos totales, NMENSAJES y VISITAS respectivamente. Tal y como se observa en las figuras 2 y 3. Una vez sumados, se les aplica un factor de corrección que indica la atracción general que ha causado este debate en los usuarios, con la fracción $NMENSAJES_k/VISITAS_k$. Después entra en juego la fecha de creación del tema. $FECHACREACION_k$ es un timestamp, y dentro de la función *calcTimeRate()* se realiza una conversión a meses, y se compara con el timestamp del mes actual. Y a partir de esta diferencia, se aplica la función 2. El objetivo de esta función es medir la antigüedad del debate, decrementando su valor conforme la diferencia con la actualidad aumenta. $DiffDate_k$ Es la diferencia con el presente, en meses, y el valor constante 10 indica los límites de puntuación. Esta función se trata de $x^{2/3} + y^{2/3} = 10^{2/3}$, y se encuentra representada en la figura 4. Donde se aprecia que esta constante marca tanto la puntuación máxima, como el límite de meses vista a tener en cuenta para puntuar.
Como resultado, la puntuación máxima que un debate por sus características puede alcanzar, es de 30.

$$TR(DiffDate_k) = 10^{\frac{2}{3}}\sqrt{(}10^{\frac{2}{3}} - DiffDate_k^{\frac{2}{3}}) - DiffDate_k^{\frac{2}{3}} * \sqrt{(}10^{\frac{2}{3}} - DiffDate_k^{\frac{2}{3}}) \quad (2)$$

Una vez que se han aplicado estas funciones a los debates, comienza a calcularse los ratings de los usuarios con éstos. Para ello, se aplica la función 3. Que calcula para cada usuario *i* del conjunto de usuarios *U* su atracción con el debate I_k. El parámetro $(NMENSAJES_{ik})$ indica el total de mensajes del usuario *i* en *k*, que es multiplicado por el rating del debate *k* calculado anteriormente. Después se aplica un factor para incidir en la importáncia de ese usuario en el sistema para la posterior predicción. El concepto del que parte esta idea es simple, no todos los usuarios votan por igual, no es lo mismo un usuario que comenta en lo que le interesa y le atrae, que otro que comenta en todo para hacerse notar. Por esta razón, el parámetro $NDEBATES_i$

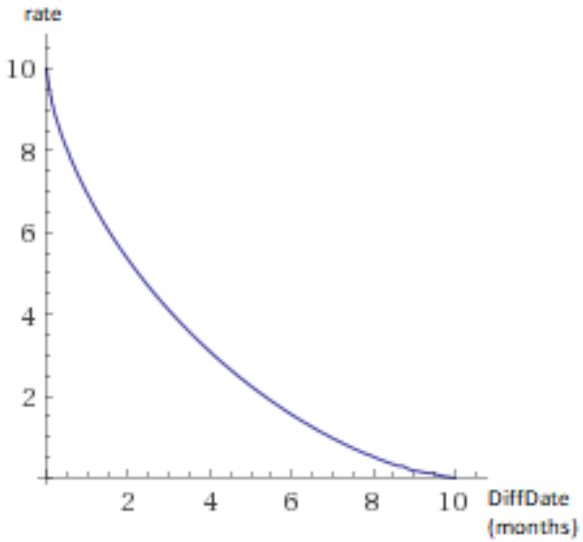

Fig. 4: Función calcTimeRate()

incluye el número de debates diferentes en los cuales a participado el usuario *i*. Y se aplica una diferencia con el rating del usuario calculado, aplicando una función logarítmica y un *min()*. Que indica que contra más debates participados, menos importará su aportación a la predicción final. Y el límite se encuentra en usuarios con más de 100 mensajes en distintos debates.

$$TU(U_i, I_k) = max(NMENSAJES_{ik} * R(I_k) - min(\log_{10}(NDEBATES_i), 2), 0) \quad (3)$$

7 IMPLEMENTACIÓN KNNEIGHBOURS

Una vez se han calculado todas las puntuaciones usuario-debate, se aplica el algoritmo de vecinos cercanos. Como se comentaba en la sección 4, esta técnica utilizará estas relaciones para realizar las predicciones, y poder recomendar los ítems.

Para su desarrollo, el diagrama de la figura 5 muestra los pasos implementados que conforman el algoritmo. Primero, estas puntuaciones se combinan formando todas las parejas posibles de debates en que un usuario a participado, *Joined Ratings*. Se realiza un join sobre las claves del usuario. Así se consigue obtener información sobre que debates están relacionados por comentarios de un mismo usuario en ambos. Después, se aplica un filtro, *Filter not Interesting*, que elimina casos duplicados y casos donde la puntuación del debate no es mayor a 0, no son interesantes. La función *Make Pairs* generaliza estos datos computados. Abstrae el usuario de la relación agrupando por la clave ítem-ítem creada en los pasos anteriores. Obteniendo un resultado de la forma indicada en la función 4. Para cada pareja de ítems I_k,I_m se obtiene una lista de tuplas con las puntuaciones de todos los usuarios que comentaron en ambos debates, que serán la entrada de datos necesaria para la parte fundamental del *kNNeighbours*, el cálculo de similitudes.

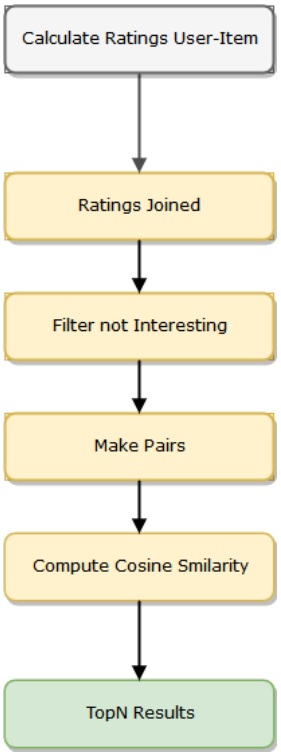

Fig. 5: Diagrama implementación kNNeighbours

$$(I_k, I_m) = (TU(U_i, I_k), TU(U_i, I_m)), \\ (TU(U_s, I_k), TU(U_r, I_m)), \\ (TU(U_t, I_k), TU(U_v, I_m))... \quad (4)$$

La creación de las similitudes entre los debates, o lo que es lo mismo, la predicción que indicará la distancia entre los vecinos y definirá que se debe recomendar, se puede realizar con diversas técnicas. En este proyecto, se ha optado por la *similitud del coseno* [4]. Esta técnica trata a las tuplas de ítems de 4 como vectores de un espacio n-dimensional, donde *n* es el número de elementos que la componen, en este caso, usuarios que realizaron comentarios en ambos temas. Así, es posible determinar su semejanza mediante el coseno del ángulo que forman estos dos vectores debate. Dando un rango de valores (0,1), siendo 1 el resultado máximo cuando se compara un vector con sí mismo. Esto se expresa como se indica en 5, donde • es el producto vectorial, y $\|I_k\|$ es la norma del vector.

$$cos(I_k, I_m) = \frac{I_k \bullet I_m}{\|I_k\| \, \|I_m\|} \quad (5)$$

El resultado obtenido además de esta predicción, contiene el número de coincidencias con el que se formó, ya que un listado de tuplas más elevado en la pereja item-item indicará un resultado más preciso. Y será usado en la parte final del algoritmo, *TopN Results*. Cuando todo este cómputo se ha realizado y guardado, la manera de utilizarlo es simple. Para determinar una recomendación sobre un debate, basta con consultar para ese mismo ítem la sentradas con una predicción más cercana a 1 realizadas con el mayor número posible de coincidencias. Para ello, la búsqueda topN asigna unos umbrales mínimos de calidad, y retorna un listado con las mejores recomendaciones.

8 Tecnologías utilizadas

8.1 Apache *Spark*

La implementación de la algorítmica descrita en los apartados anteriores se ha realizado con la herramienta de la fundación Apache, *Spark* [6]. Un sistema de cómputo general de código abierto para clústeres. Está diseñado para conseguir altos rendimientos con conjuntos de datos muy grandes. Esto es debido a que es capaz de mantener en memoria las instrucciones de los algoritmos sin necesidad de escribir los resultados cada vez que se procesa un dato. Además incorpora APIs de alto nivel para diferentes lenguajes de programación. Según los expertos es el sucesor del framework *Hadoop*. El script de ejecución de Spark se ha realizando en el lenguaje de programación Scala, un lenguaje de programación multi-paradigma, que permite tanto programación funcional como orientada a objetos. Y es ideal para este tipo de proyectos de procesamiento masivo de datos. Éstos datos se encuentran en bases de datos SQL estructuradas, y la información se extrae usando una herramienta de Spark de alto nivel para este tipo de datos, *Spark SQL* [7].
Las razones por la que se ha optado por este *framework* son principalmente dos. La primera reside en que en la empresa hasta ahora no había utilizado modelos de programación distribuida para la realización de sus procesos, y para poder implementar una aplicación con tanto análisis de datos hacía falta un sistema capaz de soportar este cómputo. Y Spark fué escogido como la opción predilecta debido al movimiento de grandes empresas como Microsoft, Amazon o Google al integrarlo en sus productos de *Big Data*. La segunda razón está relacionada con la primera, al ser la primera aplicación de estas características, se necesitaba un *framework* flexible que pudiera ejecutarse sobre una infraestrucura que en un futuro se creara, como Apache Mesos o Hadoop YARN, la cual no existe actualmente. Y Spark es capaz de correr encima sin problemas, incluso permitiendo acceder a datos de diversas fuentes, el mismo HDFS de Hadoop o Cassandra por ejemplo.

8.2 Docker

Docker [8] es un software libre que permite desplegar, automatizar y gestionar aplicaciones en unidades aisladas que contienen los requisitos y herramientas necesarias para su ejecución, los llamados *containers*. Estos contenedores son vistos por el sistema operativo principal como un proceso más, y utilizan su kernel para la ejecución. Internamente contienen la aplicación con todas sus dependencias, y solo disponen visión del sistema de ficheros virtual. Gracias a esto y a que Docker asigna los recursos, como número de cores, tiempo CPU, memoria... entre los contenedores que se ejecutan, permite aislar completamente las aplicaciones entre ellas.
Los contenedores son creados a partir del concepto de *imágenes*, que son los paquetes que contienen el OS, aplicación y dependencias a ejecutar de forma estática. Luego, es posible levantar múltiples contenedores a partir de la misma imagen. Estas imágenes son totalmente customizables, a partir de un fichero *Dockerfile*, el usuario puede crearlas o modificarlas según las necesidades de la apli-

cación. En este proyecto se ha creado una imagen con el sistema operativo Ubuntu v16.04, a partir de una imagen obtenida del repositorio *DockerHub* [9], a la cual se le ha instalado el entorno necesario para ejecutar spark. Simplemente instalando la última versión de java, JDK8, añadiendo las librerías necesarias, y exponer los puertos para acceder desde el exterior, es posible de ejecutar. Esto se debe a que Spark se ejecuta sobre la máquina virtual de java, así que exportando el proyecto como .JAR, es suficiente. En la sección 9 se detalla el proceso de creación de los contenedores para la creación de una arquitectura clúster que ejecute el código.

8.3 Desarrollo web

Para la muestra de resultados en el portal, se han utilizado tecnologías clásicas para el desarrollo web, como PHP, JavaScript y HTML.

9 Puesta en marcha y ejecución

La máquina dónde se tiene pensado ejecutar la aplicación tiene las características hardware siguientes:

- 40x Intel(R) Xeon(R) CPU E5-2650 v3
- Cada cpu está compuesta por 10 cores, y trabajan a una frecuencia de 2.30GHz
- Cada uno cuenta con una memoria caché L3 de 25 MB
- Se disponen de 48 GB de memoria RAM DDR4 para cada CPU

En esta máquina se ejecutan todo tipo de procesos de la empresa, que requieren unas librerías y unas versiones de software determinadas, y como no se dispone de ninguna infraestructura para la ejecución de la aplicación con Spark, se optó por la utilización de contenedores Docker. Como ya se vió en la sección anterior, es posible aislar la aplicación del resto con esta tecnología, y hasta es posible crear una estructura de clúster. Para correr en modo clúster una aplicación Spark, figura 6, cada una es vista como un proceso independiente, y está coordinada por un objeto del programa principal, el *SparkContext*, que es el *driver* del programa. Una vez el driver se ha conectado al *Cluster Manager*, se realiza el submit del trabajo con sus necesidades indicadas. Este manáger puede ser Apache Mesos, YARN de Hadoop o uno propio incluido con Spark, llamado *standalone*. En este proyecto se ha utilizado éste último, ya que permite de manera sencilla la inicialización y puesta en marcha del clúster. Este manager asignará los recursos pedidos por el driver, encontrará los *workers* para el trabajo, y les enviará el código y las dependencias necesarias para la ejecución. Finalmente, el driver se comunica con los workers para asignarles las tareas.

Para la recreación del clúster con Docker, se ha realizado tal y como indica la figura 7. Donde se han levantado tres contenedores, uno para el master, y uno para cada worker. Para el deploy, se lanza el driver desde un proceso de un worker dentro del cluster. Esto es así porqué es preferible que se encuentre dentro de la red local, ya que organiza las tareas de los workers.

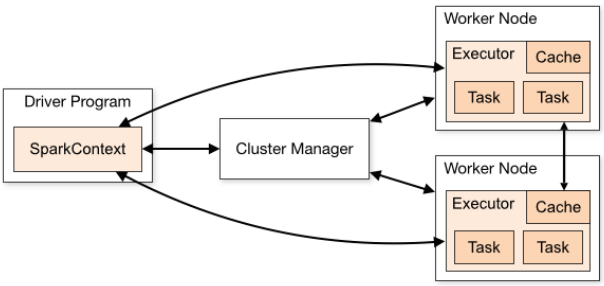

Fig. 6: Spark Cluster Mode

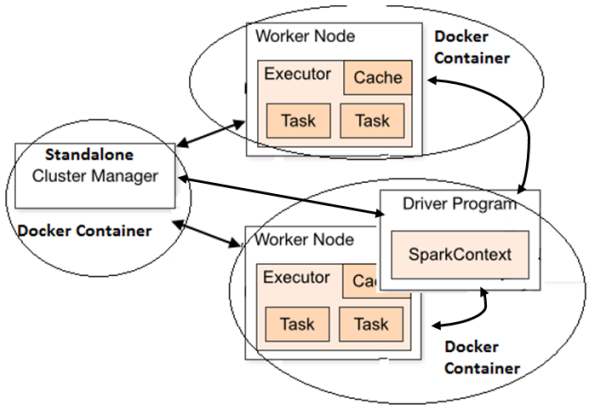

Fig. 7: Spark Cluster Setup Docker

Para que la creación de los contenedores se realice de manera automática, en docker se puede definir un fichero .yml de configuración. En él, se indica todo lo necesario para levantar el clúster, desde la asignación del master, los workers y sus recursos, la red, el mapeo de puertos... Una vez se levantan los containers, es posible ver su estado en la UI web, figura 8. Aquí se muestra un resumen de los workers y sus estados, además de las aplicaciones y drivers a ejecutar, ejecutándose y finalizadas. Pudiendo ver su estado en tiempo real.

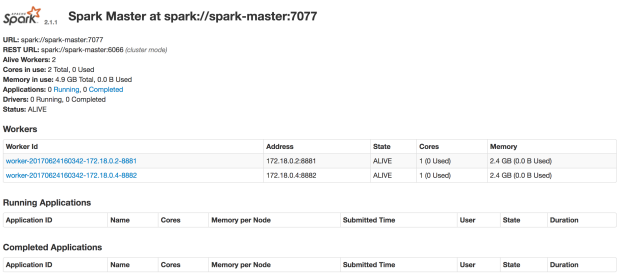

Fig. 8: Spark Website UI

[H] La figura 9 muestra la aplicación ejecutándose en el clúster comentado anteriormente, el driver se ejecuta en un worker, mientras que la aplicación lo hace en el otro.

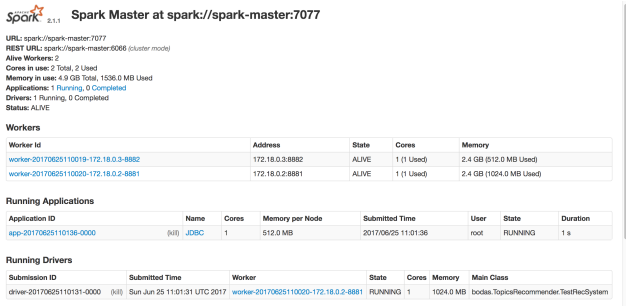

Fig. 9: Spark Job Running

10 Resultados finales

10.1 Datos obtenidos

Una vez la aplicación finaliza su ejecución, los datos resultantes son guardados en las bases de datos de la empresa. En la figura 10 se muestran un subconjunto de datos obtenidos. La primero columna de la izquierda, `ID_TEMA`, indica el debate del cual se quiere buscar una recomendación, el que actualmente se está visitando en la web, mientras que la columna `ID_TEMA_REL` es el debate que será recomendado. La columna `SCORE` es el resultado final de los cálculos de similitud entre ambos debates, expresados tal y como se indica en las secciones 6 y 7, siendo 1.0 la similutud máxima. La última columna, `OCURRENCES` es un valor muy importante, ya que indica si es una predicción interesante o no. Marca la fidelidad del campo `SCORE`, muestra el número total de coincidencias con las que se calculó, el número de tuplas de la relación de 4. Cuanto más alto, mejor precisión. Y esto es lo que ocurre en la imagen 10, se aprecia una anomalía en varios casos, en los cuales se indica una relación de similitud máxima, 1.0. Para esos resultados, el número de `OCURRENCES` con las que se calculó es bajo. Por contrario, cuando éste empieza a crecer, ya no ocurre, figura 11.

ID_TEMA	ID_TEMA_REL	SCORE	OCURRENCES
335462	337882	1.0000	3
336088	336674	1.0000	4
335522	341002	0.9114	3
333972	339004	1.0000	3
335832	338662	0.9655	15
335544	339660	1.0000	2
335132	336488	0.9131	4
335306	336608	0.9160	5
335074	342140	1.0000	3
339198	343578	1.0000	2
333846	343780	1.0000	2
337078	341156	1.0000	6
335826	343536	0.9424	10
335168	338064	1.0000	4
335200	343766	1.0000	4
334144	341066	0.9266	2

Fig. 10: Ejemplo resultados finales

ID_TEMA	ID_TEMA_REL	SCORE	OCURRENCES
335822	336762	0.9380	163
335846	342204	0.9830	162
335816	336762	0.9442	162
335832	336762	0.9339	161
335818	336762	0.9616	160
335814	342310	0.5954	142
335828	342310	0.6753	139
335834	342310	0.6616	137
335822	342310	0.5934	137
335826	342310	0.5944	137
335820	342310	0.5930	136
335830	342310	0.5969	136
335832	342310	0.5996	136
335824	342310	0.5963	134

Fig. 11: Ejemplo resultados para valores *OCURRENCES* altos

Si se observa a detalle los datos, hay ciertas similitudes que a pesar de que se han formado a partir de un número pequeño de coincidencias, no muestran una puntuación 1.0. Esto se debe a la manera en como la relación usuario-ítem es calculada, fórmula 3. Este error se forma en la parte donde se aplica la función *min()* y el logaritmo. En un principio, la idea que se buscaba era diferenciar como puntúan los usuarios, dando más importáncia a quien ha comentado menos en la comunidad, partiendo de que de esta manera, las relaciones serán más precisas. Sin embargo, esto crea un fenómeno peculiar cuando se dan las condiciones necesarias, que soprendentemente son bastante frecuentes para debates nuevos. Si eres un usuario muy activo en la comunidad, tendrás muchas posibilidades de responder muy rápido a los nuevos debates de los demás usuarios, y según como se calcula el rating del debate y del usuario en él, todos los que comenten con más de 100 respuestas realizadas, ponderan igual. Así que para debates con muy pocas respuestas, hay muchas probabilidades de que sean de usuarios con un largo historial en la web, por tanto, si se relaciona ese debate con otro de las mismas características, el resultado es similitud total.
Para evitar este problema, se aplica la técnica *topN Results* comentada anteriormente. Consiste en devolver para cada consulta de recomendación, un listado de resultados posibles que cumplen ciertas condiciones. Para este proyecto se han establecido dos *thresholds*, o umbrales, para asegurar la calidad de la recomendación. Debe tener una `SCORE` mayor a *0.94*, y se debe haber calculado a partir de más de 10 `OCURRENCES`. De esta manera se evitan en gran medida los 1.0. Sin embargo esto automáticamente afecta a los nuevos debates, que deben esperar hasta conseguir tener un cierto número de respuestas.

10.2 Muestra de resultados en la web, diseño

La muestra de los resultados de la aplicación en la web, se ha realizado de manera que resulte cómoda y atractiva para su uso por parte de los usuarios. Se ha implementado en tres lugares distintos. Primero, al navegar por un

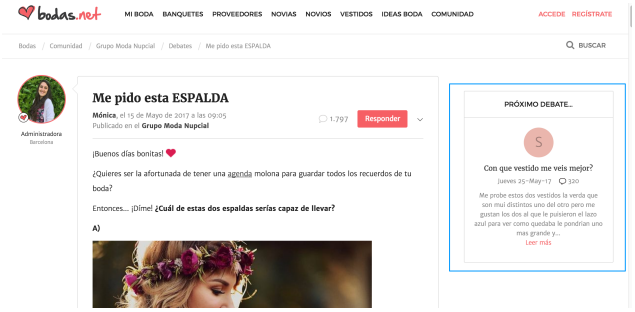

Fig. 12: Debates relacionados en *desktop*

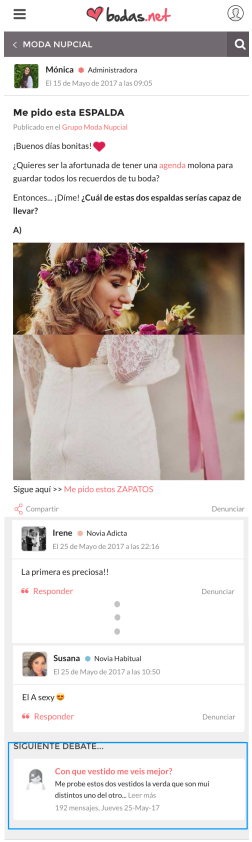

Fig. 13: Debates relacionados en *mobile topics*

Fig. 14: Debates relacionados en *mobile photos*

debate, se muestran los debates relacionados tanto en la versión de escritorio, figura 12, como en la versión móvil de la web, figura 13. Eligiendo una única opción como recomendación.

Además también se han añadido sugerencias de debates en un apartado de fotografías que aparecen dentro de debates en el portal para la versión móvil, figura 14. De esta manera, al acabar la lectura de los comentarios o de observar las fotos, es muy fácil hacer click en las sugerencias.

Destacar que el sistema de recomendación aún no está en línea de producción, no es visible para los usuarios. En un futuro próximo se incluirá.

11 Líneas futuras

Como líneas de trabajo futuras, en primer lugar habría que añadir el proyecto a producción, y comenzar a evaluar el rendimiento que se consigue. En segundo lugar, se debe revisar el problema de las recomendaciones de debates nuevos. Modificar y encontrar una solución mejor para las aportaciones de las calificaciones de los usuarios en el debate, evitando así esta penalización. Una vez se hubiera arreglado, existe un gran margen de ampliación del proyecto, ya que se podría ampliar su zona de influencia más allá de los foros de debates. Enlazar la recomendación con proveedores de vestidos, zapatos, joyería, banquetes... Según el debate visitado, sería posible la personalización en ese sector.

Además, cuando el sistema esté en marcha, y los usuarios comiencen a darse cuenta de su existencia, habrá que ir mejorando las medidas para defender al sistema frente a quien intente atacar al sistema para lograr su mal funcionamiento. Añadiendo aún más robustez.

12 Conclusiones

En este artículo se ha hablado del proyecto de creación de un sistema de recomendación para la web de la empresa *bodas.net*. Se ha realizado una introducción a este campo, dando a conocer sus orígenes y objetivos, así como las diferentes técnicas que existen para su creación. Después se han detallado los objetivos y la metodología utilizada, además de la definición del problema a tratar. A continuación, se ha explicado con detenimiento todos los algoritmos y funciones aplicadas para el cálculo de los ratings. Razonando cada elección. De la misma manera se ha explicado la implementación de la técnica seleccionada, un filtrado colaborativo a partir de vecinos cercanos. Seguidamente se ha tratado el *framework* elegido para la imple-

mentación del sistema, *Spark*, así como sus características principales. Además de comentar que la aplicación se ejecutará en un entorno de contenedores *Docker* creando un clúster, usando como *cluster manager* al modo *standalone* de Spark. Para acabar, se han mostrado y analizado los resultados obtenidos, detallando sus características. Y se ha capturado la estrategia de uso para la visualización en el portal, tanto para la versión de escritorio como para la de móvil.

AGRADECIMIENTOS

A mi família, que sin su apoyo este trabajo no hubiera sido posible. Así como a David y a Jose, que me ayudaron siempre que lo necesitaba. Y a mi tutor Juan Carlos, por guiarme durante todo el proceso.

REFERENCES

[1] B. Schwartz, *The Paradox of Choice, Why more is less*. ECCO, New York, 2004.

[2] a. F. R. T. Mahmood, "Improving recommender systems with adaptive conversational strategies," *Proceedings of the 20th ACM Conference on Hypertext and Hypermedia*, pp. 73–82, 2009.

[3] "Bodas.net about us." `https://www.bodas.net/aboutus/aboutus.php`.

[4] F. R. Lior Rokach Bracha Shapira Paul B. Kantor, *Recommender Systems Handbook.* Springer, 2011.

[5] J. S. D. Frankowski J. Herlocker S. Sen, "Collaborative filtering recommender systems," *The Adaptive Web*, p. 291–324, 2007.

[6] "Apache spark." `https://spark.apache.org/docs/latest/`.

[7] "Apache sparksql." `https://spark.apache.org/docs/latest/sql-programming-guide.html`.

[8] "What is docker?." `https://www.docker.com/what-docker`.

[9] "Dockerhub ubuntu." `https://hub.docker.com/_/ubuntu/`.