数据挖掘实验
Netflix Prize

小组成员

高志伟　2120150985

王子硕　2120151044

刘云洋　2120151016

张露露　2120151064

I. 项目背景

A. Netflix 简介

Netflix 是一家在线影片租赁提供商。公司能够提供 Netflix 超大数量的 DVD，能够让顾客快速方便的挑选影片，同时免费递送。Netflix 已经连续五次被评为顾客最满意的网站。可以通过 PC、TV 及 iPad、iPhone 收看电影、电视节目。NETFLIX 成功的秘诀其实就是一个小小的创意：租一次，随便看；不限租期。1997 年，NETFLIX 首创这种商业模式，用户支付固定的月租费，在 NETFLIX 网站上选择自己想看的电影之后，这些电影光碟就会寄到你家，收到之后你爱看多久就看多久，没有期限，等到看满意后再用 NETFLIX 附上的免费信封寄回，然后下一张电影光碟就会接着寄来。

B. Netflix 大赛

NETFLIX 设计开发了一个电影推荐系统叫 Cinematch[1]，这是一个智能预测系统，它能够根据用户以前的评分数据预测到这位顾客可能喜欢什么样的主题和风格，等新电影发行后马上为相应的用户群 体进行推荐。可是这个推荐系统的智能水平有限，准确度不高，不能令人满意。

采用传统的方法，NETFLIX 可以雇用新的计算机软件设计人员，改进已有的 Cinematch 系统，或者另起炉灶，开发新的电脑软件系统。但是，NETFLIX 使用了更好的方法。

2006 年， NETFLIX 对外宣布，他们要设立一项大赛，公开征集电影推荐系统的最佳电脑算法，第一个能把现有推荐系统的准确率提高 10%的参赛者将获得一百万美元的奖金。

Netflix 大奖赛从 2006 年 10 月份开始，Netflix 公开了大约 1 亿个 1—5 的匿名影片评级，数据集仅包含了影片名称。评价星级和评级日期，没有任何文本评价的内容。比赛要求参赛者预测 Netflix 的客户分别喜欢什么影片，要把预测的效率提高 10%以上。2009 年 9 月 21 日，来自全世界 186 个国家的四万多个参赛团队经过近三年的较量，终于有了结果。一个由工程师和统计学家组成的七人团队夺得了大奖，拿到了那张百万美元的超大支票。

II. 算法设计

A. 算法背景

1992 年，Goldberg 等人在构建邮件过滤系统 Tapestry 的过程中，采用了一种社会过滤的思想，提出了协同过滤算法[2]。1994 年，GroupLens 实验室在其发表的文中提出了基于用户的

协同过滤算法。该算法基于如下假设：如果两个用户对某些项目的评分比较相似，则可认为这两个用户兴趣度相似，就可以推断这两个用户对其它项目的评分也会比较相似。协同过滤算法首先通过用户—项目评分矩阵求出任意两个用户之间的相似度，对于某个目标用户，从中选出若干个与其最相似的用户，然后根据这些相似用户对项目的评分，预测目标用户对该未知项目的评分，根据评分高低产生推荐列表[3][4]。协同过滤算法在推荐时仅仅需要用户对项目的评分，而不必考虑项目的具体内容，具有很强的普适性。

B. 用户—项目评分模型

在协同过滤推荐系统中，定义用户对项目评分矩阵如下：

表 2-1 用户对项目评分数据表

	$Item_1$	……
$User_1$	$r_{1,1}$	……
……	……	……
$User_i$	$r_{i,1}$	……
……	……	……
$User_m$	$r_{m,1}$	……

C. 用户间相似性度量公式

度量用户 i 和用户 j 的相似性，首先要求出它们共同评分 的项目，然后根据不同的度量公式，求出这它们之间的相似度 sim(i，j)。常见的相似度度量公式如下。

（1） 余弦相似性

用户对项目的评分可以看成是一个 n 维空间上的向量， 用户 i 和用户 j 的评分就是两个向量，它们之间的相似度可以用这两个向量间夹角的余弦值表示，其值越大，表示越相似。假设用户 i 和用户 j 的评分向量分别为和，它们之间的相似度 sim(i，j)为式（2-1）。

$$sim(i,j) = \cos(\vec{i} \cdot \vec{j}) = \frac{\vec{i} \cdot \vec{j}}{\|\vec{i}\| \cdot \|\vec{j}\|} \quad (2\text{-}1)$$

（2） 修正的余弦相似性(sper 相关系数)

弦相似性的度量方法存在一个缺陷，度量过程中并未 考虑不同用户的评分尺度问题。为了修正这一缺陷，提出修正的余弦相似性的度量方法，主要做法是在计算过程中减去用户对项目评分的平均值 9。定义两用户 i 和

j 共同评分过的项目集合为 $I_{i,j}=I_i\cap I_j$ 其中 Ii 和 Ij 分别为用户 i 和用户 j 评分过 的项目集合，则它们的相似度 sim(i，j) 为式（2-2）。

$$sim(i,j)=\frac{\sqrt{\sum_{s\in I_{i,j}}(r_{i,s}-\bar{r}_i)^2}\cdot\sqrt{\sum_{s\in I_{i,j}}(r_{j,s}-\bar{r}_j)^2}}{\sum_{s\in I_{i,j}}(r_{i,s}-\bar{r}_i)\cdot(r_{j,s}-\bar{r}_j)}$$

(2-2)

相似度值范围[-1，1]值越大，则用户 i 和 j 兴趣爱好越接近。

（3） 相关相似性(Pearson 相关系数)

两用户 i 和 j 共同评分过的项目集合为 Ii，j ，它们的平均评分别为和，则它们的相似度 sim(i，j) 为式（2-3）。

$$sim(i,j)=\frac{\sum_{s\in I_{i,j}}(r_{i,s}-\bar{r}_i)}{\sqrt{\sum_{s\in I_{i,j}}(r_{i,s}-\bar{r}_i)^2}\cdot}$$

(2-3)

相似度值范围[-1，1]值越大，则用户 i 和 j 兴趣爱好越接近。本论文中使用 Pearson 相关系数求解相似度。

D. 最近邻居

对于目标用户 u，根据它与其他所有用户的相似度 sim(i，j)，选取最接近它的部分用户作为它的最近邻居选取方法有多种，我们采用的方式是将相似度降序排序，选择前 k 个作为该用户的最近邻居。

E. 评分预测

根据目标用户 u 的最近邻居集合 NSu，一般采用加权平均的预测策略，用户 u 对未评分项目 i 的预测评分为（2-4）

$$p_{u,i}=\bar{r}_u+\frac{\sum_{n\in NS_u}sim(u,n)\cdot(r_{n,i}-\bar{r}_n)}{\sum_{n\in NS_u}sim(u,n)}$$

(2-4)

其中，和是用户 u 和用户 n 的平均评分。

III. 环境配置

A. 环境配置

系统：	windows / liunx
开发语言：	ava
JDK 版本：	JDK1.7
IDE：	IntelliJ IDEA 15
项目属性：	Maven 工程

B. 项目编译和运行

（1）项目编译并导出 JAR

方法一：安装好 Maven 后，进入项目目录下，运行以下命令：

```
mvn clean
mvn package
```

在 target 目录下会生成可运行的架包 gzw-1.0-SNAPSHOT.jar。

方法二：导入 eclipse 或者

IntelliJ IDEA，导出可运行架包

（2）项目运行

在 JAR 存在的目录下，运行一下命令：

java -Xms1024M -Xmx4096M -jar ***.jar inputdir outputdir

***.jar 表示生成的架包

inputdir：输入的目录

outputdir: 输出目录

（3）程序输入

输入包括两部分：

训练集：针对每个电影的评分记录见附件中 training_set_small

测试集：需要预测的用户-电影集合见附件中 queries-small.txt

IV. 项目结构及源代码

A. 项目结构

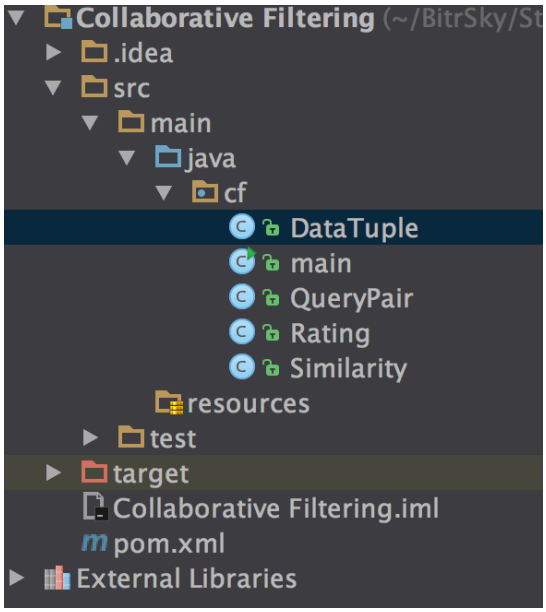

图 4-1 项目结构

如图 1，项目为 Maven 工程，包含 5 个 java 文件，main 是整个项目的入口。下面对每个 class 逐一介绍。

B. 源代码说明

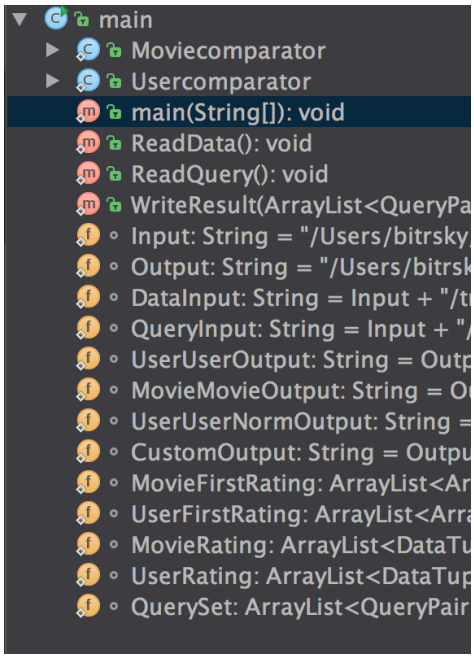

图 4-2 main.class 图示

main.Class 提供了训练集的

输入，测试文件的输入以及结果的输出，另外还定义了两个比较器，movie 之间的比较和 User 之间的比较。

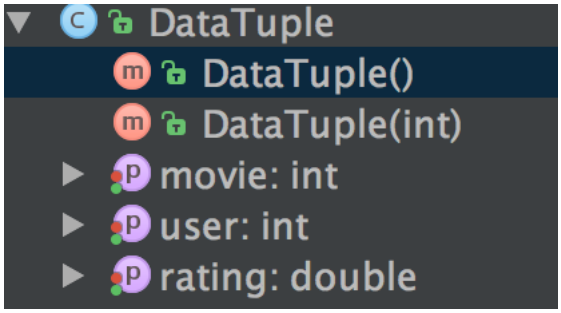

图 4-3 DataTuple.class 图示

DataTuple.class 是对每条评分记录的实例化，里面保存了用户电影以及评分

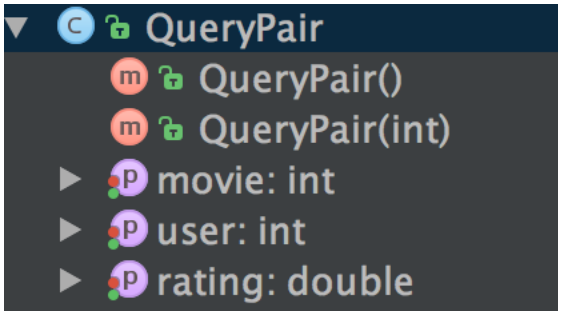

图 4-4 QueryPair.class 图示

QueryPary.class 记录了测试集中需要预测的用户电影对，Rating 为预测的分值。

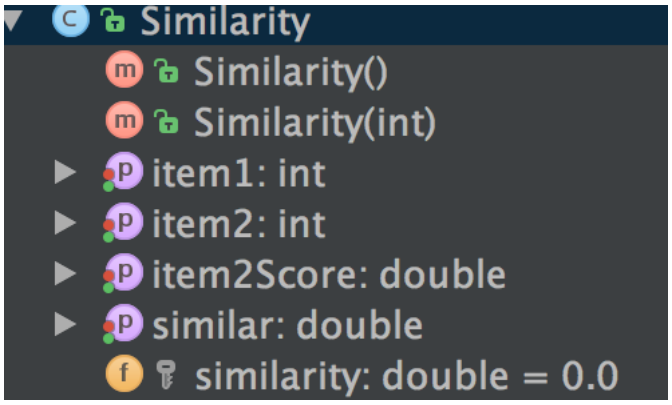

图 4-5 Similarity.class 图示

Similarity.class 表示 Item 之间的相似度，item 可以表示电影，也可以表示用户。

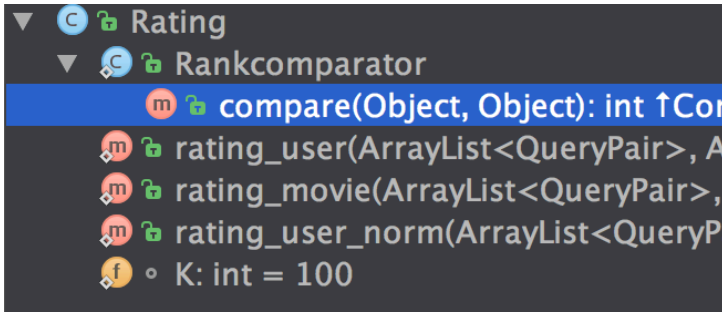

图 4-6 Rating.class 图示

Rating.class 中实现了基于电影的协同过滤预测方法[5]（rating_movie）和基于 user 的协同过滤预测方法（rating_user 和 rating_user_norm），K 表示 KNN 算法中的 K 值，初始设为 100.

V. 实验结果

实验结果见：

https://github.com/BitrSky/DataMiningPro

VI. 参考文献

[1] 高山冰. 大数据背景下 Netflix 的创新与发展研究[J]. 新闻界, 2014(8):66-69.

[2] Tak&#, cs, G&#, Pil&#, et al. Matrix factorization and neighbor based algorithms for the netflix prize problem[C]// ACM Conference on Recommender Systems, Recsys 2008, Lausanne, Switzerland, October. 2008:267-274.

[3] Koren Y. The BellKor solution to the Netflix Grand Prize[J]. Netflix Prize Documentation, 2009.

[4] Koren Y, Bell R, Volinsky C. Matrix Factorization Techniques for Recommender Systems[J]. IEEE Computer, 2009, 42(8):30-37.

[5] 吴金龙. Netflix Prize 中的协同过滤算法[D]. 北京大学, 2010.