第9章 大数据分析综合案例

（PPT版本号：2022--V1）

CONTENTS

9.1
案例任务

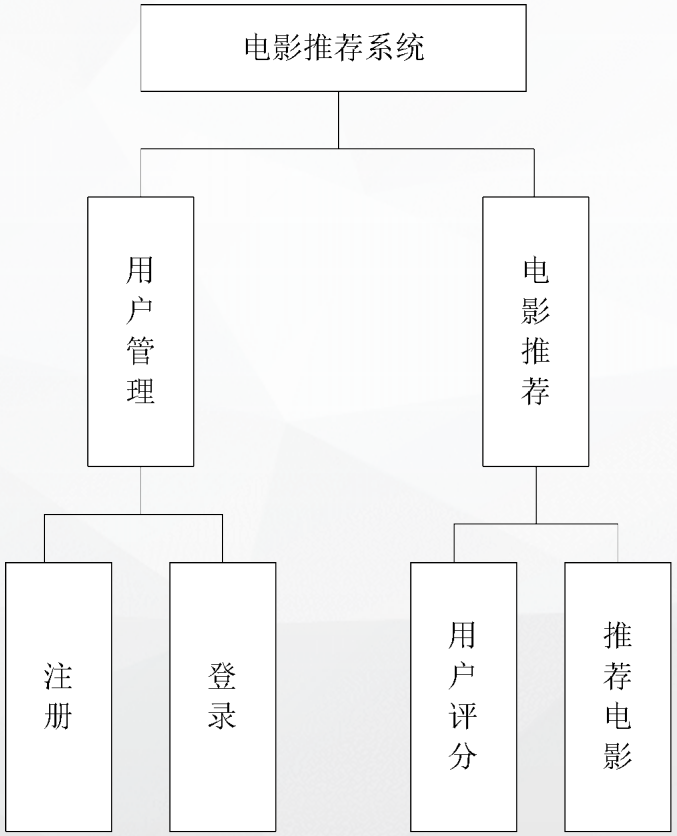
9.1 案例任务

网络上的电影信息在迅速增长，随着时间的推进，电影信息越来越多，用户面对如此庞大的数据，将变得无所适从。电影推荐系统可以根据用户的喜好（用户对一些电影的评分），向用户推荐可能感兴趣的电影，这样可以给用户创造一个良好的电影信息推荐体验，让用户不再漫无目的地去寻找符合自己口味的电影。

电影推荐系统的功能包括用户管理功能和电影推荐功能（如图所示）。用户管理功能是系统的基础功能，包括注册功能和登录功能。注册是第一次使用本系统的用户的必要步骤，用户进入注册界面，按要求填写相关信息，即用户自定义登录账号和密码，完成新用户注册。在用户成功注册后，即可使用系统的登录功能，用户可在登录界面填写账号和密码，验证成功后可进入推荐系统。

9.1 案例任务

电影推荐功能是系统的核心功能，用户登录成功以后，系统自动随机挑选一些电影呈现给用户，由用户根据自己个人喜好对电影进行评分，然后，系统会根据用户的评分信息，调用Spark程序计算出用户最可能感兴趣的几部电影，并在网页中为用户呈现精美的电影图片。



9.2 系统设计

9.2.1 系统总体设计

9.2.2 数据库设计

9.2.3 系统网站的设计

9.2.4 算法设计

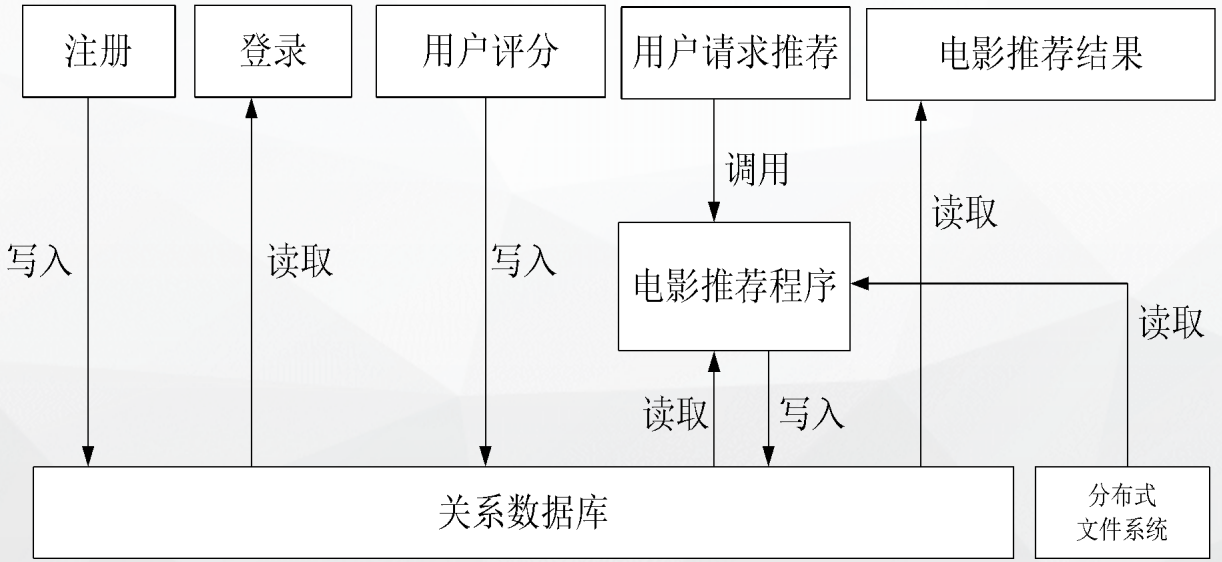
9.2.1 系统总体设计

电影推荐系统的设计开发工作包括网站、电影推荐程序和数据库三个部分：

- 网站：搭建一个网站，提供用户管理和电影推荐功能；
- 电影推荐程序：开发电影推荐程序，结合大规模历史数据集和用户个人喜好，为用户推荐其可能感兴趣的电影；
- 数据库：设计一个关系数据库，存放用户信息、电影信息、用户评分信息和电影推荐结果信息。

9.2.1 系统总体设计

下图描述了网站、电影推荐程序和数据库三个部分之间的关系。当用户注册时，新用户的用户名和密码会被写入到数据库中；当用户登录时，系统会到数据库读取用户名和登录密码进行验证；当用户评分时，用户对多部电影的评分信息会被写入到数据库；当用户请求推荐电影时，网页会调用电影推荐程序，电影推荐程序会读取数据库中的该用户的个性化评分数据，并从分布式文件系统中读取大规模历史评分数据，计算得到推荐结果（比如5部用户最感兴趣的电影），并把推荐结果写入到数据库；最后，由网页程序从数据库中读取推荐结果呈现到网页中。



9.2.2数据库设计

本系统使用关系数据库保存用户信息、电影信息、用户评分信息和电影推荐结果信息。需要创建一个数据库movierecommend，并在数据库中创建4个表，即电影信息表movieinfo、用户信息表user、用户评分表personalratings以及电影推荐结果表recommendresult。 各个表的字段如下：

（1）电影信息表movieinfo：电影ID、电影名称、电影上映时间、电影导演、主要演员、电影宣传海报、电影的平均评分、参与电影评分的人数、电影简介、电影类型。

（2）用户信息表user：用户ID、用户名、用户登录密码。

（3）用户评分表personalratings：用户ID、电影ID、用户对电影的评分、评分时间。

（4）电影推荐结果表recommendresult：用户ID、电影ID、电影评分、电影名称。

9.2.3 系统网站的设计

如图所示，电影推荐系统网站主要包括首页、登录页面、注册页面、用户评分页面和电影推荐结果页面。

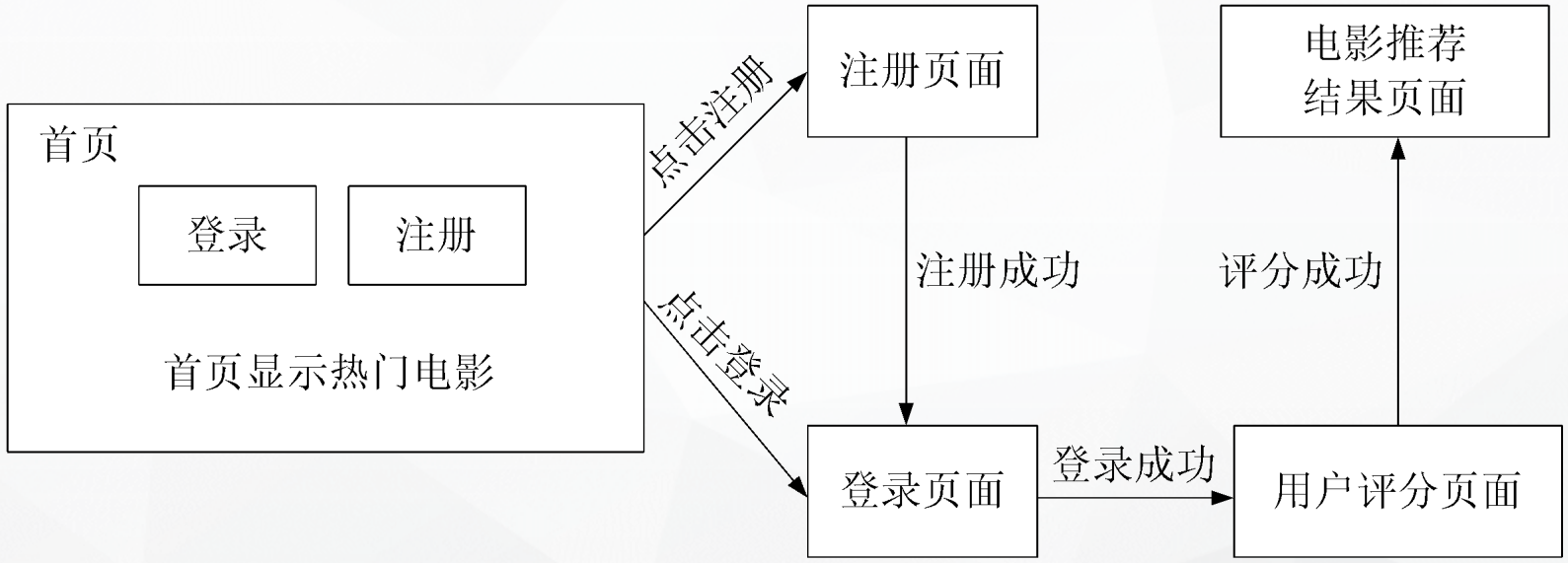


图 电影推荐系统的网页跳转示意图

9.2.3 系统网站的设计

在推荐系统网站首页中（如图所示），会滚动显示当前热播的几部电影。同时，在页面的右上角，提供了“登录”和“注册”按钮，当用户点击时，分别会跳转到“登录”和“注册”页面。

9.2.3 系统网站的设计

在用户评分成功以后，系统会调用后端的电影推荐程序计算得到电影推荐结果，并把推荐结果写入到数据库。然后，系统从数据库中读取推荐结果，并以可视化方式呈现在“电影推荐结果页面”上（如图所示）。

9.2.4算法设计

1.算法的选择

电影推荐程序的设计，是电影推荐系统设计的核心。电影推荐程序需要完成推荐功能，因此，可以选择经典的推荐算法——协同过滤。基于ALS矩阵分解的协同过滤算法就是典型的基于模型的协同过滤算法。基于模型的协同过滤算法是通过已经观察到的所有用户给产品的打分，来推断每个用户的喜好并向用户推荐适合的产品。本系统就采用该算法来预测某个用户对某部电影的评分，并把评分高的电影推荐给该用户。

9.2.4算法设计

2. ALS算法的基本原理

在实际应用中，用户和商品的关系可以抽象为一个三元组<User,Item,Rating>

用户对物品的评分行为可以表示成一个评分矩阵A(m*n)，表示m个用户对n个物品的评分情况

	v1	v2	v3	v4	v5
u1	3	5	4	?	1
u2	4	?	3	3	1
u3	3	4	5	3	2
u4	4	4	3	2	1
u5	2	4	?	1	2
u6	?	5	4	1	2

其中，矩阵A的每个元素A_{ij}，表示用户u_i对物品v_j的评分

9.2.4算法设计

•在实际应用场景中，一方面，m和n的数值都很大，矩阵A的规模很容易就会突破1亿项，这时，传统的矩阵分解方法面对如此巨大的数据量往往会显得“无能为力”；另一方面，用户不会对所有的物品都进行评分，因此，在矩阵A中会不可避免地存在一些“缺失值”（表中用问号标记的地方），这意味着A是一个稀疏矩阵，里面会存在很多空值。基于模型的协同过滤算法就是根据已经观察到的用户、物品信息来预测矩阵A中的“缺失值”。

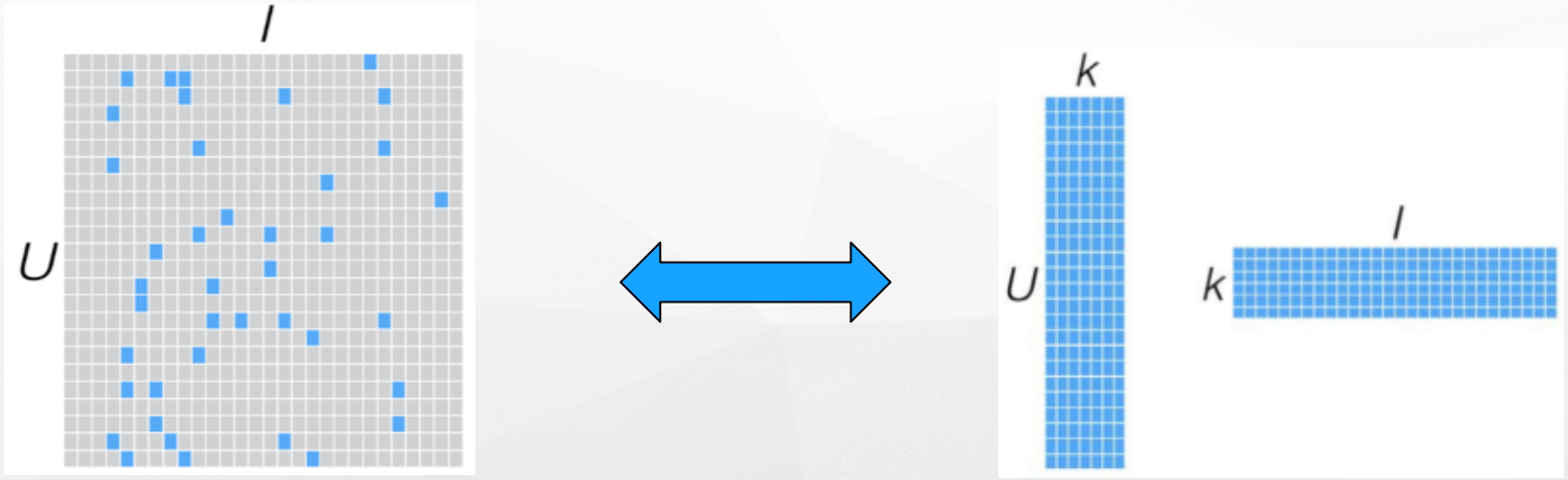
•“用户-物品”矩阵A中的元素A_{ij}是用户给予物品的显式偏好，例如，用户根据自己的喜好对电影进行评分。然而，在现实世界中使用时，我们常常只能访问隐式反馈（如意见、点击、购买、喜欢以及分享等），而无法获得用户对物品的直接评分。基于ALS矩阵分解的协同过滤算法，不是直接对评分矩阵进行建模，而是根据隐式反馈（如意见、点击、购买、喜欢以及分享等），来衡量用户喜好某个物品的置信水平，从而得到“用户-物品”矩阵A中的“缺失值”。

9.2.4算法设计

例如，将用户(User)对物品(Item)的评分矩阵分解为两个矩阵：一个是用户对物品隐含特征的偏好矩阵，另一个是物品所包含的隐含特征的矩阵。具体而言，将用户-物品的评分矩阵分解成两个隐含因子矩阵*P*和*Q*，从而将用户和物品都投影到一个隐含因子的空间中去。即对于*R*(*m*×*n*)的矩阵，ALS旨在找到两个低维矩阵*P*(*m*×*k*)和矩阵*Q*(*n*×*k*)，来近似逼近*R*(*m*×*n*)：

$$R_{m\times n} \simeq P_{m\times k} Q^{T}_{n\times k}$$

其中，*k*<<min(*m*, *n*)。这里相当于降维了，矩阵*P*和*Q*也称为低秩矩阵。



9.2.4算法设计

ALS是求解的著名算法，基本思想是：固定其中一类参数，使其变为单类变量优化问题，利用解析方法进行优化；再反过来，固定先前优化过的参数，再优化另一组参数；此过程迭代进行，直到收敛。ALS即“最小交替二乘法”中的“交替”，就是指我们先随机生成P0，然后固定P0去求解Q0，再固定Q0求解P1，这样交替进行下去。因为每步迭代都会降低重构误差，并且误差是有下界的，所以ALS一定会收敛。

9.2.4算法设计

3.使用ALS算法预测用户对电影的评分

在本案例中，我们可以获得一个电影评分数据集，里面包含了大量用户对不同电影的评分，用户和商品的关系三元组<*User*,*Item*,*Rating*>中，User就是参与打分的电影观众，Item就是电影，Rating就是观众对电影的打分。在我们获取到的电影评分数据集中，只包含了大量用户对他已经看过的电影的评分，对于没有看过的电影，用户是无法评分的，因此，由这个数据集构建得到的矩阵肯定是一个稀疏矩阵。我们可以使用电影评分数据集对ALS算法进行训练，训练的过程就是寻找低秩矩阵*P*和*Q*、使*P*和*Q*尽可能地逼近*R*的过程。算法训练过程达到收敛以后，得到一个模型，然后，就可以使用这个模型进行评分预测，也就是给定一个用户和一部电影，该模型就会预测该用户可能给该电影打多少分。

9.3
技术选择

9.3技术选择

为了实现之前给出的系统设计方案，需要选择确定相关的实现技术具体如下：
（1）数据集。通过网络爬虫从网络获得电影评分数据集，这里使用Scrapy爬虫。
（2）操作系统。在构建大数据分析系统时，一般建议采用Linux系统，Linux系统有许多发行版，这里选择Ubuntu。
（3）关系数据库。在电影推荐系统中，需要使用关系数据库来存储用户信息、电影信息、用户评分信息和电影推荐结果信息。目前，市场上有许多关系数据库产品，比如Oracle、SQL Server、DB2、MySQL等，这里选择开源数据库产品MySQL。
（4）分布式文件系统。用来保存大量的电影评分数据，这里选择Hadoop的HDFS。
（5）ETL。用来把通过网络爬虫得到的、保存在文本文件中的数据，经过数据清洗以后加载到分布式文件系统中，这里采用开源ETL工具Kettle。

9.3技术选择

（6）分布式计算框架。对于机器学习算法而言，传统的机器学习算法，由于技术和单机存储的限制，只能在少量数据上使用。在面对大规模数据集时，需要实现分布式机器学习算法，因此，这里采用机器学习框架Spark MLlib。MLlib提供了ALS算法的分布式实现，也就是说，开发人员不需要编写ALS算法的实现细节，只要调用MLlib提供的ALS算法的API，传入训练数据（电影评分数据集），MLlib就会自动完成模型训练，训练得到的模型就可以用于预测某个用户对某部电影的评分。

（7）编程语言。Spark MLlib支持Java、Scala和Python，这里采用Scala语言。

（8）开发工具。调试Spark程序，需要一款高效的开发工具，这里选择IntelliJ IDEA。

（9）网站。网站开发采用Node.js。Node.js是一个JavaScript运行环境，是一个基于Chrome JavaScript运行时建立的平台，用于方便地搭建响应速度快、易于扩展的网络应用。Node.js使用事件驱动、非阻塞I/O模型，具备轻量和高效的特点，非常适合在分布式设备上运行数据密集型的实时应用。

9.3技术选择

系统实现技术

项目	技术
数据集	Scrapy爬虫
操作系统	Linux系统，比如Ubuntu
关系数据库	MySQL
分布式文件系统	HDFS
ETL	Kettle
分布式计算框架	Spark MLlib
编程语言	Scala
开发工具	IntelliJ IDEA
网站	Node.js

9.3技术选择

通过上述技术选择以后，本案例的数据分析全流程就十分清晰了，整体过程如图所示。

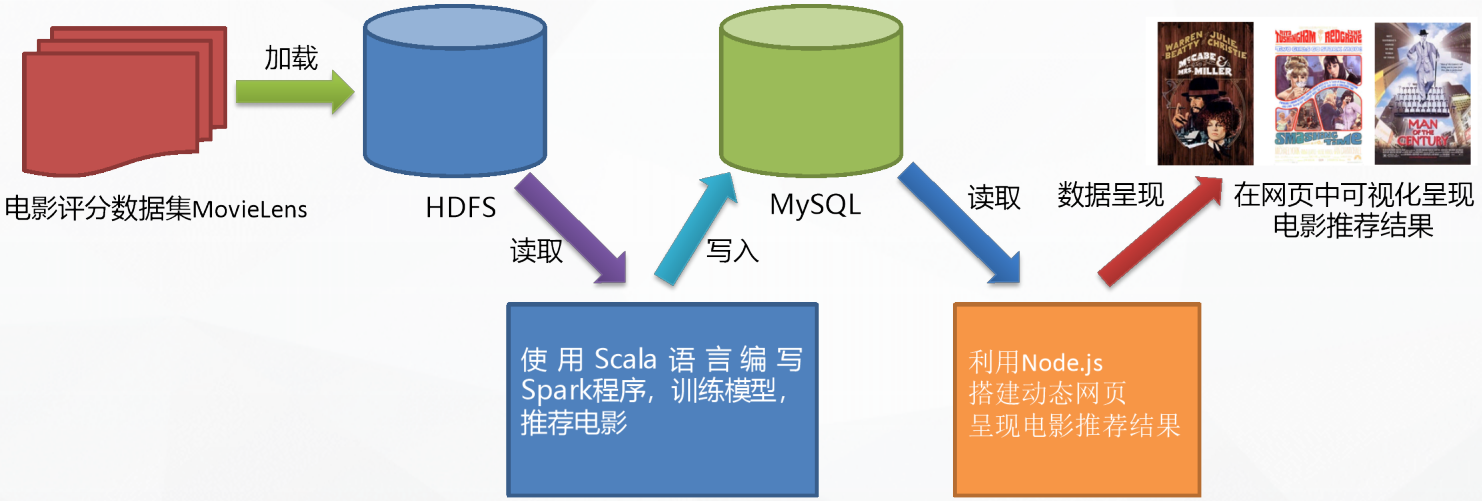


图 本案例的数据分析整体过程

9.4
系统实现

9.4 系统实现

系统实现是一个比较复杂的过程，这里只简单介绍系统实现过程中所涉及到的一些任务，关于系统实现的详细细节可以参考《大数据实训案例之电影推荐系统（Scala版）》。系统实现涉及的主要任务如下：

（1）搭建环境。安装Linux系统、JDK、关系型数据库MySQL、大数据软件Hadoop、大数据软件Spark、开发工具IntelliJ IDEA、ETL工具Kettle和Node.js；

（2）采集数据。编写Scrapy爬虫从网络上获取电影评分数据；

（3）加载数据。使用ETL工具Kettle对数据进行清洗后加载到分布式文件系统HDFS中。

（4）存储和管理数据。使用分布式文件系统HDFS和关系数据库MySQL对数据进行存储和管理。

（5）处理和分析。使用Scala语言和开发工具IntelliJ IDEA，编写Spark MLlib程序，根据HDFS中的大量数据进行模型训练（即使用数据对ALS算法进行训练得到模型），然后，使用训练得到的模型进行电影评分预测，并为用户推荐评分高的电影。

（6）可视化。使用Node.js搭建网站，接受用户访问，并以可视化方式呈现电影推荐结果。

9.5

案例所需知识和技能

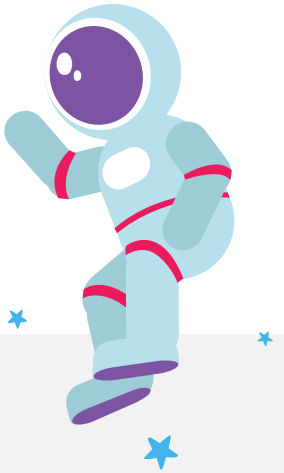
9.5 案例所需知识和技能

•总体而言，要实现本案例，我们需要掌握以下知识：Linux操作系统、关系数据库、JDK基本知识、面向对象编程、Scala编程语言、网络爬虫、数据清洗、分布式文件系统、Spark、Spark SQL、Spark Mllib、JDBC、机器学习、数据挖掘、推荐系统、协同过滤算法、ASL算法、网页应用程序开发、HTML语言、数据可视化、系统设计等。
•从专业技能的角度，我们需要掌握：Linux系统及相关软件的安装和使用方法、JDK的安装、Hadoop的安装和基本使用方法、Spark的安装和基本使用方法、MySQL数据库的安装和基本使用方法、开发工具InteliJ IDEA的安装和使用方法、Scala程序开发方法、软件项目管理工具Maven的使用方法、ETL工具Kettle的安装和使用方法、Spark SQL程序的开发方法、ALS算法的使用方法、Spark MLlib程序开发方法、Node.js的安装和使用Node.js开发动态网页的方法等。

本章小结

学习大量的大数据理论和技术，是为了帮助我们解决实际具体问题的。而如何对所学知识进行融会贯通，有效地应用到问题的解决中，通常是一个复杂繁琐的过程，这个过程不仅需要细致认真的思考，更需要不断的摸索实践。为了帮助读者搭建起理论联系实际的桥梁，本章给出了一个简单而具有代表性的案例，这个案例是一个复杂系统的“微缩版”，但是，“麻雀虽小、五脏俱全”，通过这个案例的学习，相信可以帮助读者建立对大数据分析全流程的“感知性”认识，为以后更好、更有针对性地开展各门专业课的学习，打下坚实的基础。

大数据与统计系 | 大数据系列课程



大数据

大数据与统计系数字教师网
http://hssj.wxcgi.com/