
 <p>Universidad de los Andes Facultad de Ingeniería</p>	<p>Ingeniería de Sistemas y Computación Maestría ISIS4518 – Sistemas de Recomendación – Sección 01 http://sistemas.uniandes.edu.co/~isis4518 Semestre: 2015-10 Horario: Martes – 18:00 a 20:50</p>	
---	---	---

Laboratorio 1

Descripción y objetivo

El laboratorio está diseñado para familiarizar al estudiante con varios *frameworks* para desarrollo de modelos de recomendación: el *framework* recommender101 y el *framework* Apache Mahout. Ambos implementan diferentes algoritmos de recomendación, el segundo además provee algunas funcionalidades de aprendizaje de máquina.

Los objetivos del laboratorio son:

- Familiarizarse con la estructura típica de archivos que representan la matriz de utilidad usada por los algoritmos de recomendación
- Crear un algoritmo de recomendación simple dentro de la estructura del *framework*
- Utilizar la implementación de un algoritmo de recomendación visto en clase.

Material

- Framework Recommender101
 - disponible en: <http://ls13-www.cs.uni-dortmund.de/homepage/recommender101/index.shtml>
- Dataset Movielens100k
 - disponible en: <http://files.grouplens.org/datasets/movielens/ml-100k.zip>
- Proyecto eclipse test-recommender101 disponible en Sicua+.
- Eclipse - IDE

Metodología

Realice el laboratorio en los grupos previstos para el trabajo práctico del curso. Se realiza una entrega por grupo.

Entregable

La entrega de resultados del laboratorio consta de dos partes: El software desarrollado y un informe de laboratorio.

Realice un informe donde documente su desarrollo y avance de cada uno de los puntos del laboratorio. Inicialmente muestre, mediante imágenes de pantalla, su logro de los objetivos de cada punto.

En los puntos en los cuales se espera que usted explore y experimente, realice una pequeña descripción de lo que encuentra y obtiene.

Documente los resultados obtenidos en los desarrollos realizados y haga un breve análisis sobre los resultados. En particular, documente sus hallazgos con respecto al alcance y diferencia entre los *frameworks* revisados.

Formato y hora de entrega

Para la presentación del informe de laboratorio utilice la plantilla de documentos disponible en Sicua+.

Realice su entrega del laboratorio de la siguiente forma:

- Archivo en formato **zip**, nombrado de la siguiente forma: **Lab1_NN_login1_login2.zip**, donde NN es el número del grupo y luego se encuentran los login uniandes de los integrantes del grupo.
- Contenido del archivo:
 - Proyectos eclipse modificados durante el laboratorio
 - Documento de informe en formato **pdf**, nombrado de la siguiente forma: **Lab1_NN_login1_login2.pdf**

Fecha límite de entrega: Febrero 17, 20:50 en Sicua+.

Desarrollo

1. Importar datos en modelo

El *framework* Recommender101 sirve para realizar la evaluación fuera de línea de diferentes algoritmos de recomendación. Es distribuido como un proyecto de *eclipse*, descargue el proyecto e impórtelo en *eclipse*.

Importe en eclipse el proyecto llamado `test-recommender101` e incluya en su `build path` el proyecto `recommender101-core`.

Para evaluar un sistema de recomendación es necesario un conjunto de datos, en este caso descargue el conjunto de datos (*dataset*) Movielens100k y descomprima el directorio `ml100k` en el directorio `data` del nuevo proyecto.

El conjunto de datos Movielens100k tiene 100,000 ratings (entre 1-5) de 943 usuarios para 1682 películas, lea el archivo README para obtener más información acerca de este conjunto de datos.

El archivo `u.data` es el archivo principal del conjunto de datos y contiene los *ratings*, uno por línea. Este archivo representa la matriz de utilidad que usan los algoritmos de recomendación basados en filtrado colaborativo.

El primer paso es cargar este archivo a una estructura en memoria principal, para esto se utiliza la clase `DefaultDataLoader` para cargar los *ratings* a un objeto de tipo `DataModel`. Verifique la documentación de estas 2 clases.

Cree los objetos de tipo `DefaultDataLoader` (*loader*) y `DataModel` (*model*), invoque el método `setFileName(String file)` con la ruta del archivo `u.data` sobre el *loader*. Invoque el método `loadData(DataModel Model)` sobre el *loader* pasando como parámetro el *model*. De esta forma los *ratings* quedan en memoria.

2. Añadir ratings al modelo

Añada sus ratings al modelo. El método que permite añadir *ratings* a *model* es `addRating(int user, int item, byte value)`. Utilice como *rating* un valor entre 1 a 5 (enteros), utilice un id de usuario nuevo (i.e 1000). Para ver cuáles películas están disponibles puede consultar el archivo `u.item`. Puede correr el `main` de la clase `ItemInfoLoader` del proyecto `test-recommender101` para saber cuáles son las películas con más y menos *ratings* en el *dataset*.

3. Crear recomendador simple

El primer recomendador a crear es un recomendador que siempre recomienda los ítems que tienen más *ratings*, al crear un recomendador debe crear una nueva clase que extienda la clase `AbstractRecommender`. Los métodos que debe implementar en la nueva clase son:

```
/**
 * Predicts a rating for the user
 * @param user
 * @param item
 * @return the rating value
 */
public float predictRating(int user, int item);

/**
 * Generates a ranked list of recommendations
 * @param user
 * @return the ranked list of items (in descending order)
 */
public List<Integer> recommendItems(int user);

/**
 * An init method which will be called by the instantiating class after object
 * creation
 */
public abstract void init() throws Exception;
```

Implemente un nuevo recomendador que devuelva la lista de ítems más populares (los que más ratings tienen) y otro que devuelva la lista de ítems con mejor rating promedio. La única personalización que se va a hacer sobre la lista que retornan estos es retirar de la lista los ítems que el usuario ya ha visto. Verifique los servicios que la clase `ItemInfoLoader` del proyecto `test-recommender101` brinda.

4. Uso de recomendador basado en similitud de usuarios y similitud de ítems

La clase `NearestNeighbors` implementa los algoritmos de filtrado colaborativo basado en usuario-usuario o ítem-ítem. El recomendador utiliza por defecto la similitud de correlación de Pearson para calcular el grupo de los vecinos que va a utilizar al momento de hacer una predicción. Instancie 2 recomendadores, uno para cada modelo (usuario-usuario e ítem-ítem) y genere su lista personalizada de recomendación. Compare su resultado con los recomendadores implementados anteriormente.

5. Uso de framework Apache Mahout

El *framework* Apache Mahout es distribuido mediante Maven Central, verifique el archivo `.pom` del proyecto para ver las dependencias incluidas para usar el *framework*.

El tutorial disponible en <https://mahout.apache.org/users/recommender/userbased-5-minutes.html> le ayuda a crear rápidamente un recomendador basado en usuario. Añada sus *ratings* al modelo. Revise el paquete `org.apache.mahout.cf.taste.impl.similarity` y describa su contenido. Considere el uso de varios tipos de similitud de usuario y genere listas de recomendación para su usuario. El mecanismo para la formación del vecindario considerado en el ejemplo es basado en un *threshold*. Explore el uso del mecanismo de formación de vecindario provisto por la clase `NearestNUserNeighborhood`.