

**NANCHANG UNIVERSITY**

**数据挖掘课程设计报告**



**题 目:** 朱文星

**学 院：** 软件学院

**班 级：** 信安153班

**学 号：** 8001715103

**姓 名：** 朱文星

**起讫日期：** 2017. 12.11— 2018. 1. 16

**任课教师：** 刘伯成 职称： 讲师

**部分管主任：** 邹春华

**完成时间：** 2017.1.8

**目 录**

一、 项目需求分析 - 19 -

**1.1项目介绍** - 19 -

**1.2功能需求** - 19 -

二、 系统分析与设计 - 19 -

**2.1本程序需解决的关键技术问题** - 19 -

**2.2项目流程** - 19 -

**2.3功能模块** - 19 -

三、 项目设计与实现 - 20 -

**3.1工程文件组织结构** - 20 -

**3.2项目设计** - 20 -

**3.3项目实现** - 20 -

五、 设计日志 - 21 -

六、 个人小结 - 21 -

七、 参考文献 - 22 -

# 项目需求分析

## **1.1项目介绍**

本项目是建立在数据挖掘和Apriori算法基础上对电影推荐系统的实现，采用pandas加载数据集，代码采用python语言，简洁明了的实现了Apriori算法和关联规则的抽取，整个过程分为两大阶段。

首先，借助Apriori算法寻找数据中的频繁项集。

然后，根据找到的频繁项集，生成关联规则。由于数据集较大，Apriori算法就很有必要，并且由于时间复杂度呈指数级增长，我们需要寻找更巧妙的解决方案。这种现象在数据挖掘中很常见，虽然可以用最笨的方法穷尽所有情况来解决问题，但在数据量很大的情况下必须要使用更灵活、更快捷的算法。

所以，我们用一部分数据作为训练集以发现关联规则，在剩余数据——测试集上进行测试，寻找出最合适的关联规则。

## **1.2功能需求**

产品推荐技术是门大生意。网店经常用它向潜在用户推荐他们可能购买的产品。好的推荐算法能带来更高的销售业绩。每年有几百万乃至几千万用户进行网购，向他们推荐更多的商品，潜在收益着实可观。

产品推荐问题被人们研究了多年，但它一直不温不火，直到2007年到2009年间，Netflix公司推出数据建模大赛，并设立Netflix Prize奖项之后，才得到迅猛发展。该竞赛意在寻找比Netflix公司所使用的预测用户为电影打分的系统更准确的解决方案。最后获奖队伍以比现有系统高10个百分点的优势胜出。虽然这个改进看起来不是很大，但是Netflix公司却能借助它实现更精准的电影推荐服务，从而多赚上百万美元。

# 项目分析与设计

## **2.1本项目需解决的关键技术问题**

Apriori算法的实现，抽取关联规则。

## **2.2项目流程**

### 获取数据集

import os

import pandas as pd

data\_folder = os.path.join(os.path.expanduser("~"), "Data",

"ml-100k")

ratings\_filename = os.path.join(data\_folder, "u.data")

确保ratings\_filename指向解压后得到的文件夹中的u.data文件。

### 用pandas加载数据集

加载数据集时，把分隔符设置为制表符，告诉pandas不要把第一行作为表头（header=None），设置好各列的名称。代码如下：

all\_ratings = pd.read\_csv(ratings\_filename, delimiter="\t",

header=None, names = ["UserID", "MovieID", "Rating", "Datetime"])

虽然本章用不到，还是稍微提一下，你可以用下面的代码解析时间戳数据。

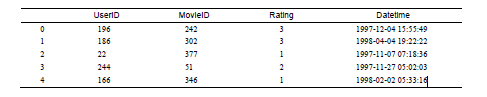
all\_ratings["Datetime"] = pd.to\_datetime(all\_ratings['Datetime'],

unit='s')

运行下面的代码，看一下前五条记录。

all\_ratings[:5]

输出结果如下。



### Apriori算法的实现

Apriori算法是亲和性分析的一部分，专门用于查找数据集中的频繁项集。基本流程是从前一步找到的频繁项集中找到新的备选集合，接着检测备选集合的频繁程度是否够高，然后算法像下面这样进行迭代。

(1) 把各项目放到只包含自己的项集中，生成最初的频繁项集。只使用达到最小支持度的项目。

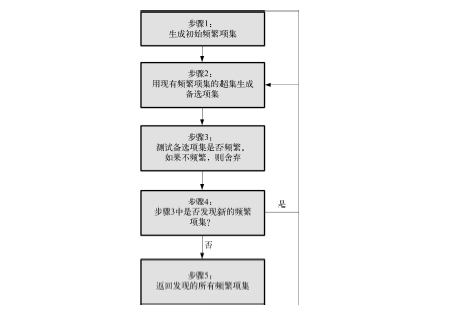
(2) 查找现有频繁项集的超集，发现新的频繁项集，并用其生成新的备选项集。

(3) 测试新生成的备选项集的频繁程度，如果不够频繁，则舍弃。如果没有新的频繁项集，就跳到最后一步。

(4) 存储新发现的频繁项集，跳到步骤(2)。

(5) 返回发现的所有频繁项集。

整个过程表示如下。



### 抽取关联规则

Apriori算法结束后，我们得到了一系列频繁项集，这还不算是真正意义上的关联规则，但是很接近了。频繁项集是一组达到最小支持度的项目，而关联规则由前提和结论组成。我们可以从频繁项集中抽取出关联规则，把其中几部电影作为前提，另一部电影作为结论组成如下形式的规则：如果用户喜欢前提中的所有电影，那么他们也会喜欢结论中的电影。

每一个项集都可用这种方式生成一条规则。

## **2.3功能模块**

1. 获取数据集
2. 用pandas加载数据集
3. Apriori算法的实现
4. 抽取关联规则
5. 将剩余数据与关联规则进行评估比较

# 项目设计与实现

## **3.1工程文件组织结构**

DataDig.py MovieRecommend.py

## **3.2项目设计**

利用python IDE环境进行编程：

具体分为以下几个步骤：

**1、获取数据集**

import os

import pandas as pd

data\_folder = os.path.join(os.path.expanduser("~"), "Data",

"ml-100k")

ratings\_filename = os.path.join(data\_folder, "u.data")

确保ratings\_filename指向解压后得到的文件夹中的u.data文件。

**2、用pandas加载数据集**

加载数据集时，把分隔符设置为制表符，告诉pandas不要把第一行作为表头（header=None），设置好各列的名称。代码如下：

all\_ratings = pd.read\_csv(ratings\_filename, delimiter="\t",

header=None, names = ["UserID", "MovieID", "Rating", "Datetime"])

虽然本章用不到，还是稍微提一下，你可以用下面的代码解析时间戳数据。

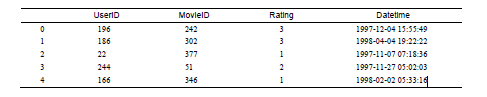
all\_ratings["Datetime"] = pd.to\_datetime(all\_ratings['Datetime'],

unit='s')

运行下面的代码，看一下前五条记录。

all\_ratings[:5]

输出结果如下。



**3、Apriori算法的实现**

Apriori算法是亲和性分析的一部分，专门用于查找数据集中的频繁项集。基本流程是从前一步找到的频繁项集中找到新的备选集合，接着检测备选集合的频繁程度是否够高，然后算法像下面这样进行迭代。

(1) 把各项目放到只包含自己的项集中，生成最初的频繁项集。只使用达到最小支持度的项目。

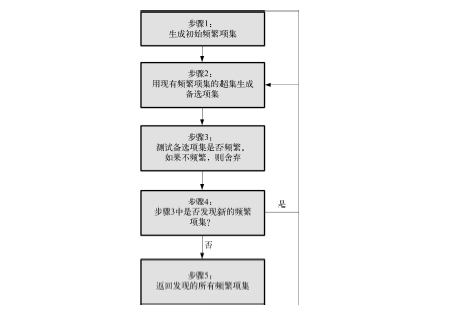
(2) 查找现有频繁项集的超集，发现新的频繁项集，并用其生成新的备选项集。

(3) 测试新生成的备选项集的频繁程度，如果不够频繁，则舍弃。如果没有新的频繁项集，就跳到最后一步。

(4) 存储新发现的频繁项集，跳到步骤(2)。

(5) 返回发现的所有频繁项集。

整个过程表示如下。



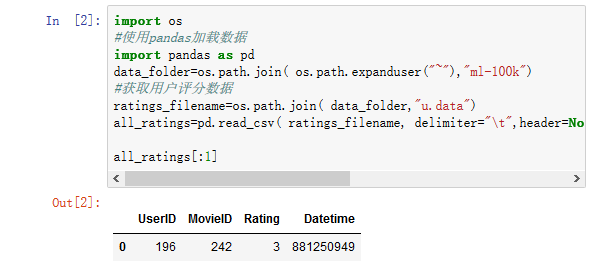
**4、抽取关联规则**

Apriori算法结束后，我们得到了一系列频繁项集，这还不算是真正意义上的关联规则，但是很接近了。频繁项集是一组达到最小支持度的项目，而关联规则由前提和结论组成。我们可以从频繁项集中抽取出关联规则，把其中几部电影作为前提，另一部电影作为结论组成如下形式的规则：如果用户喜欢前提中的所有电影，那么他们也会喜欢结论中的电影。

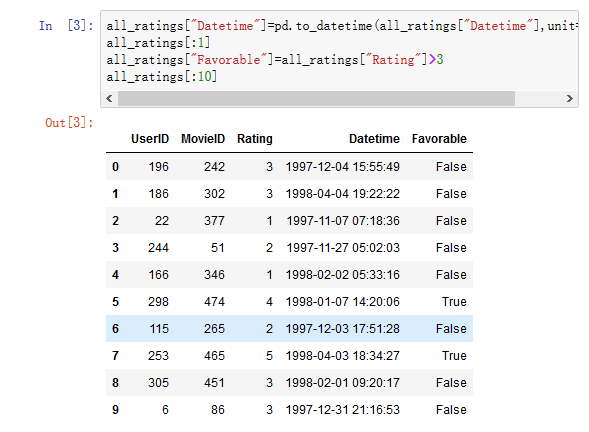
每一个项集都可用这种方式生成一条规则。

## **3.3项目实现**

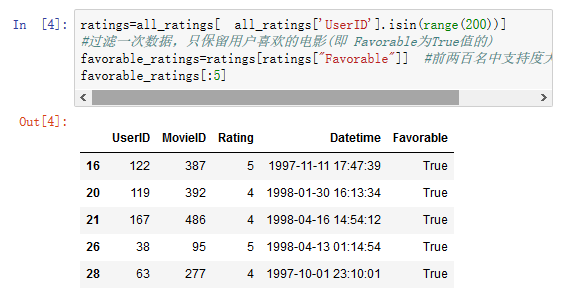
### 获取数据集



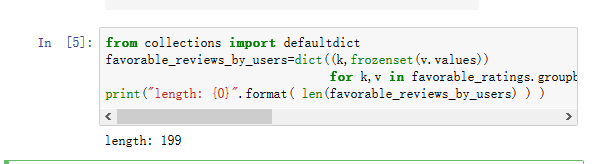
### 新增一列，用来存用户对某个电影是否喜欢 ( 如果评分大于3)



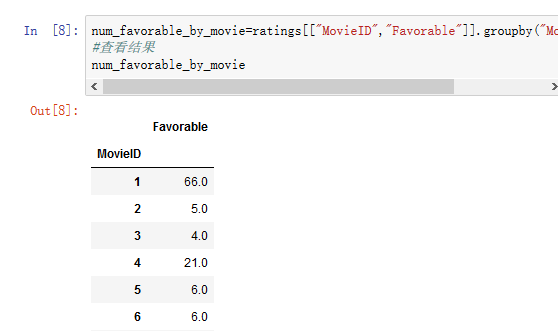
### 从数据集中取前200名用户的打分数据作训练集



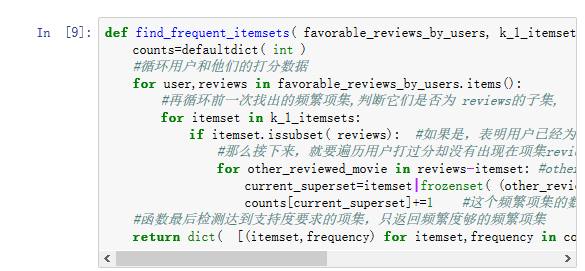
### 因为要生成频繁项集，所以我们只对打分超过一次的用户感兴趣，所以按照UserID分组，并遍历每个用户看过的每一部电影，存到一个字典中



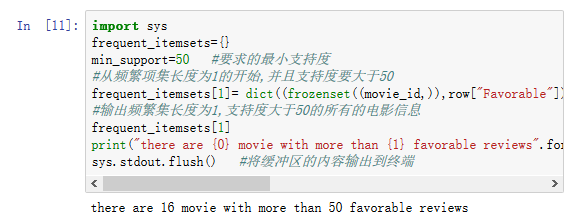
### 再创建一个数据框，存入每部电影评价分为3分以上的人数( 即 Favorable为True)的数量



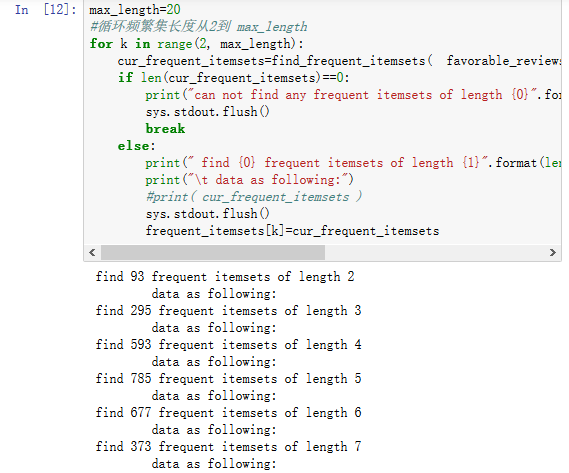
### 创建一个函数，它接收新发现的频繁项集，创建超集，检测频繁程度

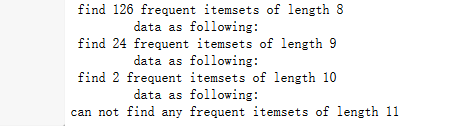


### 创建一个字典，存不同长度的频繁项集，从频繁项集长度为1的开始,并且支持度要大于50。

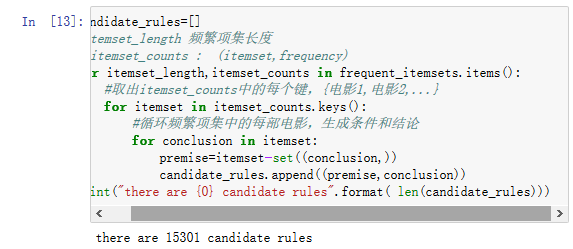


### 定义要找的频繁集的最大长度，循环频繁集长度从2到 max\_length。

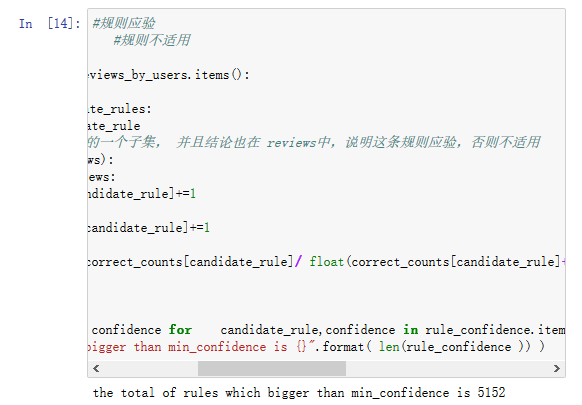




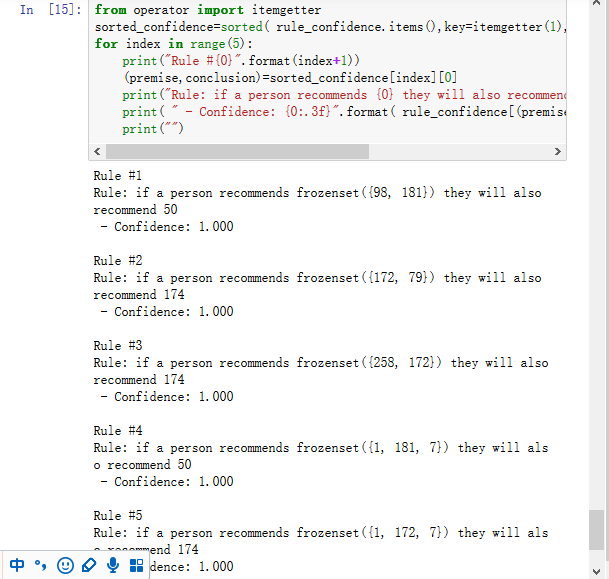
### 计算关联规则的数目



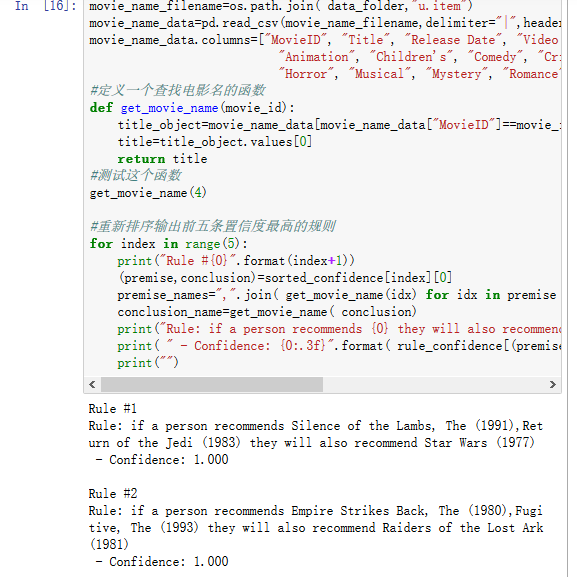
### 计算每条规则的置信度，先用两个字典存规则应验, 规则不适用数量，计算满足置信度的规则

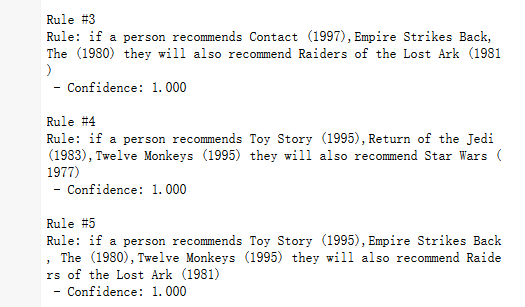


### 输出满足置信度的前五条规则

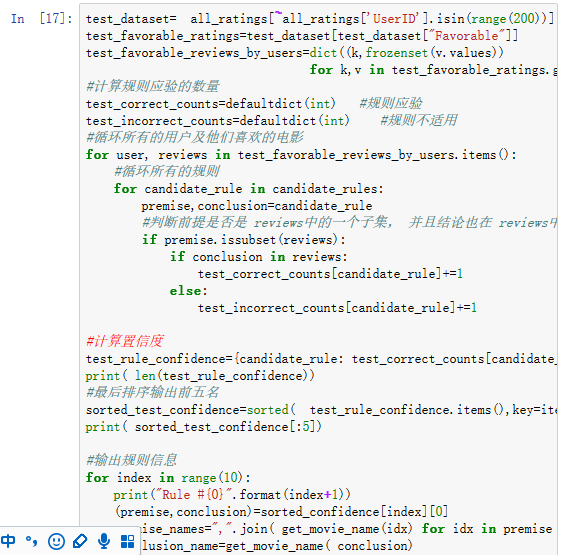


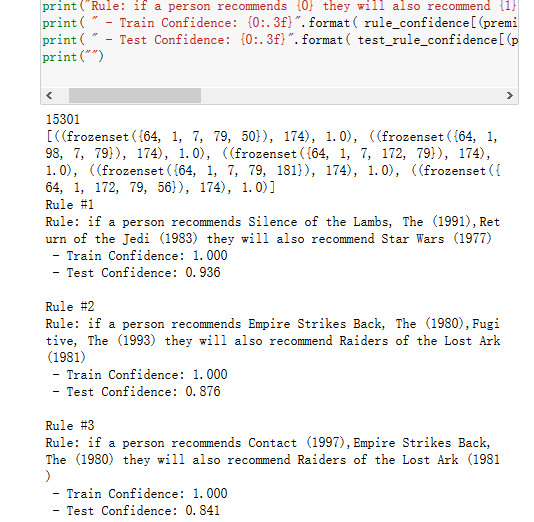
### 加载电影的名字，重新排序输出前五条置信度最高的规则

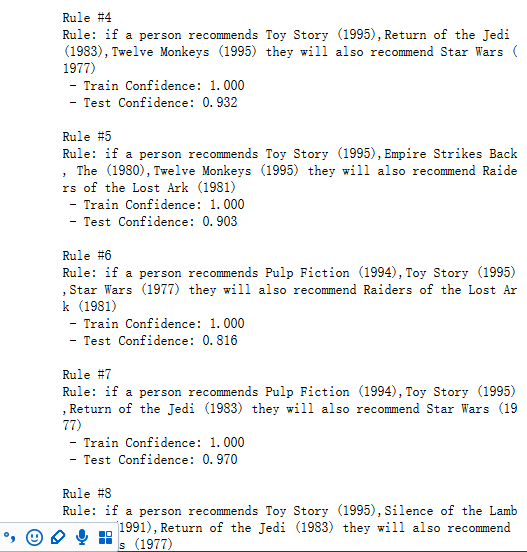


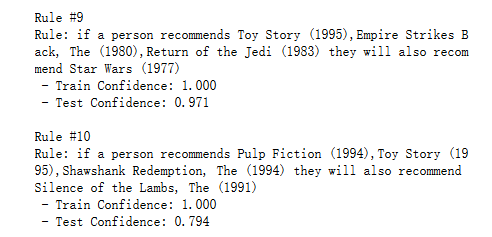


### 评估: 寻找最好的规则，抽取所有没有用于训练的数据作为测试集, 训练集数据用了前200名用户的打分数据，测试集用其它的数据即可









# 项目拓展

## **4.1 拓展目的**

1.为了更好地体会电影推荐系统的思想

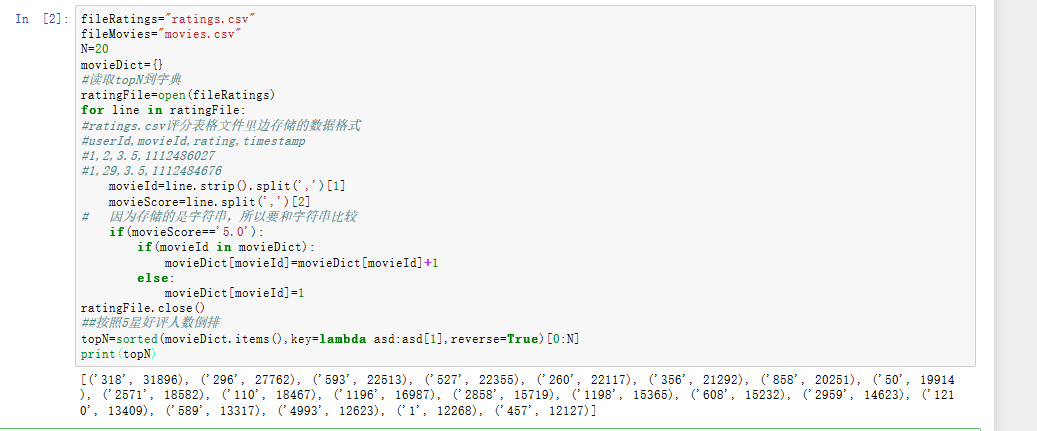
2.找出ml-20m.zip数据集中的五星好评最多的电影。

## **4.2 拓展数据**

本次电影推荐用的数据集是ml-100k.zip,所以我进行项目拓展是对ml-20m.zip包进行了代码设计，用到了movies.csv和ratings.csv文件。

## **4.3 拓展步骤**

### 1.根据movieId和movieScore求出五星好评最多的电影。



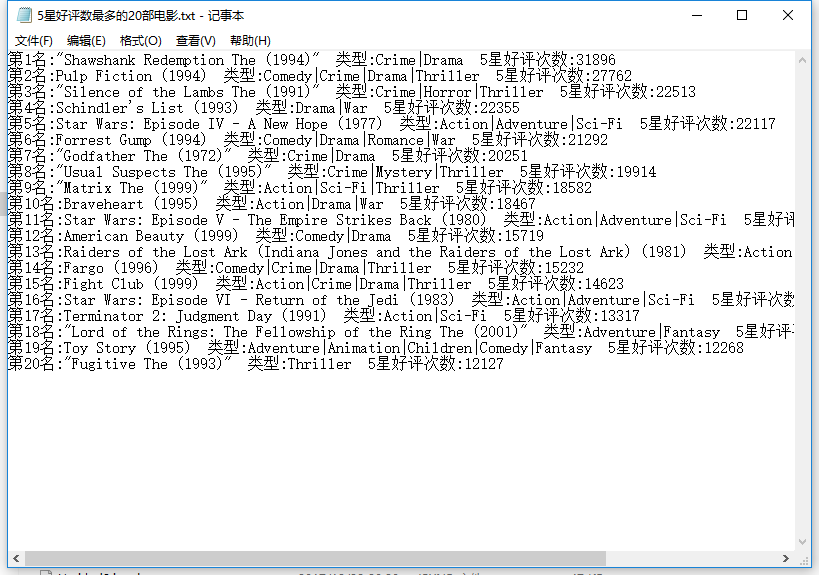


### 根据id获取电影名称





### 存入txt文件



## **4.4 拓展结果分析**

就像电影推荐系统一样，电影的评分同样也是人们关注的重点，评分高低有时直接影响了观众的观看与否，这也便捷了观众筛选电影，节省了观众的时间，还提高了观众看电影的热情，这也是为什么豆瓣评分如此火爆的原因之一。同时电影评分的高低，也反映了了一部电影，一个导演，一群演员的演技的好坏，而五星好评的多少明显决定了电影评分的高低，所以电影的评分的数据挖掘和电影推荐系统的数据挖掘一样重要。

# 个人小结

刚开始对数据挖掘的印象就是，学长学姐口中的一门实训课，就连平常做作业来说，因为平常的概念太深奥，就会觉得无法应用到实际生活中，但是通过此次的大作业，我现在觉得数据挖掘是一门很有用的学科，我发现它不仅和人工智能接上了轨，而且对python各种库的开发也做到了极致，此次我做的仅仅是一个电影推荐系统，但是却让我立刻想到，原来数据挖掘就存在我们生活的方方面面，举个最简单的例子，现在我们只要看新闻，看电影，甚至网上购物，相应的网站或者APP就会大量推送你最近浏览过或购买过的同类网页或商品，所以通过这次大作业，我受益颇多，但可能是因为我先前的不重视和并没有系统学过python的类库学习，学习起来有些吃力，此次电影推荐系统本来想将其应用到网页当中，但实在因为时间的匮乏和相关知识的不足，而无法进行更深一步应用。

既然无法进行纵向扩展，我就进行了横向扩展，对与电影推荐系统同等重要的电影评分系统进行了拓展，采用的是另一个数据集，求出其五星好评数排名前二十的电影，也就是反映出电影评分的高低。

但是知识的匮乏和时间的不足，让我无法将其应用的实际生活中，但是以后我一定会更加努力的学习这方面的内容，毕竟现在人工智能已经越来越普及，未来将是人工智能的未来。

# 参考文献

Robert Layton【澳】.Python数据挖掘入门与实践. 北京：人民邮电出版社，2016.7