UNIVERSIDAD DE GRANADA

Máster Universitario en Ingeniería Informática

Gestión de Información en la Web

Desarrollo de un Sistema de Recomendación basado en Filtrado Colaborativo

Alumno
Marvin Matías Agüero Torales
DNI
4423998
E-mail
maguero@correo.ugr.es

2016-2017

Contenido

Introducción

En este trabajo se construirá, partiendo de cero y en el lenguaje Java, un sistema de recomendación de películas basado en filtrado colaborativo (usuario - usuario). Para tal fin, se desarrollará una aplicación que muestre al usuario 20 películas al azar y éste las evalúe asignándole un valor de 1 estrella (*), indicando que no le gusta nada, hasta 5 estrellas, si es una de sus películas favoritas.

Una vez obtenidas las valoraciones del usuario, el sistema procederá a calcular el vecindario, es decir, el grupo de usuarios que más se parecen a él en cuanto a las películas vistas y a las valoraciones dadas.

Finalmente, la aplicación predirá la valoración para el usuario activo de todas las películas que han visto sus vecinos y que no ha visto el usuario activo, y le mostrará aquellas predichas con cuatro o cinco estrellas.

Objetivos

- Entender el proceso de recomendación basado en filtrado colaborativo.
- Ser capaz de plasmarlo en un programa de ordenador.

Metodología y herramientas

Para llevar acabo el desarrollo de este trabajo, como se mencionó, se ha empleado Java, que por ser un lenguaje de programación multiplataforma y multi-propósito, resulta ideal para este trabajo. El IDE (Integrated Development Environment) utilizado es Eclipse Neon.2. Como un elemento más que innovador, más bien extra, agregué como para hacer una comparativa de nuestro cálculo, Mahout y su sistema de recomendación.

La práctica se realizará con la colección MovieLens[1], compuesta por 100.000 valoraciones de 943 usuarios sobre 1682 películas. Se trabajará con los ficheros u.data, con todas las valoraciones (formato: idUser idMovie valoración timestamp), y u.item, con información de las películas (idMovie | título |…).

Sistema de Recomendación basado en Filtrado Colaborativo

Desde cero

El Sistema de Recomendación basado en Filtrado Colaborativo consta de un recomendador que obtiene las valoraciones de películas de un usuario, además de su vecindario, a partir de la base de datos MovieLens, optando utilizar la correlación de Pearson[2]. Luego en base a lo anterior, sugiere las películas recomendadas.

Las películas presentadas al azar al usuario se obtienen del fichero u.item y los vecindarios se crearán a partir de las valoraciones que ha hecho el usuario activo y de las contenidas en u.data. Se consideran vecindarios de tamaño 10.

Para ello las películas se almacenan en un vector de objetos, donde un objeto está compuesto por atributos como ID de la película, el título de la misma y su URL. Mientras que las valoraciones, se almacenan en un objeto con un diccionario, donde su clave es el ID del usuario y su valor, otro

1 https://grouplens.org/datasets/movielens/100k/
2 http://www.statisticshowto.com/what-is-the-pearson-correlation-coefficient/

diccionario, en el que la clave es el ID de la película y el valor de la valoración del usuario. Se ha creado además otro objeto, con los atributos ID de usuario y la valoración media de todas la películas evaluadas.

Para poder obtener el vecindario, además de las valoraciones extraídas de la base de datos se necesitan valoraciones del usuario activo, como el número de vecinos más cercanos definido. Ya con estos datos se puede calcular la similitud de todos los usuarios extraídos de la fuente de MovieLens, con ayuda de la correlación de Pearson en base a la valoración del usuario para la película y su propia valoración media:

$$[sim(u,v) = \frac{\sum(r(u,i) - \bar{r}(u))(r(v,i) - \bar{r}(v))}{\sqrt{\sum(r(u,i) - \bar{r}(u))^2}\sqrt{\sum(r(v,i) - \bar{r}(v))^2}}]$$

Finalmente, ya obtenidos todos los usuarios, se ordenan[3], tomando los primeros. Entonces, pasamos a calcular la recomendación en base a la fuente de datos de películas en base a la valoración del usuario para la película, la valoración media del usuario, la similitud entre usuarios. Solo se muestran las 20 películas con más valoración, a partir de 4 estrellas o puntos.

Mahout

Al igual que el apartado anterior, pero utilizando Taste[4] de Mahout[5], implementaremos un sistema de recomendación basado en filtrado colaborativo. Antes que nada debemos convertir nuestro proyecto Java a uno Maven para agregar las dependencias de Mahout al pom.xml:

```
<dependency>
    <groupId>org.apache.mahout</groupId>
    <artifactId>mahout-mr</artifactId>
    <version>0.10.0</version>
</dependency>
```

Acto seguido, debemos convertir el fichero u.data a uno separados por comas, para que Mahout pueda trabajar con él, una vez migremos el contenido de este fichero a un CSV, debemos agregar la valoraciones del usuario activo al mismo fichero con el ID disponible: 944 (puesto que existen 943 usuarios). Ya hecho todo esto, se procede a hacer el cálculo sobre el usuario activo 944, con los mismos parámetros utilizados en el sistema de recomendación previamente creado.

Una estructura muy básica para implementar un sistema de recomendación “user based” en Mahout sería la siguiente:

```
DataModel model = new FileDataModel(new File("/path/to/dataset.csv"));
UserSimilarity similarity = new PearsonCorrelationSimilarity(model);
UserNeighborhood neighborhood = new ThresholdUserNeighborhood(0.1, similarity, model);
UserBasedRecommender recommender = new GenericUserBasedRecommender(model, neighborhood,
similarity);
List recommendations = recommender.recommend(1, 3);
for (RecommendedItem recommendation : recommendations) {
  System.out.println(recommendation);
}
```

En nuestro caso, limitamos el vecindario a 10, las recomendaciones a 20 y la posibilidad de poder elegir entre una función de similaridad u otra, así como también las estadísticas del recomendador (precisión, evaluación, etc.), que calcula la diferencia entre las puntuaciones predichas y las reales para los usuarios (pero la ejecución es más lenta).

3 http://stackoverflow.com/a/1283722
4 https://sourceforge.net/projects/taste/
5 https://mahout.apache.org/users/recommender/recommender-documentation.html

Consideraciones

Hay que tener cuidado con la selección de objetos, vectores y matrices que se utilicen para los cálculos. Primero he intentado hacerlo completamente con vectores de objetos, pero resultó ineficiente en tiempo y pocas facilidades de esa estructura, por lo que utilizar diccionarios me resultó mejor, al menos en mi caso.

También cabe agregar, que al utilizar una herramienta como Mahout, los tiempos se desarrollo se acortan considerablemente, logrando muy buenos resultados.

Manual de usuario

El programa se ejecutará en línea de comandos. Para obtener las puntuaciones se lo puede hacer introduciendo las valoraciones para cada película, o hacerlo aleatoriamente (0, manual, 1, automática). Cabe recordar que las películas se presentan de forma aleatoria. También se puede modificar para el sistema de recomendación sobre Mahout, la función a utilizar siendo los valores posibles lo siguientes: PEARSON, COSINE, SPEARMAN, EUCLIDEAN, TANIMOTO, LIKELIHOOD. Por ejemplo, se puede ingresar lo siguiente:

Para no solicitar valorar películas y calcular con Mahout con la función de similaridad de Coseno:

java -jar runnable.jar 1 COSINE

Para valorarlas y calcular con Mahout con la función de similaridad de Correlación de Pearson:

java -jar runnable.jar 0 PEARSON

Tan solo hay que ejecutar el .jar e introducir las valoraciones, el programa devolverá los resultados para ambos sistemas de recomendación, de acuerdo a lo ingresado.

La estructura de programa es con el .jar y en la misma altura, es necesario tener los ficheros u.data y u.item en el directorio ml-data para que el programa los pueda leer.

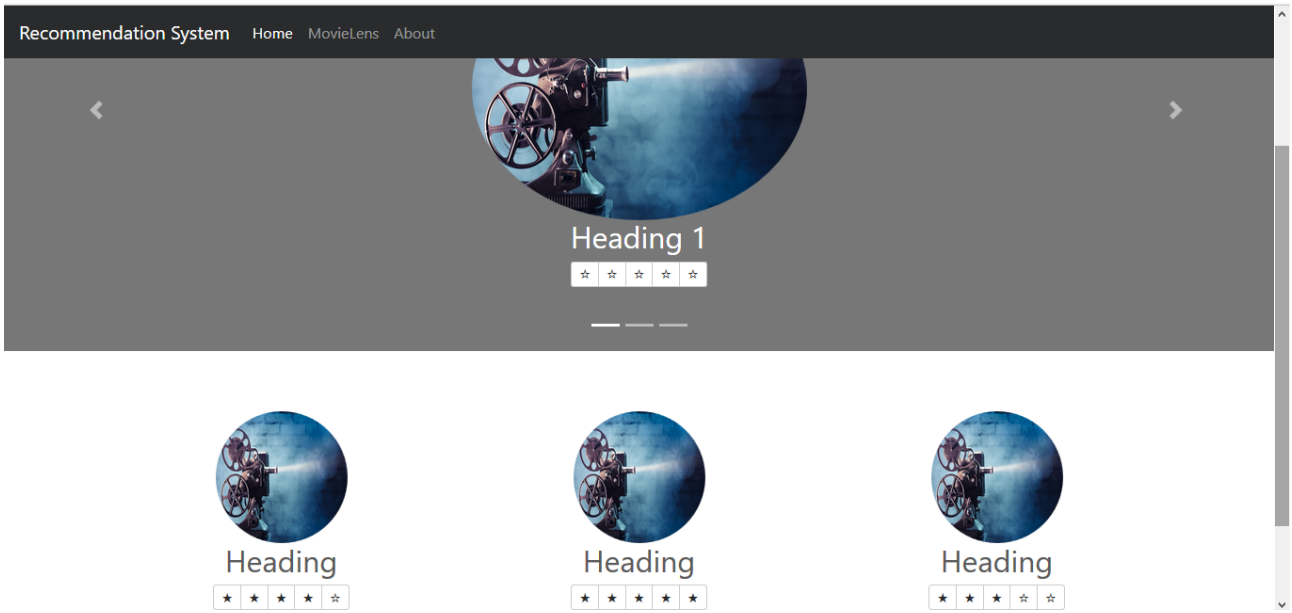
Si se desea cambiar cualquiera de los otros parámetros (número de valoraciones de usuario activo, número de vecinos más cercanos, número de recomendaciones, puntuación base para recomendar, activar estadísticas de Mahout, función de similitud para Mahout) hay que cambiar necesariamente el código y recompilar. Esta sería una salida del programa:

```
-------------------------------------------------------------------
Hi! user ... Please say you rating value for the next movies ...
-------------------------------------------------------------------
Movie title => Happy Gilmore (1996)
Please, enter a rating [1 (worst) to 5 (best)]:
4
Movie title => Fatal Instinct (1993)
Please, enter a rating [1 (worst) to 5 (best)]:
5
Movie title => Little Women (1994)
Please, enter a rating [1 (worst) to 5 (best)]:
3
Movie title => 'Til There Was You (1997)
Please, enter a rating [1 (worst) to 5 (best)]:
4
Movie title => Excess Baggage (1997)
Please, enter a rating [1 (worst) to 5 (best)]:
5
Movie title => Foreign Correspondent (1940)
Please, enter a rating [1 (worst) to 5 (best)]:
2
Movie title => Doors, The (1991)
Please, enter a rating [1 (worst) to 5 (best)]:
4
Movie title => Frighteners, The (1996)
Please, enter a rating [1 (worst) to 5 (best)]:
1
Movie title => Casablanca (1942)
Please, enter a rating [1 (worst) to 5 (best)]:
```

```
1
Movie title => Magnificent Seven, The (1954)
Please, enter a rating [1 (worst) to 5 (best)]:
3
Movie title => Midnight Dancers (Sibak) (1994)
Please, enter a rating [1 (worst) to 5 (best)]:
3
Movie title => Robin Hood: Prince of Thieves (1991)
Please, enter a rating [1 (worst) to 5 (best)]:
3
Movie title => Three Colors: Red (1994)
Please, enter a rating [1 (worst) to 5 (best)]:
3
Movie title => Usual Suspects, The (1995)
Please, enter a rating [1 (worst) to 5 (best)]:
4
Movie title => Stephen King's The Langoliers (1995)
Please, enter a rating [1 (worst) to 5 (best)]:
3
Movie title => Mrs. Parker and the Vicious Circle (1994)
Please, enter a rating [1 (worst) to 5 (best)]:
1
Movie title => Tombstone (1993)
Please, enter a rating [1 (worst) to 5 (best)]:
2
Movie title => Blood & Wine (1997)
Please, enter a rating [1 (worst) to 5 (best)]:
4
Movie title => Dead Man Walking (1995)
Please, enter a rating [1 (worst) to 5 (best)]:
3
Movie title => Temptress Moon (Feng Yue) (1996)
Please, enter a rating [1 (worst) to 5 (best)]:
2
---------------------------------------------------------------------
Thanks! See our recommendations for you:
---------------------------------------------------------------------
Movie => Spitfire Grill, The (1996) | Rating => 5.0
Movie => Cold Comfort Farm (1995) | Rating => 5.0
Movie => Horseman on the Roof, The (Hussard sur le toit, Le) (1995) | Rating => 5.0
Movie => Carried Away (1996) | Rating => 4.807017543859649
Movie => Secrets & Lies (1996) | Rating => 4.807017543859649
Movie => Godfather, The (1972) | Rating => 4.558681185722928
Movie => Speed 2: Cruise Control (1997) | Rating => 4.308333333333334
Movie => Con Air (1997) | Rating => 4.308333333333334
Movie => Jumanji (1995) | Rating => 4.308333333333334
Movie => Beauty and the Beast (1991) | Rating => 4.308333333333334
Movie => Mortal Kombat (1995) | Rating => 4.308333333333334
Movie => Alice in Wonderland (1951) | Rating => 4.308333333333334
Movie => Star Trek: Generations (1994) | Rating => 4.308333333333334
Movie => Clueless (1995) | Rating => 4.308333333333334
Movie => Mask, The (1994) | Rating => 4.308333333333334
Movie => Batman Forever (1995) | Rating => 4.308333333333334
Movie => Seven (Se7en) (1995) | Rating => 4.308333333333334
Movie => Speed (1994) | Rating => 4.226747311827957
Movie => Mrs. Doubtfire (1993) | Rating => 4.226747311827957
Movie => Star Trek IV: The Voyage Home (1986) | Rating => 4.226747311827957
---------------------------------------------------------------------
MAHOUT:
---------------------------------------------------------------------
Thanks! See our others recommendations for you:
---------------------------------------------------------------------
Function Similarity => PEARSON
---------------------------------------------------------------------
Movie => Chinatown (1974) | Rating => 5.0
Movie => Jane Eyre (1996) | Rating => 5.0
Movie => Rear Window (1954) | Rating => 4.75
Movie => Local Hero (1983) | Rating => 4.6666665
Movie => My Best Friend's Wedding (1997) | Rating => 4.6666665
Movie => Psycho (1960) | Rating => 4.6666665
Movie => This Is Spinal Tap (1984) | Rating => 4.6666665
Movie => To Kill a Mockingbird (1962) | Rating => 4.6666665
Movie => Taxi Driver (1976) | Rating => 4.6
Movie => One Flew Over the Cuckoo's Nest (1975) | Rating => 4.6
Movie => Citizen Kane (1941) | Rating => 4.5
Movie => Shining, The (1980) | Rating => 4.5
Movie => Philadelphia Story, The (1940) | Rating => 4.5
Movie => Princess Bride, The (1987) | Rating => 4.5
```

```
Movie => Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1963) | Rating => 4.5
Movie => Raging Bull (1980) | Rating => 4.5
Movie => Clockwork Orange, A (1971) | Rating => 4.5
Movie => GoodFellas (1990) | Rating => 4.5
Movie => Butch Cassidy and the Sundance Kid (1969) | Rating => 4.5
Movie => Full Metal Jacket (1987) | Rating => 4.5
---------------------------------------------------------------------
RMSE: 1.1634013915801678
Precision: 0.035280373831775715
Recall: 0.035525365338667535
F1 Score: 0.035402445743601206
```

Se ha creado un pequeño bosquejo *no funcional* de como podría ser este sistema sobre una plataforma Web:



Conclusiones

Los sistemas de recomendación son muy utilizados en la Web: Amazon, Netflix, Facebook, son algunos ejemplos. Los algoritmos implementados para estos, generalmente tienden a tener buenos resultados. Desde luego el usuario busca que los objetos recomendados sean satisfactorios, aunque a veces no sea así, es por ello que mientras más datos tengamos de los usuarios, seremos más certeros. El algoritmo óptimo de recomendación depende de la naturaleza de los datos y del escenario en cuestión.

Hay mucho que mejorar, más partiendo desde un sistema hecho desde cero, y para el lenguaje utilizado en este trabajo (Java), existen opciones como Taste de Mahout, que es mucho más amigable y con una comunidad detrás; o neo4j[6] junto con Cypher[7], una base de datos orientada a grafos y su lenguaje de consultas respectivamente; ambos con muchas más opciones para experimentar con sistemas de recomendación y filtros colaborativos casi al vuelo (como pudimos comprobar con el primero).

Bibliografía

Fernandez Luna, J. M. (2017). Sistemas de Recomendación, Gestión de Información en la Web, Máster en Ingeniería Informática, Dpto. Ciencias de la Computación e Inteligencia Artificial, Universidad de Granada.
Illinois Institute of Technology. Item Based Recommenders. Recuperado el 17 de Mayo de 2017 de

6 https://neo4j.com/developer/guide-build-a-recommendation-engine/
7 https://neo4j.com/developer/cypher-query-language/

http://ir.cs.georgetown.edu/cs422/files/DM-ItemRecommenders.pdf.
Skategui (2017). Recommendation algorithms with Apache Mahout. Recuperado el 17 de Mayo de 2017 de https://blog.guillaumeagis.eu/recommendation-algorithms-with-apache-mahout/.

Anexos

Se adjunta en un .zip el proyecto Java con la siguiente estructura:

- MovieLens
  - src
    - engine → Contiene los fuentes necesarios para crear el Sistema de Recomendación basado en Filtrado Colaborativo "Usuario-Usuario"
    - hadoop → Implementación de Taste de Hadoop para un Sistema de Recomendación basado en Filtrado Colaborativo "Usuario-Usuario"
    - system → Contiene el fuente **main** que ejecuta el *Recomendador*
  - *pom.xml → archivo con las dependencias del proyecto*
  - ml-data → contiene los archivos u.item, u.data y u.data.csv
  - mockup → boceto web del sistema de recomendación
  - demo → .zip con .jar ejecutarble del sistema de recomendación por líneas de comandos

Disponible en https://github.com/mmaguero/MovieLensRecommender