Team Project Homework2_implemention for Video Recommender System-4git

---

团队项目作业2 视频推荐系统的实现

一、概述

本次团队作业为java作业，使用的IDE为eclipse，作业参考需求分析得出的内容，实现推荐系统的两大子系统——离线数据处理系统和在线推荐引擎。

推荐系统是可以依照人们的兴趣为人们提供符合兴趣的推荐项目。例如下表中，用户如果喜欢《玩具总动员》，则推荐系统会推荐出《星球大战4》、《独立日》、《星球大战6》、《碟中谍》、《欢乐糖果屋》、《回到未来》等电影。希望本次作业帮助同学们锻炼编程思维，提高编程能力，提高团队能力和加深友谊。

```
Inputed Movie:
1 - Toy Story (1995)::Adventure|Animation|Children|Comedy|Fantasy
Recommender movies:
260 - Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)::Action|Adv
780 - Independence Day (a.k.a. ID4) (1996)::Action|Adventure|Sci-Fi|War
1210 - Star Wars: Episode VI - Return of the Jedi (1983)::Action|Adventure|Sci
648 - Mission: Impossible (1996)::Action|Adventure|Mystery|Thriller
1073 - Willy Wonka & the Chocolate Factory (1971)::Children|Comedy|Fantasy|Mus
1270 - Back to the Future (1985)::Adventure|Comedy|Sci-Fi
588 - Aladdin (1992)::Adventure|Animation|Children|Comedy|Musical
32 - 12 Monkeys (Twelve Monkeys) (1995)::Sci-Fi|Thriller
356 - Forrest Gump (1994)::Comedy|Drama|Romance|War
1196 - Star Wars: Episode V - The Empire Strikes Back (1980)::Action|Adventure
736 - Twister (1996)::Action|Adventure|Romance|Thriller
1265 - Groundhog Day (1993)::Comedy|Fantasy|Romance
3114 - Toy Story 2 (1999)::Adventure|Animation|Children|Comedy|Fantasy
1198 - Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark)
```

作业资源:
文件/recommender_system_data.zip
    movies.dat 电影编号Movie ID对应的电影名称和信息
    ratings_train.dat 所有用户的电影评分记录
    ratings_train_test.txt 为软件开发所提供的小型的评分记录子集

提示：.dat文件可以使用记事本打开，若看不见文件后缀名的，请在系统中设置显示文件后缀名。

二、提交内容

团队作业提交到组长的git云端代码库中：
1. 离线数据处理系统recommender_data_process项目文件夹，提交位置：组长git云端/用户名/作业项目/team_project/recommender_data_process；
2. 在线推荐引擎recommender_online项目文件夹，提交位置：组长git云端/用户名/作业项目/team_project/recommender_online；

原创性提示：
作业严禁抄袭。 布置作业是为了通过作业加深对知识的认识并得到实践锻炼。 请不要复制其他同学的文字和截图。作业可以沟通和讨论，而就算讨论内容相同，写入作业的表达，文字的编排，修饰词的使用，一定都是不同的。作业会使用查重系统检查，如检查出雷同作业、抄袭作业的，则该次作业成绩将降等或取消，严重的将通报教务。

三、离线数据处理系统（作业最快时间18小时，建议时间48小时）

1.读入网站用户给电影评分的历史数据（参考知识：文件IO，file_io项目）

Movie Lens网站的数据记录了用户对影片的评分。推荐系统的离线模块要从数据仓库取得这些数据，并加以处理。这些数据包含了用户编号，电影编号，用户评分和评分时间。数据在Movie Lens网站上已经经过整理，规范整洁。数据记录的每一行记录都代表了一次用户的评分：

UserID::MovieID::Rating::Timestamp

例如，数据1::185::5::838983525，代表了用户1看了第185号电影《Net》1995年上映（电影对应关系可以在电影与ID对应数据中查到），评分为5。时间戳数据，目前不需要使用。

2.将用户评分历史数据处理成两个映射（参考知识：字符串split方法，map_java项目，ArrayList）
我们需要设计两个映射（maps）用来查找邻居。它们分别是item_to_user 映射和user_to_item 映射。将原始数据处理成两个映射的过程中，可以使用java中自带的字符串split方法，将用户评分数据切割开来。

item_to_user映射的关键值（Key）为电影的标号（MovieID），对应的Value是所有看过这个电影的用户（UserID）和他们的评分（Rating）。item_to_user映射的格式是：

MovieID: (UserID1, Rating), (UserID2, Rating), (UserID3, Rating)...

其中，MovieID为映射的Key，映射的Value为用户ID和评分信息：(UserID1, Rating), (UserID2, Rating), (UserID3, Rating)...，可以使用一个ArrayList作为映射Value的数据结构。

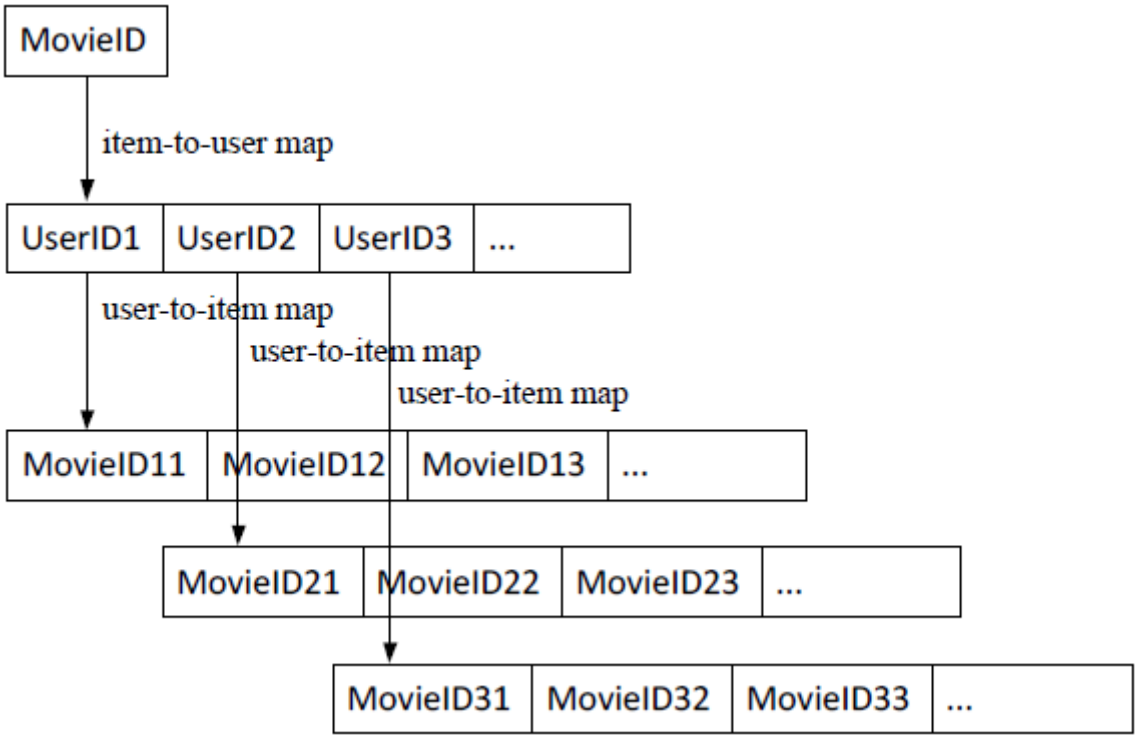
user_to_item映射的Key值是用户的标号UserID，Value是这个用户所看过电影的标号（MovieID）和用户对这部电影的评分。这个user_to_item映射的格式是：

UserID: (MovieID1, Rating), (MovieID2, Rating), (MovieID3, Rating)...

该映射的存储方式和item_to_user映射相类似。

可以将处理完成的两个映射使用文件io功能存成txt文档，作为中间结果，保存在硬盘上，保留中间处理的数据以便程序异常时使用。

3.邻居查找（参考知识：map_java项目，ArrayList）
使用以上这两个映射来找到一部电影的邻居。
首先从影片电影1出发，查找所有看了这部电影并做出评分的用户，从这些用户出发，再找到他们看过并且评分过的所有电影，这些电影便是电影1的候选邻居。

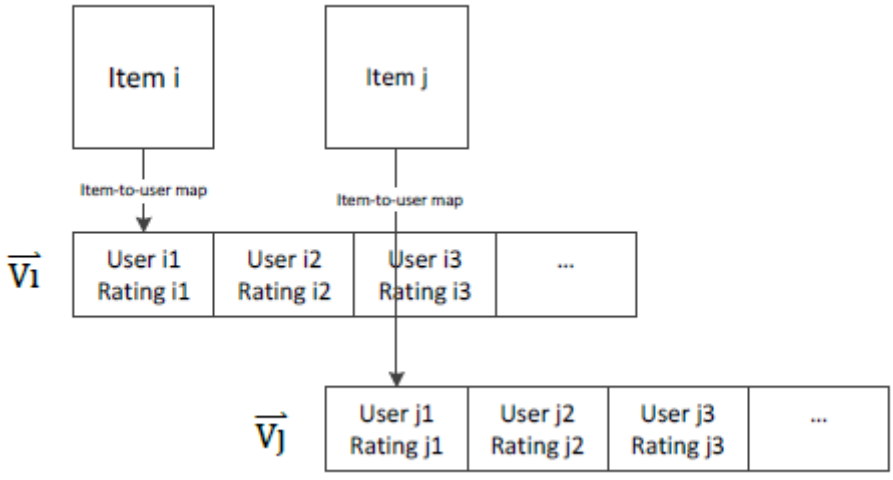


如图所示，MovieID的邻居通过先查找item_to_user映射，再查找user_to_item映射，得到集合{MovieID11, MovieID12, MovieID13..., MovieID21, MovieID22, MovieID23... MovieID31, MovieID32, MovieID33...}，这个集合便是MovieID的邻居。

4. 相似度计算（参考知识：similarity_java项目）
当邻居关系确定后，计算两两邻居的相似度。假设我们有两部电影Item i和Item j，如果以Item i为Key去item-to-user map中查找，则得到Value结果为看过item I这部电影的所有用户和他们的评分。我们将这些用户的评分构成向量Vi,。同理，得到看过电影Item j所有用户评分可以构成向量Vj。

使用余弦相似度计算公式计算两部电影的相似度：

$$\frac{\overrightarrow{V_I}\cdot\overrightarrow{V_J}}{|\overrightarrow{V_I}|\cdot|\overrightarrow{V_J}|}$$



注意：向量的维度补0问题。假设有位用户看过Item i电影评分为5，而他没有看过Item j电影，因此这位用户在Vj中的评分就不存在（因为没有观看记录）。这会引起两个向量的维度数目不同（Vi中有评分5，而Vj中没有评分，此时Vi的维度比Vj的维度多了1），而我们知道计算两个向量的相似度需要它们的维度数相同。于是，为了匹配Vi和Vj向

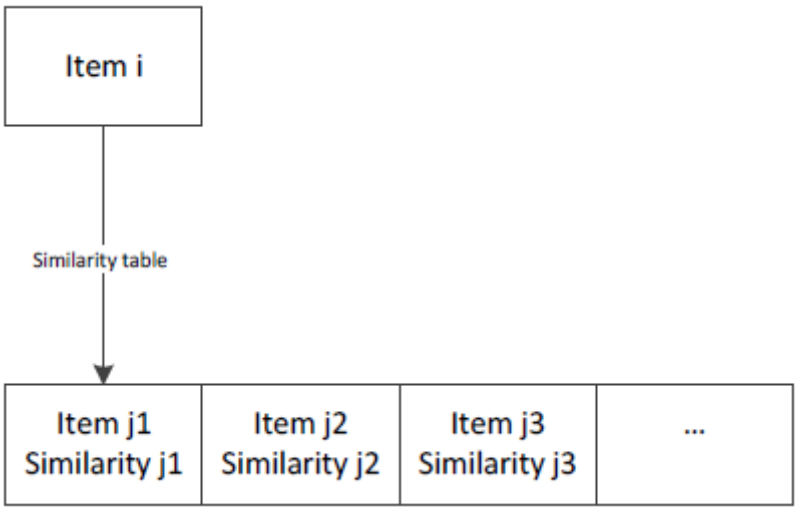
量的维度数，我们需要将这位用户的评分以值0的形式插入到Vj中，来使Vj的维度匹配Vi。反之，如果有用户看过Item j电影而未看过Item i电影，则需要加入值0维度到Vi向量中，以使Vi的维度数匹配Vj的维度数。

将计算得到的相似度结果存成 txt文档，这个txt文档中所包含的数据叫做sim table。存储格式的参考格式可以为：MovieID:MovieID1,simValue1　MovieID2,simValue2　MovieID3,simValue3...中间的大空隔为制表符"\t"，至此，离线数据处理任务便完成了。

这条sim table数据表示电影MovieID和MovieID1的相似度为simValue1，MovieID和MovieID2的相似度为simValue2，MovieID和MovieID3的相似度为simValue3等等。

四、在线推荐引擎（作业最快时间6小时，建议时间18小时）

在线推荐系统的的基本推荐功能实现较容易理解。将一个给定的输入item i输入到推荐引擎中，使用item i作为键值，在加载到电脑内存中的相似度表（simTable）中找到与item i对应的相似度较高的的其他item_id。在推荐过程中，所有的查询均在计算机内存中操作，所以推荐结果返回的速度是非常快的。查询相似项目的返回值是一个列表，列表中的每个元素包含了相似项目item id和相似度值Similarity，如图所示。最后，系统对相似值向量按照相似度值进行排序，并返回前15名的推荐结果。（参考知识：文件IO，map_java项目）



1. 加载sim table数据到系统内存，存储成为一个map，key为MovieID，value为与这个MovieID的相似的若干电影和它们与这个MovieID电影的相似度，格式为：
MovieID: (MovieID1, simValue1), (MovieID2, simValue2), (MovieID3, simValue3)...

2. 输入一个MovieID做为key，查找sim table map，返回相似度最高的15个MovieID。

3. 为这15个MovieID匹配电影名称的信息并显示，最终完成截图如图所示，系统输入一个MovieID号码，显示出:
(1) 该MovieID的号码，以及对应的电影名称和信息。
(2) 推荐的15个MovieID号码，以及对应的电影名称和信息

截图如图所示：

```
Inputed Movie:
1 - Toy Story (1995)::Adventure|Animation|Children|Comedy|Fantasy
Recommender movies:
260 - Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)::Action|Adv
780 - Independence Day (a.k.a. ID4) (1996)::Action|Adventure|Sci-Fi|War
1210 - Star Wars: Episode VI - Return of the Jedi (1983)::Action|Adventure|Sci
648 - Mission: Impossible (1996)::Action|Adventure|Mystery|Thriller
1073 - Willy Wonka & the Chocolate Factory (1971)::Children|Comedy|Fantasy|Mus
1270 - Back to the Future (1985)::Adventure|Comedy|Sci-Fi
588 - Aladdin (1992)::Adventure|Animation|Children|Comedy|Musical
32 - 12 Monkeys (Twelve Monkeys) (1995)::Sci-Fi|Thriller
356 - Forrest Gump (1994)::Comedy|Drama|Romance|War
1196 - Star Wars: Episode V - The Empire Strikes Back (1980)::Action|Adventure
736 - Twister (1996)::Action|Adventure|Romance|Thriller
1265 - Groundhog Day (1993)::Comedy|Fantasy|Romance
3114 - Toy Story 2 (1999)::Adventure|Animation|Children|Comedy|Fantasy
1198 - Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark)
```