PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

Sistemas Recomendadores

Informe intermedio de proyecto

Desarrollo de framework de recomendación 'pylibrec'

Autor:

Integrante: ***Gabriel Sepúlveda Villalobos.***

- domingo 23 de octubre de 2016 -

1. Análisis exploratorio de los datos

Para evaluar el presente proyecto, se utilizará el conjunto de datos denominado *ml-100k*, el cual consiste en un conjunto de ratings de películas, los cuales fueron recolectados durante un período de siete mese desde el sitio web de *MovieLens*. A continuación se listan algunas de sus principales características:

- 100.000 ratings en escala de 1 a 5, de 943 usuarios sobre 1682 películas (ítems).
- Cada usuario ha evaluado al menos 20 películas.
- Contiene información demográfica de los usuarios: edad, género, ocupación y código postal.
- Contiene descripción completa de películas: título, fecha de lanzamiento, URL *IMDb* y género.

A continuación se presenta el análisis estadísticos realizado sobre el conjuntos de datos de entrenamiento, que serán utilizados para el establecimiento de modelos de recomendación. La siguiente tabla, resume los principales datos obtenidos del conjunto de datos de entrenamiento para predicciones de ratings.

Tabla 1.- Información estadística del conjunto de datos de entrenamiento.

Parámetro	Valor
Número de usuarios	943
Número de ítems	1.682
Número de ratings	100.000
Valor mínimo de rating	1
Valor máximo de rating	5
Paso entre valores de ratings	1
Densidad de matriz de ratings	0.06304732
Sparsity de matriz de ratings	0.9369527
Valor medio de número de ratings por usuario	106.0456 +- 100.9309
Valor medio de promedio de ratings por usuario	3.588051 +- 0.44511
Mayor cantidad de ratings por usuario	737
Identificador de usuario con más ratings	405
Mayor cantidad de ratings por ítem	583
Identificador de ítem con más ratings	50 (*Star Wars (1977)*)

Además de obtener estos datos estadísticos, gracias a la información adicional provista por el dataset, es posible complementar el análisis con información cualitativa. Como por ejemplo, para el caso del identificador de ítem con más ratings, fue posible saber que corresponde a la película *Star Wars*.

Para complementar estos datos, la siguiente figura muestra como se distribuyen los usuarios, en función del número de ítems evaluados por ellos.

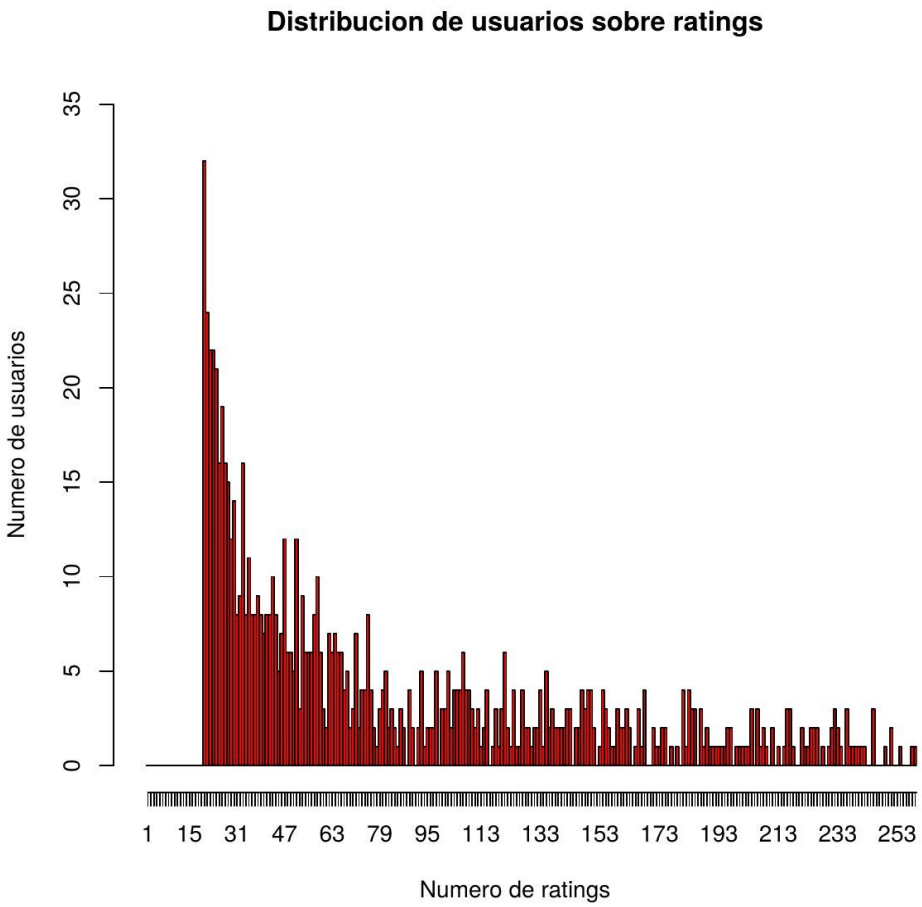

Figura 1. Distribución de usuarios sobre ratings del conjunto de datos de entrenamiento.

Al observar la figura 1, se aprecia un tipo de distribución característica para estas variables. Aquí se puede ver que la mayoría de los usuarios realiza pocas evaluaciones, concentrándose estos en las cantidades más bajas de número de ratings.

Además, tal como se señala en la descripción del dataset, se comprueba que todos los usuarios con menos de 20 ratings han sido eliminados del conjunto.

Por otra parte, la figura 2 presenta la distribución de ratings en función de los posibles valores que pueden ser asignados a cada ítem.

De ella se desprende que existe una alta concentración de votos con valores 3 y 4, lo cual es coincidente con el valor medio de promedio de ratings por usuarios, el cual tiene un valor de 3.58 como se presenta en la tabla 1.

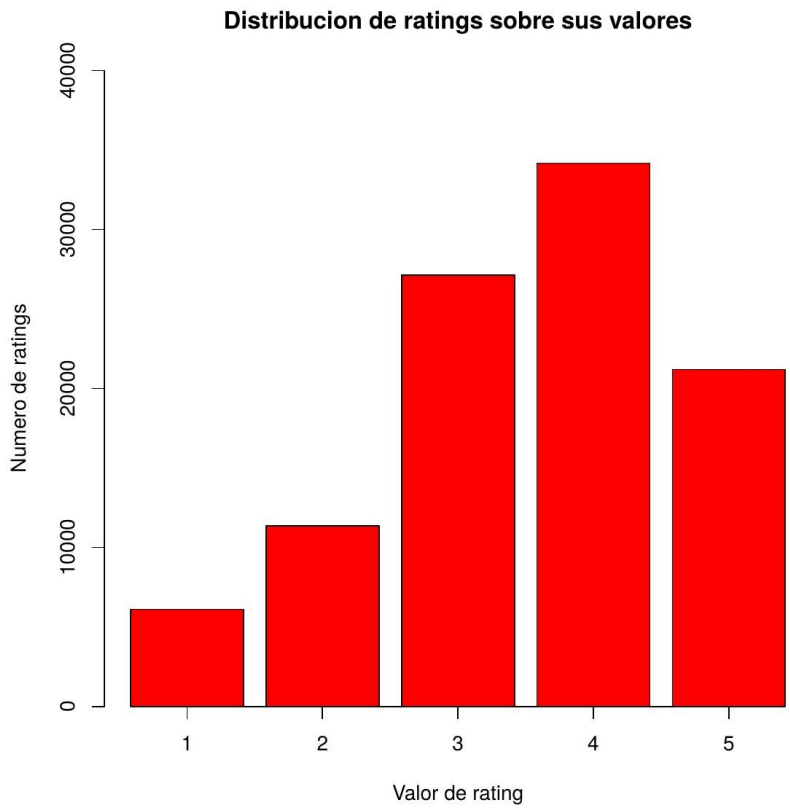

Figura 2. Distribución de ratings sobre sus posibles valores del conjunto de datos de entrenamiento.

2. Estado de avance

En esta sección, se exponen los principales avances en el desarrollo del proyecto que se encuentra en construcción, y que consiste en la elaboración de una biblioteca para sistemas de recomendación denominada *pylibrec*.

Al comenzar este proyecto, el primer paso fue plantear un diseño general con los módulos de software que debería componer la biblioteca, el cual permita describir la relación que ellos tienen entre sí.

Por esta razón, la siguiente figura presenta un diagrama con la estructura diseñada, y que actualmente se encuentra en desarrollo.

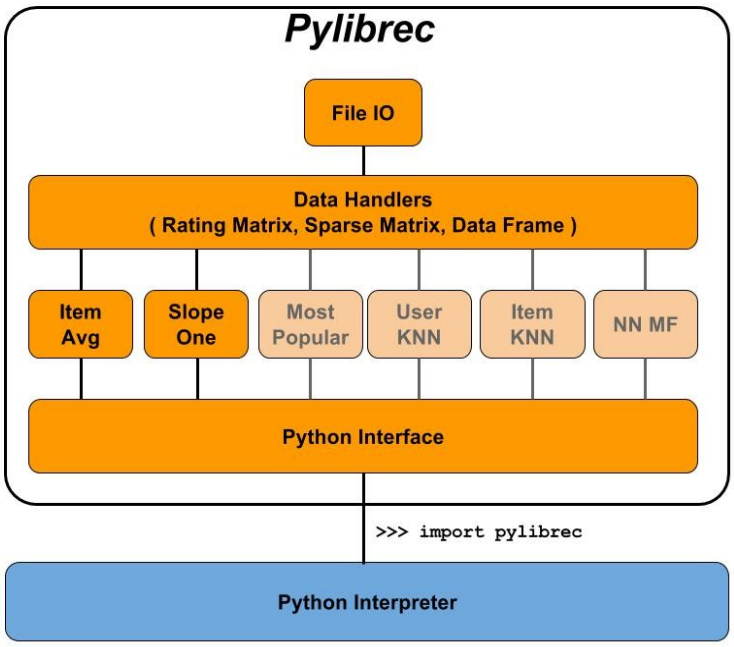

Figura 3. Diseño general de biblioteca *pylibrec*.

En primer lugar, se puede ver la existencia de una interacción directa entre *pylibrec* (bloque superior) con el intérprete de *Python* (bloque inferior azul), a través de la simple importación de un módulo.

Si enfocamos nuestra atención en el módulo superior, que representa la biblioteca *pylibrec*, se puede ver que en color naranjo, se muestran todos los componentes relevantes que se han propuesto como objetivo final de la implementación. De ellos, los cuadros de color más marcado (naranjo obscuro), son los que se encuentran desarrollados y en etapa de prueba. Por otra parte, los de color más suave (naranjo claro), representan los componentes cuya implementación no ha sido iniciada.

Como ya se mencionó en la propuesta, para maximizar el rendimiento de los algoritmos, la totalidad de los módulos han sido y serán implementados en lenguaje *C++*. En concordancia con esto, para poder

generar una biblioteca dinámica que pueda ser importada como módulo *Python*, se ha utilizado la *API Python/C*, la cual viene incluida en la instalación estándar de *Python*.

A continuación se presenta una descripción general de cada uno de los módulos implementados.

2.1 File IO

Este componente representa el punto de entrada y salida de datos, ya que permite la lectura de archivos que contienen el conjunto de entrenamiento y/o prueba, además de la escritura de predicciones de ratings y/o rankings, si así se desea.

Este otorga un buen nivel de flexibilidad en cuando a formatos de textos, debido a que sus campos pueden estar separados por cualquier tipo de carácter, el cual debe ser especificado por el usuario a través de la interfaz de alto nivel.

2.2 Data handlers

Este módulo contiene una serie de estructuras de datos, las que permiten tener un acceso homogéneo a los datos de ratings (independencia de formato de archivo), y con un mayor nivel de abstracción. Ellas serán directamente utilizadas por los algoritmos de recomendación que procesarán y generarán los datos de ratings y/o rankings para los pares usuario/ítem.

2.3 Item Average

Este módulo implementa el algoritmo de predicción de ratings *Item Average*, el cual asignará valores de ratings a los ítems, según el valor promedio de ratings recibidos por el ítem en cuestión en el pasado.

2.4 Slope One

Este contiene la implementación del algoritmo de *Slope One*, el cual realiza predicciones de ratings en base al cálculo de las desviaciones promedio entre varios ítems, con el fin de minimizar el error de predicción obtenido al utilizar un modelo lineal de pendiente uno.

2.5 Python Interface

En esta primera etapa de desarrollo, este módulo ha sido el que ha concentrado la mayor parte de los esfuerzos realizados, debido a que su definición es clave para establecer la factibilidad y utilidad de la propuesta. Además, su elaboración permite establecer con claridad los requerimientos, responsabilidades y capacidades que puede brindar cada componente interno de la biblioteca al usuario final.

Para su implementación, se ha utilizado la *API Python/C* provista por el paquete estándar de *Python*, la cual permite definir estructuras de bajo nivel para la interacción directa con el intérprete de *Python*. Con ella se podrán definir, tanto funciones al más puro estilo C, como clases de mayor complejidad, para permitir el manejo y creación de todo tipo de datos abstractos.

Con el trabajo realizado hasta ahora, es posible obtener una instancia de recomendación desde la biblioteca *pylibrec*, para generar predicciones de ratings utilizando los dos métodos especificados anteriormente.

Esto se puede realizar desde el intérprete de Python utilizando la *API* implementada, tal como se detalla a continuación.

- Recommender(algorithm, dataset, dlmchar = ‘,’, header = False, usercol = 0, itemcol = 1, ratingcol = 2)

  - *algorithm*: nombre de algoritmo que se desea instanciar: ‘*ItemAvg*’, ‘*SlopeOne*’.
  - *dataset*: nombre del archivo que contiene los datos de entrenamiento: *userId*, *itemId* y *rating* requeridos.
  - *dlmchar*: caracter delimitador de campos dentro del archivo dataset.
  - *header*: Indicador de existencia de linea de cabecera al comienzo del archivo dataset.
  - *usercol*: índice de posición donde se encuentra la columna de identificadores de usuario dentro del archivo dataset.
  - *itemcol*: índice de posición donde se encuentra la columna de identificadores de ítems dentro del archivo dataset.
  - *ratingcol*: índice de posición donde se encuentra la columna de ratings dentro del archivo dataset.
  - *Retorno*: instancia de clase *Recommender*, definida por los parámetros pasados en su construcción.

- Recommender.train()
  - Inicia procedimiento de entrenamiento de modelo de recomendación según los parámetros entregados en la construcción del objeto.

- Recommender.test(dataset, outputfile = None, dlmchar = ‘,’, header = False, usercol = 0, itemcol = 1, ratingcol = -1)
  - *dataset*: nombre del archivo que contiene los datos de prueba: *userId* y *itemId* requeridos.
  - *outputfile*: nombre de archivo de salida donde se desea que se escriba el resultado de las predicciones. De no ser especificado, ningún archivo será escrito y el resultado puede ser obtenido a través del valor de retorno del método.
  - *dlmchar*: caracter delimitador de campos dentro del archivo dataset.

- *header*: Indicador de existencia de linea de cabecera al comienzo del archivo dataset.
- *usercol*: índice de posición donde se encuentra la columna de identificadores de usuario dentro del archivo dataset.
- *itemcol*: índice de posición donde se encuentra la columna de identificadores de ítems dentro del archivo dataset.
- *ratingcol*: índice de posición donde se encuentra la columna de ratings dentro del archivo dataset. Este parámetro ha sido definido para uso futuro, y servirá como indicador para el cálculo de métricas de rendimiento.
- *Retorno*: lista de 3-tuplas que contienen el resultado de las predicciones de ratings realizadas sobre el dataset, y cuyos campos son (‘userId’, ‘itemId’, predicción).

◆ Recommender.predict(itemId, userId)
- *itemId*: identificador de ítem.
- *userId*: identificador de usuario.
- *Retorno*: predicción de rating para *itemId* y *userId*, según el modelo entrenado.

A continuación se presenta un ejemplo práctico de utilización de la biblioteca sobre el intérprete de *Python*.

```
#! /usr/bin/env python

import pylibrec

if __name__ == '__main__':

   mp = pylibrec.Recommender( 'ItemAvg',
                              '../ml-100k/u1.base',
                              dlmchar = '\t',
                              header = False,
                              usercol = 0,
                              itemcol = 1,
                              ratingcol = 2 )

   print 'training...'
   mp.train()

   print 'testing...'
   predlist = mp.test( '../ml-100k/u1.test',
                       dlmchar = '\t',
                       header = False,
                       usercol = 0,
                       itemcol = 1,
                       output_file = 'predictions.csv' )

   pred = mp.predict( '457', '443' )
   print 'user 457, item 443, prediction ' + str( pred )
```

Luego de su ejecución, se obtiene la siguiente salida por pantalla, y la generación del archivo de predicciones *predictions.csv*.

```
$ ./test.py
training...
testing...
user 457, item 443, prediction 3.7933883667
```

A continuación se presenta un extracto del archivo *predictions.csv*.

```
...
100,750,3.88119
100,752,3.35484
100,905,3.31818
100,990,3.07407
101,1034,2.7619
101,1047,2.81132
101,1051,3.27778
101,1057,2.33333
...
```

Para mayor detalle, esta biblioteca puede ser descargada desde el repositorio creado para su mantenimiento.

https://github.com/PUC-RecSys-Class/RecSys-Proyecto-Grupo15

3. Problemas encontrados

El problema más importante encontrado hasta ahora, fue el de elegir la herramienta a utilizar para crear la interfaz entre el código implementado en *C/C++*, y el intérprete *Python*.

Para su construcción, el primer paso fue evaluar algunas de las herramientas disponibles, entre las cuales se consideraron *Cython*, *ctypes* y *Python/C API*.

En primer lugar, *Cython* es un lenguaje de programación basado en *Pyrex*, que permite el desarrollo de extensiones en lenguaje *C*, pero utilizando una sintaxis similar a *Python*. Al realizar algunas pruebas con este método, se ve que por razones de portabilidad, el código fuente generado en C es demasiado extenso y confuso, lo cual resta legibilidad y control sobre el diseño de detalles. Estas razones motivan la exploración de nuevas alternativas.

Otra herramienta analizada fue c*type*, el cual a grades rasgos, consiste en un módulo de *Python* que permite importar bibliotecas dinámicas, para luego poder ejecutar las funciones que esta contiene a través de una interfaz del tipo objeto-método. Sin embargo, para poder cargar la biblioteca que se desea utilizar, se debe especificar su nombre y ubicación de forma explícita, lo cual lo hace poco transparente su uso al usuario final.

A continuación se plantea un ejemplo sencillo de su utilización.

```
#! /usr/bin/env python

import ctypes

if __name__ == '__main__':

   mylib = ctypes.CDLL( './library.so' )
   mylib.say_hello()
```

Finalmente, al evaluar la *API Python/C*, se encuentran todas las características deseadas, ya que permite la creación de funciones y estructuras de datos complejas, encapsuladas dentro de un módulo que es posible cargar de forma estándar desde el intérprete de *Python*. Además, permite mantener todo el procesamiento e interfaz en lenguaje *C/C++*, lo que representa un punto importante a la hora de buscar mejorar el rendimiento de los algoritmos.

Su único defecto en comparación con las otras dos herramientas analizadas, es su mayor complejidad a la hora de programar, lo cual se ve traducido en un mayor cuidado en el uso correcto de la memoria al administrar las referencias de los objetos utilizados. Sin embargo, la interfaz a desarrollar ha sido pensada como una entidad acotada, en la cual el espacio a errores debería minimizarse.

4. Plan de avance (actividades por realizar).

Dado que en esta primera etapa, el avance se ha concentrado mayormente en el desarrollo de interfaces entre lenguajes de programación, y en el establecimiento de la arquitectura de la biblioteca, las tareas por venir se enfocan con mayor fuerza en los algoritmos de recomendación y sus métricas.

A continuación, se presenta una lista de puntos con las tareas que falta por realizar, antes de liberar una primera versión oficial de la biblioteca, según los objetivos propuestos inicialmente.

- Implementación de generación de rankings.
- Implementación de generación de predicciones de ratings para algoritmos: *User KNN*, *Item KNN*, *Non-Negative MF*.
- Cálculo de métricas de rendimiento.
- Generación de paquete para instalación automática a través herramientas como *pip* o *easy_install*.
- Evaluación de biblioteca utilizando el dataset de *MovieLens*, y comparación de resultados y utilización de recursos, con los obtenidos por herramienta *librec*.