中国人民大学信息学院

大数据实践

基于 Spark Mllib 的电影推荐系统 报告

报告人：陶友贤（2017100955）

Github: https://github.com/taoyouxian/Xspace/wiki/big_data_practice

1 前言

互联网的出现和普及给用户带来了大量的信息，满足了用户在信息时代对信息的需求，但随着网络的迅速发展而带来的网上信息量的大幅增长，使得用户在面对大量信息时无法从中获得对自己真正有用的那部分信息，对信息的使用效率反而降低了，这就是所谓的信息超载问题。

解决信息超载问题一个非常有潜力的办法是推荐系统，它是根据用户的信息需求、兴趣等，将用户感兴趣的信息、产品等推荐给用户的个性化信息推荐系统，如图 1 所示。和搜索引擎相比推荐系统通过研究用户的兴趣偏好，进行个性化计算，由系统发现用户的兴趣点，从而引导用户发现自己的信息需求。一个好的推荐系统不仅能为用户提供个性化的服务，还能和用户之间建立密切关系，让用户对推荐产生依赖。

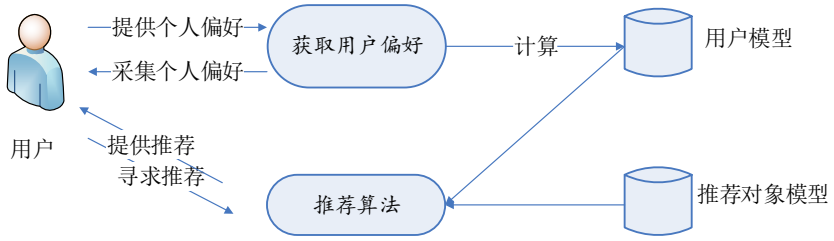

图 1 推荐系统通用模型

本文中，笔者只是根据当前实时推荐系统的应用价值，其系统架构图（见附件图 8），简单实现了初版本的基于 Spark Mllib 电影推荐系统应用，仅仅实现用户的定向推荐（相当于架构图中的“离线”部分）。

2 系统设计

系统设计将分别从推荐算法设计、数据集选择、数据分析等方面介绍。

（1）推荐算法设计

在推荐算法设计上，推荐系统主要采用的系统推荐方法有基于内容推荐，协同过滤推荐，基于关联规则推荐，基于效用推荐，基于知识推荐，组合推荐等。这里，我们主要采用协同过滤算法进行数据分析。协同过滤推荐技术是推荐系统中应用最早和最为成功的技术之一。它一般采用最近邻技术，利用用户的历史喜好信息计算用户之间的距离，然后利用目标用户的最近邻居用户对商品评价的加

权评价值来预测目标用户对特定商品的喜好程度，系统从而根据这一喜好程度来对目标用户进行推荐。协同过滤最大优点是对推荐对象没有特殊的要求，能处理非结构化的复杂对象，如音乐、电影。其算法的推荐过程如下图 2 所示。

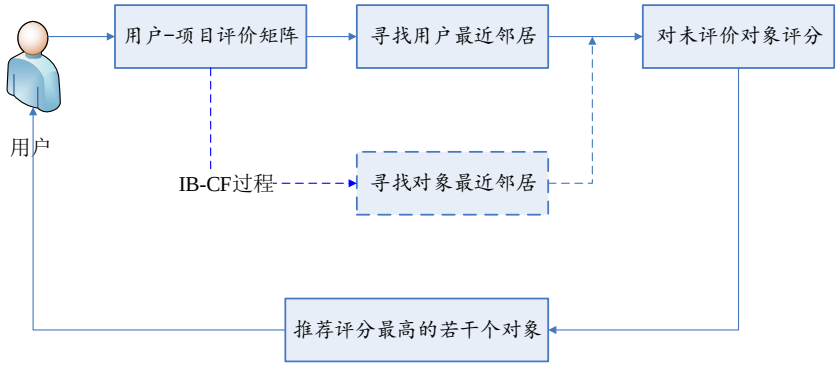

图 2 基于用户的协同过滤算法推荐过程

Spark MLlib 当前支持基于模型的协同过滤，其中用户和商品通过一小组隐性因子进行表达，并且这些因子也用于预测缺失的元素。MLLib 使用交替最小二乘法（ALS）来学习这些隐性因子。在这个模型里， MLlib 中的实现有如下参数：

- numBlocks 是用于并行化计算的分块个数（设置为-1 时 为自动配置）；
- rank 是模型中隐性因子的个数；
- iterations 是迭代的次数；
- lambda 是 ALS 的正则化参数；
- implicitPrefs 决定了是用显性反馈 ALS 的版本还是用隐性反馈数据集的版本；
- alpha 是一个针对于隐性反馈 ALS 版本的参数，这个参数决定了偏好行为强度的基准。

本文使用的基于矩阵分解的模型如下图 3 所示

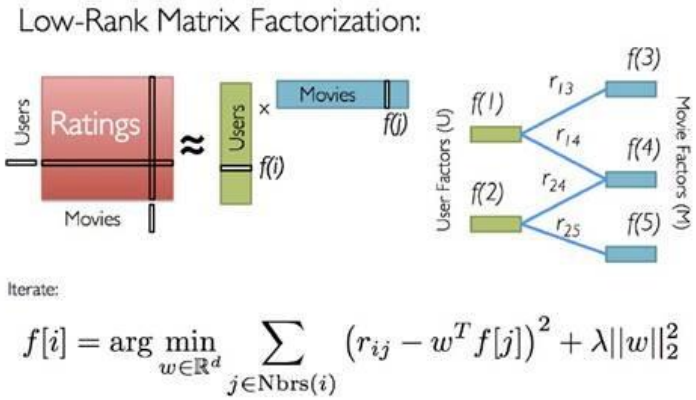

图 3 基于矩阵分解的模型图

（2）数据集选择

在数据集选择上，考虑到推荐系统的应用性和简单性，这里使用 GroupLens Research（http://grouplens.org/datasets/movielens/）提供的数据，该数据为一组从 20 世纪 90 年末到 21 世纪初由 MovieLens 用户提供的电影评分数据，这些数据中包括电影评分、电影元数据（风格类型和年代）以及关于用户的人口统计学数据（年龄、邮编、性别和职业等）。根据不同需求该组织提供了不同大小的样本数据，不同样本信息中包含三种数据：评分、用户信息和电影信息。

（3）数据分析

在数据分析上，对上述数据集，我们对此分析进行如下步骤：

1. 装载如下两种数据：
   a) 装载样本评分数据，其中最后一列时间戳除 10 的余数作为 key，Rating 为值；
   b) 装载电影目录对照表（电影 ID->电影标题）
2. 将样本评分表以 key 值切分成 3 个部分，分别用于训练(60%，并加入用户评分),校验(20%)， and 测试(20%)
3. 训练不同参数下的模型，并再校验集中验证，获取最佳参数下的模型
4. 用最佳模型预测测试集的评分，计算和实际评分之间的均方根误差

根据用户评分的数据，推荐前十部最感兴趣的电影（注意要剔除用户已经评分的电影）

3 系统部署

系统部署将分别从集群配置中的软硬件资源、测试数据说明等方面介绍，集群软硬件环境如表 3.1 所示。

表 3.1 集群软硬件配置

设备	配置
CPU	Intel Core i7-3770 3.40GHz, 4 cores
内存	16 GB
磁盘	2TB 7200RPM SATA _ 1
网络	1 Gb Ethernet
操作系统	CentOS release 6.4 x86-64 (Final)

Hadoop	2.7.3
Spark	2.1.1

本实验部署在共 5 个节点的 Hadoop 分布式集群上，其中 1 个节点作为主节点，其他 4 个节点作为从节点。每个节点 CPU 核数为 4 个，内存为 16G，操作系统为 CentOs6.4。每个节点拥有 2TB 的磁盘空间，且都是台式机；此外，Java 版本为 1.8，Hadoop 版本为 2.7.3，与 Hadoop 集群配套的 Spark 版本为 2.1.1。

在测试数据上，在 MovieLens 提供的电影评分数据分为三个表：评分、用户信息和电影信息，在该系列提供的附属数据提供大概 6000 位读者和 100 万个评分数据，具体位置为/data/class8/movielens/data 目录下，对三个表数据说明可以参考该目录下 README 文档。

1.评分数据说明（ratings.data)

该评分数据总共四个字段，格式为 UserID::MovieID::Rating::Timestamp，分为为用户编号:: 电影编号:: 评分:: 评分时间戳，其中各个字段说明如下：

用户编号范围 1~6040
电影编号 1~3952
电影评分为五星评分，范围 0~5
评分时间戳单位秒
每个用户至少有 20 个电影评分

2.用户信息(users.dat)

用户信息五个字段，格式为 UserID::Gender::Age::Occupation::Zip-code，分为为用户编号:: 性别:: 年龄:: 职业::邮编，其中各个字段说明如下：

用户编号范围 1~6040
性别，其中 M 为男性，F 为女性
不同的数字代表不同的年龄范围，如：25 代表 25~34 岁范围
职业信息，在测试数据中提供了 21 中职业分类
地区邮编

3.电影信息(movies.dat)

电影数据分为三个字段，格式为 MovieID::Title::Genres，分为为电影编号:: 电影名:: 电影类别，其中各个字段说明如下：

电影编号 1~3952

由 IMDB 提供电影名称，其中包括电影上映年份

电影分类，这里使用实际分类名非编号，如：Action、Crime 等

4 实验及结果

（1） 启动 Hadoop/Spark 集群

在主节点的 Hadoop 和 Spark 目录下的 sbin 文件夹下启动 start-all.sh 脚本，实现集群的运行。

（2） 进行用户评分，生成用户样本数据

由于该程序中最终推荐给用户十部电影，这需要用户提供对样本电影数据的评分，然后根据生成的最佳模型获取当前用户推荐电影。用户可以使用 /home/hadoop/upload/class8/movielens/bin/rateMovies 程序进行评分，最终生成 personalRatings.txt 文件，如图 4 所示：

```
[hadoop@hadoop1 ~]$ cd /home/hadoop/upload/class8/movielens
[hadoop@hadoop1 movielens]$ bin/rateMovies
Please rate the following movie (1-5 (best), or 0 if not seen):
Toy Story (1995): 5
Independence Day (a.k.a. ID4) (1996): 4
Dances with Wolves (1990): 5
Star Wars: Episode VI - Return of the Jedi (1983): 3
Mission: Impossible (1996): 4
Ace Ventura: Pet Detective (1994): 5
Die Hard: With a Vengeance (1995): 5
Batman Forever (1995): 4
Pretty Woman (1990): 3
Men in Black (1997): 5
Dumb & Dumber (1994): 3
[hadoop@hadoop1 movielens]$ ll
total 16
drwxrwxrwx 2 hadoop hadoop 4096 Mar 12 17:03 bin
drwxrwxrwx 2 hadoop hadoop 4096 Mar 12 17:03 data
-rw-rw-r-- 1 hadoop hadoop  242 Aug 21 10:54 personalRatings.txt
-rwxrw-rw- 1 hadoop hadoop  561 Jul  2  2014 personalRatings.txt.template
[hadoop@hadoop1 movielens]$
```

图 4 用户评分展示图

（3） 设置运行环境

在 IDEA 运行配置中设置 MovieLensALS 运行配置，需要设置输入数据所在文件夹和用户的评分文件路径：

- 输入数据所在目录：输入数据文件目录，在该目录中包含了评分信息、用户信息和电影信息，这里设置为 home/hadoop/upload/class8/movielens/data/
- 用户的评分文件路径：前一步骤中用户对十部电影评分结果文件路径，在这里设置为/home/hadoop/upload/class8/movielens/personalRatings.txt

（4） 执行并观察输出

- 输出 Got 1000209 ratings from 6040 users on 3706 movies，表示本算法中计算数据包括大概 100 万评分数据、6000 多用户和 3706 部电影；

- 输出 Training: 602252, validation: 198919, test: 199049，表示对评分数据进行拆分为训练数据、校验数据和测试数据，大致占比为 6:2:2；
- 在计算过程中选择 8 种不同模型对数据进行训练，然后从中选择最佳模型，其中最佳模型比基准模型提供 22.30%

```
RMSE (validation) = 0.8680885498009973 for the model trained with rank = 8, lambda = 0.1, and numIter = 10.
RMSE (validation) = 0.868882967482595 for the model trained with rank = 8, lambda = 0.1, and numIter = 20.
RMSE (validation) = 3.7558695311242833 for the model trained with rank = 8, lambda = 10.0, and numIter = 10.
RMSE (validation) = 3.7558695311242833 for the model trained with rank = 8, lambda = 10.0, and numIter = 20.
RMSE (validation) = 0.8663942501841964 for the model trained with rank = 12, lambda = 0.1, and numIter = 10.
RMSE (validation) = 0.8674684744165418 for the model trained with rank = 12, lambda = 0.1, and numIter = 20.
RMSE (validation) = 3.7558695311242833 for the model trained with rank = 12, lambda = 10.0, and numIter = 10.
RMSE (validation) = 3.7558695311242833 for the model trained with rank = 12, lambda = 10.0, and numIter = 20.
The best model was trained with rank = 12 and lambda = 0.1, and numIter = 10, and its RMSE on the test set is 0.8652326018300565.
The best model improves the baseline by 22.30%.
```

- 利用前面获取的最佳模型，结合用户提供的样本数据，最终推荐给用户如下影片：

```
Movies recommended for you:
 1: Bewegte Mann, Der (1994)
 2: Chushingura (1962)
 3: Love Serenade (1996)
```

```
 4: For All Mankind (1989)
 5: Vie est belle, La (Life is Rosey) (1987)
 6: Bandits (1997)
 7: King of Masks, The (Bian Lian) (1996)
 8: I'm the One That I Want (2000)
 9: Big Trees, The (1952)
10: First Love, Last Rites (1997)
```

如上所述，其在 IDEA 中的参数配置、训练模型及其运行结果如下图 5、6、7 所示：

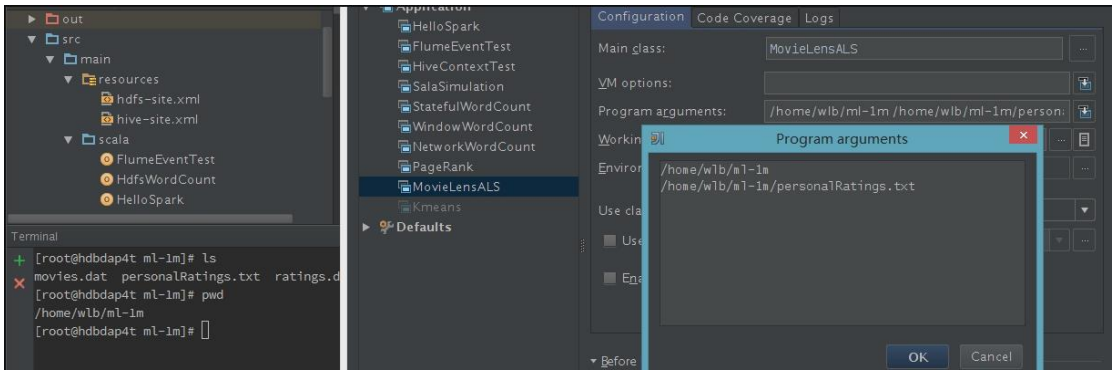

图 5 参数设置示意图

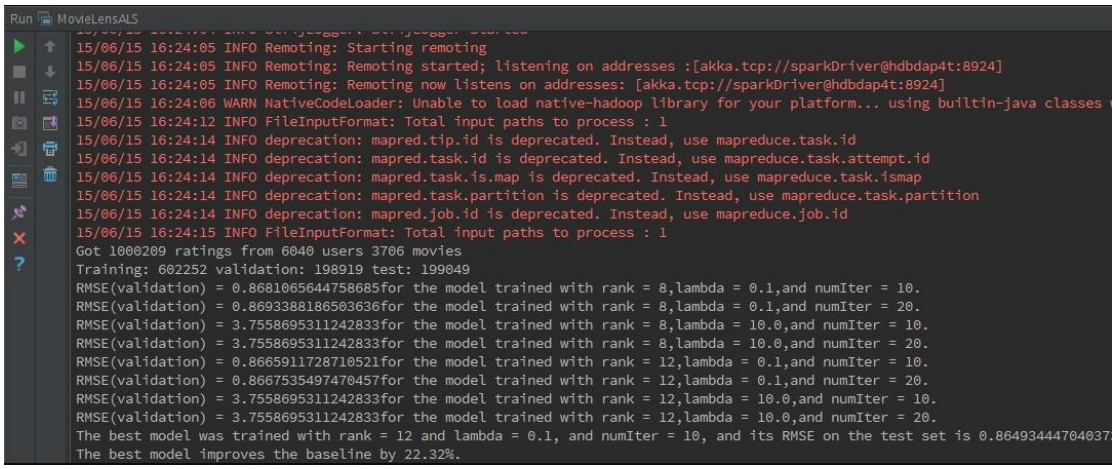

图 6 模型训练示意图

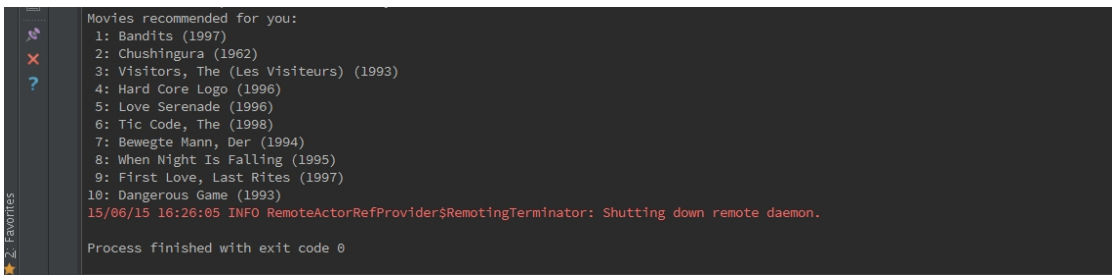

图 5 运行结果示意图

5 总结

首先，从最开始的教学到后来的实验任务，可以说我真的学习到了很多，从最开始对这类问题的不熟悉，之后真正动手去收集资料并进行系统的部署与程序设计，才真正了解自己学习中的优缺点，学习到了大数据下系统部署的难点与挑战，一方面是 Linux 系统下面不断运行的一些命令，另一方面是写程序的过程中也是不断挑战自己的过程，需要在集群上不断测试，查看日志。当然，在这个过程中，自己不断学习和调试程序也是很享受的。

我将老师系统部署的手册和自己亲自实践中遇到的问题都放置在 github 上，也是对自己学习的一个积累吧，同时，我所在的项目组也在开发基于大数据下的实时分析系统，主要是围绕 Presto 进行的系统设计，也是对老师上课内容的一个补充。

6 附件

- 关于测试数据说明：

1. 使用的 ratings.dat 的数据样本如下所示：

```
1::1193::5::978300760
1::661::3::978302109
1::914::3::978301968
1::3408::4::978300275
1::2355::5::978824291
1::1197::3::978302268
1::1287::5::978302039
1::2804::5::978300719
```

2. 使用的 users.dat 的数据样本如下所示：

```
1::F::1::10::48067
2::M::56::16::70072
```

```
3::M::25::15::55117
4::M::45::7::02460
5::M::25::20::55455
6::F::50::9::55117
7::M::35::1::06810
8::M::25::12::11413
```

3. 使用的 movies.dat 的数据样本如下所示：

```
1::Toy Story (1995)::Animation|Children's|Comedy
2::Jumanji (1995)::Adventure|Children's|Fantasy
3::Grumpier Old Men (1995)::Comedy|Romance
4::Waiting to Exhale (1995)::Comedy|Drama
5::Father of the Bride Part II (1995)::Comedy
6::Heat (1995)::Action|Crime|Thriller
7::Sabrina (1995)::Comedy|Romance
8::Tom and Huck (1995)::Adventure|Children's
```

- 基于 Spark Mllib 的电影实时推荐系统

本文实现了简单的电影推荐系统应用，但是，仅仅实现用户的定向推荐，在实际应用中价值不是非常大。如果体现价值，最好能够实现实时或者准实时推荐。其架构图如下图 4 所示：

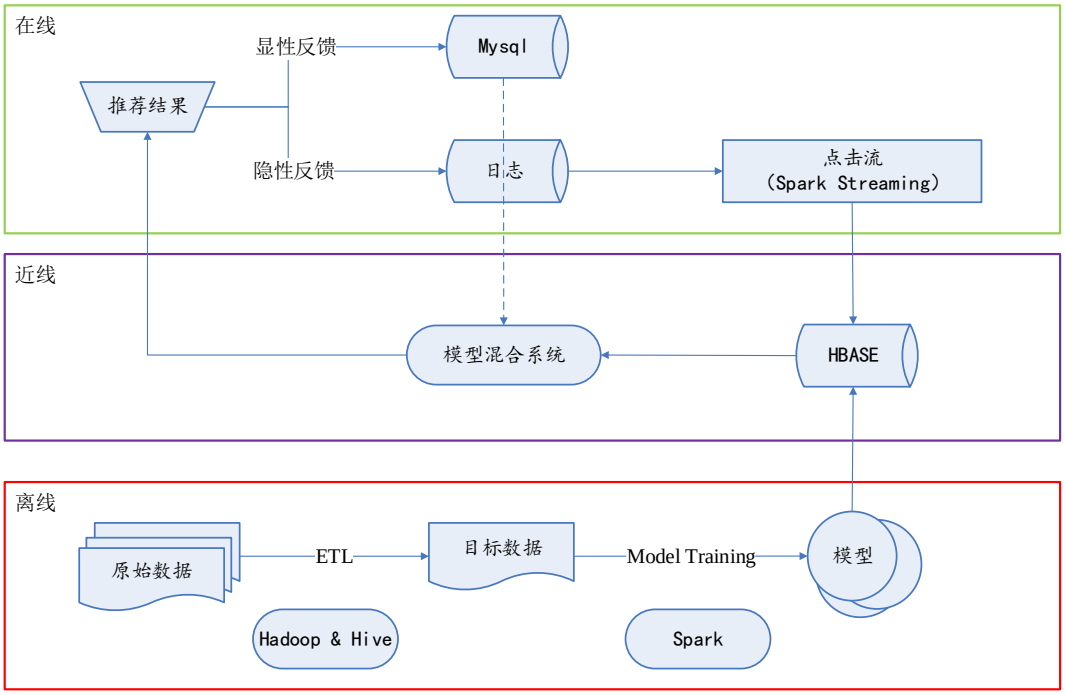

图 8 电影实时推荐架构图

- 架构图分为三层：离线、近线和在线。
- 离线部分：主要实现模型的建立。原始数据通过 ETL 加工清洗，得到目标数据，目标业务数据结合合适的算法，学习训练模型，得到最佳的模型。
- 近线部分：主要使用 HBase 存储用户行为信息，模型混合系统综合显性反馈和隐性反馈的模型处理结果，将最终的结果推荐给用户。
- 在线部分：这里，主要有两种反馈，显性和隐性。显性反馈理解为用户将商品加入购物车，用户购买商品这些用户行为；隐性反馈理解为用户在某个商品上停留的时间，用户点击哪些商品这些用户行为。这里，为了实现实时/准实时操作，使用到 Spark Streaming 对数据进行实时处理。（有可能是 Flume+Kafka+Spark Streaming 架构）