Contents

Capítulos **Página**

Sistemas de Recomendación

Uayeb Caballero Rodríguez

Universidad Nacional Autónoma de Honduras, Ciudad Universitaria, Honduras

July 17, 2017

A recommender system is an Information Retrieval technology that improves access and proactively recommends relevant items to users by considering the users' explicitly mentioned preferences and objective behaviors. A recommender system is one of the major techniques that handle information overload problem of Information Retrieval by suggesting users with appropriate and relevant items. Today, several recommender systems have been developed for different domains however, these are not precise enough to fulfil the information needs of users. Therefore, it is necessary to build high quality recommender systems. In designing such recommenders, designers face several issues and challenges that need proper attention. This paper investigates and reports the current trends, issues, challenges, and research opportunities in developing high-quality recommender systems. If properly followed, these issues and challenges will introduce new research avenues and the goal towards fine-tuned and high-quality recommender systems can be achieved.

1 Introduccion

Los sistemas de recomendaciones son herramientas que generan recomendaciones sobre un determinado objeto de estudio, a partir de las preferencias y opiniones dadas por los usuarios. El uso de estos sistemas se está poniendo cada vez más de moda en Internet debido a que son muy útiles para evaluar y filtrar la gran cantidad de información disponible en la Web con objeto de asistir a los usuarios en sus procesos de búsqueda y recuperación de información. En este trabajo realizaremos una revisión de las características y aspectos fundamentales relacionados con el diseño, implementación y estructura de los sistemas de recomendaciones analizando distintas propuestas que han ido apareciendo en la literatura al respecto.

2 Collaborative Filtering

El Filtrado colaborativo (FC) es una técnica utilizada por algunos sistemas recomendadores. En general, el filtrado colaborativo es el proceso de filtrado de información o modelos, que

usa técnicas que implican la colaboración entre múltiples agentes, fuentes de datos, etc. [2] Las aplicaciones del filtrado colaborativo suelen incluir conjuntos de datos muy grandes. Los métodos de filtrado colaborativo se han aplicado a muchos tipos de datos, incluyendo la detección y control de datos (como en la exploración mineral, sensores ambientales en áreas grandes o sensores múltiples, datos financieros) tales como instituciones de servicios financieros que integran diversas fuentes financieras, o en formato de comercio electrónico y aplicaciones web 2.0 donde el foco está en los datos del usuario, etc. Esta discusión se centra en el filtrado colaborativo para datos de usuario, aunque algunos de los métodos y enfoques pueden aplicarse a otras aplicaciones.

En el enfoque más reciente, el filtrado colaborativo es un método para hacer predicciones automáticas (filtrado) sobre los intereses de un usuario mediante la recopilación de las preferencias o gustos de información de muchos usuarios (colaborador). El Filtrado colaborativo se basa, en que si una persona A tiene la misma opinión que una persona B sobre un tema, A es más probable que tenga la misma opinión que B en otro tema diferente que la opinión que tendría una persona elegida azar. Por ejemplo, un sistema de recomendación basado en el filtrado colaborativo para televisión podría hacer predicciones acerca de los programas que le gustarían a un usuario a partir de una lista parcial de los gustos de ese usuario (gustos o disgustos). Notese que estas predicciones son específicas para el usuario, pero utilizan la información obtenida de muchos usuarios. Esto difiere del enfoque más simple de otorgarle una puntuación promedio (poco específico) para cada elemento de interés, por ejemplo sobre la base de su número de votos.

2.1 Índice Jaccard

El índice de Jaccard (IJ) o coeficiente de Jaccard (IJ) mide el grado de similitud entre dos conjuntos, sea cual sea el tipo de elementos.
La formulación es la siguiente:
$J(A,B) = |A \cap B| / |A \cup B|$

Es decir, la cardinalidad de la intersección de ambos conjuntos dividida por la cardinalidad de su unión. Siempre toma valores entre 0 y 1, correspondiente este último a la igualdad total entre ambos conjuntos. En ecología se usa para medir la similitud, disimilitud o distancias que existen entre dos estaciones de muestreo, con una formulación equivalente [1]

Ahora miremos como aplicar esta formula con R, En primer lugar declararemos un dataframe de peliculas con sus rankings

```
a <- data.frame(user = "A", HP1 = 4, HP2 = NA, HP3 = NA,
                TW=1, SW1 = 1, SW2= NA, SW2 = NA)
b <- data.frame(user = "B", HP1 = 5, HP2 = 5, HP3 = 4,
                TW=NA, SW1 = NA, SW2= NA, SW2 = NA)
c <- data.frame(user = "C", HP1 = NA, HP2 = NA, HP3 = NA,
                TW=2, SW1 = 4, SW2= 5, SW2 = NA)
d <- data.frame(user = "D", HP1 = NA, HP2 = 3, HP3 = NA,
                TW=NA, SW1 = NA, SW2= NA, SW2 = 3)
```

```
movies.rating <- rbind(a,b,c,d)
```

Ahora la funcion que trabaja especificamente con un dataframe definido como el anterior:

```
Jaccard <- function(data.set, user1, user2){

  u1 <- subset(data.set, user == user1)
  u2 <- subset(data.set, user == user2)

  u1 <- u1[, !(names(u1) %in% c("user"))]
  u2 <- u2[, !(names(u2) %in% c("user"))]

  union_ <- 0
  intersect_ <- 0

  for(name_ in names(u1)){

    if( !is.na(u1[,name_]) & !is.na(u2[,name_])){
      intersect_ <- intersect_ + 1
    }

    if( !is.na(u1[,name_]) | !is.na(u2[,name_])){
      union_ <- union_ + 1
    }

  }

  intersect_/union_

}
```

Ahora cuando probamos la similitud de A con B y A con C obtendriamos los siguientes resultados:

```
> Jaccard(data.set = movies.rating, user1 = "A", user2 = "B")
[1] 0.2
> Jaccard(data.set = movies.rating, user1 = "A", user2 = "C")
[1] 0.5
```

2.2 Basado en memoria

Este mecanismo utiliza los datos de las evaluaciones de los usuarios para calcular la similitud entre los usuarios o elementos. Esto se utiliza para hacer recomendaciones. Este fue de los primeros mecanismos y se usa en muchos sistemas comerciales. Es fácil de implementar y es eficaz. Ejemplos típicos de este mecanismo son los FC basados en el vecino más cercano y

recomendaciones de los N-primeros basadas en elementos o usuarios. [3] Por ejemplo, en los enfoques basados en el usuario, el valor de la evaluación del usuario u con respecto al elemento i se calcula como una agregación de las evaluaciones de usuarios similares para el elemento:
$r_{u,i} = agrr_{u' \in U} r_{u',i}$
Donde U denota el conjunto de los mejores N usuarios que son más similares al usuario u con respecto a i. Algunos ejemplos de la función de agregación incluye:

$$r_{u,i} = \frac{1}{N} \sum_{u' \in U} r_{u',i}$$
$$r_{u,i} = k \sum_{u' \in U} sim(u, u') r_{u',i}$$
$$r_{u,i} = \bar{r_u} + k \sum_{u' \in U} sim(u, u')(r_{u',i} - \bar{r_{u'}})$$

Donde k es un factor de normalización se define como $k = 1/\sum_{u' \in U} |sim(u, u')|$ y $\bar{r_u}$ es el promedio de las evaluaciones del usuario u para todos los elementos que ha evaluado.

El algoritmo basado en vecindad calcula la similitud entre dos usuarios o elementos, produce una predicción para el usuario tomando el promedio ponderado de todas las evaluaciones. Calcular la similitud entre los elementos o usuarios es una parte importante de este enfoque. Múltiples mecanismos tales como la correlación de Pearson y la similitud basada en el coseno entre vectores sirven para esto. La similitud de correlación de Pearson de dos usuarios x, y se define como

$$sim(x, y) = \frac{\sum_{i \in I_{xy}} (r_{x,i} - \bar{r_x})(r_{y,i} - \bar{r_y})}{\sqrt{\sum_{i \in I_{xy}} (r_{x,i} - \bar{r_x})^2 \sum_{i \in I_{xy}} (r_{y,i} - \bar{r_y})^2}}$$

donde Ixy es el conjunto de elementos evaluados por ambos usuarios. El enfoque basado en coseno define la similitud entre dos usuarios x e y como:

$$\text{sim}(x, y) = \cos(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{||\vec{x}|| \times ||\vec{y}||} = \frac{\sum_{i \in I_{xy}} r_{x,i} r_{y,i}}{\sqrt{\sum_{i \in I_x} r_{x,i}^2} \sqrt{\sum_{i \in I_y} r_{y,i}^2}}$$

El algoritmo de recomendación basado en los N-primeros usuarios identifica los k más similares al usuarios actual usando el modelo de similitud basado en vectores. Después de que los k usuarios más similares se encuentran, sus correspondientes matrices de usuario-elementos se agregan para identificar el conjunto de elementos que se recomiendan. Un método popular para encontrar a los usuarios similares es el, que implementa el en tiempo lineal. Las ventajas de este enfoque incluyen: la interpretación de los resultados, lo cual es un aspecto importante de los sistemas de recomendación, es fácil de crear y utilizar; los nuevos datos se pueden agregar fácilmente y de forma incremental, no es necesario considerar el contenido de los elementos que se recomiendan, y mecanismo escala adecuadamente con elementos reevaluados. Hay varias desventajas con este enfoque. En primer lugar, depende de las puntuaciones de las personas. En segundo lugar, su rendimiento disminuye cuando los datos son dispersos, lo cual es frecuente en elementos relacionados con la web. Esto evita la escalabilidad de este enfoque y tiene problemas con grandes conjuntos de datos. A pesar de que puede manejar eficientemente los nuevos usuarios porque cuenta con una estructura de datos, añadiendo nuevos elementos se hace más complicada, ya que la representación general

se basa en un espacio vectorial específico. Para ello sería necesario incluir el nuevo elemento y volver a insertar todos los elementos de la estructura.
Para hacer experimentos en R con esto necesitamos de los siguientes paquetes

```
install.packages("recommenderlab")
install.packages("recosystem")
```

Para probar el codigo

```
library(recommenderlab)
library(recosystem)

#Building model
model <- Recommender(real_ratings, method = "IBCF",
                     param=list(normalize = "center",
                                method="Cosine", k=350))

#Making predictions
prediction <- predict(model, real_ratings[1:5], type="ratings")
as(prediction, "matrix")[,1:8]
```

Y como resultado

```
       m1       m2       m3       m4       m5       m6       m7       m8
u1       NA       NA 4.000000       NA       NA 4.400783       NA 3.73582
u2 4.264205 3.185612 3.172429 3.202051 3.581299 4.014510 3.253853 3.00000
u3 4.034784 3.794960 4.475967       NA 3.477293 4.171651 4.000000 5.00000
u4 4.319038       NA       NA       NA       NA 4.318060       NA
NA
u5 3.561104 1.783488 1.000000 2.638740 1.524747       NA 2.851991 3.00000
```

3 Reducción de Dimensiones

Para conjuntos de datos de alta dimensión (es decir, con número de dimensiones más de 10), reducción de la dimensión se realiza generalmente antes de la aplicación de un K-vecinos más cercanos (k-NN) con el fin de evitar los efectos de la maldición de la dimensionalidad. [4]

Extracción de características y la reducción de la dimensión se puede combinar en un solo paso utilizando análisis de componentes principales (PCA), análisis discriminante lineal (LDA), o análisis de la correlación canónica (CCA) técnicas como un paso pre-procesamiento seguido por la agrupación de K-NN en vectores de características en el espacio reducido dimensión. En aprendizaje automático este proceso de pocas dimensiones también se llama incrustar.

Para conjuntos de datos muy altas dimensiones (por ejemplo, cuando se realiza la búsqueda de similitud de secuencias de vídeo en vivo, datos de ADN o de alta dimensión Series de tiempo) ejecutar un rápido 'aproximada' Búsqueda K-NN usando hash sensibles de localidad, "proyecciones aleatorias", "bocetos" u otras técnicas de búsqueda de similitud de alta dimensión de la VLDB caja de herramientas podría ser la única opción viable.

Para Entrar a nuestra tecnica aplicada SVD es importante tenes claro los siguientes conceptos:

3.1 Rango de una matriz

El rango de una matriz es el número máximo de columnas (filas respectivamente) que son linealmente independientes. El rango fila y el rango columna siempre son iguales: este número es llamado simplemente rango de A (prueba más abajo). Comúnmente se expresa como rg(A). El número de columnas independientes de una matriz A de m filas y n columnas es igual a la dimensión del espacio columna de A. También la dimensión del espacio fila determina el rango. El rango de A será, por tanto, un número no negativo, menor o igual que el mínimo entre m y n:

$$\mathbb{A} \in M_{mxn} \rightarrow 0 \leq rang(\mathbb{A}) \leq min(m, n)$$

3.2 Eigen valores y eigen espacios

Formalmente, se definen los vectores propios y valores propios de la siguiente manera: Sea $A : V \rightarrow V$ n operador lineal en un cierto $\mathbb{K}$-espacio vectorial V y v un vector no nulo en V. Si existe un escalar c tal que

$$\mathbf{A}\mathbf{v} = c\mathbf{v}, \qquad \mathbf{v} \neq \mathbf{0}, c \in \mathbb{K},$$

entonces decimos que v es un vector propio del operador A, y su valor propio asociado es c. Observe que si v es un vector propio con el valor propio c entonces cualquier múltiplo diferente de cero de v es también un vector propio con el valor propio c. De hecho, todos los vectores propios con el valor propio asociado c junto con 0, forman un subespacio de V, el espacio propio para el valor propio c. Observe además que un espacio propio Z es un subespacio invariante de A, es decir dado w un vector en Z, el vector Aw también pertenece a Z.

3.3 Proceso de ortogonalización de Gram-Schmidt

El método de Gram-Schmidt se usa para hallar bases ortogonales (Espacio Euclideo no normalizado) de cualquier base no euclídea. En primer lugar tenemos que:

$\mathbf{v} - \frac{\langle \mathbf{v}, \mathbf{u} \rangle}{\langle \mathbf{u}, \mathbf{u} \rangle} \mathbf{u} = \mathbf{v} - \mathrm{proy}_{\mathbf{u}}(\mathbf{v})$

Es un vector ortogonal a $\mathbf{u}$. Entonces, dados los vectores $\mathbf{v}_1, \ldots, \mathbf{v}_n$, se define:

$\mathbf{u}_1 = \mathbf{v}_1,$

$\mathbf{u}_2 = \mathbf{v}_2 - \frac{\langle \mathbf{v}_2, \mathbf{u}_1 \rangle}{\langle \mathbf{u}_1, \mathbf{u}_1 \rangle} \mathbf{u}_1,$

$\mathbf{u}_3 = \mathbf{v}_3 - \frac{\langle \mathbf{v}_3, \mathbf{u}_1 \rangle}{\langle \mathbf{u}_1, \mathbf{u}_1 \rangle} \mathbf{u}_1 - \frac{\langle \mathbf{v}_3, \mathbf{u}_2 \rangle}{\langle \mathbf{u}_2, \mathbf{u}_2 \rangle} \mathbf{u}_2,$

Generalizando en k:

$\mathbf{u}_k = \mathbf{v}_k - \sum_{j=1}^{k-1} \frac{\langle \mathbf{v}_k, \mathbf{u}_j \rangle}{\langle \mathbf{u}_j, \mathbf{u}_j \rangle} \mathbf{u}_j \mathbf{u}_k = \mathbf{v}_k - \sum_{j=1}^{k-1} \frac{\langle \mathbf{v}_k, \mathbf{u}_j \rangle}{\langle \mathbf{u}_j, \mathbf{u}_j \rangle} \mathbf{u}_j$

3.4 Descomposición en valores singulares

En álgebra lineal, la descomposición en valores singulares de una matriz real o compleja es una factorización de la misma con muchas aplicaciones en estadística y otras disciplinas.

Dada una matriz real $A \in \Re^{m \times n}$, los autovalores de la matriz cuadrada, simétrica y semidefinida positiva $A^T A \in \Re^{n \times n}$ son siempre reales y mayores o iguales a cero. Teniendo en cuenta el producto interno canónico vemos que:

$\left(A^T A\right)^T = A^T \left(A^T\right)^T = A^T A$. O sea que es simétrica

$(Ax, Ax) = x^T A^T A x = ||Ax||^2 \geq 0$ es decir $A^T A$ es semidefinida positiva, es decir, todos sus autovalores son mayores o iguales a cero.

Si λ_i es el i-ésimo autovalor asociado al i-ésimo autovector, entonces $\lambda_i \in \Re$. Esto es una propiedad de las matrices simétricas.

3.5 SVD con R

```
A = as.matrix(data.frame(c(4,7,-1,8), c(-5,-2,4,2), c(-1,3,-3,6)))
A
```

```
     c.4..7...1..8. c..5...2..4..2. c..1..3...3..6.
[1,]              4              -5              -1
[2,]              7              -2               3
[3,]             -1               4              -3
[4,]              8               2               6
```

Ahora para aplicar SVD podemos usar la funcion incluida en R

```
A.svd <- svd(A)
A.svd
```

```
d
[1] 13.161210  6.999892  3.432793

u
           [,1]       [,2]        [,3]
[1,] -0.2816569  0.7303849 -0.42412326
[2,] -0.5912537  0.1463017 -0.18371213
[3,]  0.2247823 -0.4040717 -0.88586638
[4,] -0.7214994 -0.5309048  0.04012567

v

           [,1]       [,2]       [,3]
[1,] -0.8557101 0.01464091 -0.5172483
[2,]  0.1555269 -0.94610374 -0.2840759
[3,] -0.4935297 -0.32353262  0.8073135
```

3.6 SVD con R, paso a paso

```
ATA <- t(A) %*% A
ATA

##                  c.4..7...1..8. c..5...2..4..2. c..1..3...3..6.
## c.4..7...1..8.              130            -22              68
## c..5...2..4..2.             -22             49              -1
## c..1..3...3..6.              68             -1              55

ATA.e <- eigen(ATA)
v.mat <- ATA.e$vectors
v.mat

##            [,1]       [,2]       [,3]
## [1,]  0.8557101 -0.01464091 -0.5172483
## [2,] -0.1555269  0.94610374 -0.2840759
## [3,]  0.4935297  0.32353262  0.8073135

v.mat[,1:2] <- v.mat[,1:2] * -1
v.mat

##            [,1]       [,2]       [,3]
## [1,] -0.8557101  0.01464091 -0.5172483
## [2,]  0.1555269 -0.94610374 -0.2840759
## [3,] -0.4935297 -0.32353262  0.8073135

AAT <- A %*% t(A)
AAT

##      [,1] [,2] [,3] [,4]
## [1,]   42   35  -21   16
```

```
## [2,]   35   62  -24   70
## [3,]  -21  -24   26  -18
## [4,]   16   70  -18  104
```

```
AAT.e <- eigen(AAT)
u.mat <- AAT.e$vectors
u.mat
```

```
##             [,1]       [,2]        [,3]       [,4]
## [1,] -0.2816569  0.7303849 -0.42412326 -0.4553316
## [2,] -0.5912537  0.1463017 -0.18371213  0.7715340
## [3,]  0.2247823 -0.4040717 -0.88586638  0.0379443
## [4,] -0.7214994 -0.5309048  0.04012567 -0.4426835
```

```
u.mat <- u.mat[,1:3]
r <- sqrt(ATA.e$values)
r <- r * diag(length(r))[,1:3]
r
```

```
##          [,1]     [,2]     [,3]
## [1,] 13.16121 0.000000 0.000000
## [2,]  0.00000 6.999892 0.000000
## [3,]  0.00000 0.000000 3.432793
```

```
svd.matrix <- u.mat %*% r %*% t(v.mat)
svd.matrix
```

```
##      [,1] [,2] [,3]
## [1,]    4   -5   -1
## [2,]    7   -2    3
## [3,]   -1    4   -3
## [4,]    8    2    6
```

4 Conclusión

La cantidad de algoritmos para construir sistemas de recomendacion son incontables, hoy en dia con el constante crecimiento de Internet y con las distintas necesidades para retener los usuarios, los recommendation engines deben de ser mas precisos hasta el punto de dar la experiencia de personalizacion por usuario. tener este tipo de paradigmas hace que el costo computacional sea elevado en el caso de querer implementar tecnicas en base a filtros colaborativos, por lo que para datasets pequeños se recomienda su uso. lo bueno de este tipo de tecnicas es que su interpretabilidad es mas rapida. para el caso de datasets muy grandes se recomiendan tecnicas que involucran reduccion de dimensiones como el caso de Descomposicion Espectral, SVD o ACP. Estos por su tecnica son mas rapidos de procesar pero mas dificiles de interpretar.

References

[1] Real, R., Vargas, J. M. (1996). The probabilistic basis of Jaccards index of similarity. Systematic biology, 45(3), 380-385

[2] Terveen, Loren; Hill, Will (2001). Beyond Recommender Systems: Helping People Help Each Other

[3] Xiaoyuan Su, Taghi M. Khoshgoftaar, A survey of collaborative filtering techniques, Advances in Artificial Intelligence archive, 2009.

[4] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, Uri Shaft (1999) "Cuando es" vecino más cercano "significativa? ". Base de datos Teoría-ICDT99 , 217-235