基于协同过滤的电影推荐算法

Table of Contents

[基于协同过滤的电影推荐算法 1](#_Toc528518305)

[系统说明 1](#_Toc528518306)

[算法介绍 3](#_Toc528518307)

[按照用户访问记录加权的协同过滤推荐算法 3](#_Toc528518308)

[功能介绍 3](#_Toc528518309)

[主要步骤 3](#_Toc528518310)

[基于用户电影评分的协同过滤推荐算法 4](#_Toc528518311)

[功能介绍 4](#_Toc528518312)

[主要步骤 4](#_Toc528518313)

[基于NMF非负矩阵分解的推荐算法 5](#_Toc528518314)

[功能介绍 5](#_Toc528518315)

[主要步骤 6](#_Toc528518316)

[基于Baseline的推荐算法 6](#_Toc528518317)

[功能介绍 6](#_Toc528518318)

[主要步骤 7](#_Toc528518319)

[基于Item评分的协同过滤算法个性化电影推荐系统 7](#_Toc528518320)

[功能介绍 7](#_Toc528518321)

[主要步骤 8](#_Toc528518322)

[数据说明 8](#_Toc528518323)

[三、结果说明 10](#_Toc528518324)

[四、代码说明 15](#_Toc528518325)

# 系统说明

主要完成了UserBased 与ItemBased两类推荐算法。包含：

1, 按照用户访问记录加权的协同过滤算法个性化电影推荐系统;

2, 按照用户评价加权的协同过滤算法个性化电影推荐系统;

3, 基于NMF非负矩阵分解的推荐算法个性化电影推荐系统;

4, 基于Baseline的推荐算法个性化电影推荐系统;

5, 基于Item评分的协同过滤算法个性化电影推荐系统.

针对用户输入的user id，系统会选择用户所选的算法进行推荐。在算法执行过程中会提示用户输入相应的参数，最后在Terminal中打印出电影推荐列表。

**使用方式**：

主程序为driver.py, 使用时Terminal中输入: python driver.py --help 查看帮助。python driver.py --method=N执行指定算法, N {1, 2, 3, 4, 5}.

**Requirements**:

\* python 2.7

\* numpy

\* scipy

\* pandas

**Framework**：

为了尽可能的低耦合（添加新的算法时不用修改其他已有的部分），保证系统能很容易的扩展，我们采用了如下的framework：



CFBase是所有算法的抽象模块，算法实现时都需要extends此类。除了实现CFBase中的3个主要接口外，在driver中的switch\_method中也要对应填入实现类以及补充对新增算法的描述，其他代码无需修改。

CFBase主要由4个接口组成：

1. initEnv：初始化环境，包含读取数据，用户录入所选算法的参数等。（可选）
2. compute：计算用户间的相似度矩阵或向量（必须实现）
3. recommend：计算用户想要获取的被推荐的电影（必须实现）
4. showResult：展现结果，一般是打印输出。已经提供默认实现，只需要按照格式传入参数即可使用。（可选，已提供默认实现）

下面分别对这5种算法进行介绍。

**符号定义：**

Rm: Rating matrix, 用户对电影的评分矩阵，每一行代表一个用户，每一列代表一部电影。

SM: Similiarty Matrix, 通过Rm计算得到的用户间的相似度矩阵或向量。

# 算法介绍

## 按照用户访问记录加权的协同过滤推荐算法

### 功能介绍

通过计算不同用户间的共同打分历史重叠个数得到不同用户的相似度。根据相似度推荐用户感兴趣的电影。

### 主要步骤

1. initEnv：按访问电影的记录进行协同过滤，数据集格式为M\*N用户评分矩阵。每一行代表某一用户的电影评分记录。每一列代表所有用户对该电影的评分。
2. compute：当试图为某一用户ID进行相关推荐时，按该用户与其他用户叠加重合程度由上向下进行权值排序，生成用户簇。用户间的相似度计算举例：评分矩阵Rm(Rating matrix)如下(每一行代表一个用户，每一列代表一部电影)：

那么与的相似度为 = (0, 1, 4, 2, 0, 0) dot\_and (3, 0, 1, 4, 0, 1) = 2. (dot\_and 由dot pruducts而来，本意是采用逻辑与的形式进行点乘)。

③recommend：在用户簇中由上而下抽取指定数目的相关用户，检索相关用户中还曾访问过哪些除了被检索用户曾经访问过的电影，依照用户簇中的权重对这些电影进行叠加，最终生成电影列表，即协同过滤推荐列表。

补充：对于整个评分矩阵V来说，可以采用公式：来计算不同用户之间的相似度矩阵。最终结果是一个M\*M的矩阵，矩阵中第i,j个元素代表user i与user j的相似度即共同打分过的电影。

对SM（SimMatrix）矩阵，用户, 参数n\_knn(number of k-nearest neighbors), 寻找的top k个近邻。

表示将的前k近邻向量累加得到每部电影的总权重。则，对用户推荐的电影集合为：

## 基于用户电影评分的协同过滤推荐算法

### 功能介绍

本算法在基于用户电影访问记录的算法的基础上进行改进的产物。我们试图改进出一种更为精确的推荐算法，于是在数据集中利用了上一个算法中没有利用到的参数——电影评分，在对用户簇进行相似度筛选计算时，不单单考虑电影间相同的访问记录，也同样将电影评分的相似程度纳入考查范围之内。这样即使两个用户之间的电影访问记录高度重叠，一旦他们对影片的评价相差甚远，两个用户的相似度也会有很大差别。

### 主要步骤

改进型的基于用户电影评分的协同过滤推荐算法主要原理如下：

1. initEnv：与按照用户访问记录加权的协同过滤推荐算法相同，获取用户的评分矩阵Rm。
2. compute：当试图为某一用户ID进行相关推荐时，获取该用户在Rm中对应的记录与其他用户逐一进行相似度计算，相似度的计算考量了叠加电影数目和评分两个维度，然后根据相似度由上而下进行权值排序，生成用户簇。
3. recommend：以后的步骤同上一个算法基本相同，均为根据权值对用户曾评价过的以外的电影进行叠加计算，最后生成电影推荐列表。

对于相似度的计算，我们采用了余弦相似度计算作为我们计算用户间相似程度的依据，公式如下图所示：

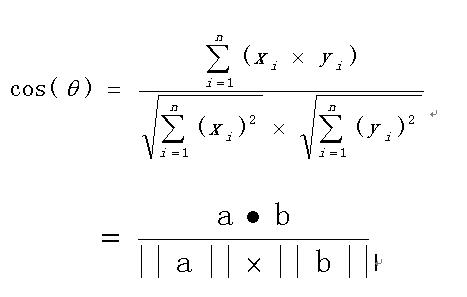


图1-1 余弦相似度计算公式

利用向量间两个夹角的余弦值作为衡量两个个体间差异大小的标准，余弦值越接近1，表示两个向量越接近0度，也就是两个向量越相似。

补充：系统同时提供三种相似度计算方式：Cosine，Pearson，Jaccard（这个方法用在此处是不是合适，后续还会分析）。

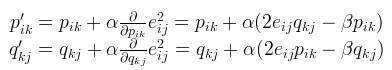
## 基于NMF非负矩阵分解的推荐算法

### 功能介绍

前两种算法的思路简单，实现也比较容易，但是会产生一种用户-评分数据过于稀疏的问题。即如果将用户-评分数据看做一个矩阵的话，这个矩阵存在大量的空值。因此，为了将空值部分用已有值估算出来，从而大幅提升推荐算法的精确性，PMF概率矩阵分解算法应运而生。我们在算法的研究过程中采用了PMF算法的分支——NMF非负概率矩阵分解算法，这个算法的核心思想是机器学习中的回归问题。它首先将一个N\*M维的用户-评分矩阵分为N\*K维的P矩阵与K\*M维的Q矩阵，K是一个参数，需要自行调节，在10~100之间。然后，定义一个风险参数，用于表示衡量矩阵分解的好坏程度，即：

http://static.oschina.net/uploads/space/2015/1007/001409_ZOCC_1451225.png

利用这个风险函数，通过梯度下降的方法，不断地减少损失值直到不能减少为止，即非缺失值的e(i,j)总和最小。相应的P，Q更新方式也就可以得出，其中：



这样就可以学习到新的P，Q矩阵，将P X Q可得到新的估算矩阵，从而将缺失的用户-评分填入了通过计算得到的估计值，大大增加了推荐算法的精确性。

### 主要步骤

1. initEnv：获取用户的评分矩阵Rm及对应参数
2. compute：分解Rm矩阵为P，Q。按照NMF算法进行梯度下降更新、计算eij、更新Pik和Qkj（保证非负）以及计算风险损失等操作；

②recommend：根据用户的user id，计算P矩阵中针对user id所在的行与Q  
矩阵中所有的列的点乘，返回用户要获取的降序排序(descending order)的推荐列表。

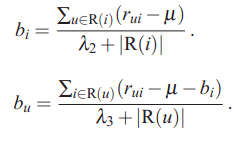
## 基于Baseline的推荐算法

### 功能介绍

在学习了解了NMF算法后，我们又学习并研究了另一种更加简单易行的baseline推荐算法。与NMF相比，baseline的一大显著特点就是算法简单容易实现，基本模型公式为：

http://static.oschina.net/uploads/space/2015/1007/003726_RonL_1451225.png

bui为baseline模型预估分数；μ为已知投票数据中投票的均值；bu为用户打分相对于平均值的偏差，可正可负，意义为用户u整体的兴趣偏好，偏差越高，表明该用户相对于其他用户对电影的评价越积极，同时更乐于接受其他用户评分高的电影；bi为该电影被打分时，相对于平均值的偏差，意义为该电影受欢迎的程度，偏差越高，表明该电影相对于其他电影评分越高，越受欢迎。在basline模型中，μ可以直接统计得到，而bi和bu可以利用



求得，式中u∈R(i)表示评价过电影i的所有用户，|R(i)|为其集合的个数，为用户u在电影i上的评分；同理，i∈R(u)表示用户u所评价过的所有电影，|R(u)|为其集合的个数。这个模型虽然简单，但其中已包含了用户的个性化信息与电影的个性化信息，面对大数据时反而有简单轻便的优点。

### 主要步骤

1. initEnv：获取Rm矩阵，，
2. compute：对数据矩阵进行迭代操作，从而按照baseline算法对μ值进行计算以及对bi和bu值进行更新等操作，最终得到SM（BUI）向量，代表用户user i对所有电影的打分。
3. recommend：将SM中用户未评分的前n部电影进行推荐

## 基于Item评分的协同过滤算法个性化电影推荐系统

### 功能介绍

前4中算法都为UserBased的算法实现，在计算用户间的相似度时面对用户量少的情况能够很快的得出结果，用户量多时则较为耗时。同时，对于用户评分记录较少的情况，由于背景知识不足，算法的推荐效果并不理想。与UserBased相对应，ItemBased推荐算法从Item出发，计算Item之间的相似度，之后根据用户打过分的Item，计算出与该Item最为相似的其他项，从而向用户推荐结果。当然，ItemBased在Item数量较多时也会存在计算量大的问题。

对于Item之间的相似度计算我们默认采用Pearson相关系数公式：

 其中表示用户u对物品i的打分，表示第i个物品打分的平均值. 通过以上公式，计算两两item之间的相似度，最终获得到一个M\*M的相似度矩阵IS(Item Similiarty matrix)，其中第i,j个元素值即为。

根据得到的IS矩阵，对用户u已打分的item的分数进行加权求和，权值为各个item与item i的相似度，然后对所有item相似度的和求平均，计算得到用户u对物品i打分：

其中为item i与item j的相似度，为用户u对item j的打分。

白话版：Rm评分矩阵如下（包含电影A, B, C, D, E, F， 用户u1, u2, u3, u4）：

想要计算u1对A的评分，

首先根据u1的历史记录计算A与其他电影的相似度. sim(A, B) =-0.69, sim(A, C) = -0.26, sim(A, D) = 0.41

然后计算A的评分：= (-0.69 \* 1 + -0.26 \* 4 + 0.41 \* 2) / |{B, C, D}| = - 0.3

当然，在这个例子中计算出来的是负数，说明用户可能对电影A不感兴趣。同时也说明计算相似度函数的选择是应具有解释性的。

对于相似度公式的具体应用，我们采用第三方工具包scipy.stats.stats#pearsonr(ui, uj).

### 主要步骤

1. initEnv：加载数据，获取评分矩阵Rm
2. compute：计算不同item之间的相似度矩阵IS
3. recommend：根据用户的打分历史，计算未打分的item，计算前n个推荐的item

# 数据说明

（一）数据来源

我们使用的电影数据为MovieLens数据集。这个数据集经常用来做推荐系统、机器学习算法的测试数据集，尤其在推荐系统领域，很多著名论文都是基于这个数据集的。同时，它也是某次具有历史意义的推荐系统竞赛所用的数据集。文件结构如下图所示：

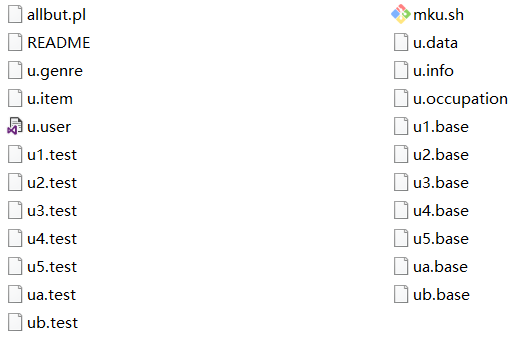


图2-1 MovieLens数据集文件结构

（二）规模

我们使用的数据集为MovieLens 100K规模的数据，即10万条评分数据，涵盖了1682部电影，以及943名评分者。如下图所示：

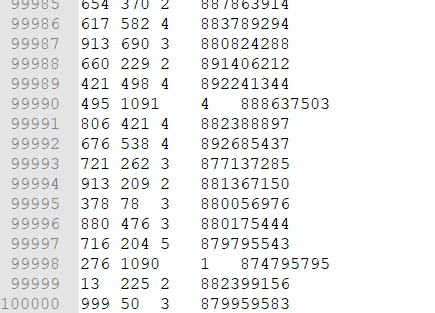


图2-2 评分记录共10万条

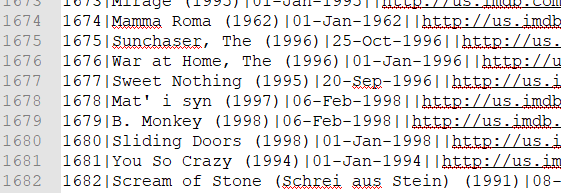


图2-3 涵盖电影共1682部

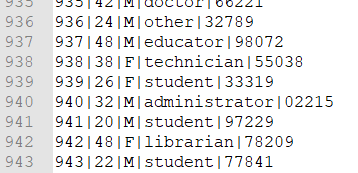


图2-4 涵盖评分者943名

由此可以说明，小组所选用的数据集规模足够庞大，足以支持本次小学期电影协同过滤推荐算法的应用，消除了因为数据规模较小所产生的偶然性和不确定性。

（三）数据格式

本数据集的数据格式均以纯文本形式展现，并通过制表符、换行符和|间隔符等，将文本形式的文件简单组合成类似于表格形式的展现。这样的数据格式使得读取与切片变得十分简单，降低了工作量。

（四）内容

数据集所包含的文件中，我们使用到的主要有u.data，u.item和u.user三个文件，其中u.data为用户对电影的评分记录（10万条），u.item为电影的相关信息（1682部），u.user为评分者的相关信息（943名），下面具体介绍这些文件中所包含的内容。

对于u.data，第一列显示的内容为评分者的ID号，第二列显示的内容为所评价的电影的ID号，第三列显示的内容为评分，第四列显示的内容为评分时的时间戳，如下图所示：

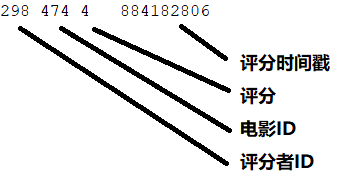


图2-5 u.data内容

对于u.item，第一列显示的内容为电影的ID号，第二列显示的内容为电影名称，第三列显示的内容为电影发行日期，第四列显示的内容为电影在imdb网站上所对应的链接，如下图所示：

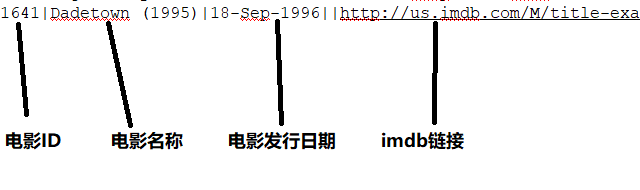


图2-6 u.item内容

对于u.user，第一列显示的内容为用户的ID号，第二列显示的内容为用户的年龄，第三列显示的内容为用户的性别，第四列显示的内容为用户的职业，第五列显示的内容为用户的邮政编码，如下图所示：

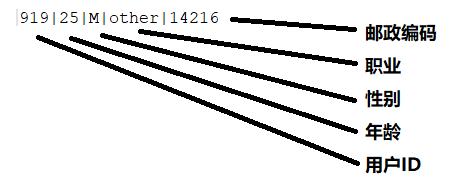


图2-6 u.user内容

三、结果说明

（一）基于用户电影访问记录的协同过滤推荐算法

执行程序，进入主界面，提示输入要查询的用户ID，如图所示：

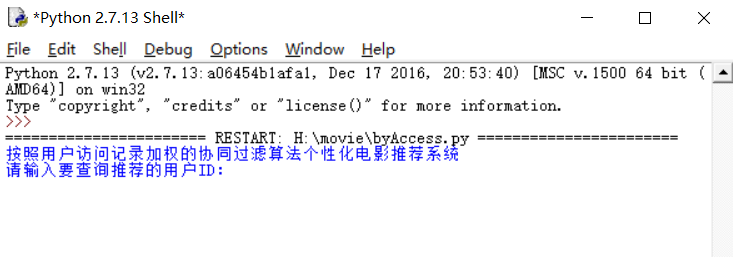


图3-1 主界面下提示输入要查询的用户ID

在主界面下输入要查询的用户ID后按下回车，提示输入要列入计算的相关用户数量，数量越多结果越精准，但计算量也越大，如图所示：

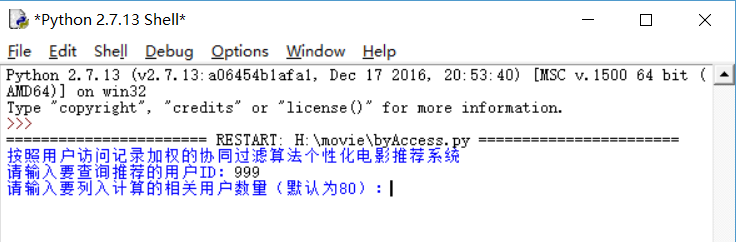


图3-2 程序提示输入要列入计算的相关用户数量

输入列入计算的相关用户数量后按下回车，提示输入要显示的推荐结果数量，可根据喜好输入，如图所示：

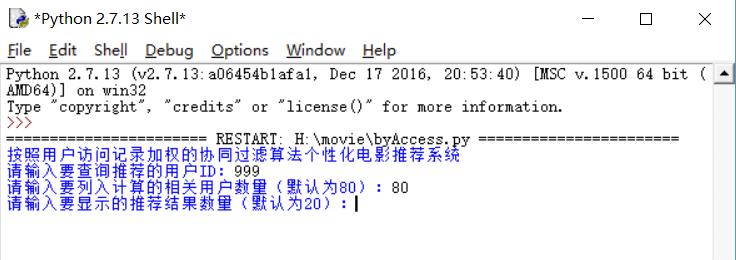


图3-3 程序提示输入要显示的推荐结果数量

输入要显示的推荐结果数量后按下回车，显示要查询的用户所访问的电影和推荐的电影结果，如图所示：

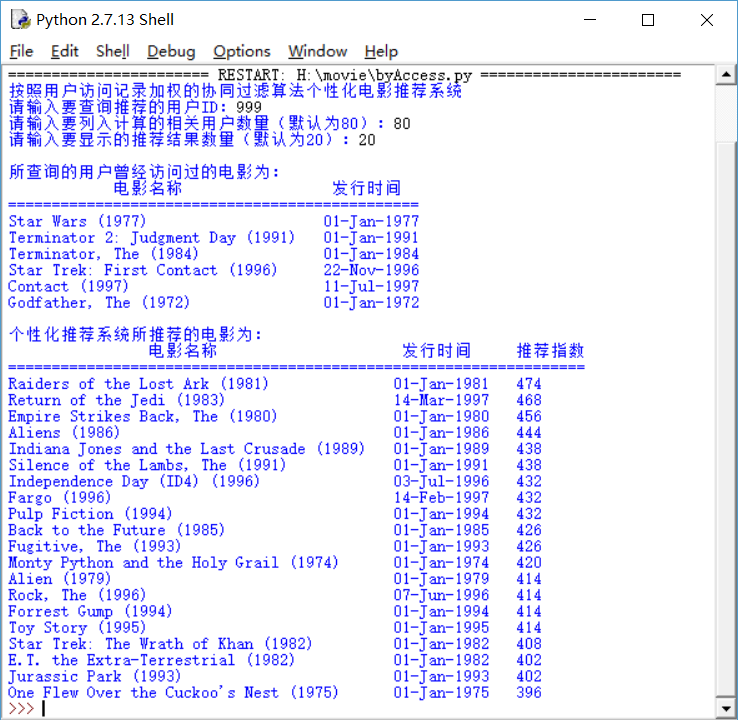


图3-4 程序显示要查询的用户所访问的电影和推荐的电影结果

我们对程序运行结果进行分析时，特意选取了喜好观看动作类、冒险类、科幻类电影的用户作为数据样本，用户曾观看的电影有《星球大战》、《终结者》、《星际迷航》等电影，而电影推荐结果中的大部分电影同样是这样类型的影片，且知名度较高，因此在这个方面上可以印证，推荐算法的设计总体上是成功的。

（二）基于用户电影评分的协同过滤推荐算法

由于本算法是在上一个算法的基础上进行修改的，因此界面运行结果与上一个算法相似，只是在相同数据下最终输出的电影推荐结果有一定的不同。首先，执行程序，进入主界面，提示输入要查询的用户ID，如图所示：

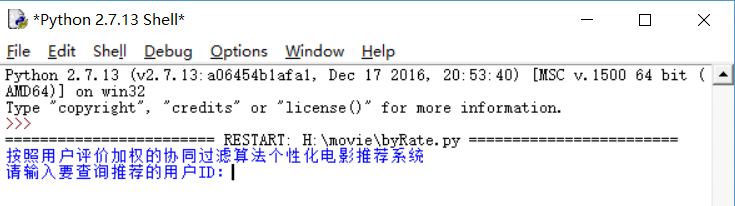


图3-5 主界面下提示输入要查询的用户ID

在主界面下输入要查询的用户ID后按下回车，提示输入要列入计算的相关用户数量，数量越多结果越精准，但计算量也越大，如图所示：

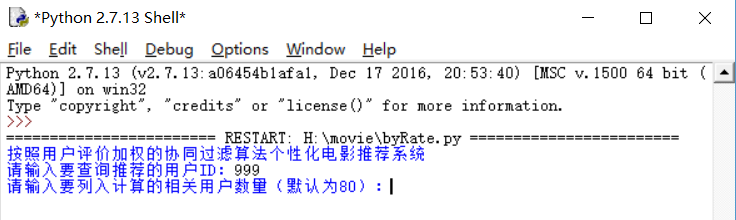


图3-6 程序提示输入要列入计算的相关用户数量

输入列入计算的相关用户数量后按下回车，提示输入要显示的推荐结果数量，可根据喜好输入，如图所示：

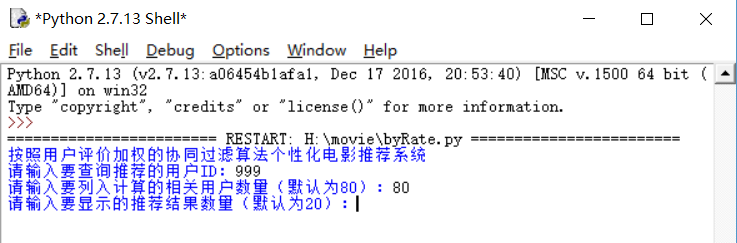


图3-7 程序提示输入要显示的推荐结果数量

输入要显示的推荐结果数量后按下回车，显示要查询的用户所评价的电影和推荐的电影结果，如图所示：

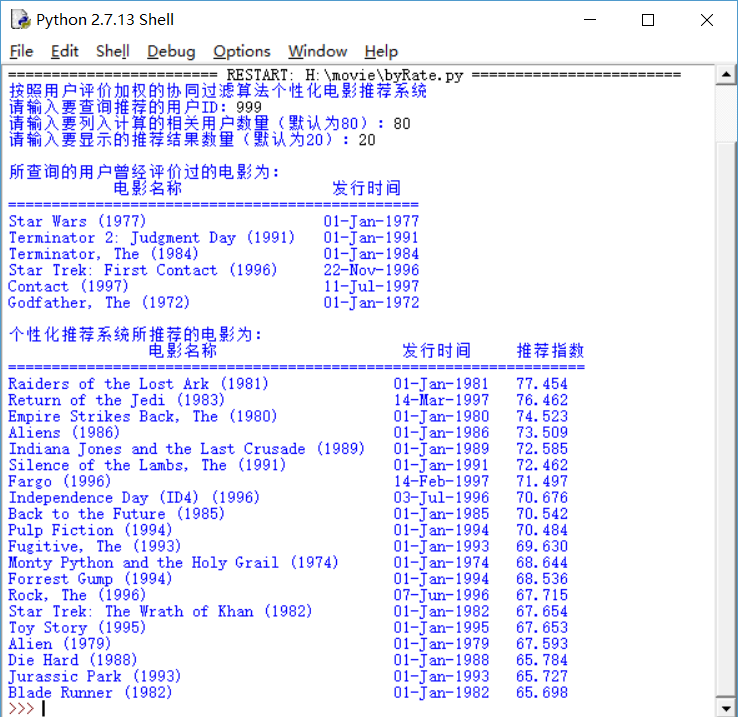


图3-8 程序显示要查询的用户所评价的电影和推荐的电影结果

对比算法（一）和算法（二）的运行结果可以发现，新旧两版算法的推荐结果在前几个结果时几乎完全一致，即使是后面的结果，也有相当大的重复几率。这可能是因为我们所使用的数据集有10万条记录，可以充分进行计算与评估，避免了小数据集的偶发性与不确定性；另一方面，由于两个数据算法在推荐结果上的高度一致性，因此也可以使两个算法互相印证对方的有效性，即算法设计在总体上是成功的。

（三）基于NMF非负矩阵分解的推荐算法

首先运行程序，在JAVA的控制台下会出现可输入文本界面，如图所示：

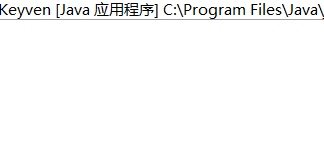


图3-9 JAVA控制台下的文本输入界面

在文本输入界面中输入我们用来测试的用户-评分矩阵5 3 0 1 ; 4 0 0 1 ; 1 1 0 5 ; 1 0 0 4 ; 0 1 5 4，其中0代表空白的用户-评分记录，按下回车后在控制台输出了原矩阵与经过NMF计算的含预测值的矩阵，如图所示：

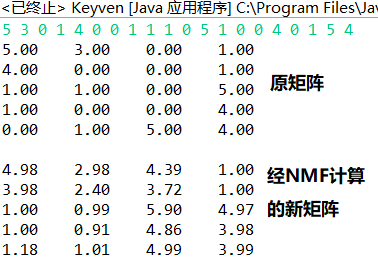


图3-10 NMF输出结果

根据输出结果可以发现，NMF算法在保留了原有既有数据的基础上，对矩阵中为0值的位置进行了NMF计算，并填入了预测值。

（四）基于Baseline的推荐算法

首先运行程序，在JAVA的控制台下会出现可输入文本界面，如图所示：

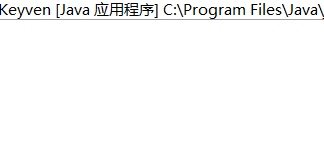


图 3-11 JAVA控制台下的文本输入界面

在文本输入界面中输入我们用来测试的用户-评分矩阵5 3 0 1 ; 4 0 0 1 ; 1 1 0 5 ; 1 0 0 4 ; 0 1 5 4，其中0代表空白的用户-评分记录，按下回车后在控制台输出了经过baseline计算的含预测值的矩阵，如图所示：



图3-12 baseline输出结果

通过NMF算法与baseline算法输出结果的对比，可以发现二者的输出结果基本相近，baseline算法的输出结果相对更加稳定，波动较小；而NMF算法所得预测值的波动较大，稳定性较差。

（五）基于Item评分的协同过滤算法个性化电影推荐系统

程序运行图：



图3-13 ItemBased运行图

python driver.py --method=5 运行item-based方法，根据提示依次输入用户ID，要显示的推荐结果数量以及采用的相似度计算方式获取最终结果。

四、代码说明

代码已在项目中体现，此处略。