Filtrado Colaborativo y Sistemas de Recomendación

Sergio Manuel Galán Nieto
Inteligencia en Redes de Comunicaciones
5º Ingeniería de Telecomunicación
Universidad Calos III de Madrid
sergio.galan@gmail.com

ABSTRACT

Los sistemas de recomendación basados en algoritmos de filtrado colaborativo utilizan las valoraciones de los usuarios sobre ciertos elementos del conjunto total para predecir valoraciones en el resto de los elementos y recomendar los de mayor valoración predicha. Este documento ofrece una descripción de los sistemas de filtrado colaborativo. Intenta cubrir los multiples aspectos que comprenden: Las diversas aproximaciones estadisticas: Basados en modelo, basados en datos. También algunos métodos de evaluación, y los problemas relativos a la implementación. Ademas se presenta información acerca del uso de la librería de filtrado "Taste" que contiene implementaciones de varios algoritmos de filtrado colaborativo.

General Terms

Filtrado, DataMining, IR

Keywords

Filtrado colaborativo, Filtrado de información, Filtros sociales, Mineria de datos, Sistemas de recomendación,Web Social

1. INTRODUCCIÓN

Antes de la llegada de Internet un consumidor de cualquier tipo de producto tenía un acceso limitado a la información relacionada tanto con el producto en si como con otras posibles opciones. La publicidad se convertía asi en prácticamente la única forma de dar a conocer un producto, y el problema del usuario era como conseguir una información veraz. En el caso de productos culturales como la musica, la radio o las revistas especializadas actuaban como unicos difusores de lo nuevo. Ahora la situación se ha invertido totalmente. De la escasez de información se ha pasado a la saturación. De disponer de algunas estanterias con CDs y Videos en el centro comercial mas cercano se ha pasado a tener acceso a una cantidad inagotable de creaciones culturales en tiendas online o en redes P2P. Ahora el problema ha tornado en como separar lo que queremos de lo que no queremos encontrar.

Una aproximación para intentar ofrecer a cada persona lo que busca es mediante el ánalisis del contenido. Para ello se hace una representación del contenido de cada elemento del conjunto y se compara con una representación del contenido que el usuario está buscando. Este filtrado basado en contenido es efectivo principalmente para encontrar documentos textuales en función de unos criterios de búsqueda. Sin embargo es mas dificil parametrizar de forma automática contenidos multimedia.

Aqui es donde estan comenzando a jugar un papel importante los sistemas de recomendación. La idea que subyace tras ellos es encontrar usuarios con gustos similares a los de otro determinado y recomendar a éste cosas que desconoce pero que gustan a aquellos con los que se tiene similitud. Es decir, un sistema de recomendación es un amigo virtual cuyos gustos son una mezcla de los gustos de miembros de la comunidad de usuarios con gustos similares a los de uno mismo. En lugar de intentar aplicar inteligencia artificial para analizar el contenido y relacionarlo entre sí, deja a la comunidad la tarea de evaluar cada elemento y se utilizan algoritmos de minería de datos para encontrar utilidad a las relaciones. Se construye asi una inteligencia colectiva que permite al usuario recibir información adaptada a sus gustos, incluso a veces cosas que no estaba esperando encontrar pero que le resultan útiles.

En este trabajo se intenta proporcionar una panorámica general acerca de los sistemas de recomendación asi como resultados obtenidos del trabajo con algunos de los algoritmos estadísticos estudiados.

2. HISTORIA Y SITUACION ACTUAL

En un comienzo los sistemas de recomendación eran conocidos tan sólo como filtros colaborativos y los primeros trabajos datan de principios de los años 90. El termino fue acuñado en 1992 para un sistema de filtrado de correo electrónico no automatizado. En 1994 se desarrolló el primer "workshop" en Berkeley donde se vio la utilidad en diversas áreas de los primeros algoritmos simples de este tipo. También se identificaron algunas cuestiones importantes para el desarrollo de estos algoritmos: Escalabilidad, Viabilidad económica, puntuaciones implicitas y explicitas... Uno de los grupos de investigación pioneros en el desarrollo del filtrado colaborativo fue el proyecto GroupLens de la universidad de Minesota que aún permanece muy activo y que ha propor-

IRC'07, Enero, 2007, Leganés, Madrid, Spain.

cionado una gran parte de la base algoritmica de muchos sistemas de recomendación. Fueron los primeros en introducir el filtro colaborativo automático usando un algoritmo de busqueda de vecinos para proporcionar predicciones en los grupos de noticias de USENET. De este grupo de investigación partió tambien la iniciativa empresarial NetPerceptions, despejando gran parte de las dudas acerca de la viabilidad económica de estos proyectos. En la actualidad es un campo que se encuentra muy activo y genera un gran número de publicaciones y congresos todos los años. Y es que el filtrado colaborativo es un aspecto de gran importancia dentro de las redes sociales y la pequeña revolución que ha supuesto la llamada "Web 2.0". Algunos ejemplos actuales de uso son:

- Recomendaciones en tiendas on-line. Partiendo de un producto se recomiendan otros productos que han interesado a los usuarios que compraron dicho producto. La web pionera en este tipo de recomendaciones fue Amazon.com
- Filtrado de noticias. Se construye un perfil que almacena las noticias que un usuario consulta.
- Recomendaciones musicales, de libros, de peliculas. En los últimos años han surgido decenas de webs de este tipo entre las que destacan "last.fm" y "MyStrands", esta última de origen español. En estos servicios, cada vez que un usuario escucha una canción se envía su información a la base de datos del sistema, el cual las utiliza para generar nuestras recomendaciones, pero las funcionalidades que ofrecen crecen constantemente. Por ejemplo last.fm ofrece radios personalizadas para cada usuario en funcion de las recomendaciones que reciba y MyStrands organiza fiestas en las que la musica se elije automáticamente de forma colaborativa en funcion de los gustos de los asistentes. El modelo de negocio de estas empresas, es además de la publicidad, el de acuerdos con tiendas on-line para enlazar directamente las recomendaciones con su servicio de venta. También el de proporcionar a las compañías -discográficas en este caso- análisis de tendencias musicales, de nuevos artistas, etc...
- Búsqueda de personas afines en comunidades. En webs como meneame.net se tienen en cuenta las noticias que cada usuario ha votado para generar una lista de vecinos con similares intereses.

3. APROXIMACIÓN TEORICA AL PROBLEMA DE LA RECOMENDACIÓN

El objetivo es sugerir nuevos elementos a un usuario basándose en sus elecciones anteriores y en elecciones de gente con similar historial de ratings o valoraciones. Existen dos formas de recoger estas valoraciones. Una es de forma explicita, es decir el usuario asigna una puntuación a cada elemento que será un valor numérico discreto entre un máximo y un mínimo. La segunda forma es recoger las valoraciones implícitamente, extrayendo la información pertinente de las acciones del usuario. Por ejemplo el tiempo que pasa leyendo una determinada pagina web, los enlaces que sigue, el número de veces que se escucha una canción, esta sería una aproximación mas clasica de minería de datos. Una vez que se tiene suficiente información del usuario se pasa a la fase de predicción y recomendación. Predicción hace referencia a estimar que valoración daría el usuario a cada elemento mientras que recomendación se refiere a extraer los N elementos más recomendables (Top-N recommendation)

Algoritmos de filtrado colaborativo basados en memoria, o algoritmos de vecinos cercanos(Nearest Neighbour).

Utilizan toda la base de datos de elementos y usuarios para generar predicciones. Primeramente emplean tecnicas estadísticas para encontrar a vecinos, es decir usuarios con un historial de valoraciones sobre los elementos similar al usuario actual. Una vez que se ha construido una lista de vecinos se combinan sus preferencias para generar una lista con los N elementos mas recomendables para el usuario actual. Entre sus inconvenientes se encuentra la necesidad de disponer de un número mínimo de usuarios con un número minimo de predicciones cada uno, incluido el usuario para el que se pretende realizar la recomendación.

Algoritmos de filtrado colaborativo basados en Modelo.

Desarrollan primero un modelo de los ratings del usuario. Tratan el problema como un problema de predicción estadistica y calculan el valor esperado para cada item en función de los ratings anteriores. Para ello se utilizan distintos algortimos de aprendizaje Clustering o redes neuronales como las Redes de Funciones de Base Radial (RBFN). Por ejemplo utilizando clustering se trata de clasificar a un usuario en particular dentro de una clase de usuarios y a partir de ahí se estiman las probabilidades condicionadas de esa clase hacia los elementos a evaluar.
En general, ante las consultas responden más rapido que los basados en memoria, pero por contra necesitan de un proceso de aprendizaje intensivo.

3.1 Otros aspectos

El consumo de memoria y de CPU de cualquier sistema de filtrado e información es muy elevado al tratar con muchos datos. La optimización de los algoritmos para mejorar su rendimiento es uno de los principales campos de investigación dentro del filtrado colaborativo. Algunas caracterictas deseables en estos sistemas suponen una modificación constante de los datos, lo que hace necesarios algoritmos que tengan una coste de actualización bajo. Por ejemplo los nuevos elementos han de aparecer en el sistema lo antes posible. Tambien se requiere una mejora continua del perfil del usuario. Es decir que el usuario perciba que el "esfuerzo" de evaluar nuevos elementos se vea compensado con unas mejoras en las recomendaciones que se obtienen.

Además de los aspectos técnicos la implantación de sistemas de recomendación puede plantear problemas sociales como la privacidad de los usuarios. Es un asunto delicado ya que para tener un buen funcionamiento es necesario conocer la máxima información posbible del usuario. Una primera solución es mantener tan sólo la información relacionada con las votaciones y conocer del usuario sólo un seudónimo, pero esto choca con el modelo de negocio basado en publicidad que se utiliza abundantemente en la actualidad.

Por ultimo mencionar también que existen investigaciones dirigidas a comprender la relacion entre los usuarios y los sistemas de recomendacion. En [7] se estudia como influencian al usuario las recomendaciones a la hora de dar su propia opinion. Psicológicamente es conocido desde hace

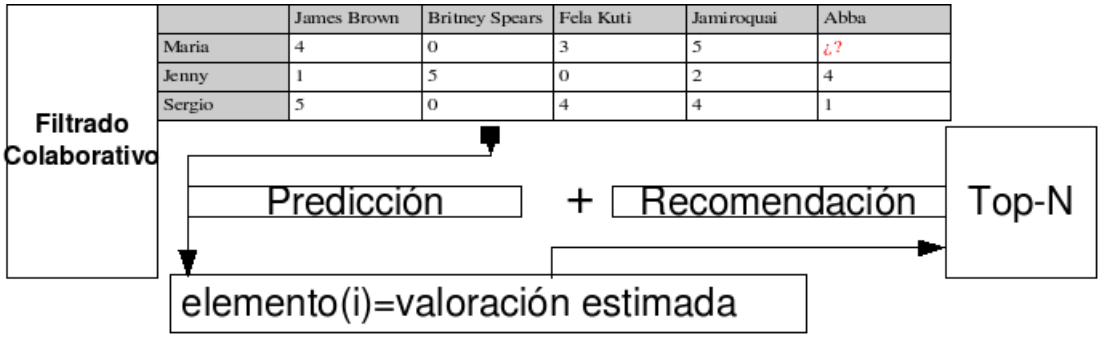

Figure 1: Modelo de un sistema de filtrado colaborativo

tiempo como la opinion de los demas modifica la opinión propia. Ahora tambien se percibe como el mostrar predicciones del voto de un usuario influencia el propio voto futuro.

4. ALGORITMOS

4.1 Algoritmos basados en vecinos cercanos.

Fueron los primeros algoritmos de filtrado colaborativo en implementarse. En primer lugar es necesario medir los parecidos de todos los usuarios con el usuario actual. Pueden utilizarse distintas medidas: [1]

Coeficiente de Correlación de Pearson. Se deriva de las formulas d eregresión lieal, y asume que la relación entre elementos es lineal, los errores indpendientes y la distribución tiene varianza constante y media 0. Estas suposiciones normalmente no se producen realmente con lo que hay que valorar como afectan a la bondad de los resultados, pero en un gran número de casos el el rendimiento utilizando Pearson es apropiado. El peso que se asigna al usuario u para predecir al usuario activo a viene dado por: ($r_{a,i}$es la votacion del usuario a al elemento i)

$$w_{a,u} = \frac{\sum_{i=1}^{m}(r_{a,i} - \bar{r}_a) * (r_{u,i} - \bar{r}_u)}{\sigma_a \sigma_u}$$

Otras formas son las basadas en vectores (medida del coseno), las medidas de correlación basadas en entropia, correlación de *Ringo* o correlación de *Spearman.*

Una vez se tienen los pesos de correlación de cada algorimo hay que saber como de confiables son estos pesos. Es posible tener un alto grado de correlación con vecinos con los que se comparten pocos elementos valorados por el usuario actual, pero con igual valoración. El uso de estos pesos proporciona una estimaciones malas puesto que para tener una idea real de la correlación cuantas son necesarios cuantos mas votos compartidos mejor. En los casos de pocas muestras es recomendable disminuir el factor de correlación en función del número de votos compartidos.

Para intentar mejorar más los pesos de la correlación entre usuarios se puede entrar a trabajar con la varianza de los elementos que cada usuario ha votado. Si un elemento es votado positivamente por un gran porcentaje de la población, el que dos usuarios compartan esa votación dice poca información acerca de la correlación entre usuarios. Lo contrario pasa en el caso de elementos que sean votados positiva o negativamente por pocos usuarios. Para tomar en cuenta este hecho se añade a la formula de la correlación de Pearson un termino con la varianza del elemento.

Selección de Vecinos

En sistemas con pocos usuarios podría ser factible trabajar con todos los usuarios como vecinos tan sólo multiplicando cada uno por el peso de su correlación. Sin embargo en los sistemas actuales que poseen miles de usuarios esta vía no es posible, por ello hay que tomar un subconjunto de los usuarios, lo que no sólo mejora la eficiencia sino también la efectividad. Para elegir el número de vecinos se puede:

1. Establecer un umbral de correlación y tomar todos los que superen dicho umbral. El problema es que puede haber usuarios que no tengan muchos vecinos con correlación alta. Por tanto el número de vecinos sería bajo y esto provocará que el número de elementos sobre el que la vecindad de un usuario puede opinar es también bajo.

2. Tomar “n” vecinos siempre, teniendo en cuenta que un número demasiado alto puede diluir la influencia de los vecinos con mas peso y un número demasiado bajo provoca los mismos problemas que el método anterior

Recomendación

La forma básica consiste en tomar las notas dadas por cada vecino y proporcionarlas a su correlación con el usuario actual:

$$p_{a,i} = r_a + \frac{\sum_{u=1}^{n}(r_{u,i} - \bar{r}_u) * w_{a,u}}{\sum_{u=1}^{n} w_{a,u}}$$

Una posible mejora es la de introducir una variable que dependa de lo optimista o pesismista que sea cada usuario. Unos usuarios votan por norma mas alto que otros. Para proporcionarlos basta con normalizar respecto a la media de puntuación de cada usuario. De esta forma el predictor quedaría como:

$$p_{a,i} = r_a + \sigma_a \frac{\sum_{u=1}^{n} \frac{(r_{u,i} - \bar{r}_u)}{\sigma_u} * w_{a,u}}{\sum_{u=1}^{n} w_{a,u}}$$

4.2 Algoritmos basados en elementos.

En lugar de buscar similitudes entre usuarios buscan cercanías entre elementos.[3] [4] El procedimiento consiste en seleccionar los los elementos que un usuario determinado ha votado y despues comprobar como de similar es cada uno del

resto de los elementos del sistema, para terminar recomendando los más parecidos. Existen distintas formas de evaluar la similitud entre elementos pero el procedimiento genérico consiste en tomar dos elementos x1, x2 y despues calcular su similitud a partir de todos los usuarios que han votado ambos elementos. En teoría es la misma aproximación que la que se tenia con algoritmos basados en vecinos cercanos. La ventaja es que en el caso de los elementos la similitud entre ellos es menos variable que la similitud entre usuarios, lo que permite pre-computar estas similitudes y hace el proceso mucho mas rápido.

Similitudes Basadas en Coseno.

Se considera cada elemento como un vector dentro de un espacio vectorial de m dimensiones y se calcula la similitud como el coseno del ángulo que forman. Es decir si tendemos dos vectores x1, x2 consistentes en un array cuyos elementos son las votaciones recibidas de cada usuario. Su similitud será pues:

$$cos(x1, x2) = \frac{\vec{x1} \cdot \vec{x2}}{\|\vec{x2}\| \Delta \|\vec{x1}\|}$$

Similitudes Basadas en Correlación.

Considerando el conjunto de votaciones de los usuarios con votos sobre el elemento x1 y x2, se utiliza el coeficiente de correlación de Pearson. Siendo U ese conjunto, u cada usuario R la valoración sobre un elemento y $\bar{R}_x$ la valoración media de ese elemento

$$sim(x1, x2) = \frac{\sum_{u \in U} \left(R_{u,x1} - \bar{R}_{x1}\right)\left(R_{u,x2} - \bar{R}_{x2}\right)}{\sqrt{\sum_{u \in U} \left(R_{u,x1} - \bar{R}_{x1}\right)^2} \sqrt{\sum_{u \in U} \left(R_{u,x2} - \bar{R}_{x2}\right)^2}}$$

Similitudes Basadas en Coseno Ajustado.

En las similitudes basadas en coseno no se tienen en cuenta las diferencias entre las escalas de los usuarios. Para eso se incluye en la fórmula el parámetro $\bar{R}_u$que indica la valoración media de ese usuario. Tambien se denomina transformación z:

$$sim(x1, x2) = \frac{\sum_{u \in U} \left(R_{u,x1} - \bar{R}_{u}\right)\left(R_{u,x2} - \bar{R}_{u}\right)}{\sqrt{\sum_{u \in U} \left(R_{u,x1} - \bar{R}_{u}\right)^2} \sqrt{\sum_{u \in U} \left(R_{u,x2} - \bar{R}_{u}\right)^2}}$$

Cálculo de la predicción

Una vez que se elige un método para computar la relación entre elementos quedan por calcular los elementos que son más parecidos a los del usuario. De nuevo existen varias aproximaciones.

Suma con pesos (Weighted Sum).

Se toman todos los elementos que el usuario ha votado. Se toma un elemento "x1" y para ese elemento se suman todos los coeficientes de similitud entre ese elemento y los elementos votados por el usuario, proporcionados al valor del voto. Siendo N cada elemento votado por el usuario $S_{i,N}$La similitud entre los elementos i y N y $R_{u,N}$la valoración del usuario del elemento N:

$$Pu, i = \frac{\sum_N \left(S_{i,N} * R_{u,N}\right)}{\sum_N \left(|S_{i,N}|\right)}$$

Regresión.

Similar al modelo anterior, pero en lugar de sumar directamente las notas de los elementos similares se utiliza una aproximación basada en la recta de regresión. Con este meétodo se intenta compensar un problema que se da al evaluar las similitudes mediante medidas del coseno o la correlación, y es que vectores con alta similitud pueden encontrarse distantes en sentido euclídeo. Se utiliza la misma fórmula que en el caso de la suma proporcionada pero sustituyendo $R_{u,N}$por:$\bar{R}'_N = \alpha \bar{R} i + \beta + \epsilon$

4.3 Predictores "Slope-one"

A la hora del cálculo de la predicción para un usuario U se tiene en cuenta tanto la información de usuarios que tienen en comun la votación de algún elemento U como la información del resto de los elementos votados.[8]

Partiendo de dos Arrays v_iy w_i de longitud n se busca obtener la mejor predicción de w a partir de v. Tendrá la forma $f(x) = x + b$ y deberá minimizar el error cuadrático medio $\sum_i (v_i + b - w_i)^2$de donde derivando se obtiene que

$$b = \frac{\sum w_i - v_i}{n}$$

Es decir, la diferencia media entre ambos arrays. Partiendo de esto podemos diseñar las predicciones. A partir de un conjunto X de datos de entrenamiento se toman dos elementos cualquiera i,j con votaciones u_i u_jy se calcula la desviación media entre ambos (Solo se consideran usuarios que hayan votado tanto i como j:

$$desv_{j,i} = \sum \frac{u_j - u_i}{n}$$

Con lo que se tiene un array simétrico precalculado que es posible actualizar con cada nuevo elemento que se añada al sistema.

La predicción para un usuario u_j a partir del resto de usuarios será (Con R_j el conjunto de items relevantes):

$$P(u)_j = \frac{1}{total(R_j)} \sum_{i \in R_j} (desv_{j,i} + u_i)$$

4.3.1 Predictor "Slope-one" con pesos.

El predictor anterior no tiene en cuenta el número de notas que se han tomado. No es igual predecir la nota de un usuario sobre un item L a partir de los ratings de ese usuario de otros elementos J K si hay muchos mas usuarios que tienen el par de votos J-L que el par K-L. El elemento J es mucho mejor predictor que el elemento K en este caso. Analíticamente esta idea se introduce en la ecuación con el factor $c_{j,i}$que es el número de evaluaciones de los items j,i

$$P(u)_j = \frac{\sum_{i \in S(u) - \{j\}} (desv_{j,i} + u_i) c_{j,i}}{\sum_{i \in S(u) - \{j\}} c_{j,i}}$$

4.3.2 Predictor "Slope-one" Bi-Polar.

Trabaja dividiendo la predicción en dos partes. Usa el algoritmo anterior una vez para obtener una predicción de los elementos que han gustado al usuario y de los que no han gustado.

El primer problema que plantea este sistema es establecer el umbral a partir del que se considera que un elemento gusta o disgusta. La idea intuitiva es establecer un umbral que sea

la mitad de la escala de evaluación. Si la escala va de 1 a 10 los elementos por debajo de 5 se considerarían como evaluados negativamente y los otros son evaluados positivamente. Esta aproximación sería la adecuada si las evaluaciones de los usuarios fueran distribuidas uniformemente, Sin embargo el comportamiento real de los usuarios indica que existe un porcentaje elevado de votaciones superiores a la mitad de la escala. Por ello el valor del umbral se establece como la media de todas las notas dadas por el usuario.

En la práctica este procedimiento supone doblar el número de usuarios, pero a la vez también reduce el número de elementos en el cálculo de las predicciones.

5. MÉTODOS DE EVALUACIÓN DE RENDIMIENTO.

Evaluar el rendimiento de los algoritmos de recomendación no es trivial. Primero porque diferentes algoritmos pueden ser mejores o peores dependiendo del dataset elegido. La mayoría de estos algoritmos están diseñados para el dataset de películas de GroupLens donde existe un número de elementos mucho menor al número de usuarios y votos. En una situación inversa el comportamiento puede diferir totalmente.

También los objetivos del sistema de recomendación pueden diversos. Un sistema puede diseñarse para estimar con exactitud la nota que daria un usuario a un elemento, mientras otro puede tener como principal objetivo el no proporcionar recomendaciones erróneas. Es decir puede haber múltiples tipos de medidas: Que las recomendaciones cubran todo el espectro de elementos del conjunto(cobertura), que no se repitan, que sean explicables... Sin embargo el principal objetivo de un sistema de recomendación no es directamente cuantificable: la satisfacción del usuario. En muchos casos conseguir un error cuadrático menor al elegir un algoritmo u otro no es apreciado por el usuario. Sin embargo hay muchos otros parámetros que pueden influir en esa satisfacción: La sensación de credibilidad que ofrezca el sistema, la interfaz de usuario, la mejora del perfil al incluir nuevos votos...

En cualquier caso las medidas de precisión pueden dar una primera idea de como de bueno es el algoritmo nuclear del sistema de recomendación. Existen dos tipos de métodos de evaluación:

- *Métodos estadisticos.* El parámetro de evaluación mas utilizado es el error medio absoluto(Mean Absolute Error, MAE). Mide la desviación de las recomendaciones predichas y los valores reales. A menor MAE mejor predice el sistema las evaluaciones de los usuarios.

$$MAE = \frac{\sum_{i=1}^{N} p_i - q_i}{N}$$

  El MAE sin embargo puede dar una idea distorsionada del algoritmo para el caso de sistemas que tienen como objetivo encontrar una lista de buenos elementos recomendables. El usuario tan sólo está interesado en los N primeros elementos de la lista. El error que se cometa al estimar el resto le es indiferente. Tampoco es recomendable en sistemas en los que la salida deba de ser una decisión binaria de si/no. Por ejemplo, con una escala de 1 a 10 si el umbral está siguado en 5, utilizando MAE se obtendría un mayor error al errar de 9 a 5 que al errar de 5 a 4, lo cual no es cierto a lahora de medir el error de salida.
  Sin embargo es un tipo de error estadisticamente muy estudiado y sencillo de comprender. Posee muchas variaciones, como el *error cuadrático medio* que persigue penalizar los mayores errores o el *error absoluto normalizado* que facilita la tearea de establecer comparaciones entre pruebas con diferentes datasets.

- Metricas de decisión. Evaluan cómo de efectivo es un sistema de predicción ayudando al usuario a seleccionar los elementos mayor calidad, es decir con que frecuencia el sistema de recomendación efectua recomendaciones correctas. Para ello asumen que el proceso de predicción es binario: o el elemento recomendado agrada al usuario o no lo agrada. Sin embargo en la práctica se plantea el problema de evaluar esto. Una posible solución es la de dividir el conjunto de datos en dos conjuntos, entrenamiento y test. Se trabaja con con el conjunto de entrenamiento y posteriormente se evalúa el resultado comparando las recomendaciones proporcionadas con las del conjunto de test. Aun siendo a veces útil esta técnica, hay que tener en cuenta que los resultados dependen fuertemente del porcentaje de elementos relevantes que el usuario haya votado. La mas conocida de estas métricas es la de “Precision and Recall” y es utilizada en muchos tipos de sistemas de recuperación de información. Precisión es la probabilidad de que un elemento seleccionado sea relevante y *Recall* es la probabilidad de que sea seleccionado un elemento relevante, aunque en los sistemas de recomendación la“relevancia” es algo totalmente subjetivo. De cara al usuario esta métrica es mas intuitiva, puesto que establecer que un sistema tiene una precisión del 90 % significa que de cada 10 elementos recomendados 9 serán buenas recomendaciones, algo que no queda claro proporcionando valores de error cuadrático medio.
  ROC Reciver operating characteristic) es otra medida muy utilizada. Proporciona una idea de la potencia de diagnóstico de un sistema de filtrado. Las curvas ROC dibujan la especifidad(Probabilidad de que un elemento malo del conjunto sea rechazado por el filtro) y la sensitividad (probabilidad de que un elemento bueno al azar sea aceptado). Si un elemento es bueno o malo viene dado por las valoraciones de los usuarios. Las curvas se dibujan variando el umbral de predicción a partir del cual se acepta un elemento. El área bajo la curva se va incrementando si cuando el filtro es capaz de retener mas elementos buenos y menos malos.

6. EXPERIENCIA PRÁCTICA.

Para probar algunos de los algoritmos comentados anteriormente se va a hacer uso de la librería de de filtros colaborativos “Taste”. En cuanto al dataset a utilizar se trata del dataset de peliculas del grupo GroupLens.

6.1 Dataset.

El dataset de peliculas de groupLens “MovieLens” es el dataset de referencia, utilizado en la mayoría de las publicaciones sobre filtros colaborativos junto con el dataset de *Compaq Research*, que hoy dia ya no se encuentra disponible. Su estructura es la que aparece en la figura 2 Un ejemplo

de la tabla de usuarios sería:
1::F::1::10::48067 El campo de sexo puede tener los valores F/M La edad se agrupa por rangos:
* 1: "Under 18" * 18: "18-24" * 25: "25-34" * 35: "35-44" * 45: "45-49" * 50: "50-55" * 56: "56+" Y la ocupación se expresa mediante un código

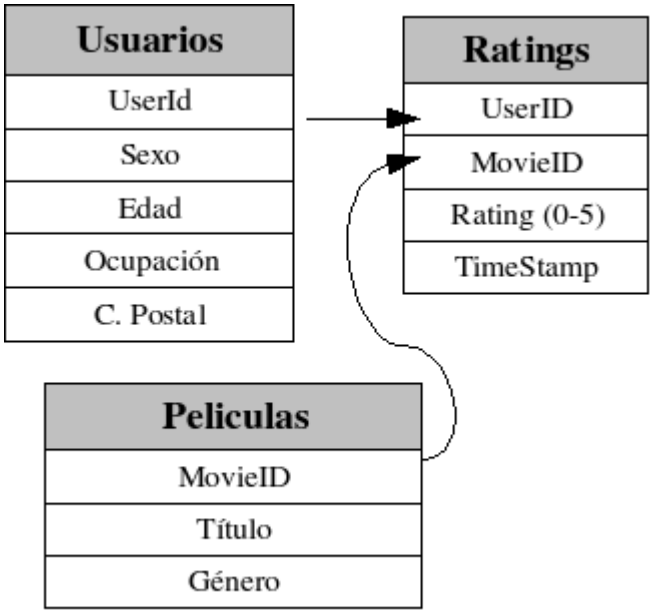

Figure 2: Esquema de campos del dataset de peliculas de GroupLens

Para la tabla de peliculas: *3949::Requiem for a Dream (2000)::Drama*
Y por ultimo la tabla de puntuaciones: *1227::3786::3::1009223209.* La marca de tiempo tiene el formato indicado en la función time() de unix, el número de segundos desde el 1 de enero de 1970.

El dataset original contiene un millón de evaluaciones de 6040 usuarios pero debido a que los requisitos de memoria para trabajar con todo el dataset superan al hardware disponible para las pruebas, se ha reducido el número a unas 200000 evaluaciones de 1228 usuarios. Los datos fueron recopilados durante el año 2000 asi que sólo contienen peliculas hasta esa fecha. El usuario 1228 contiene contiene notas propias para comprobar de forma subjetiva los resultados.

6.2 Taste

Taste es un motor de filtros colaborativos escrito en JAVA. Puede Integrarse en las aplicaciones de diversas maneras. Si la aplicación propia está programada en java, pueden incluirse las clases de TASTE de manera normal. En el caso de programar aplicaciones J2EE se puede utilizar un bean sin estado para utilizarlo en nuestra aplicacion. También soporta una interface para *WebServices SOAP* y puede funcionar como servidor externo, respondiendo a peticiones HTTP.

Esta ultima será la forma en la que se utilice aqui. Entre los ficheros de TASTE viene incluido un codigo de ejemplo ya preparado para funcionar con el dataset de MOVIELENS. Se ha trabajando a partir de ese codigo modificando lo necesario para cambiar el tipo de recomendador.

6.2.1 Trabajo con Taste

Para trabajar con cualquiera de los algoritmos que implementa Taste es necesario disponer de una gran cantidad de memoria RAM, se recomienda reservar 1GB para la aplicación. Sin embargo debido al hardware disponible se tuvo que reducir el tamaño del dataset dado que sólo podian reservarse alrededor de 500 MB para el uso de Taste. El codigo viene preparado para ejecutarse en un servidor de servlet como ApacheTomcat. Primeramente fue necesario copiar los Datasets y despues generar un archivo war contiendo todo, lo que fue hecho con: ANT BUILD-GROUPLENS-EXAMPLE La creación de un sistema de recomendación basado en Taste comprende varias capas, como se muestra en la figura 3.

Modelo de datos. El modelo de datos se implementa a traves de la interfaz "DataModel" que viene con varias implementaciones preparadas para obtener datos desde distintas fuentes. MYSQLJDBCDATAMODEL se usa para bases de datos MySql. FILEDATAMODEL es la usada para leer datos provenientes ficheros CSV (comma sepparated Values). Para leer el dataset de MovieLens se necesita expandir esta clase y tratar el fichero previamente, lo que se hace desde el fichero "GroupLensDataModel.java"

Transformaciones

Taste implementa varias transformacionesn que pueden aplicarse a los valores proporcionados por un usuario para tratar de mejorar la calidad de las recomendaciones. Para realizar esto es necesario utilizar el metodo
dataModel.addTransform(PreferenceTransform);
Las transformaciones implementadas son:

ZScore Normaliza los datos del usuario ajustando la media y la varianza para evitar el problema ya comentado de que unos usuarios puntuan mas alto que otros.

InverseUserFrequency Potencia los elementos raros. Es decir considera que un elemento que han votado pocas personas es mas importante que otro que es compartido entre mas gente.

CaseAmplification Amplifica los valores superiores y disminuye los inferiores.

Medidas de similitud

En el caso de trabajar con similitudes entre usuarios pueden usarse las medidas basadas en coseno, en o en las formulas de Pearson o Spearman. La implementación se realiza de la siguiente forma:
UserCorrelation userCorrelation = new CosineMeasureCorrelation(DataModel dataModel);
Pero en lugar de CosineMeasureCorrelation pueden utilizarse *PearsonCorrelation* o *SpearmanCorrelation*
Si por contra el sistema trabaja con similitudes entre elementos las medidas de similitud son solo las de Pearson y la del coseno. Sin embargo lo mas rápido es pasar una lista de relaciones precompiladas:
ItemCorrelation itemCorrelation = new PearsonCorrelation(DataModel dataModel);

Búsqueda de vecinos

Hay que implementar la interfaz *UserNeighborhood.* Como se vio en el apartado 4.1 se puede optar por pedir una lista de

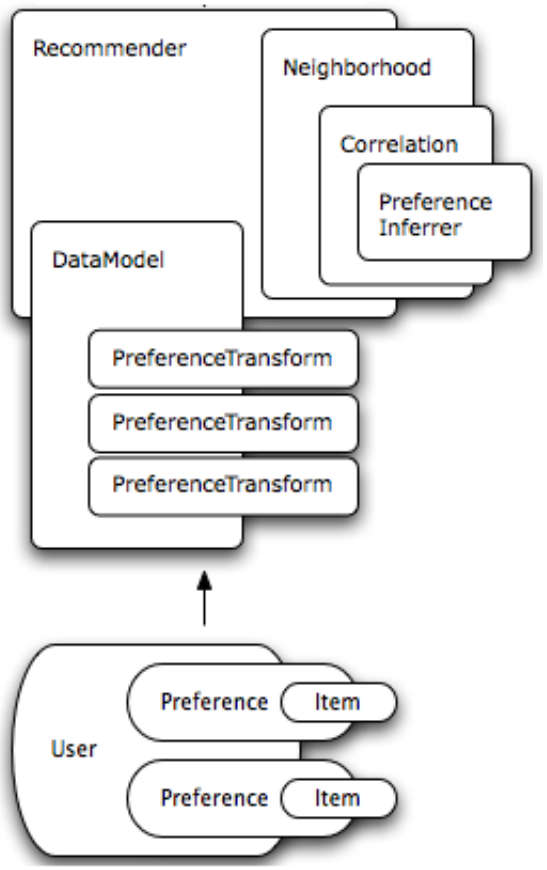

Figure 3: Estructura de una aplicación con Taste. Fuente:http://taste.sourceforge.net/

vecinos de tamaño fijo o una lista que supere un determinado umbral. Para ello *Taste* implementa:

1. NearestNUserNeighborhood Devuelve listas de vecinos de tamaño fijo indicado en el parámetro n del constructor: *UserNeighborhood neighborhood = new NearestNUserNeighborhood(n, userCorrelation, model);*

2. ThresholdUserNeighborhood. Devuelve listas de vecinos que superen el umbral de correlación establecido en el parametro umbral: *UserNeighborhood neighborhood = new NearestNUserNeighborhood(umbral, userCorrelation, model);*

Generar Recomendaciones

Taste incorpora un gran número de recomendadores. Ademas de por Items y por Usuarios podemos encontrar:

TreeClusteringRecommender: Primeramente Agrupa a los usuarios y despues genera una lista de recomendaciones teniendo en cuenta el grupo al que pertenece el usuario.

SlopeOneRecommender. Es del tipo explicado en la sección 4.3, Se implementa tanto el modelo con pesos como sin pesos. Para su implementación no es necesario el procedimiento de busqueda de vecinos y de cálculo de similitudes. Basta con utilizar las clase recommender con el modelo de datos utilizado.
Recommender recommender = new SlopeOneRecommender(model);
Recommender cachingRecommender = new CachingRecommender(recommender);

Evaluar resultados.

Se incorpora la interfaz *RecommenderEvaluator* para facilitar la tarea de realizar las evaluaciones. Con taste vienen dos metodos ya programados:

RMSRecommenderEvaluator Calcula el parámetro RMS (root mean Squared) Para ello evalua la diferencia entre lo predicho y lo real. El rms es la raiz cuadrada de la media de la diferencia al cuadrado.

AverageAbsoluteDifferenceRecommenderEvaluator Calcula la diferencia media absoluta entre lo predicho y lo real.

Para estas evaluaciones se toma un porcentaje de datos de prueba y otro porcentaje de datos de Test, parámetros estos que se indican en el metodo *"evaluate"*:
evaluate(recommenderBuilder, dataModel, double ***trainingPercentage****, double* ***evaluationPercentage****)*
Para ir comprobando el error absoluto sobre el dataset de movielens se ha hecho uso de la clase *GroupLensRecommenderEvaluatorRunner* dentro del fichero de ejemplo. Para su uso el comando es: ANT EVAL-GROUPLENS-EXAMPLE

Algunos resultados obtenidos

Tras introducir las puntuaciones de peliculas propias en el dataset se modificó el codigo fuente para ir comprobando los algoritmos. La interfaz de control es mediante un navegador web pasando parámetros por la url:
http://127.0.0.1:8080/taste/RecommenderServlet ?userID=1228&howMany=20&debug=true
El top20 de las peliculas para el usuario es:
5.0 1230 Annie Hall (1977) Comedy|Romance
5.0 1244 Manhattan (1979) Comedy|Drama|Romance
5.0 2571 Matrix The (1999) Action|Sci-Fi|Thriller
5.0 260 Star Wars: Episode IV - A New Hope (1977) Action
5.0 2959 Fight Club (1999) Drama
5.0 296 Pulp Fiction (1994) Crime|Drama
5.0 3481 High Fidelity (2000) Comedy
5.0 541 Blade Runner (1982) Film-Noir|Sci-Fi
5.0 741 Ghost in the Shell (Kokaku kidotai) (1995) Animation
5.0 858 Godfather The (1972) Action|Crime|Drama
4.0 1089 Reservoir Dogs (1992) Crime|Thriller
4.0 111 Taxi Driver (1976) Drama|Thriller
4.0 1221 Godfather: Part II The (1974) Action|Crime
4.0 1483 Crash (1996) Drama|Thriller
4.0 2076 Blue Velvet (1986) Drama|Mystery
4.0 2664 Invasion of the Body Snatchers (1956) Horror|Sci-Fi
4.0 3186 Girl Interrupted (1999) Drama
4.0 3556 Virgin Suicides The (1999) Comedy|Drama
4.0 3569 Idiots The (Idioterne) (1998) Comedy|Drama
3.0 1078 Bananas (1971) Comedy|War
Al aplicar la normalización (transformacion z dataModel.addTransform(new ZScore());) se produce un cambio en la nota de las peliculas:

1.1061487 1230 Annie Hall (1977) Comedy|Romance
1.1061487 1244 Manhattan (1979) Comedy|Drama|Romance
0.31955406 3556 Virgin Suicides The (1999) Comedy|Drama
0.31955406 3569 Idiots The (Idioterne) (1998) Comedy|Drama

Trabajando con el algoritmo *slopeone* se obtiene un error medio de 0.814 sin utilizar ningún tipo de transformación y sin embargo aumenta hasta 3.614 utilizando la transformación z y hasta 1.62 utilizando el refuerzo de peliculas raras, algo que no coincide con lo esperado. Subjetivamente no puedo dar una impresión completa puesto que desconozco muchas de las peliculas recomendadas. Sin embargo, algunas de las conocidas no se adaptan a lo esperado mientras que otras sí.

Este tipo de errores es probable que sean debidos a estar trabajando con un dataset reducido.

Al utilizar el algoritmos de vecinos mas cercanos las recomendaciones si parecen ser mas adaptadas al gusto del usuario de prueba.

Recommendations:

5.0 1408 Last of the Mohicans The (1992) Action|Romance|War
5.0 3363 American Graffiti (1973) Comedy|Drama
5.0 593 Silence of the Lambs The (1991) Drama|Thriller
5.0 1374 Star Trek: The Wrath of Khan (1982) dventure|Sci-Fi
5.0 50 Usual Suspects The (1995) Crime|Thriller
5.0 1233 Boat The (Das Boot) (1981) Action|Drama|War
5.0 3114 Toy Story 2 (1999) Animation|Children's|Comedy
5.0 2599 Election (1999) Comedy
5.0 2355 Bug's Life A (1998) Animation|Children's|Comedy
5.0 1104 Streetcar Named Desire A (1951) Drama
5.0 1198 Raiders of the Lost Ark (1981) Action|Adventure
5.0 1291 Indiana Jones and the Last Crusade (1989) Adventure
5.0 2580 Go (1999) Crime
5.0 356 Forrest Gump (1994) Comedy|Romance|War
5.0 1278 Young Frankenstein (1974) Comedy|Horror

Parece ser que por algún tipo de problema en el software no han podido obtenerse resultados cuantitativos de los algoritmos basados en Items y en vecinos cercanos.

Conclusiones

Los sistemas de recomendación son ya ampliamente utilizados en Internet y debido a la gran cantidad de información de todo tipo que nos rodea, su presencia y utilidad es de esperar que aumente en el futuro.
Los algoritmos en los que se sustentan están bastante probados y matemáticamente pueden conseguir buenos resultados. No obstante aún hay que pulir muchos desperfectos. De cara al usuario no importa conseguir el mínimo error cuadrático prediciendo la evaluación de una película, sino que se busca que los elementos recomendados sean satisfactorios. Por esto el procedimiento de evaluación del comportamiento de un sistema de filtros colaborativos no es sencillo, no puede limitarse a datos estadísticos.

Para obtener un mejor nivel de calidad es necesario introducir más elementos externos ademas de los cálculos de correlación entre usuarios. Una vez se obtiene una lista de elementos posibles para su recomendación habría que introducir nuevos parámetros adaptado al usuario. En el caso de las películas por ejemplo, criterios como el país de procedencia, la fecha de edición o la temática pueden ser muy útiles de incorporar a la hora de generar la última lista al usuario.

En cualquier caso aunque hay camino por andar, los filtros colaborativos tienen actualmente una enorme funcionalidad y el número de usuarios que hacen uso de ellos no deja de crecer día a día.

7. REFERENCES

[1] Jonathan Herlocker, Joseph A. Konstan Al Borchers John Riedl.
An Algorithmic framework for performing collaborative Filtering.

[2] Sean Owen
Taste: Collaborative Filtering for Java.
http://sourceforge.net/projects/taste/

[3] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl.
Item-Based Collaborative Filtering Recommendation Algorithms WWW10, ACM 1-58113-348-0/01/0005. MAY 2005

[4] Mukund Deshpande Georoge Karypis
Item-Based Top-N Recommendation Algorithms

[5] Jonathan Herlocker, Joseph A. Konstan John Riedl
Evaluating Collaborative Filtering Recommender Systems

[6] Ian Soboroff, Charles Nicholas, and Michael Pazzani.
Workshop on Recommender Systems: Algorithms and Evaluation

[7] Dan Cosley, Shyong K. Lam, Istvan Albert, Joseph A. Konstan, John Riedl.
Is Seeing Believing? How Recommender Interfaces Affect Users Opinions. ACM 1-58113-630-7/03/0004, 2003

[8] Daniel Lemire, Anna Maclachlan
Slope One Predictors for Online Rating-Based Collaborative Filtering, SDM05. February 7, 2005