Convirtiendo los datos en información

Práctica 3

Sistemas de Información

Víctor Fernández Fernández
Mikayel Mardanyan Petrosyan

1 Decisiones de implementación

1. **¿Cómo medir la similitud de los usuarios?**

   Hay varios tipos de medidas que seleccionar a la hora de calcular la similitud entre usuarios (Manhattan, Euclidea, Minkowski, correlación de Pearson, medida del coseno, ...), ¿cuál de estas se debe usar en nuestro caso?

   Las 3 primeras medidas se usan cuando la matriz de datos es densa, la siguiente cuando los valores pueden estar inflados (por ejemplo, debido a que cada valoración es dada por una persona distinta) y la última cuando la matriz es dispersa.

   No existe un umbral estipulado que marque cómo marcar una matriz como densa o dispersa, aunque Wikipedia[1] indica que una matriz es dispersa cuando más de la mitad de la matriz tiene huecos o ceros. En nuestro caso, en las matrices de entrada, los ceros representan el 15,6%, 16,2%, 17,6% y 18,6% del total de la matriz, respectivamente, por lo que podemos considerar que las matrices son densas.

   Es por ello que debemos utilizar alguna de las 3 primeras medidas. Podemos descartar el coeficiente de Pearson, ya que suponemos que la valoración del efecto de los medicamentos es realizada por un sanitario y no está inflada.

   En cuanto a cuál de estas 3 utilizar, podemos descartar la distancia de Minkowski ya que ningún artículo es más determinante que otro. Y entre los dos restantes, elegimos la distancia de Manhattan (o CityBlock distance) animados por este post[2].

2. **Tipo de filtrado**

   Hay dos tipos de filtrado para realizar el filtrado colaborativo:

   - Basado en usuarios
   - Basado en artículos

   ¿Cuál se debe usar?

   El filtrado basado en usuarios es adecuado cuando:

   - Las matrices son densas.

- Hay muchos datos en relación al número de artículos y estos no son cambiantes.

El filtrado basado en artículos es adecuado cuando:

- Las valoraciones son cambiantes.

Teniendo en cuenta la información anterior y la del apartado anterior podemos asegurar que el filtrado adecuado es el basado en usuarios.

3. **¿Qué implementación de *UserNeighborhood* usar?**

   La interfaz *UserNeighborhood* tiene 3 implementaciones: *CachingUserNeighborhood*, *NearestNUserNeighborhood* y *ThresholdUserNeighborhood*. De estas la más adecuada es *ThresholdUserNeighborhood*, ya que busca la similitud con todos los posibles usuarios y esto es lo que el enunciado pide.

2 Filtrado colaborativo

Para realizar el filtrado colaborativo es necesario cargar los datos en un objeto *DataTable*. Para ello, los datos deben estar en el formato {*id, objeto, valoracion*}. Es por ello que el primer paso a realizar es disponer los datos en este formato. Para ello se crea un nuevo fichero *datos_filtrado_colaborativo_todos.csv* con los datos en el formato pedido. [3]

Una vez cargados los datos, se puede realizar la recomendación como muestra la documentación. [4] [5] [6] [7]

3 Reglas de asociación

En este caso, la información de los ficheros CSV debe cargarse en un objeto del tipo *Instances*. Dicha carga es simple cuando la información se encuentra en un solo fichero y en el formato deseado, pero en nuestro caso, se encuentra repartida en varios y con valores del 0 al 5. La forma más sencilla sería recorrer todos los ficheros, sustituyendo los valores dados por los deseados de uno en uno y guardar la información en un único fichero. Sin embargo, este pre-procesamiento requiere tiempo, y se decidió investigar si había una manera más rápida de hacerlo.

Y así es. Se creó un objeto *Instances* para cada fichero y después se unieron todos en uno. A continuación, se aplicaron varios filtros: un primer filtro para eliminar la primera columna con los *id*s, ya que estos no son necesarios; otro para convertir los números menores que 3.5 en *missing values*, denotados mediante el símbolo *?* [1] y los mayores que 3.5 en 1 (equivalente a *t (true)*); y un último filtro para convertir los datos en nominales. Para las reglas de fallo el segundo filtro mencionado se invirtió, y además fue necesario un filtro adicional para convertir los ceros en valores mayores que 3.5 para que estos contaran como *missing*. [8] [9] [10] [11] [12] [13] [14]

La interfaz gráfica de Weka nos permitió jugar con los distintos filtros para obtener el adecuado.

[1] Ya que el filtro solo admitía valores numéricos y no permitía introducir el símbolo *?* directamente, los valores se intercambiaron por *Double.NaN*

4 Agrupamiento

En este apartado los datos también se debían guardar en un objeto *Instances*, pero estos no se encontraban en un CSV sino en una base de datos. Al igual que en el apartado anterior, esto se puede resolver con un pre-procesamiento, pasando los datos a un fichero CSV para luego cargarlos, pero esta no es la mejor manera.

El objeto *Instances* es un conjunto de objetos *Instance*, los cuales almacenan cada línea del CSV, es decir, los datos de cada paciente (edad, IMC, forma física, ...). Para crear un objeto *Instance* se usó la implementación *DenseInstance*, pasando como parámetro al constructor de este un array con 20 valores, uno por cada atributo del paciente y un decimal (1.0), indicando que todos tienen el mismo peso. Además, hay que indicar al objeto *Instances* cuáles son los atributos, especificando el nombre (las cabeceras de las columnas que habría en la primera fila del CSV) y los valores que puede tomar. Esta información se encuentra en la variable *attInfo*.

Los atributos *edad* y *IMC* son de tipo *double*, el valor por defecto. *Sexo* podrá tomar los valores {*V, M*}; *forma física* y *hepatitis*, los valores {*0, 1, 2, 3*}; y el resto, *booleans*. Es necesario indicar el rango de *forma física* y *hepatitis* para no obtener pacientes con hepatitis del tipo 1.5 o con forma física 2.4, ya que nos parecía que estos atributos debían tener valores discretos, no continuos.

También creemos necesario mencionar, para facilitar la corrección, que a la hora de guardar los pacientes prototipo en un CSV, los valores de los atributos se pueden obtener de dos maneras. Los valores de los atributos de un paciente prototipo vienen representados mediante un array de números *double*. En el caso de los atributos de tipo *double*, estos valores representan los valores en sí, mientras que en el resto de atributos, representan el índice del array de valores que puede tomar dicho atributo. [15]

References

[1] Sparse matrix
Fuente: https://en.wikipedia.org/wiki/Sparse_matrix

[2] Manhattan vs. Euclidean distance
Fuente: https://datascience.stackexchange.com/questions/20075/when-would-one-use-manhattan-distance-as-opposed-to-euclidean-distance/20231#20231

[3] DataModel interface
Fuente: https://mahout.apache.org/docs/0.13.0/api/docs/mahout-mr/org/apache/mahout/cf/taste/model/DataModel.html

[4] UserBasedRecommender interface
Fuente: https://mahout.apache.org/docs/0.13.0/api/docs/mahout-mr/org/apache/mahout/cf/taste/recommender/class-use/UserBasedRecommender.html#org.apache.mahout.cf.taste.impl.recommender

[5] UserBasedRecommender in 5 minutes
Fuente: https://mahout.apache.org/users/recommender/userbased-5-minutes.html

[6] UserNeighborhood interface
Fuente: https://mahout.apache.org/docs/0.13.0/api/docs/mahout-mr/org/apache/mahout/cf/taste/neighborhood/UserNeighborhood.html

[7] UserSimilarity interface
Fuente: https://mahout.apache.org/docs/0.13.0/api/docs/mahout-mr/org/apache/mahout/cf/taste/similarity/UserSimilarity.html

[8] Instances class
Fuente: https://weka.sourceforge.io/doc.stable-3-8/weka/core/Instances.html

[9] Instance class
Fuente: https://weka.sourceforge.io/doc.stable-3-8/weka/core/Instance.html

[10] Remove filter
Fuente:https://weka.sourceforge.io/doc.stable-3-8/weka/filters/unsupervised/attribute/Remove.html

[11] NumericCleaner class
Fuente: https://weka.sourceforge.io/doc.stable-3-8/weka/filters/unsupervised/attribute/NumericCleaner.html

[12] NumericToNominal class
Fuente: https://weka.sourceforge.io/doc.stable-3-8/weka/filters/unsupervised/attribute/NumericToNominal.html

[13] NumericTransform class
Fuente: https://weka.sourceforge.io/doc.stable-3-8/weka/filters/unsupervised/attribute/NumericTransform.html

[14] AssociationRule class
Fuente: https://weka.sourceforge.io/doc.stable-3-8/weka/associations/AssociationRule.html

[15] Attribute class
Fuente: https://weka.sourceforge.io/doc.dev/weka/core/Attribute.html