Universidad Autónoma de Madrid, Escuela Politécnica Superior

Grado en Ingeniería Informática

Búsqueda y Minería de Información 2013-2014

Práctica 5: Sistemas de Recomendación y Análisis de Redes Sociales

Fechas

- Comienzo: 4 de abril
- Entrega: 28 de abril (hora límite 23:55)

Objetivos

Un primer objetivo de esta quinta práctica es implementar y evaluar sistemas de recomendación. En concreto, se van a desarrollar:

- Algoritmos de recomendación basada en contenido
- Algoritmos (heurísticas) de recomendación basada en filtrado colaborativo
- Métricas de evaluación de sistemas de recomendación

Un segundo objetivo es realizar un análisis sencillo de redes sociales. En concreto, se van a desarrollar:

- Métricas topológicas en grafos usadas para el análisis de redes sociales
- Generación de modelos aleatorios de redes sociales
- Visualización de grafos asociados a redes sociales

Conjuntos de datos

Sistemas de recomendación

Los algoritmos de recomendación y su evaluación se van a desarrollar usando el conjunto de datos de puntuaciones (*ratings*) de películas del sistema MovieLens[1] extendido en el 2nd Workshop on Information Heterogeneity and Fusion in Recommender Systems[2] (HetRec'11).

El conjunto de datos está accesible a través de las siguientes URLs:

http://ir.ii.uam.es/hetrec2011/datasets.html

http://www.grouplens.org/node/462

Y contiene:

- 855.598 *ratings* asignados por 2.113 usuarios a 10.197 películas
- 47.957 anotaciones asignadas a las películas por los usuarios; usando un total de 13.222 etiquetas (*tags*)
- Meta-información sobre las películas: títulos en español, años de estreno, directores, actores, géneros cinematográficos, localizaciones, URLs de imágenes de cartel
- Identificadores de las películas en los sistemas Internet Movie Database[3] (IMDb) y RottenTomatoes[4]

El fichero **user_ratedmovies.dat** contiene los perfiles de usuario basados en ratings, formados por tuplas [*u*, *r*, *i*], cada una de las cuales indicando que un usuario *u* asignó el rating *r* a la película *i*.

El fichero **user_taggedmovies.dat** contiene acciones de etiquetado de los usuarios, formados por tuplas [*u*, *t*, *i*], cada una de las cuales indica que un usuario *u* asignó la etiqueta *t* a la película *i*.

El fichero **movie_tags.dat** contiene las etiquetas más veces asignadas a las películas por usuarios de MovieLens, incluyendo más usuarios que los que aparecen en el conjunto de datos proporcionado.

---

[1] **MovieLens** movie recommender system
GroupLens research group, University of Minnesota, http://movielens.umn.edu

[2] **2nd Workshop on Information Heterogeneity and Fusion in Recommender Systems**
Information Retrieval Group, Universidad Autónoma de Madrid, http://ir.ii.uam.es/hetrec2011

[3] **Internet Movie Database** (IMDb), http://www.imdb.com

[4] **RottenTomatoes** movie reviews, http://www.rottentomatoes.com

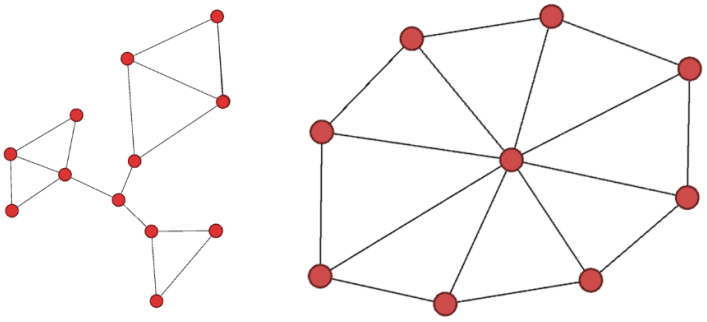
Análisis de redes sociales

En esta práctica se van a analizar un conjunto de redes sociales con diferentes características. El formato de grafo de esas redes sociales es un fichero separado por comas (CSV) que lista los arcos del grafo, mediante el identificador del nodo fuente y el identificador del nodo destino. Nótese que alguno de estos grafos usa identificadores no numéricos, por lo que es mejor usan un identificador de tipo String para la representación.

Grafos de ejemplo pequeños

En la práctica se proporcionan dos grafos de pequeño tamaño para poder comprobar el correcto funcionamiento de los algoritmos implementados. Estos grafos se adjuntan en **small.zip**.

En la siguiente figura se muestra la representación gráfica de ambos grafos. Se pide mostrar en la memoria los resultados de las métricas topológicas en ambos grafos. En small.zip se adjuntan las imágenes fuente para facilitar la presentación.



small 1 small 2

Grafos de colecciones reales

Se adjuntan dos grafos generados con colecciones reales:

- **Facebook4K:** 4000 nodos asociados a usuarios identificadores numéricos
- **Twitter10K:** 10000 nodos asociados a usuarios identificados con sus nombre de usuario en Twitter

Nótese que el grafo de Twitter es dirigido (la propiedad *follower* es asimétrica). Para simplificar se puede asumir que las relaciones son no dirigidas. Como parte opcional, se puede hacer la distinción de relaciones dirigidas y modificar los algoritmos necesarios para calcular las métricas de acuerdo a grafos dirigidos.

Grafos generados

Se deberán generar dos grafos siguiendo los algoritmos de Barabási y Erdös. El número de nodos utilizado queda a elección propia. Se recomienda utilizar las clases provistas en la librería JUNG[5]: `BarabasiAlbertGenerator` y `ErdosRenyiGenerator`.

Ejercicios

Ejercicio 1: Recomendación basada en contenido [2.5 puntos]

Se pide desarrollar en Java un método de recomendación basada en contenido. Sobre el conjunto de datos proporcionado, dicho método ha de utilizar perfiles de usuario y descripciones de películas representados mediante vectores de pesos numéricos asociados a **etiquetas sociales**.

Se tendrá que decidir el algoritmo para asignar pesos a las etiquetas de usuario y de película, así como el método de recomendación basada en contenido que determine la potencial "afinidad" de un usuario por una película.

La implementación es libre, pero se valorará que se haga un diseño de clases flexible y extensible. Se sugiere consultar la implementación de sistemas de recomendación del proyecto Apache Mahout[6], pero esto no implica que el diseño que se realice en la práctica deba ser tan detallado como el de Mahout.

El método de recomendación se deberá probar con etiquetas del conjunto de datos MovieLens-HetRec'11 descrito arriba. En concreto, se deberá probar con el fichero **user_taggedmovies.dat**. En función del método de recomendación desarrollado, podrían usarse también los ficheros **movie_tags.dat** y/o **user_ratedmovies.dat**.

[5] **JUNG**, Java Universal Network/Graph Framework, http://jung.sourceforge.net

[6] **Apache Mahout recommender documentation** https://cwiki.apache.org/confluence/display/MAHOUT/Recommender+Documentation

El resultado de este ejercicio constará de:

- Las **clases Java** asociadas al método de recomendación, incluyendo un **main** para prueba del mismo. Como mínimo, el main pedirá por línea de comandos el identificador de un usuario, y visualizará por pantalla tanto el perfil (películas etiquetadas y etiquetas asignadas) como las recomendaciones basadas en contenido generadas para ese usuario [2 puntos]
- Una sección en la **memoria** de la práctica donde se describa el algoritmo de establecimiento de pesos a las etiquetas de usuario y película, el método de recomendación, y (brevemente) el diseño/implementación desarrollados [0.5 puntos]

Ejercicio 2: Recomendación basada en filtrado colaborativo [2.5 puntos]

Se pide desarrollar en Java un método heurístico de filtrado colaborativo basado en usuario. Sobre el conjunto de datos proporcionado, dicho método ha de utilizar perfiles de usuario representados mediante vectores de ***ratings***.

Se tendrán que decidir las heurísticas con las que calcular similitudes entre perfiles de usuario y agregar dichas similitudes para determinar la "afinidad" de un usuario por una película. En la agregación se utilizará un tamaño de ***vecindario*** establecido previamente.

Al igual que en el ejercicio 1, la implementación es libre, pero de nuevo se valorará que se haga un diseño de clases flexible y extensible.

En este caso, el método de filtrado colaborativo se deberá probar con *ratings* del fichero **user_ratedmovies.dat**. En función del método de recomendación desarrollado, podrían usarse también los ficheros **movie_tags.dat** y/o **user_taggedmovies.dat**.

El resultado de este ejercicio constará de:

- Las **clases Java** asociadas al método de recomendación, incluyendo un **main** para prueba del mismo. Como mínimo, el main pedirá por línea de comandos el identificador de un usuario, y visualizará por pantalla tanto el perfil (películas evaluadas y ratings asignados) como las recomendaciones colaborativas generadas para ese usuario [2 puntos]
- Una sección en la **memoria** de la práctica donde se describa el modelo de recomendación, y (brevemente) el diseño/implementación desarrollados [0.5 puntos]

Ejercicio 3: Evaluación de recomendaciones basadas en filtrado colaborativo [2 puntos]

Se pide desarrollar en Java un programa que evalúe el método de filtrado colaborativo desarrollado en el ejercicio 2 computando las métricas MAE y RMSE, para varios tamaños de vecindario, y usando el 80% de los *ratings* de cada usuario para construcción del método de recomendación (entrenamiento, *training*) y el 20% de los ratings restantes para evaluación del mismo (*test*).

El resultado de este ejercicio constará de:

- Las **clases Java** asociadas a la evaluación del método de filtrado colaborativo [1.5 puntos]
- Una sección en la **memoria** de la práctica donde se describa (brevemente) la implementación desarrollada, y se reporten y analicen los valores de MAE y RMSE obtenidos para los diferentes tamaños de vecindario probados [0.5 puntos]

Ejercicio 4: Análisis de redes sociales [3 puntos]

Se pide desarrollar en Java las siguientes métricas topológicas de un nodo individual en un grafo:

- **Coeficiente de clustering de un nodo**
- **PageRank de un nodo**

la siguiente métrica para pares de vértices:

- **Arraigo (*embededness*) de los arcos**

y las siguientes métricas para grafos completos:

- **Coeficiente de clustering global**
- **Coeficiente de asortatividad (*assortativity*)**

Se sugiere que la implementación de las métricas se haga sobre la estructura de grafo de JUNG, aunque esto no es obligatorio; se permite usar estructuras de grafo propias o de otras librerías Java. El siguiente tutorial da la información necesaria para utilizar esta representación:

http://www.grotto-networking.com/JUNG/JUNG2-Tutorial.pdf

Con las clases desarrolladas, se pide calcular las métricas anteriores para las 6 redes sociales de la práctica.

- Para el coeficiente de clustering y PageRank de nodos se deberán generar dos ficheros **bmi1314_XXXX_p5_YY_nodeClustering.txt** y **bmi1314_XXXX_p5_YY_nodePageRank.txt** con los top 10 valores de las métricas ordenados de mayor a menor.

  El formato de una línea de fichero deberá ser `<red_social> <id_nodo> <valor_metrica>`, donde `red_social` será uno de los siguientes valores: `facebook4K`, `twitter10K`, `barabasi`, `erdos`, `small1`, `small2`, según corresponda.

- Para el arraigo de arcos se deberá generar un ficheros **bmi1314_XXXX_p5_YY_edgeEmbededness.txt** con los top 10 valores de la métrica ordenados de mayor a menor.

  El formato de una línea de fichero deberá ser `<red_social> <id_nodo1> <id_nodo2> <valor_metrica>`, donde `red_social` será uno de los siguientes valores: `facebook4K`, `twitter10K`, `barabasi`, `erdos`, `small1`, `small2`, según corresponda.

- Para los coeficientes de clustering y asortatividad de grafos se deberán generar dos ficheros **bmi1314_XXXX_p5_YY_graphClustering.txt** y **bmi1314_XXXX_p5_YY_graphAssortativity.txt** con los valores de las métricas para cada red social de la práctica.

  El formato de una línea de fichero deberá ser `<red_social> <valor_metrica>`, donde `red_social` será uno de los siguientes valores: `facebook4K`, `twitter10K`, `barabasi`, `erdos`, `small1`, `small2`, según corresponda.

El resultado de este ejercicio constará de:

- Las **clases Java** asociadas al cómputo de las métricas topológicas [2 puntos]
- Una sección en la **memoria** de la práctica donde se reporten los resultados volcados en los ficheros de texto y donde se represente y analice la distribución de grado (número de nodos por grado frente a grado) de las redes `facebook4K`, `twitter10K`, `barabasi` y `erdos` [1 punto]

Ejercicio opcional: Generación de modelos aleatorios de redes sociales [2 puntos extra]

Se pide desarrollar en Java un programa que genere un modelo aleatorio de red social. El modelo a desarrollar es de elección personal.

Se pide visualizar redes sociales generadas mediante la herramienta Gephi[7].

Para una visualización legible habrá que ajustar el modelo (número de nodos, densidad) y el *layout* de representación de grafo de Gephi.

El resultado de este ejercicio constará de:

- Las **clases Java** asociadas a la generación del modelo aleatorio [1.5 puntos]
- Una sección en la **memoria** de la práctica donde se incluyan imágenes de los grafos generados [0.5 puntos]

Calificación

Esta práctica se calificará con una puntuación de 0 a 10 (+ 2 puntos extra para la parte opcional) atendiendo a las puntuaciones individuales de ejercicios y apartados dadas en el enunciado. El peso de la calificación de la práctica en la calificación final de prácticas es del **25%**.

Entrega

La entrega de esta práctica consistirá en un fichero ZIP con el nombre **bmi1314_XXXX_p5_YY.zip**, donde XXXX debe sustituirse por el grupo 2461 ó 2462 según corresponda, e YY debe sustituirse por el número de pareja (01, 02, …, 10, …). Este fichero contendrá:

- Una carpeta **src** con todos los paquetes y ficheros fuente .java desarrollados.
- Los ficheros de texto generados en el ejercicio 4.
- El documento **bmi1314_XXXX_p5_YY_memoria.pdf** con las secciones especificadas en los ejercicios de la práctica.

Dicho fichero se enviará por el enlace habilitado al efecto en el curso **Moodle** de la asignatura.

[7] **Gephi**, The Open Graph Viz Platform, https://gephi.org,
Versión no instalable: https://launchpad.net/gephi/0.8/0.8.2beta/+download/gephi-0.8.2-beta.tar.gz