# Mahout是什么



* Mahout是一个算法库,集成了很多算法。
* Apache Mahout 是 Apache Software Foundation（ASF）旗下的一个开源项目，提供一些可扩展的**机器学习领域经典算法的实现**，旨在帮助开发人员更加方便快捷地创建智能应用程序。
* Mahout项目目前已经有了多个公共发行版本。Mahout包含许多实现，包括聚类、分类、推荐过滤、频繁子项挖掘。
* 通过使用 Apache Hadoop 库，Mahout 可以有效地扩展到Hadoop集群。
* Mahout 的创始人 Grant Ingersoll 介绍了机器学习的基本概念，并演示了如何使用 Mahout 来实现文档集群、提出建议和组织内容。



# Mahout能做什么

## 2.1、推荐引擎

在目前采用的机器学习技术中，推荐引擎是最容易被一眼认出来的，也是应用范围最广的。服务商或网站会根据你过去的行为为你推荐书籍、电影或文章。

在部署了推荐系统的电子商务网站中，**亚马逊大概是最有名**的。亚马逊基于用户的交易行为和网站记录为你推荐你可能喜欢的商品。

而facebook这样的社交网络则利用推荐技术为你找到最可能尚未关联的朋友。

同时，这一技术也被各大知名国内网站所使用，如腾讯、人人、京东、淘宝。

## 2.2、聚类

顾名思义，物以类聚，人以群分。聚类是把具有共同属性的物品进行归类。

Google news使用聚类技术通过标题把新闻文章进行分组，从而按照逻辑线索来显示新闻，而并非给出所有新闻的原始列表。

## 2.3、分类

分类技术决定了一个事物多大程度上从属于某种类别或类型，或者多大程度上具有或者不具有某些属性。与聚类一样，分类无处不在，但更多隐身于幕后。通常这些系统会考察类别中的大量实例，来学习推到出分类的规则。

雅虎邮箱基于用户以前对正常右键和垃圾邮件的报告，以及电子右键自身的特征，来判别到来的消息是否是垃圾邮件。

# 3、Mahout协同过滤算法

Mahout使用了Taste来提高协同过滤算法的实现，它是一个**基于Java实现的可扩展的，高效的推荐引擎**。Taste既实现了最基本的**基于用户的和基于内容的推荐算法**，同时也提供了扩展接口，使用户可以方便的定义和实现自己的推荐算法。同时，Taste不仅仅只适用于Java应用程序，它可以作为内部服务器的一个组件以**HTTP和Web Service**的形式向外界提供推荐的逻辑。Taste的设计使它能满足企业对推荐引擎在性能、灵活性和可扩展性等方面的要求。

Taste主要包括以下几个接口：

* **DataModel**是用户喜好信息的抽象接口，它的具体实现支持从任意类型的数据源抽取用户喜好信息。Taste 默认提供 **JDBCDataModel** 和 **FileDataModel**，分别支持从**数据库和文件**中读取用户的喜好信息。
* UserSimilarity 和 ItemSimilarity 。UserSimilarity 用于定义两个用户间的相似度，它是基于协同过滤的推荐引擎的核心部分，可以用来计算用户的“邻居”，这里我们将与当前用户口味相似的用户称为他的邻居。ItemSimilarity 类似的，计算Item之间的相似度。
* **UserNeighborhood**用于基于用户相似度的推荐方法中，推荐的内容是基于找到与当前用户喜好相似的邻居用户的方式产生的。UserNeighborhood 定义了确定邻居用户的方法，具体实现一般是基于 UserSimilarity 计算得到的。
* **Recommender**是推荐引擎的抽象接口，Taste 中的核心组件。程序中，为它提供一个 DataModel，它可以计算出对不同用户的推荐内容。实际应用中，主要使用它的实现类 **GenericUserBasedRecommender** 或者 **GenericItemBasedRecommender**，分别实现**基于用户相似度的推荐引擎或者基于内容的推荐引擎**。
* **RecommenderEvaluator**：评分器。
* **RecommenderIRStatsEvaluator**：搜集推荐性能相关的指标，包括准确率、召回率等等。

## 3.1、DataModel

* org.apache.mahout.cf.taste.impl.model.GenericDataModel
* org.apache.mahout.cf.taste.impl.model.GenericBooleanPrefDataModel
* org.apache.mahout.cf.taste.impl.model.PlusAnonymousUserDataModel
* org.apache.mahout.cf.taste.impl.model.file.FileDataModel
* org.apache.mahout.cf.taste.impl.model.hbase.HBaseDataModel
* org.apache.mahout.cf.taste.impl.model.cassandra.CassandraDataModel
* org.apache.mahout.cf.taste.impl.model.mongodb.MongoDBDataModel
* org.apache.mahout.cf.taste.impl.model.jdbc.SQL92JDBCDataModel
* org.apache.mahout.cf.taste.impl.model.jdbc.MySQLJDBCDataModel
* org.apache.mahout.cf.taste.impl.model.jdbc.PostgreSQLJDBCDataModel
* org.apache.mahout.cf.taste.impl.model.jdbc.GenericJDBCDataModel
* org.apache.mahout.cf.taste.impl.model.jdbc.SQL92BooleanPrefJDBCDataModel
* org.apache.mahout.cf.taste.impl.model.jdbc.MySQLBooleanPrefJDBCDataModel
* org.apache.mahout.cf.taste.impl.model.jdbc.PostgreBooleanPrefSQLJDBCDataModel
* org.apache.mahout.cf.taste.impl.model.jdbc.ReloadFromJDBCDataModel

    从类名上就可以大概猜出来每个DataModel的用途，奇怪的是竟然没有HDFS的DataModel，有人实现了一个，请参考 [MAHOUT-1579](https://issues.apache.org/jira/browse/MAHOUT-1579) 。

## 3.2、相似度

UserSimilarity 和 ItemSimilarity 相似度实现有以下几种：

* CityBlockSimilarity ：基于Manhattan距离相似度
* EuclideanDistanceSimilarity ：基于欧几里德距离计算相似度
* LogLikelihoodSimilarity ：基于对数似然比的相似度
* PearsonCorrelationSimilarity ：基于皮尔逊相关系数计算相似度
* SpearmanCorrelationSimilarity ：基于皮尔斯曼相关系数相似度
* TanimotoCoefficientSimilarity ：基于谷本系数计算相似度
* UncenteredCosineSimilarity ：计算 Cosine 相似度

## 3.3、最近邻域

UserNeighborhood 主要实现有两种：

* NearestNUserNeighborhood：对每个用户取固定数量N个最近邻居
* ThresholdUserNeighborhood：对每个用户基于一定的限制，取落在相似度限制以内的所有用户为邻居

## 3.4、推荐引擎

Recommender分为以下几种实现：

* GenericUserBasedRecommender：基于用户的推荐引擎
* GenericBooleanPrefUserBasedRecommender：基于用户的无偏好值推荐引擎
* GenericItemBasedRecommender：基于物品的推荐引擎
* GenericBooleanPrefItemBasedRecommender：基于物品的无偏好值推荐引擎

## 3.5、推荐系统评测

RecommenderEvaluator有以下几种实现：

* AverageAbsoluteDifferenceRecommenderEvaluator ：计算平均差值
* RMSRecommenderEvaluator ：计算均方根差

# Mahout协同过滤算法编程

## 创建Maven项目：详见《创建一个Maven项目》

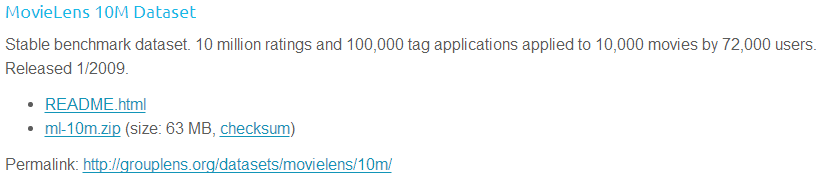
## 导入Mahout依赖

|  |
| --- |
| <**dependencies**>  <**dependency**>  <**groupId**>org.apache.mahout</**groupId**>  <**artifactId**>mahout-examples</**artifactId**>  <**version**>0.11.2</**version**>  <**exclusions**>  <**exclusion**>  <**groupId**>org.slf4j</**groupId**>  <**artifactId**>slf4j-log4j12</**artifactId**>  </**exclusion**>  <**exclusion**>  <**groupId**>commons-io</**groupId**>  <**artifactId**>commons-io</**artifactId**>  </**exclusion**>  </**exclusions**>  </**dependency**>  <**dependency**>  <**groupId**>commons-io</**groupId**>  <**artifactId**>commons-io</**artifactId**>  <**version**>2.4</**version**>  </**dependency**>  <**dependency**>  <**groupId**>junit</**groupId**>  <**artifactId**>junit</**artifactId**>  <**version**>4.11</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-jdbc</**artifactId**>  <**version**>4.2.0.RELEASE</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-core</**artifactId**>  <**version**>4.2.0.RELEASE</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-context</**artifactId**>  <**version**>4.2.0.RELEASE</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.apache.struts</**groupId**>  <**artifactId**>struts2-core</**artifactId**>  <**version**>2.2.1</**version**>  </**dependency**>  <**dependency**>  <**groupId**>javassist</**groupId**>  <**artifactId**>javassist</**artifactId**>  <**version**>3.8.0.GA</**version**>  </**dependency**>  <**dependency**>  <**groupId**>javax.servlet</**groupId**>  <**artifactId**>servlet-api</**artifactId**>  <**version**>2.5</**version**>  </**dependency**>  <**dependency**>  <**groupId**>commons-lang</**groupId**>  <**artifactId**>commons-lang</**artifactId**>  <**version**>2.4</**version**>  </**dependency**>  <**dependency**>  <**groupId**>net.sf.barcode4j</**groupId**>  <**artifactId**>barcode4j</**artifactId**>  <**version**>2.0</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.jsoup</**groupId**>  <**artifactId**>jsoup</**artifactId**>  <**version**>1.8.3</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.apache.hadoop</**groupId**>  <**artifactId**>hadoop-client</**artifactId**>  <**version**>2.6.1</**version**>  </**dependency**>  <**dependency**>  <**groupId**>io.netty</**groupId**>  <**artifactId**>netty-all</**artifactId**>  <**version**>5.0.0.Alpha2</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.apache.commons</**groupId**>  <**artifactId**>commons-lang3</**artifactId**>  <**version**>3.2</**version**>  </**dependency**>  <**dependency**>  <**groupId**>junit</**groupId**>  <**artifactId**>junit</**artifactId**>  <**version**>4.11</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.wltea.ik-analyzer</**groupId**>  <**artifactId**>ik-analyzer</**artifactId**>  <**version**>4.10.2</**version**>  </**dependency**>  <**dependency**>  <**groupId**>redis.clients</**groupId**>  <**artifactId**>jedis</**artifactId**>  <**version**>2.7.3</**version**>  </**dependency**> </**dependencies**> |

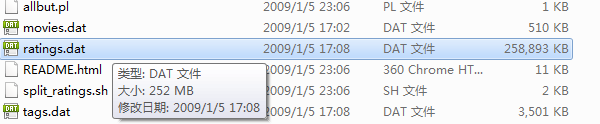
## 4.3、下载电影评分数据

下载地址：<http://grouplens.org/datasets/movielens/>

数据类别：7.2万用户对1万部电影的百万级评价和10万个标签数据



本例数据：本例中只需要使用评分数据



## 4.4、编写基于用户的推荐





## 编写基于物品的推荐

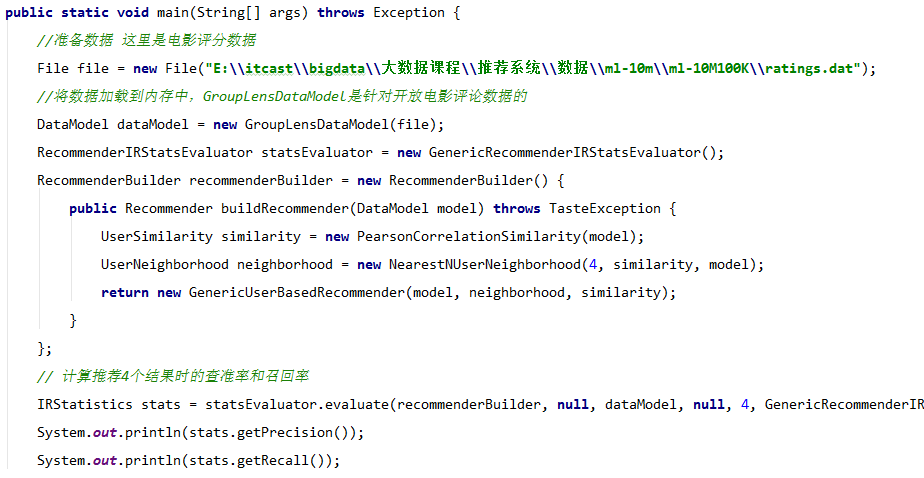




## 评估推荐模型



## 4.7、获取推荐的查准率和查全率





# Mahout运行在Hadoop集群

## 4.1、Hadoop 执行脚本

hadoop jar mahout-examples-0.9-job.jar org.apache.mahout.cf.taste.hadoop.item.RecommenderJob --input /sanbox/movie/10M.txt --output /sanbox/movie/r -s SIMILARITY\_LOGLIKELIHOOD

参数说明

* --input(path) : 存储用户偏好数据的目录，该目录下可以包含一个或多个存储用户偏好数据的文本文件；
* --output(path) : 结算结果的输出目录
* --numRecommendations (integer) : 为每个用户推荐的item数量，默认为10
* --usersFile (path) : 指定一个包含了一个或多个存储userID的文件路径，仅为该路径下所有文件包含的userID做推荐计算 (该选项可选)
* --itemsFile (path) : 指定一个包含了一个或多个存储itemID的文件路径，仅为该路径下所有文件包含的itemID做推荐计算 (该选项可选)
* --filterFile (path) : 指定一个路径，该路径下的文件包含了[userID,itemID] 值对，userID和itemID用逗号分隔。计算结果将不会为user推荐 [userID,itemID] 值对中包含的item (该选项可选)
* --booleanData (boolean) : 如果输入数据不包含偏好数值，则将该参数设置为true，默认为false
* --maxPrefsPerUser (integer) : 在最后计算推荐结果的阶段，针对每一个user使用的偏好数据的最大数量，默认为10
* --minPrefsPerUser (integer) : 在相似度计算中，忽略所有偏好数据量少于该值的用户，默认为1
* --maxSimilaritiesPerItem (integer) : 针对每个item的相似度最大值，默认为100
* --maxPrefsPerUserInItemSimilarity (integer) : 在item相似度计算阶段，针对每个用户考虑的偏好数据最大数量，默认为1000
* --similarityClassname (classname) : 向量相似度计算类
* outputPathForSimilarityMatrix ：SimilarityMatrix输出目录
* --randomSeed ：随机种子 -- sequencefileOutput ：序列文件输出路径
* --tempDir (path) : 存储临时文件的目录，默认为当前用户的home目录下的temp目录
* --threshold (double) : 忽略相似度低于该阀值的item对

## 执行结果

上面命令运行完成之后，会在当前用户的hdfs主目录生成temp目录，该目录可由 --tempDir (path) 参数设置

