

威 海 职 业 学 院

# 毕 业 设 计 任 务 书

专业           计算机应用技术          

年级       2016 级       班级       二班      

姓名       孔 倩       学号       20160101054      

      张智超             20160101056      

      牟志茹             20160101060      

      周新军             20160101076      

      邓诗松             20160101083      

威 海 职 业 学 院 教 务 处 编 印

## 毕业设计指导须知

一、毕业设计是高职教学过程中一个十分重要的环节。是锻炼学生运用所学知识正确分析和解决实际问题的一个重要方面，也是高职培养应用型专门人才的要求。

二、指导教师应为具有讲师以上或相应职称的有关专业人员，且专业对口（指所指导专业应同所聘教师专业职称相一致）。经系、教务处审查同意后，才能指导学生的毕业设计。

三、学生应以严肃认真，实事求是的态度完成设计。要独立思考，自己动手，不得抄袭或找人代笔。

四、毕业设计选题要符合专业培养目标的要求。论文（任务书）写作要做到论点明确、论据充分、论理透彻，语言准确恰当，书面整洁、字迹工整，图纸应清晰、工整，符合设计要求，符合国家有关标准和部颁标准。字数、图纸数量符合有关要求。并在规定的时间内完成。

五、答辩过程中学生要严谨认真，文明礼貌，谦虚谨慎，认真回答答辩主持人，委员等提出的问题。

六、填报有关表格时，应按项目要求逐项填实、填全、填清。

设计题目	基于 Hadoop 平台的豆瓣影视作品数据分析						
专业	计算机应用技术	年级	16 级	班级	二班	学制	三年
指导教师	李晓粉	职称		助教	班主任	彭丹	
一、小组成员组成及分工：							
学号	姓名	项目角色		工作描述			
20160101076	周新军	组长		数据收集、代码编写与调试			
20160101054	孔倩	组员		资料收集、数据整理、论文整理			
20160101056	张智超	组员		资料收集、分析报告			
20160101060	牟志茹	组员		资料整理、数据处理			
20160101083	邓诗松	组员		资料收集、平台搭建			
二、毕业设计（论文）进度计划							
起止时间		工作内容			备注		
2018/03/09--2018/03/20		确定毕业设计课题，收集资料			我们小组成员积极参与毕业论文，认真拟定毕业设计方 案、计划，遇到不懂的问题虚心向指导老师请教。通过李晓粉老师的精心指导和小组成员的团结协作，我们成功完成毕业设计任务。		
2018/03/21--2018/03/24		明确设计内容，进行小组分工					
2018/03/25--2018/03/31		数据平台搭建					
2018/04/01--2018/04/15		数据抓取					
2018/04/16--2018/04/30		数据分析与建模					
2018/05/01--2018/05/07		数据存储与处理					
2018/05/08--2018/05/15		数据可视化（制图）					
2018/05/16--2018/05/20		编写数据分析					
2018/05/21--2018/06/15		汇总与论文编写、查错					

# 答辩情况记录

答辩人	答辩题目	答辩情况			
		正确	基本正确	经提示回答正确	不正确
	答辩总体评价：				
	答辩总体情况：				
	答辩总体评价：				

	答辩总体评价：				
	答辩总体评价：				
	答辩总体评价：				
此表有主持答辩的同志填写：					
答辩委员会（或小组）评语：					
成绩：      主持答辩人签名：      职称：      月      日					

指导教师评语：

成绩：                      教师签名：                      年        月        日

系复审查意见：

成绩：              复审人签名：              职称：              公章        年        月        日

教务处终审意见：

公章                      年        月        日

# 基于 Hadoop 平台的豆瓣影视作品数据分析

## 摘 要

随着科技的发展以及生活的水平不断提高，越来越多的人喜欢去电影院看电影，也随着影视的发展，越来越多的作品出现在人们面前，其作品却是良莠不齐。为了能够帮助影视行业实现更精准的用户定位和市场分析，以生产出内容优质、符合大众口味的高质量影视产品，这就需要对以往的电影数据进行分析总结。本论文主要讲述了基于 Hadoop 平台来分析豆瓣大量的影视作品数据。

**关键词** Hadoop 数据抓取 数据分析 Python MapReduce

# 目录

0.	前言 .....	1
1.	影视作品数据分析设计需求分析 .....	2
1.1.	问题定义 .....	2
1.2.	Hadoop 概述 .....	2
1.2.1.	Hadoop 起源 .....	2
1.2.2.	Hadoop 平台的优缺点 .....	3
1.2.3.	Hadoop 生态系统 .....	3
1.2.4.	Hadoop 应用 .....	5
1.3.	Python 概述 .....	5
1.3.1.	Python 起源 .....	6
1.3.2.	Python 的优缺点 .....	6
2.	数据平台搭建 .....	9
2.1.	安装虚拟化软件 VMware Workstation Pro .....	9
2.2.	安装操作系统 Ubuntu .....	12
2.3.	配置 Ubuntu 系统环境 .....	21
2.4.	安装配置 Hadoop .....	22
3.	数据抓取 .....	25
3.1.	收集 url .....	25
3.2.	获得页面信息 .....	27
4.	数据分析与建模 .....	31
4.1.	编写统计每年作品数量的 MapReduce 程序 .....	31
4.2.	编写统计作品类型的 MapReduce 程序 .....	31
4.3.	编写作品制作地区的 MapReduce 程序 .....	33
4.4.	编写处理高评价的电影处理程序 .....	34
4.5.	编写处理低评价的电影处理程序 .....	34
4.6.	编写作品评分数量的 MapReduce 程序 .....	34
4.7.	编写作品类型平均评分的程序 .....	35
5.	数据存储与处理 .....	36
5.1.	在 HDFS 上创建 input 和 output 目录 .....	36
5.2.	将收集处理完的数据上传到 HDFS .....	36
5.3.	进行本地测试 .....	36
5.4.	使用 Hadoop 平台处理程序 .....	36
5.5.	使用 Python 程序本地处理数据 .....	36
6.	数据可视化与分析 .....	38
6.1.	导出 HDFS 上的数据 .....	38



6.2.	安装数据可视化软件 IPython .....	38
6.3.	安装依赖软件包 .....	38
6.4.	打开数据可视化软件 iPython .....	38
6.5.	所得效果图与分析.....	39
6.6.	分析报告 .....	46
7.	总结与展望.....	47
7.1.	论文总结 .....	47
7.2.	工作展望 .....	49
参考文献、资料索引 .....		50
致 谢.....		51

## 0. 前言

随着云时代的来临，大数据（Big data）也吸引了越来越多的关注。大数据目前已经成为 IT 领域最为流行的词汇，其实它并不是一个全新的概念。早在 1980 年，著名未来学家阿尔文·托夫勒便在《第三次浪潮》一书中，明确提出“数据就是财富”这一观点，并将大数据热情地赞颂为“第三次浪潮的华彩乐章”。直到现在，大数据在政府决策部门、行业企业、研究机构等得到了广泛的应用，并实际创造了价值。大数据分析相比于传统的数据仓库应用，具有数据量大、查询分析复杂等特点。本论文以 Hadoop 平台为基础以豆瓣影视作品为数据，进行的一次大数据分析。

Hadoop 是 Apache 软件基金会旗下的一个开源分布式计算平台。以 Hadoop 分布式文件系统和 MapReduce 为核心的 Hadoop 为用户提供了系统底层细节透明的分布式基础架构。HDFS 的高容错性、高伸缩性等优点允许用户将 Hadoop 部署在低廉的硬件上，形成分布式系统，MapReduce 分布式编程模型允许用户在不了解分布式系统底层细节的情况下开发并行应用程序。所以用户可以利用 Hadoop 轻松地组织计算机资源，从而搭建自己的分布式计算平台，并且可以充分利用集群的计算和存储能力，完成海量数据的处理。

Hadoop 实现了一个分布式文件系统（Hadoop Distributed File System），简称 HDFS。HDFS 有高容错性的特点，并且设计用来部署在低廉的（low-cost）硬件上；而且它提供高吞吐量（high throughput）来访问应用程序的数据，适合那些有着超大数据集（large data set）的应用程序。

本课题就是介绍什么是 Hadoop，如何搭建 Hadoop，如何使用 Hadoop 处理数据，如何使用 Python 语言编写 MapReduce 程序，以及使用 iPython 实现数据可视化，最后是数据分析报告。

准备工作主要是下载需要的软件，VMware workstation, Ubuntu16.04.iso 镜像文件, Hadoop2.7.3 安装包。

## 1. 影视作品数据分析设计需求分析

### 1.1. 问题定义

在上次观看了战狼 2 之后，开始进入了豆瓣的影视世界，对于一个新用户来说，特别喜欢豆瓣电影的影评，对于所看过的留下深刻印象的电影都在豆瓣上搜了一边，同一部电影，希望能看到与自己感受不同的观点，因为每一位观众都有着不一样的人生轨迹，看待事物的角度可能会有所不同，正是这种差异往往能引发思考。豆瓣的影视数据方面一直以来都比较权威。还有一个网站是 IMDB，我不认为 IMDB 的影评更加符合中国人。就像战狼 2 的影评，在豆瓣上评分是 7.1 分，在 IMDB 上评分是 6.2 分，豆瓣中评论的数量是 50W，而 IMDB 中只有 4K 多条。这也是选择豆瓣电影的影视数据作为数据的原因。随着不断的了解豆瓣，心中就产生了一些疑问，最受人喜欢的影视作品是有哪些，最不喜欢的影视作品又有哪些，最喜欢的影视作品类型有哪些，盛产影视作品的国家又有哪些等等。

考虑到这些数据不是特别大，使用现有的资源。最终决定使用 Hadoop 平台+Python 语言来分析豆瓣的影视作品。

### 1.2. Hadoop 概述

#### 1.2.1. Hadoop 起源

Hadoop 是一个由 Apache 基金会所开发的分布式系统基础架构。

用户可以在不了解分布式底层细节的情况下，开发分布式程序。充分利用集群的威力进行高速运算和存储。

Hadoop 实现了一个分布式文件系统（Hadoop Distributed File System），简称 HDFS。HDFS 有高容错性的特点，并且设计用来部署在低廉的（low-cost）硬件上；而且它提供高吞吐量（high throughput）来访问应用程序的数据，适合那些有着超大数据集（large data set）的应用程序。HDFS 放宽了（relax）POSIX 的要求，可以以流的形式访问（streaming access）文件系统中的数据。

Hadoop 的框架最核心的设计就是：HDFS 和 MapReduce。HDFS 为海量的数据提供了存储，则 MapReduce 为海量的数据提供了计算。

所以用户可以利用 Hadoop 轻松地组织计算机资源，从而搭建自己的分布式

计算平台，并且可以充分利用集群的计算和存储能力，完成海量数据的处理。

#### 1.2.2. Hadoop 平台的优缺点

优点：

1. 高可靠性。Hadoop 按位存储和处理数据的能力值得人们信赖。
2. 高扩展性。Hadoop 是在可用的计算机集簇间分配数据并完成计算任务的，这些集簇可以方便地扩展到数以千计的节点中。
3. 高效性。Hadoop 能够在节点之间动态地移动数据，并保证各个节点的动态平衡，因此处理速度非常快。
4. 高容错性。Hadoop 能够自动保存数据的多个副本，并且能够自动将失败的任务重新分配。
5. 低成本。与一体机、商用数据仓库以及 QlikView、Yonghong Z-Suite 等数据集市相比，Hadoop 是开源的，项目的软件成本因此会大大降低。

缺点：

1. 不能做到低延迟。由于 Hadoop 针对高数据吞吐量做了优化，牺牲了获取数据的延迟，所以对于低延迟数据访问，不适合 Hadoop，对于低延迟的访问需求，HBase 是更好的选择。
2. 不适合大量的小文件存储。由于 namenode 将文件系统的元数据存储在内存中，因此该文件系统所能存储的文件总数受限于 namenode 的内存容量，根据经验，每个文件、目录和数据块的存储信息大约占 150 字节。因此，如果大量的小文件存储，每个小文件会占一个数据块，会使用大量的内存，有可能超过当前硬件的能力。
3. 不适合多用户写入文件，修改文件。Hadoop2.0 虽然支持文件的追加功能，但是还是不建议对 HDFS 上的文件进行修改，因为效率低。对于上传到 HDFS 上的文件，不支持修改文件，HDFS 适合一次写入，多次读取的场景。HDFS 不支持多用户同时执行写操作，即同一时间，只能有一个用户执行写操作。

#### 1.2.3. Hadoop 生态系统

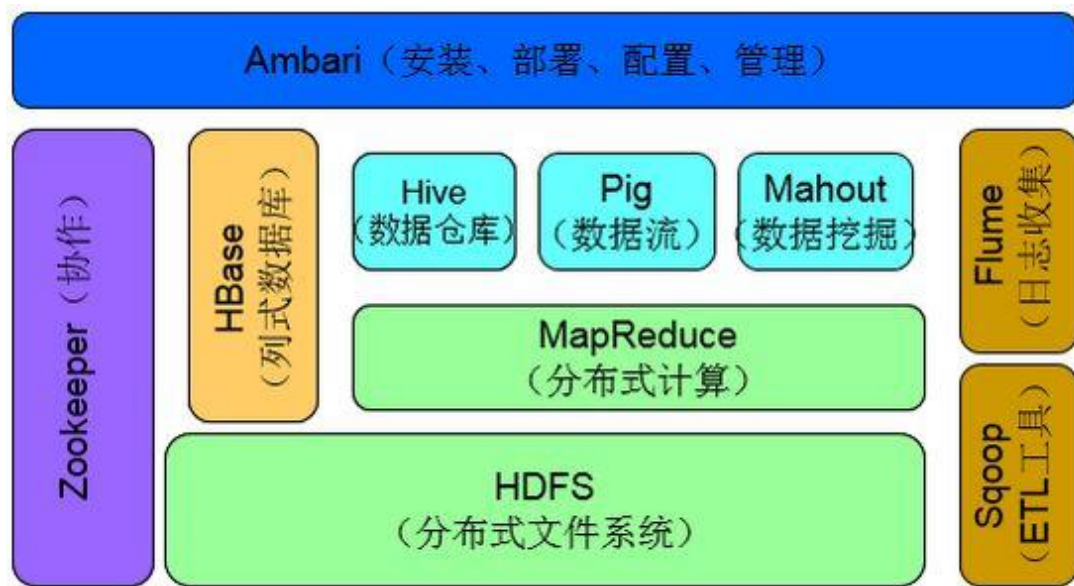


图 1-1 Hadoop 生态系统

1. Hadoop Common:

在 0.20 及以前的版本中，包含 HDFS、MapReduce 和其他项目公共内容，从 0.21 开始 HDFS 和 MapReduce 被分离为独立的子项目，其余内容为 Hadoop Common。

2. HDFS:

Hadoop 分布式文件系统 (Distributed File System) — HDFS。

3. MapReduce:

并行计算框架，0.20 前使用 `org.apache.Hadoop.mapred` 旧接口，0.20 版本开始引入 `org.apache.Hadoop.MapReduce` 的新 API。

4. HBase:

类似 Google BigTable 的分布式 NoSQL 列数据库。

5. Hive:

数据仓库工具，由 Facebook 贡献。

6. Zookeeper:

分布式锁设施，提供类似 Google Chubby 的功能，由 Facebook 贡献。

7. Avro:

新的数据序列化格式与传输工具，将逐步取代 Hadoop 原有的 IPC 机制。

8. Pig:

大数据分析平台，为用户提供多种接口。

9. Ambari:

Hadoop 管理工具，可以快捷的监控、部署、管理集群。

#### 10. Sqoop:

在 Hadoop 与传统的数据库间进行数据的传递。

#### 1.2.4. Hadoop 应用

百度:

百度在 2006 年就开始关注 Hadoop 并开始调研和使用,在 2012 年其总的集群规模达到近十个,单集群超过 2800 台机器节点, Hadoop 机器总数有上万台机器,总的存储容量超过 100PB,已经使用的超过 74PB,每天提交的作业数目有数千个之多,每天的输入数据量已经超过 7500TB,输出超过 1700TB。

阿里巴巴:

阿里巴巴的 Hadoop 集群截至 2012 年大约有 3200 台服务器,大约 30000 物理 CPU 核心,总内存 100TB,总的存储容量超过 60PB,每天的作业数目超过 1500000 个,每天 hivequery 查询大于 6000 个,每天扫描数据量约为 7.5PB,每天扫描文件数约为 4 亿,存储利用率大约为 80%,CPU 利用率平均为 65%,峰值可以达到 80%。阿里巴巴的 Hadoop 集群拥有 150 个用户组、4500 个集群用户。

腾讯:

腾讯也是使用 Hadoop 最早的中国互联网公司之一,截至 2012 年年底,腾讯的 Hadoop 集群机器总量超过 5000 台,最大单集群约为 2000 个节点,并利用 Hadoop-Hive 构建了自己的数据仓库系统 TDW,同时还开发了自己的 TDW-IDE 基础开发环境。腾讯的 Hadoop 为腾讯各个产品线提供基础云计算和云存储服务。

Facebook:

Facebook 使用 Hadoop 存储内部日志与多维数据,并以此作为报告、分析和机器学习的数据源。目前 Hadoop 集群的机器节点超过 1400 台,共计 11?200 个核心 CPU,超过 15PB 原始存储容量,每个商用机器节点配置了 8 核 CPU, 12TB 数据存储,主要使用 StreamingAPI 和 JavaAPI 编程接口。Facebook 同时在 Hadoop 基础上建立了一个名为 Hive 的高级数据仓库框架, Hive 已经正式成为基于 Hadoop 的 Apache 一级项目。此外,还开发了 HDFS 上的 FUSE 实现。个产品线提供基础云计算和云存储服务。

### 1.3. Python 概述

### 1.3.1. Python 起源

Python，是一种面向对象的解释型计算机程序设计语言，由荷兰人 Guido van Rossum 于 1989 年发明，第一个公开发行人版发行于 1991 年。Python 是纯粹的自由软件，源代码和解释器 CPython 遵循 GPL (GNU General Public License) 协议。Python 语法简洁清晰，特色之一是强制用空白符(white space)作为语句缩进。

Python 名字的来源：1989 年圣诞节期间，在阿姆斯特丹，Guido 为了打发圣诞节的无趣，决心开发一个新的脚本解释程序，做为 ABC 语言的一种继承。之所以选中 Python（大蟒蛇的意思）作为该编程语言的名字，是因为他是一个叫 Monty Python 的喜剧团体的爱好者。

ABC 是由 Guido 参加设计的一种教学语言。就 Guido 本人看来，ABC 这种语言非常优美和强大，是专门为非专业程序员设计的。但是 ABC 语言并没有成功，究其原因，Guido 认为是其非开放造成的。Guido 决心在 Python 中避免这一错误。同时，他还想实现在 ABC 中闪现过但未曾实现的东西。就这样，Python 在 Guido 手中诞生了。Python 已经成为最受欢迎的程序设计语言之一。2011 年 1 月，它被 TIOBE 编程语言排行榜评为 2010 年度语言。自从 2004 年以后，python 的使用率呈线性增长。

### 1.3.2. Python 的优缺点

优点：

1. 简单：Python 是一种代表简单主义思想的语言。阅读一个良好的 Python 程序就感觉像是在读英语一样。它使你能够专注于解决问题而不是去搞明白语言本身。

2. 易学：Python 极其容易上手，因为 Python 有极其简单的说明文档。

3. 速度快：Python 的底层是用 C 语言写的，很多标准库和第三方库也都是用 C 写的，运行速度非常快。

4. 免费、开源：Python 是 FLOSS（自由/开放源码软件）之一。使用者可以自由地发布这个软件的拷贝、阅读它的源代码、对它做改动、把它的一部分用于新的自由软件中。FLOSS 是基于一个团体分享知识的概念。

5. 高层语言：用 Python 语言编写程序的时候无需考虑诸如如何管理你的程

序使用的内存一类的底层细节。

6. 可移植性：由于它的开源本质，Python 已经被移植在许多平台上（经过改动使它能够工作在不同平台上）。这些平台包括 Linux、Windows、FreeBSD、Macintosh、Solaris、OS/2、Amiga、AROS、AS/400、BeOS、OS/390、z/OS、Palm OS、QNX、VMS、Psion、Acom RISC OS、VxWorks、PlayStation、Sharp Zaurus、Windows CE、PocketPC、Symbian 以及 Google 基于 linux 开发的 android 平台。

7. 解释性：一个用编译性语言比如 C 或 C++写的程序可以从源文件（即 C 或 C++语言）转换到一个你的计算机使用的语言（二进制代码，即 0 和 1）。这个过程通过编译器和不同的标记、选项完成。运行程序的时候，连接/转载器软件把你的程序从硬盘复制到内存中并且运行。而 Python 语言写的程序不需要编译成二进制代码，你可以直接从源代码运行程序。

8. 易于移植：在计算机内部，Python 解释器把源代码转换成称为字节码的中间形式，然后再把它翻译成计算机使用的机器语言并运行。这使得使用 Python 更加简单，也使得 Python 程序更加易于移植。

9. 面向对象：Python 既支持面向过程的编程也支持面向对象的编程。在“面向过程”的语言中，程序是由过程或仅仅是可重用代码的函数构建起来的。在“面向对象”的语言中，程序是由数据和功能组合而成的对象构建起来的。

10. 可扩展性：如果需要一段关键代码运行得更快或者希望某些算法不公开，可以部分程序用 C 或 C++编写，然后在 Python 程序中使用它们。

11. 可嵌入性：可以把 Python 嵌入 C/C++程序，从而向程序用户提供脚本功能。

12. 丰富的库：Python 标准库确实很庞大。它可以帮助处理各种工作，包括正则表达式、文档生成、单元测试、线程、数据库、网页浏览器、CGI、FTP、电子邮件、XML、XML-RPC、HTML、WAV 文件、密码系统、GUI（图形用户界面）、Tk 和其他与系统有关的操作。这被称作 Python 的“功能齐全”理念。除了标准库以外，还有许多其他高质量的库，如 wxPython、Twisted 和 Python 图像库等等。

13. 规范的代码：Python 采用强制缩进的方式使得代码具有较好可读性。而 Python 语言写的程序不需要编译成二进制代码。

缺点：



1. 单行语句和命令行输出问题：很多时候不能将程序连写成一行，如 `import sys;for i in sys.path:print i`。而 perl 和 awk 就无此限制，可以较为方便的在 shell 下完成简单程序，不需要如 Python 一样，必须将程序写入一个 .py 文件。

2. 独特的语法：这也许不应该被称为局限，但是它用缩进来区分语句关系的方式还是给很多初学者带来了困惑。即便是很有经验的 Python 程序员，也可能陷入陷阱当中。

3. 运行速度慢：这里是指与 C 和 C++相比。

## 2. 数据平台搭建

为了使分析的数据更真实,结合手中的设备,将搭建 Hadoop 伪分布式平台。实验所需要准备的软件,如图 2-1 所示:

1. VMware-workstation-full-14.1.1.28517.exe, 虚拟化程序的安装包。
2. Ubuntu-16.04.3-desktop-amd64.iso, Ubuntu linux 系统安装镜像。
3. Hadoop-2.7.3.tar.gz, Hadoop 平台文件。

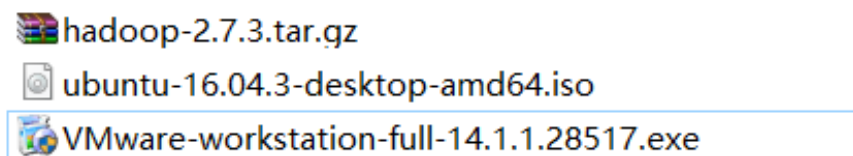


图 2-1 所需软件

### 2.1. 安装虚拟化软件 VMware Workstation Pro

在网上下载 VMware Workstation 14 Pro 安装包, 点击安装, 如图 2-2 所示:



图 2-2 下一步

勾选我接受许可协议中的条款(A), 如图 2-3 所示:

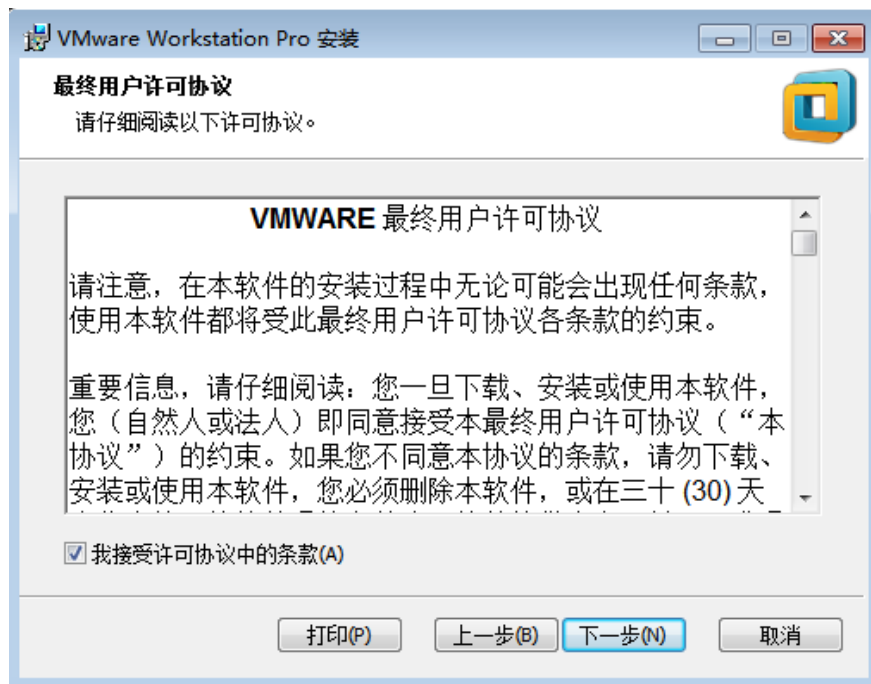


图 2-3 勾选许可协议

选择安装的目录，如图 2-4 所示：

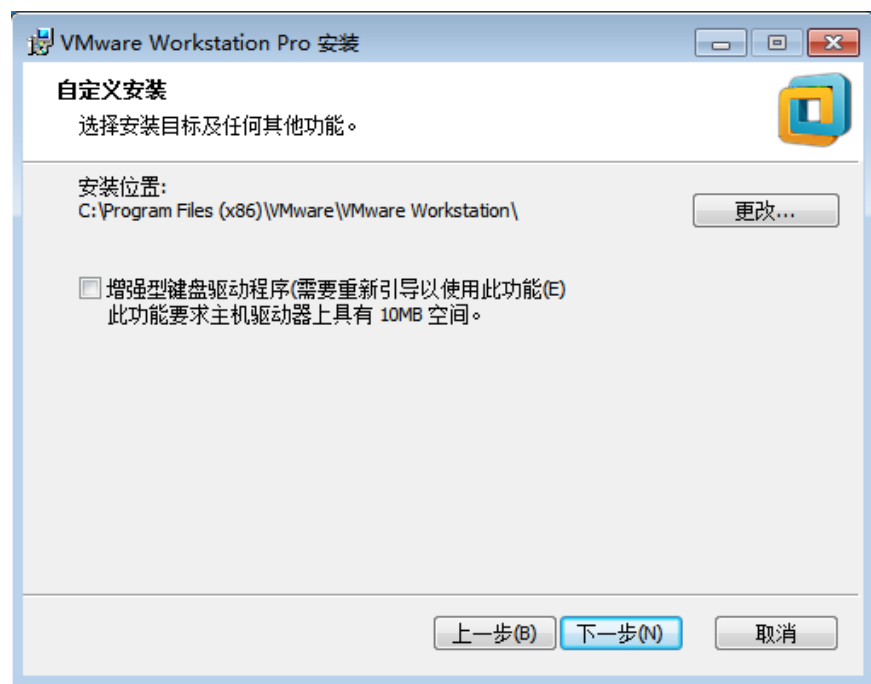


图 2-4 选择安装目录

其他选项默认，点击安装，如图 2-5 所示：

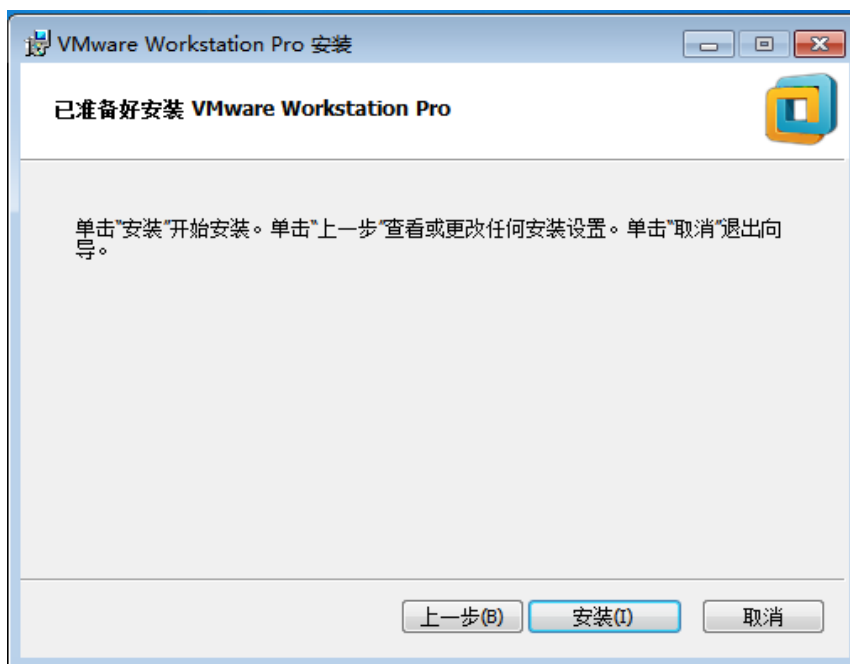


图 2-5 安装

安装完成后输入许可证密钥，如图 2-6 所示：

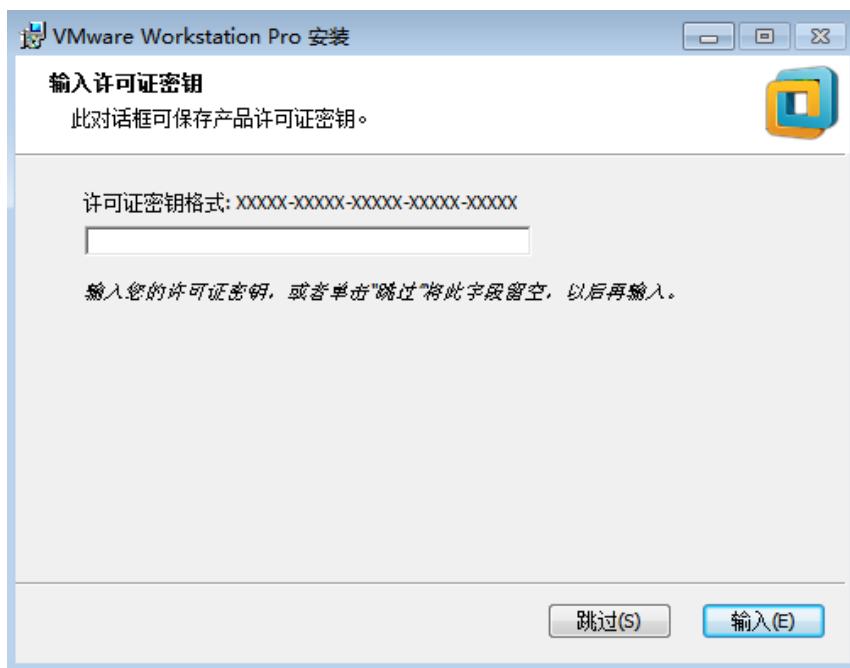


图 2-6 输入许可证

安装完成，如图 2-7 所示：

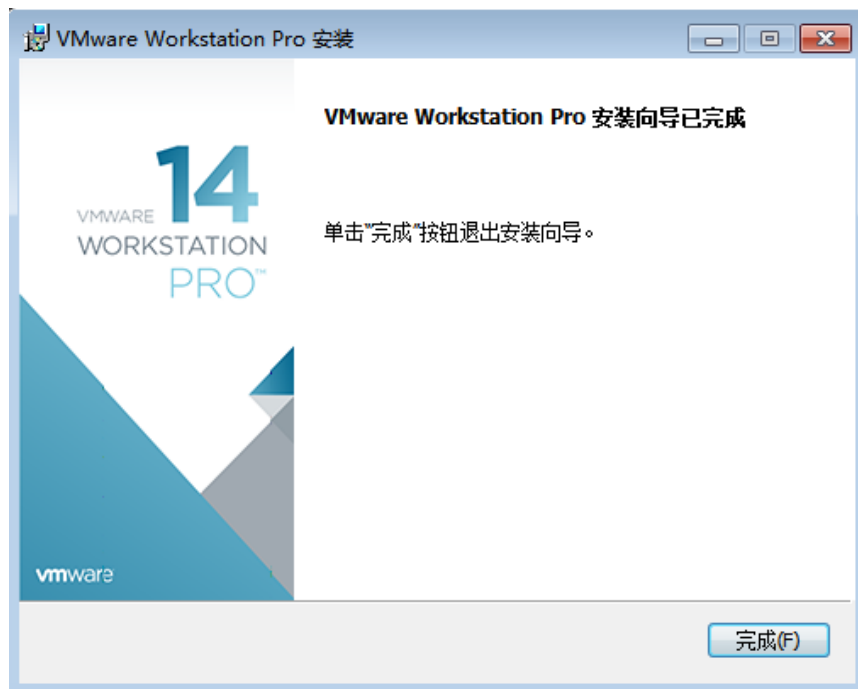


图 2-7 安装完成

## 2.2. 安装操作系统 Ubuntu

之前我们已经安装完成了虚拟化软件 VMware Workstation。我们对 Hadoop 所使用的操作系统 Ubuntu 进行安装。

打开 VMware Workstation 在主界面点击创建新的虚拟机，如图 2-8 所示：

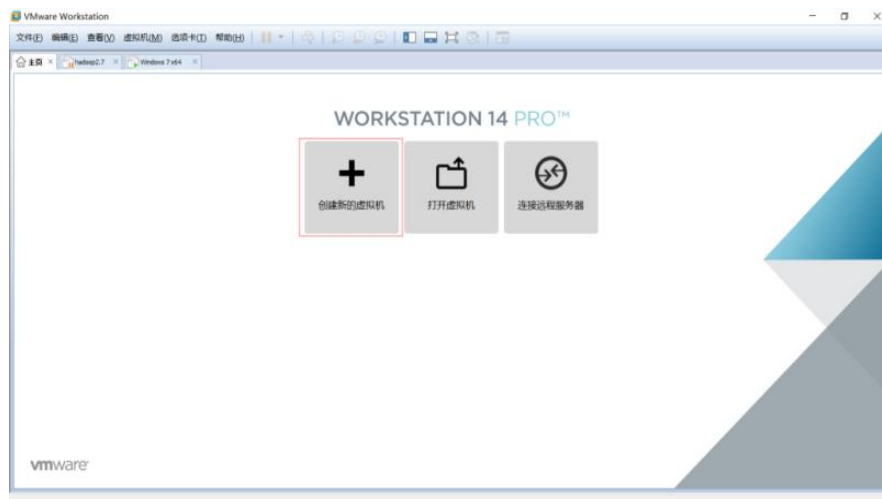


图 2-8 创建新的虚拟机

在打开的向导中选择典型，图 2-9 所示：



图 2-9 典型配置

选择稍后安装操作系统，如图 2-10 所示：



图 2-10 稍后安装操作系统

选择客户机操作系统 linux, 选择 ubuntu 64 位，如图 2-11 所示：

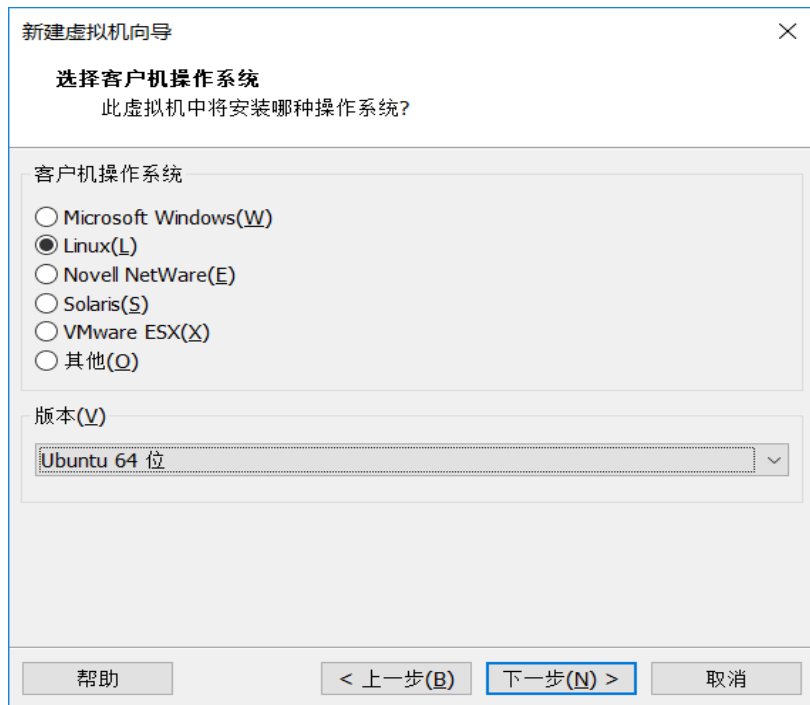


图 2-11 选择操作系统 Ubuntu

在虚拟机名字中输入Hadoop并选择在磁盘上存放的位置,如图2-12所示:

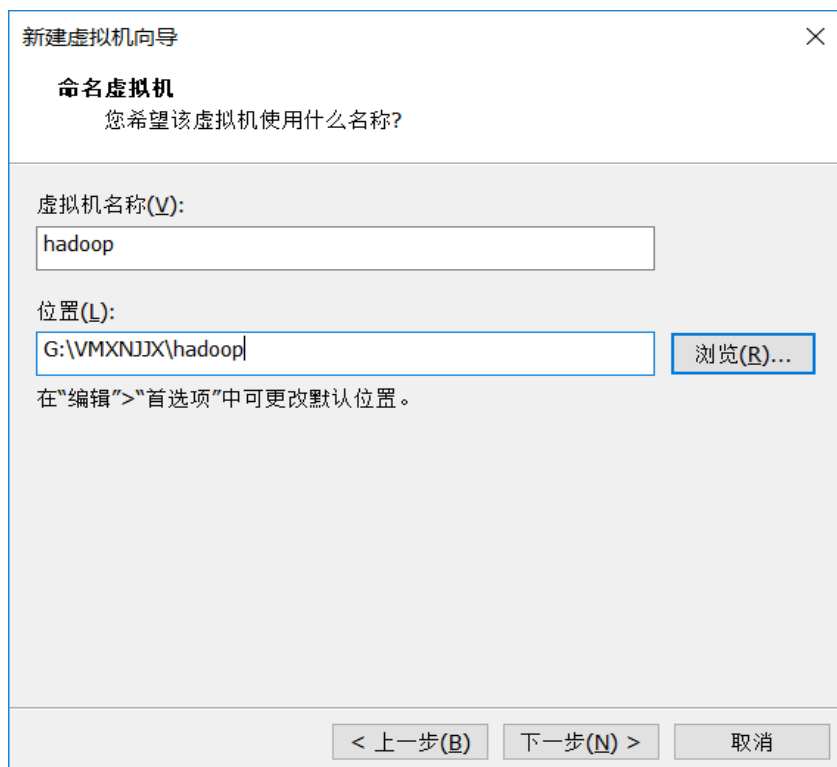


图 2-12 输入虚拟机名称选择虚拟机位置

指定磁盘大小,默认 20G,选择将虚拟磁盘拆分成多个文件,如图 2-13 所示:

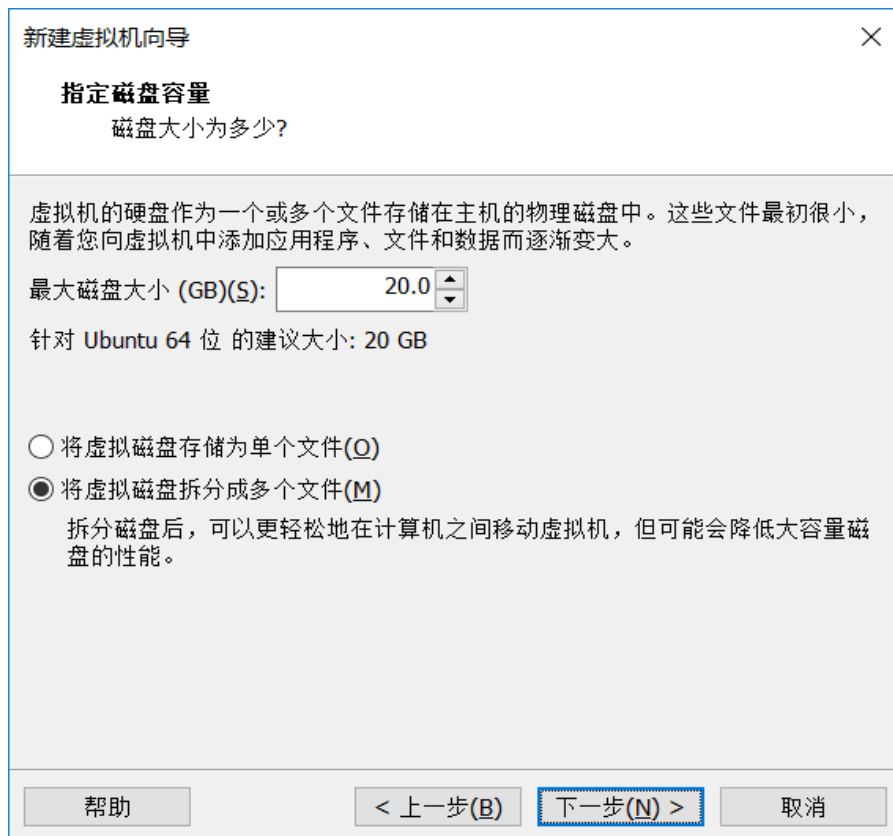


图 2-13 确定磁盘大小

点击自定义硬件进行自定义调整硬件，如图 2-14 所示：



图 2-14 自定义硬件



在新 CD/DVD (sata) 选项卡中，设备状态勾选启动时连接，连接使用 iso 镜像文件点击预览选择我们下载的 Ubuntu16.04 镜像文件，如图 2-15 所示：

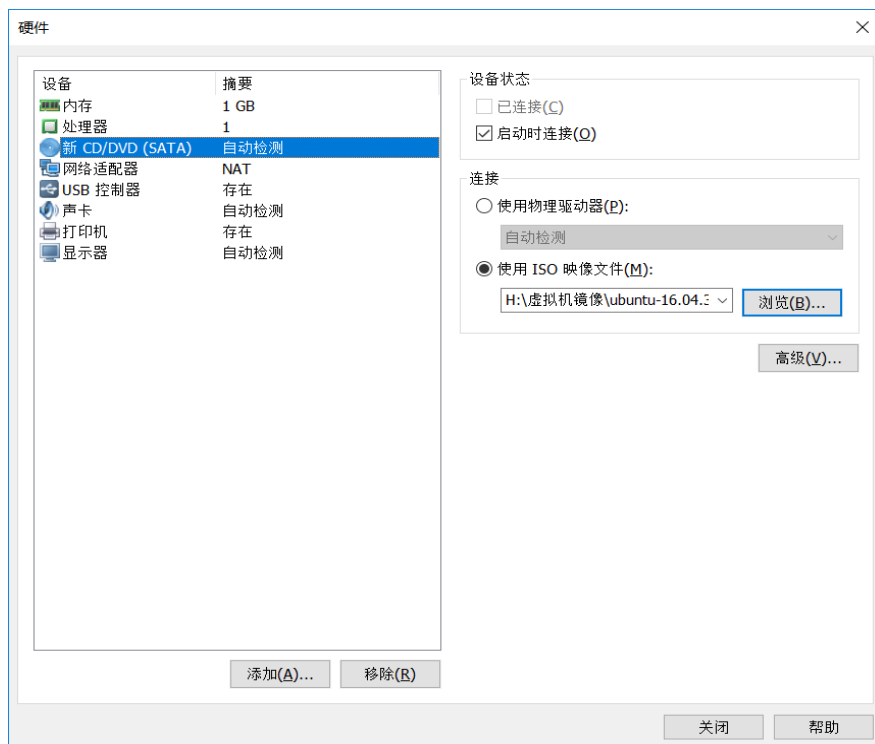


图 2-15 使用 iso 映像文件

完成，图 2-16 所示：



图 2-16 完成配置

开启虚拟机开始安装 Ubuntu16.04，如图 2-17 所示：



图 2-17 开启虚拟机

选择语言并安装 Ubuntu。如图 2-18 所示：



图 2-18 选择语言安装 Ubuntu

准备安装，如图 2-19 所示：



图 2-19 准备安装

安装类型，清除整个磁盘安装 Ubuntu，如图 2-20 所示：



图 2-20 清除整个磁盘安装 Ubuntu

确认修改，如图 3-21 所示：

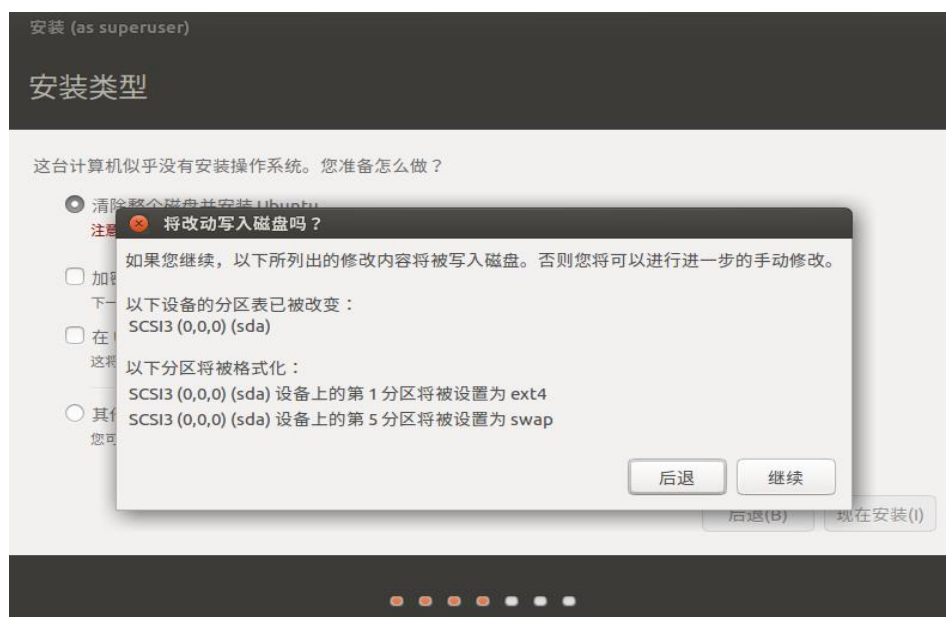


图 2-21 修改磁盘

选择时区，东八区 shanghai，如图 2-22 所示：

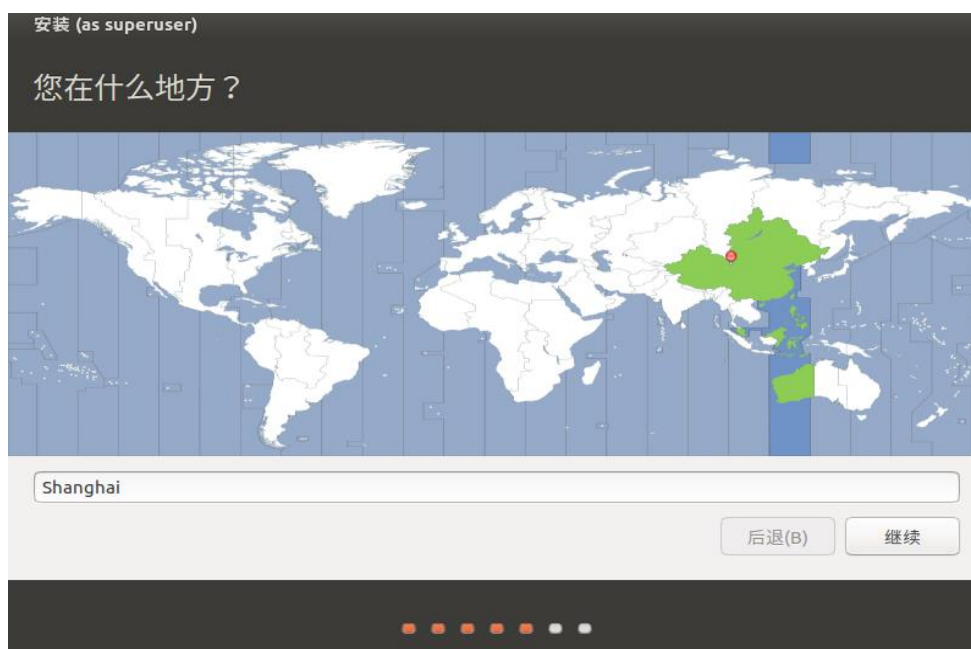


图 2-22 选择时区

选择键盘布局汉语，图 2-23 所示：

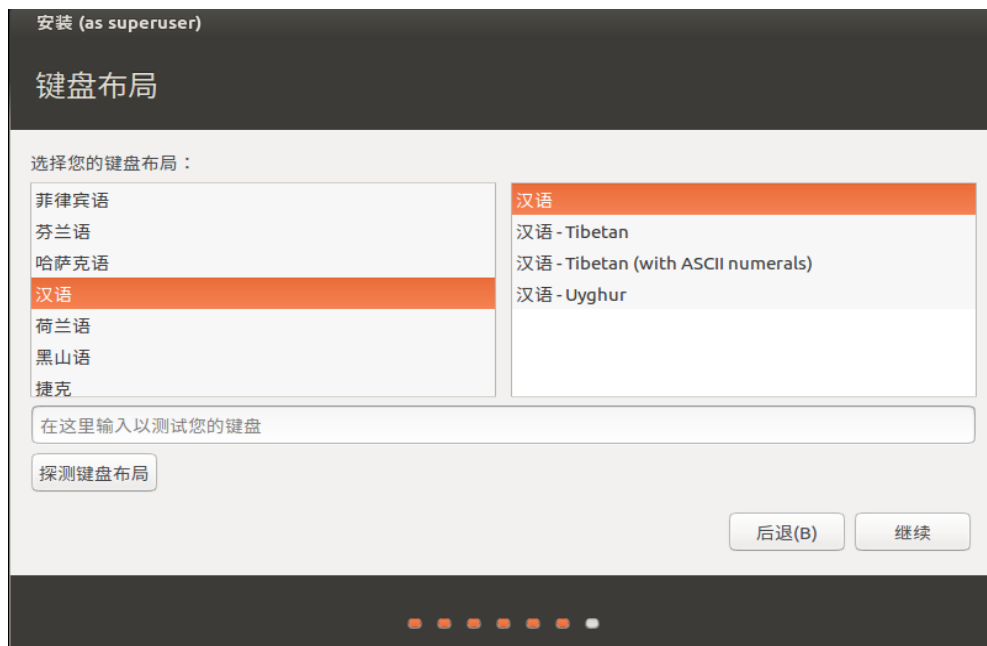


图 2-23 键盘布局

新建一个管理者 Hadoop 输入密码，确认密码，如图 2-24 所示：

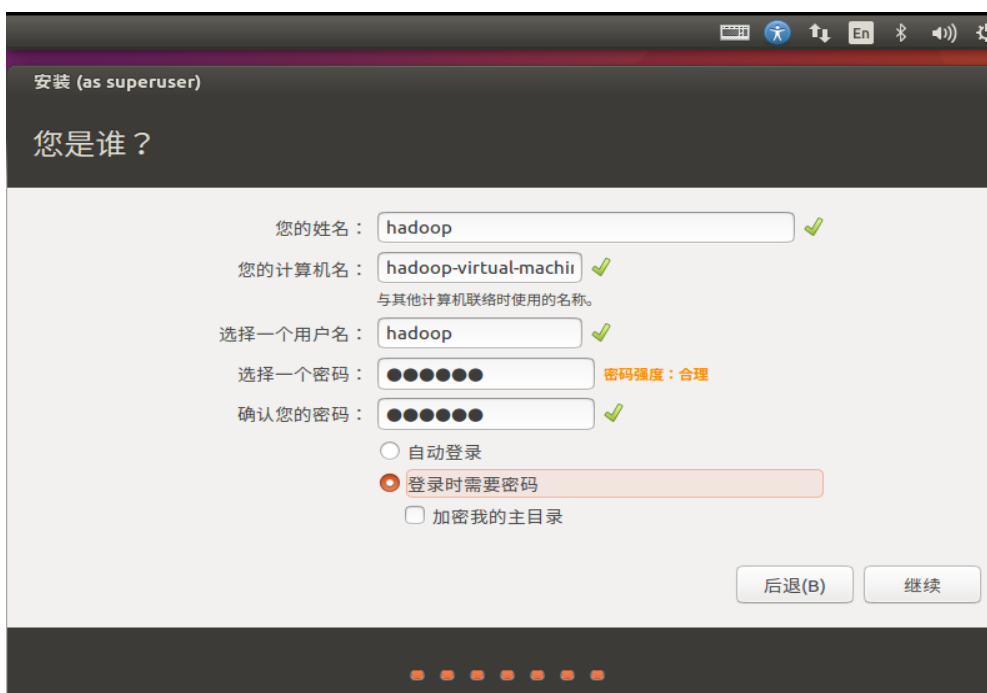


图 2-24 新建用户

开始安装，图 2-25 所示：

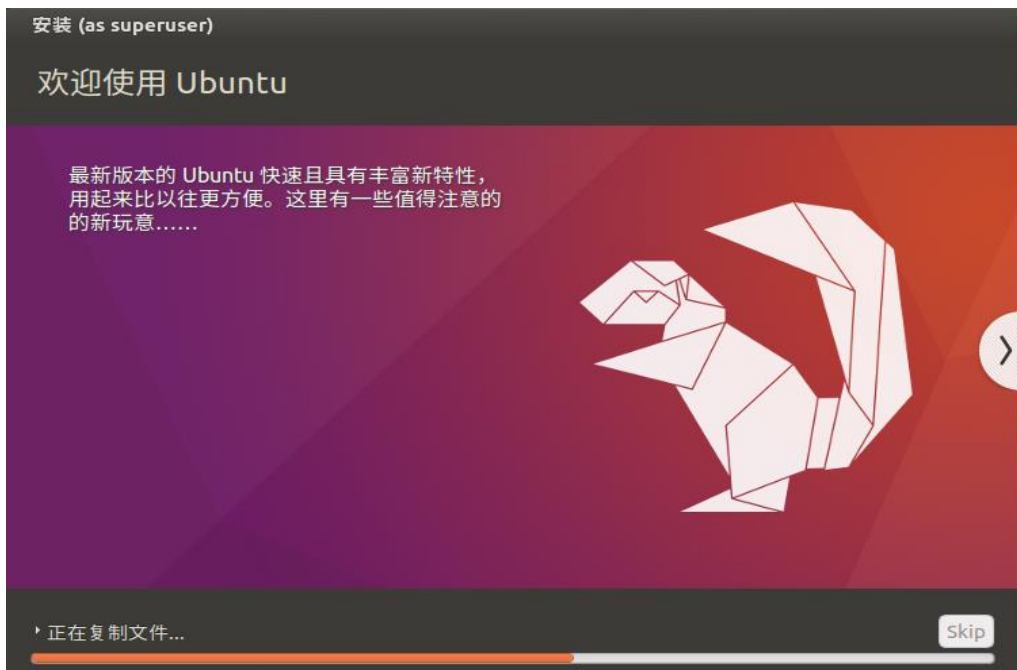


图 2-25 安装开始

安装完成后，现在重启，如图 2-26 所示：

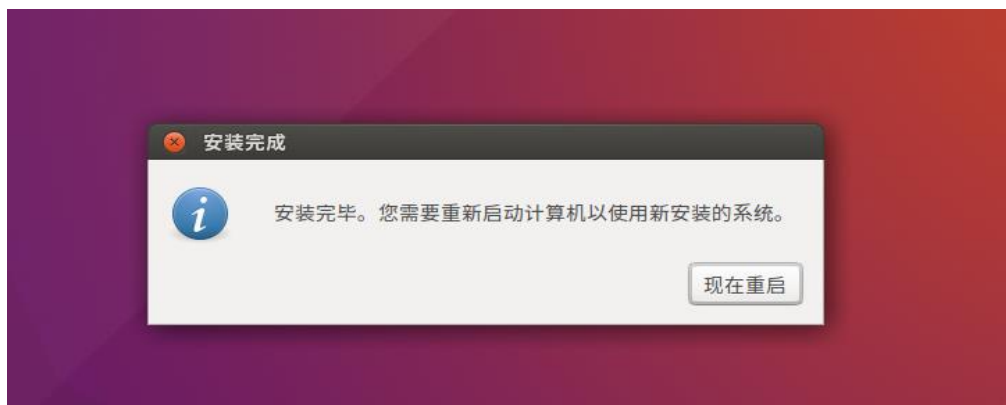


图 2-26 安装完成

## 2.3. 配置 Ubuntu 系统环境

重启系统后，打开终端，更新软件源中的所有软件列表：

```
hadoop@hadoop2:~$ sudo apt-get update
```

安装 vim 软件：

```
hadoop@hadoop2:~$ sudo apt-get install vim
```

安装 ssh 登录软件：

```
hadoop@hadoop2:~$ sudo apt-get install openssh-server
```

配置 ssh 免密钥登录：

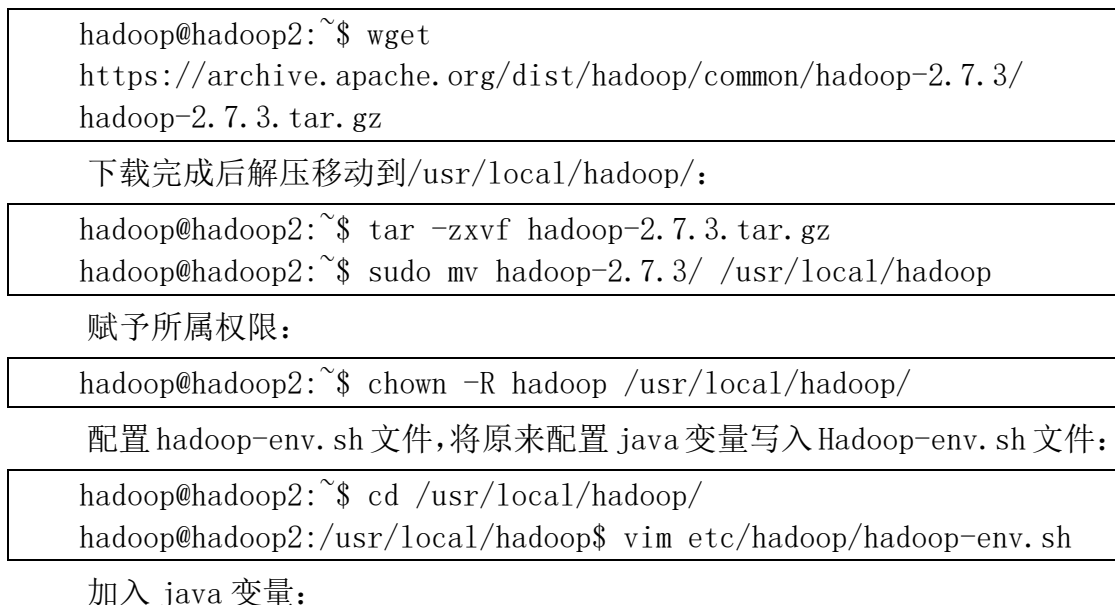
```
hadoop@hadoop2:~$ ssh-keygen -t rsa
```



图 2-36 更新配置文件

## 2.4. 安装配置 Hadoop

下载 Hadoop。Hadoop 可以通过访问 <https://archive.apache.org/dist/hadoop/common/> 下载，也可以使用 `wget` 命令进行下载：



```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```

配置 core-site.xml，将配置的内容写入  
<configuration></configuration>内：

```
hadoop@hadoop2:/usr/local/hadoop$ vim etc/hadoop/core-site.xml
```

配置信息：

```
<configuration>
<property>
<name>hadoop.tmp.dir</name>
<value>file:/usr/local/hadoop/tmp</value>
</property>
<property>
<name>fs.defaultFS</name>
<value>hdfs://localhost:9000</value>
</property>
</configuration>
```

配置 hdfs-site.xml，将配置的内容写入  
<configuration></configuration>内：

```
hadoop@hadoop2:/usr/local/hadoop$ vim etc/hadoop/hdfs-site.xml
```

配置信息：

```
<configuration>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property>
<name>dfs.namenode.name.dir</name>
<value>file:/usr/local/hadoop/tmp/dfs/name</value>
</property>
</configuration>
```

配置完成后随后对 Hadoop Namenode 进行格式化，结果出现 successfully  
表示成功。

```
hadoop@hadoop2:/usr/local/hadoop$ bin/hadoop namenode -format
```

启动 Hadoop 平台，如图 2-50 所示，验证如果输入 Jps 出现除 Jps 外五个  
进程表示成功：

```
hadoop@hadoop2:/usr/local/hadoop$ sbin/start-all.sh
```

验证：

```
hadoop@hadoop2:/usr/local/hadoop$ jps
```



85472	DataNode
85844	ResourceManager
85351	NameNode
86559	Jps
85967	NodeManager
85678	SecondaryNameNode

数据平台搭建完成。

### 3. 数据抓取

在网上收集数据使用 Python 编写的爬虫，收集数据网站是豆瓣。

抓取数据步骤：

1. 收集 url。
2. 通过 url 获得页面的信息。

#### 3.1. 收集 url

根据豆瓣电影的 tag（标签）来爬收集各影视作品的详情页的 url，例如：  
https://movie.douban.com/tag/2017，如图 3-1 所示：



图 3-1 电影标签 2017 页面

点击下方的换页按钮，获得新的页面，如图 3-2 所示：



图 3-2 电影标签 2017 页面第二页

查看前后 url 发现规律，根据标签和翻页访问站点，查看页面源代码，获得规律。如图 3-3 所示：

## 基于 Hadoop 平台的豆瓣影视作品数据分析

```
<a class="nbg" href="https://movie.douban.com/subject/5350027/" title="妖猫传">
  
d>
| valign="top">

<div class="pl2">

  <a href="https://movie.douban.com/subject/5350027/" class="">
    妖猫传
    / <span style="font-size:13px;">沙门空海 / 沙门空海之大唐鬼宴</span>
  </a>
```

图 3-3 查看源码

根据页面出现的规律编写正则表达式，使用登录豆瓣的 headers 头请求，如图 3-4 所示：

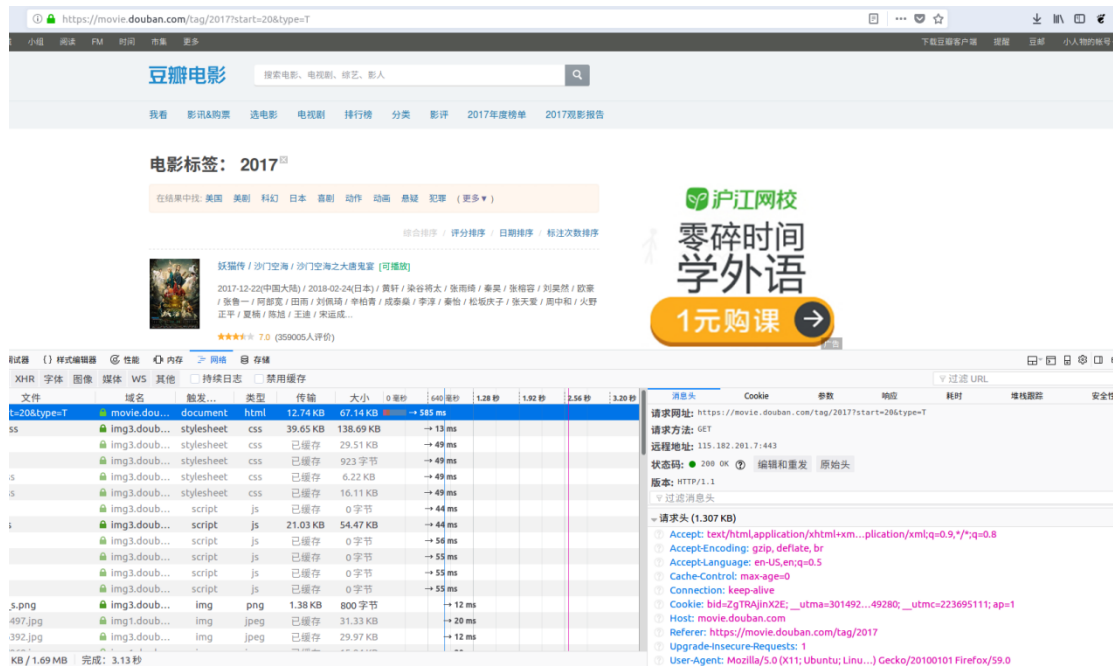


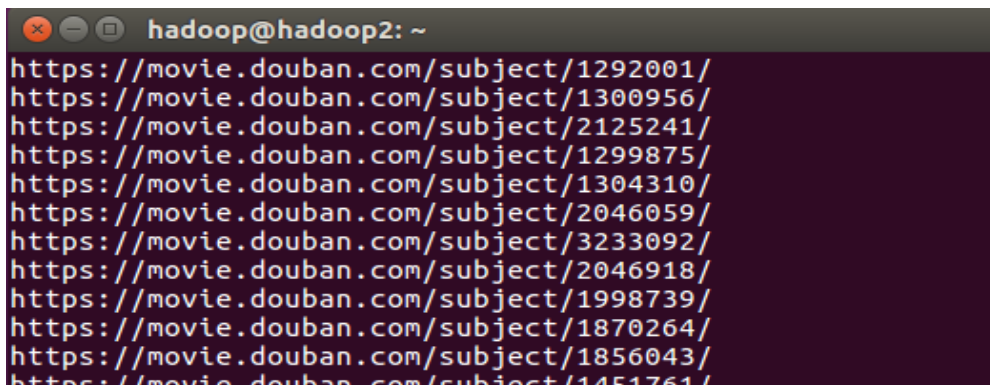
图 3-4 头请求

根据以上的信息编写爬虫程序，如图 3-5 所示。

```
hadoop@hadoop2: ~/data/daima
#coding=utf-8
import urllib2
import time
import re
import os
import random
headers={'Accept':'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8','Accept-language':'zh-CN,zh;q=0.9','Connection':'keep-alive','Cookie':'-----', 'Host':'movie.douban.com','Referer':'https://movie.douban.com/tag/','Upgrade-Insecure-Requests':'1','User-Agent':'Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:59.0) Gecko/20100101 Firefox/59.0'}
for k in range(1900,2019):
    for s in range(0,8000,20):
        url = "https://movie.douban.com/tag/"+str(k)+"?start="+str(s)+"&type=T"
        req = urllib2.Request(url,headers=headers)
        res = urllib2.urlopen(req)
        html = res.read()
        links = re.findall("<a href='\"(https://\\/.movie.douban.com/\\subject/\\.?.+?)\"',html)
        try:
            print links[0]
        except:
            break
        lin=open('/home/hadoop/links.txt','a+')
        for i in links:
            lin.write(i)
            lin.write('\\n')
        lin.close()
        time.sleep(random.randint(2,8))
```

图 3-5 爬虫程序

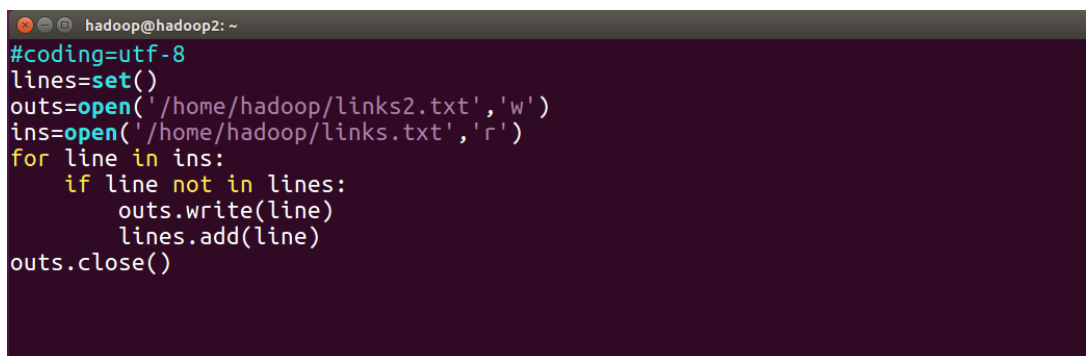
收集的 url，如图 3-6 所示。



```
hadoop@hadoop2: ~
https://movie.douban.com/subject/1292001/
https://movie.douban.com/subject/1300956/
https://movie.douban.com/subject/2125241/
https://movie.douban.com/subject/1299875/
https://movie.douban.com/subject/1304310/
https://movie.douban.com/subject/2046059/
https://movie.douban.com/subject/3233092/
https://movie.douban.com/subject/2046918/
https://movie.douban.com/subject/1998739/
https://movie.douban.com/subject/1870264/
https://movie.douban.com/subject/1856043/
https://movie.douban.com/subject/1451761/
```

图 3-6 收集的 url

为了使下一步抓取程序更加有效率对收集到的 url 进行查重。如图 3-7 所示



```
hadoop@hadoop2: ~
#coding=utf-8
lines=set()
outs=open('/home/hadoop/links2.txt','w')
ins=open('/home/hadoop/links.txt','r')
for line in ins:
    if line not in lines:
        outs.write(line)
        lines.add(line)
outs.close()
```

图 3-7 数据查重

查重前后的行数对比，如图 3-8 所示：



```
hadoop@hadoop2: ~
hadoop@hadoop2:~$ wc links.txt
79069   79068 3341224 links.txt
hadoop@hadoop2:~$ wc links2.txt
58064   58063 2452968 links2.txt
```

图 3-8 查重对比

### 3.2. 获得页面信息

根据收集的 url 访问具体电影的详情页面获得具体的电影信息，如图 3-9 -3-10 所示：

## 基于 Hadoop 平台的豆瓣影视作品数据分析

```
hadoop@hadoop2: ~
#coding=utf-8
import urllib2
import time
import sys
import os
import re
import random
cishu=0
for line in sys.stdin:
    cishu=cishu+1
    headers={'Accept':'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8','Accept-language':'zh-CN,zh;q=0.9','Connection':'keep-alive','Cookie':'-----','Host':'movie.douban.com','Referer':'https://movie.douban.com/tag','Upgrade-Insecure-Requests':'1','User-Agent':'Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:59.0) Gecko/20100101 Firefox/59.0'}
    url=line
    req=urllib2.Request(url,headers=headers)
    res=urllib2.urlopen(req)
    html=res.read()
    did=re.findall('url=http://m.douban.com/movie/subject/(.+?)',html)
    title=re.findall('"v:itemreviewed">(.*?)</span>',html)
    year=re.findall('class="year">((\d{4})\d{4})',html)
    types=re.findall('"v:genre">(.*?)</span>',html)
    state=re.findall(">制片国家/地区:</span> (.+?)<br/>",html)
    score=re.findall('"v:average">(.*?)</strong>',html)
    snum=re.findall('"v:votes">(.*?)</span>',html)
    tnum=re.findall("status=P\>全部 (.+?) 条</a>",html)
    slen=len(score)
    tlen=len(types)
    ylen=len(year)
    dlen=len(state)
    if slen==0 or ylen==0 or dlen==0:
        time.sleep(1)
        continue
    else:
        if tlen==0:
            record=did[0]+'\\t'+title[0]+'\\t'+year[0]+'\\t'+state[0]+'\\t'+score[0]+'\\t'+snum[0]+'\\t'+tnum[0]
        elif tlen==1:
            record=did[0]+'\\t'+title[0]+'\\t'+year[0]+'\\t'+types[0]+'\\t'+state[0]+'\\t'+score[0]+'\\t'+snum[0]+'\\t'+tnum[0]
        elif tlen==2:
```

图 3-9 页面抓取程序 1

```
record=did[0]+'\\t'+title[0]+'\\t'+year[0]+'\\t'+types[0]+'\\t'+types[1]+'\\t'+state[0]+'\\t'+score[0]+'\\t'+snum[0]+'\\t'+tnum[0]
        elif tlen==3:
            record=did[0]+'\\t'+title[0]+'\\t'+year[0]+'\\t'+types[0]+'\\t'+types[1]+'\\t'+types[2]+'\\t'+state[0]+'\\t'+score[0]
+ '\\t'+snum[0]+'\\t'+tnum[0]
        elif tlen==4:
            record=did[0]+'\\t'+title[0]+'\\t'+year[0]+'\\t'+types[0]+'\\t'+types[1]+'\\t'+types[2]+'\\t'+types[3]+'\\t'+state[0]
+ '\\t'+score[0]+'\\t'+snum[0]+'\\t'+tnum[0]
        elif tlen==5:
            record=did[0]+'\\t'+title[0]+'\\t'+year[0]+'\\t'+types[0]+'\\t'+types[1]+'\\t'+types[2]+'\\t'+types[3]+'\\t'+types[4]
+ '\\t'+state[0]+'\\t'+score[0]+'\\t'+snum[0]+'\\t'+tnum[0]
        elif tlen==6:
            record=did[0]+'\\t'+title[0]+'\\t'+year[0]+'\\t'+types[0]+'\\t'+types[1]+'\\t'+types[2]+'\\t'+types[3]+'\\t'+types[4]
+ '\\t'+types[5]+'\\t'+state[0]+'\\t'+score[0]+'\\t'+snum[0]+'\\t'+tnum[0]
        else:
            record=did[0]+'\\t'+title[0]+'\\t'+year[0]+'\\t'+state[0]+'\\t'+score[0]+'\\t'+snum[0]+'\\t'+tnum[0]
            info=open('/home/zhouxinjun/info.txt','a+')
            info.write(record)
            print str(cishu)+'\\t'+did[0]
            info.write('\\n')
            info.close()
            time.sleep(random.randint(2,4))
```

图 3-10 页面抓取程序 2

执行命令进行抓取，如图 3-11 所示：

```
hadoop@hadoop2: ~$ cat links2.txt | python zhuaqu2.1.py
```

图 3-11 执行抓取

获得的页面信息，如 3-12 所示：

```
hadoop@hadoop2: ~
328455 杨三姐告状 2007 剧情 中国大陆 5.7 62 14
2999306 食客 식객 2007 剧情\喜剧 韩国 6.4 3057 518
3732505 电锯女仆 Chainsaw Maid 2007 动画 日本 8.2 9174 1970
2333724 斗牛，要不要 鬥牛，要不要 2007 剧情\爱情 台湾 6.3 7305 979
2272288 女人一辈子 2007 剧情 中国大陆 7.5 379 84
2972705 好男不当兵 Kein Bund fürs Leben 2007 喜剧 德国 7.4 3638 933
1958952 椿三十郎 2007 剧情\动作\历史 日本 7.1 467 110
2059313 约书亚 Joshua 2007 剧情\惊悚\恐怖 美国 7.1 6957 1717
```

图 3-12 抓取的信息

获得的页面信息中基本包括电影在豆瓣的 Id、电影名称、电影年份、电影类型、地区、评分、评分人数、影评人数八个方面。

进行查重处理，如图 3-13 所示：

```

hadoop@hadoop2: ~
#coding=utf-8
mids=set()
outs=open('/home/hadoop/infonew.txt','w')
ins=open('/home/hadoop/info.txt','r')
for line in ins:
    mid,other= line.split('\t', 1)
    if mid not in mids:
        outs.write(line)
        mids.add(mid)
outs.close()
~

```

图 3-13 查重程序

查重的前后行数，如图 3-14 所示：

```

hadoop@hadoop2: ~$ wc info.txt
47264 497850 3965239 info.txt
hadoop@hadoop2: ~$ wc infonew.txt
47003 495105 3943643 infonew.txt

```

图 3-14 查重对比

对数据进行整理，根据抓取代码和获得的数据对数据进行整理在之前的程序中获得的是 排除了没有评分或者没有年份或者没有地区的电影，还有存在差别的是电影类别多少，存在没有类型的电影数据，将没有类型的电影数据进行类型填充为 ‘NULL’，如图 3-15 所示：

```

hadoop@hadoop2: ~
#coding=utf-8
import sys
for line in sys.stdin:
    line=line.strip()
    t=len(line.split('\t',7))
    if t==8:
        record=line
    else:
        did,title,year,state,score,snum,tnum=line.split('\t',6)
        record=did+'\t'+title+'\t'+year+'\t'+NULL+'\t'+state+'\t'+score+'\t'+snum+'\t'+tnum
    records=open('/home/hadoop/records.txt','a+')
    records.write(record)
    records.write('\n')
    records.close()

```

图 3-15 数据整理程序

整理完的数据，如图 3-16 所示：

```

hadoop@hadoop2: ~
4328455 杨三姐告状 2007 剧情 中国大陆 5.7 62 14
2999306 食客 식객 2007 剧情\喜剧 韩国 6.4 3057 518
3732505 电锯女仆 Chainsaw Maid 2007 动画 日本 8.2 9174 1970
2333724 斗牛，要不要 鬥牛，要不要 2007 剧情\爱情 台湾 6.3 7305 979
2272288 女人一辈子 2007 剧情 中国大陆 7.5 379 84
2972705 好男不当兵 Kein Bund fürs Leben 2007 喜剧 德国 7.4 3638 933
1958952 棒三太郎 2007 剧情\动作\历史 日本 7.1 467 110
2059313 约书亚 Joshua 2007 剧情\惊悚\恐怖 美国 7.1 6957 1717
2006066 诱捕2 Decoys 2: Alien Seduction 2007 科幻\惊悚\恐怖 加拿大 5.0 474 63
3017790 路易·C·K：臭不要脸 Louis C.K.: Shameless 2007 喜剧\脱口秀 美国 8.8 569 120
2382920 大过年 2007 NULL 中国大陆 8.3 428 86
2081871 红男爵 Der Rote Baron 2008 剧情\传记\战争 德国 / 英国 7.7 3657 946
1899601 蜜蜂总动员 Bee Movie 2007 喜剧\动画\家庭\冒险 美国 6.8 17843 1941
3542098 哈皮父子 2007 喜剧\动画\儿童 中国大陆 7.6 517 89
2020437 复核调查 Contre-enquête 2007 犯罪 法国 7.5 825 250
3145914 没有人是一个孤岛 Ei kukaan ole saari 2006 NULL 芬兰 8.6 68 17
2042981 火箭科学 Rocket Science 2007 剧情\喜剧 美国 7.6 2558 537
2070033 派对动物 Party Animals 2007 NULL 英国 7.7 29 12
20253812 天眼神虎 2007 动画\儿童 中国大陆 5.6 179 33

```

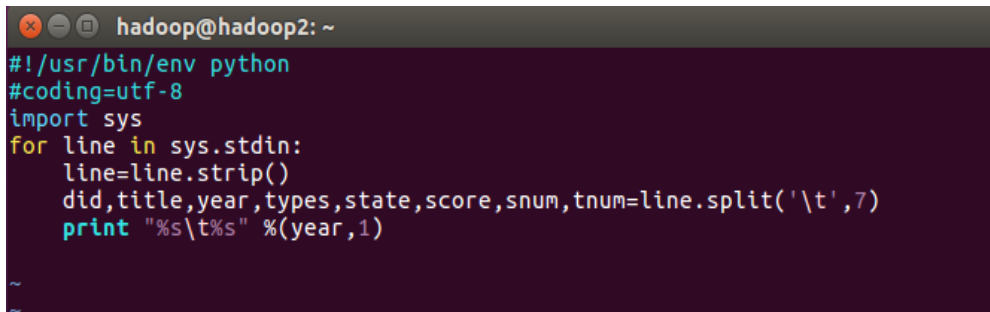
图 3-16 整理结果

数据收集与数据整理完成。

## 4. 数据分析与建模

### 4.1. 编写统计每年作品数量的 MapReduce 程序

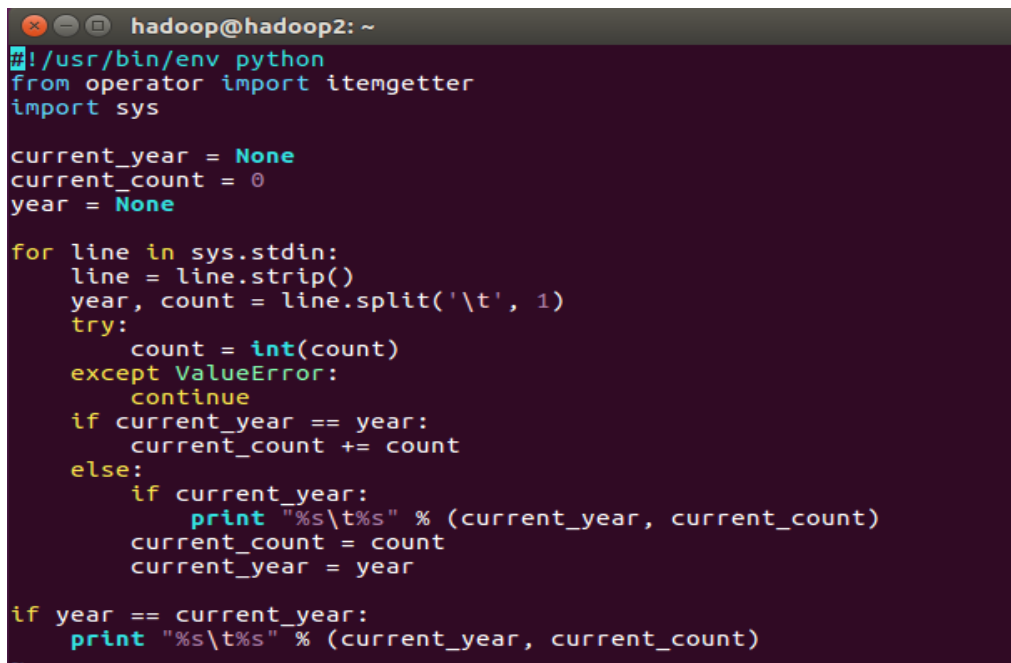
Map 程序，如图 4-1 所示：



```
hadoop@hadoop2: ~  
#!/usr/bin/env python  
#coding=utf-8  
import sys  
for line in sys.stdin:  
    line=line.strip()  
    did,title,year,types,state,score,snum,tnum=line.split('\t',7)  
    print "%s\t%s" %(year,1)  
~  
~
```

图 4-1 统计每年作品数量 Map 程序

Reduce 程序，如图 4-2 所示：



```
hadoop@hadoop2: ~  
#!/usr/bin/env python  
from operator import itemgetter  
import sys  
  
current_year = None  
current_count = 0  
year = None  
  
for line in sys.stdin:  
    line = line.strip()  
    year, count = line.split('\t', 1)  
    try:  
        count = int(count)  
    except ValueError:  
        continue  
    if current_year == year:  
        current_count += count  
    else:  
        if current_year:  
            print "%s\t%s" % (current_year, current_count)  
            current_count = count  
            current_year = year  
  
if year == current_year:  
    print "%s\t%s" % (current_year, current_count)  
~  
~
```

图 4-2 统计每年作品数量 Reduce 程序

### 4.2. 编写统计作品类型的 MapReduce 程序

Map 程序，如图 4-3 所示：



```
hadoop@hadoop2: ~
#!/usr/bin/env python
#coding=utf-8
import sys
for line in sys.stdin:
    line=line.strip()
    did,title,year,types,state,score,snum,tnum=line.split('\t',7)
    tlen=len(types.split('\t',5))
    types=types.replace('Adult','成人')
    types=types.replace('Comedy','喜剧')
    types=types.replace('Game-Show','真人秀')
    types=types.replace('News','新闻')
    types=types.replace('Reality-TV','真人秀')
    types=types.replace('Talk-Show','脱口秀')
    if tlen==1:
        type1=types
        print "%s\t%s" %(type1,1)
    elif tlen==2:
        type1,type2=types.split('\t',1)
        print "%s\t%s\n%s\t%s" %(type1,1,type2,1)
    elif tlen==3:
        type1,type2,type3=types.split('\t',2)
        print "%s\t%s\n%s\t%s\n%s\t%s" %(type1,1,type2,1,type3,1)
    elif tlen==4:
        type1,type2,type3,type4=types.split('\t',3)
        print "%s\t%s\n%s\t%s\n%s\t%s\n%s\t%s" %(type1,1,type2,1,type3,1,type4,1)
    elif tlen==5:
        type1,type2,type3,type4,type5=types.split('\t',4)
        print "%s\t%s\n%s\t%s\n%s\t%s\n%s\t%s\n%s\t%s" %(type1,1,type2,1,type3,1,type4,1,type5,1)
    else:
        type1,type2,type3,type4,type5,type6=types.split('\t',5)
        print "%s\t%s\n%s\t%s\n%s\t%s\n%s\t%s\n%s\t%s\n%s\t%s" %(type1,1,type2,1,type3,1,type4,1,type5,1,type6,1)
```

图 4-3 统计作品类型 Map 程序

Reduce 程序，如图 4-4 所示：

```
hadoop@hadoop2: ~
#!/usr/bin/env python
from operator import itemgetter
import sys

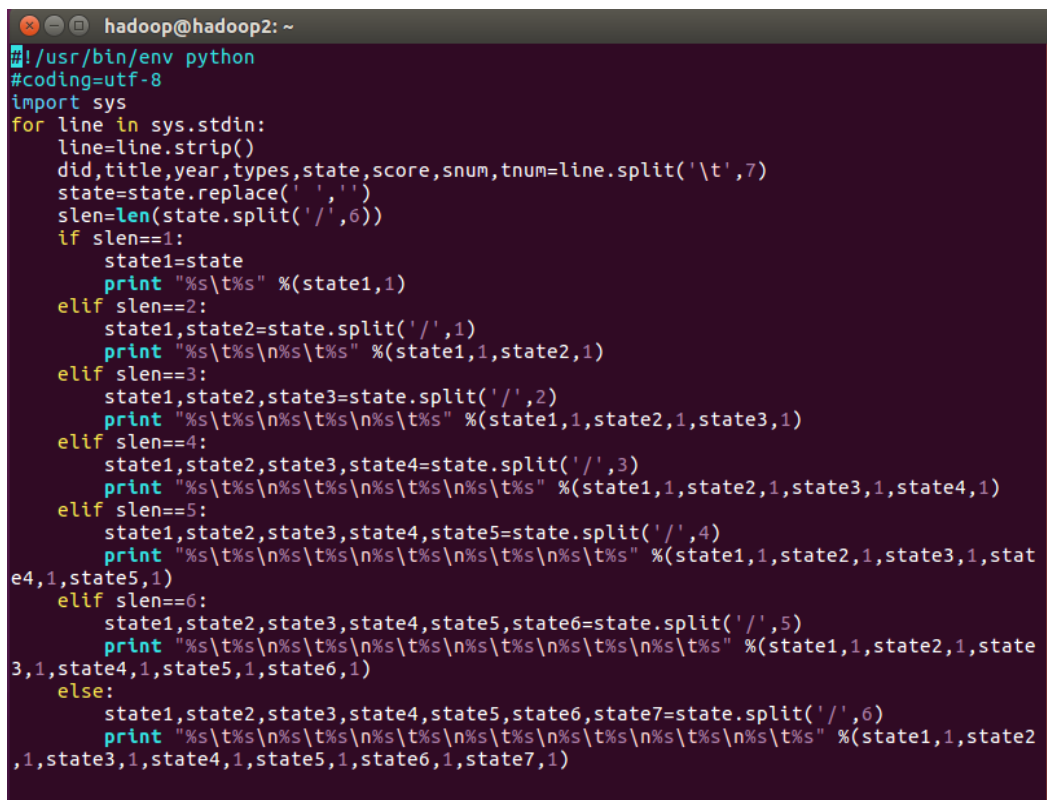
current_types = None
current_count = 0
types = None

for line in sys.stdin:
    line = line.strip()
    types, count = line.split('\t', 1)
    try:
        count = int(count)
    except ValueError:
        continue
    if current_types == types:
        current_count += count
    else:
        if current_types:
            print "%s\t%s" % (current_types, current_count)
            current_count = count
            current_types = types
        if types == current_types:
            print "%s\t%s" % (current_types, current_count)
```

图 4-4 统计作品类型 Reduce 程序

#### 4.3. 编写作品制作地区的 MapReduce 程序

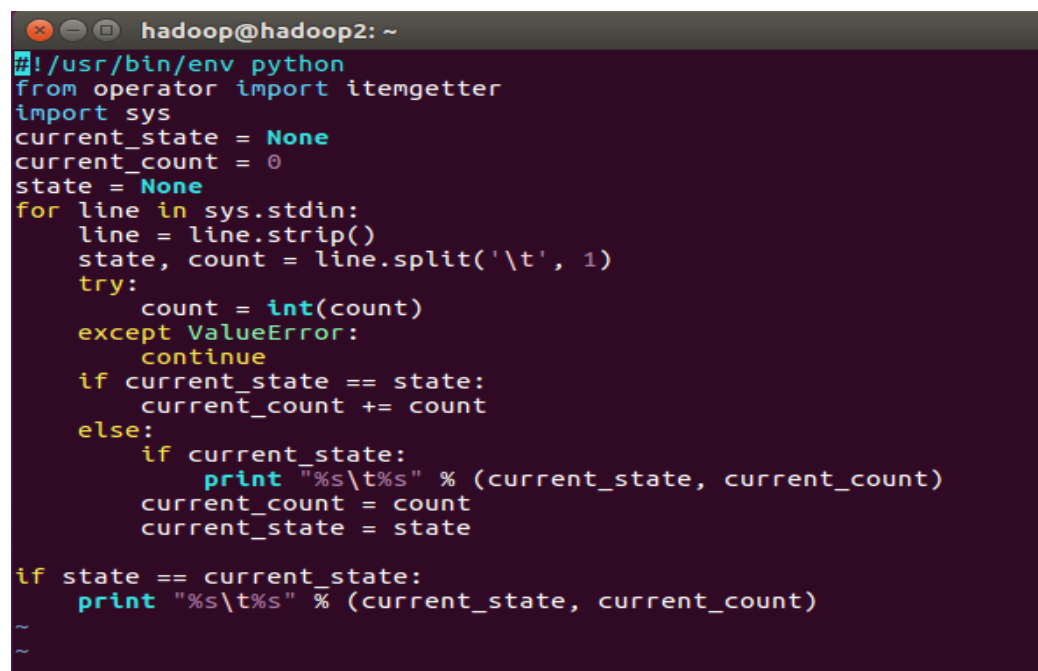
Map 程序，如图 4-5 所示：



```
hadoop@hadoop2: ~
#!/usr/bin/env python
#coding=utf-8
import sys
for line in sys.stdin:
    line=line.strip()
    did,title,year,types,state,score,snum,tnum=line.split('\t',7)
    state=state.replace(' ','')
    slen=len(state.split('/',6))
    if slen==1:
        state1=state
        print "%s\t%s" %(state1,1)
    elif slen==2:
        state1,state2=state.split('/',1)
        print "%s\t%s\n%s\t%s" %(state1,1,state2,1)
    elif slen==3:
        state1,state2,state3=state.split('/',2)
        print "%s\t%s\n%s\t%s\n%s\t%s" %(state1,1,state2,1,state3,1)
    elif slen==4:
        state1,state2,state3,state4=state.split('/',3)
        print "%s\t%s\n%s\t%s\n%s\t%s\n%s\t%s" %(state1,1,state2,1,state3,1,state4,1)
    elif slen==5:
        state1,state2,state3,state4,state5=state.split('/',4)
        print "%s\t%s\n%s\t%s\n%s\t%s\n%s\t%s\n%s\t%s" %(state1,1,state2,1,state3,1,state4,1,state5,1)
    elif slen==6:
        state1,state2,state3,state4,state5,state6=state.split('/',5)
        print "%s\t%s\n%s\t%s\n%s\t%s\n%s\t%s\n%s\t%s\n%s\t%s" %(state1,1,state2,1,state3,1,state4,1,state5,1,state6,1)
    else:
        state1,state2,state3,state4,state5,state6,state7=state.split('/',6)
        print "%s\t%s\n%s\t%s\n%s\t%s\n%s\t%s\n%s\t%s\n%s\t%s\n%s\t%s" %(state1,1,state2,1,state3,1,state4,1,state5,1,state6,1,state7,1)
```

图 4-5 作品制作地区 Map 程序

Reduce 程序，如图 4-6 所示：

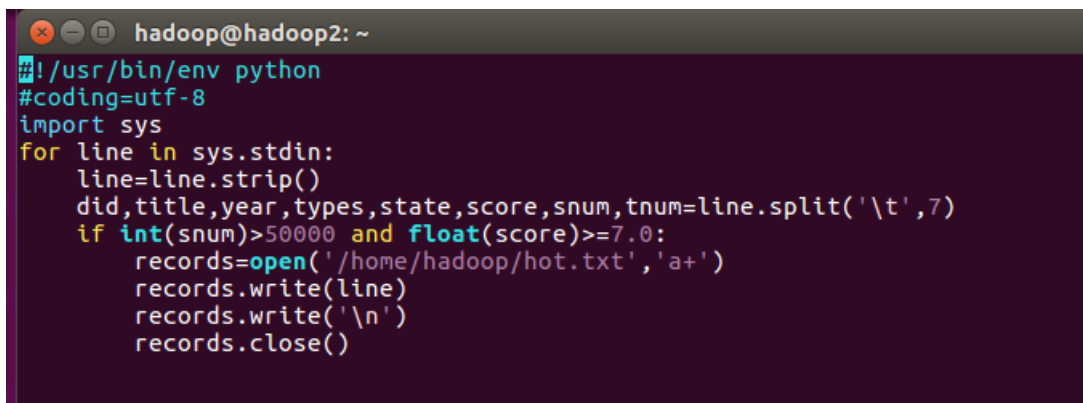


```
hadoop@hadoop2: ~
#!/usr/bin/env python
from operator import itemgetter
import sys
current_state = None
current_count = 0
state = None
for line in sys.stdin:
    line = line.strip()
    state, count = line.split('\t', 1)
    try:
        count = int(count)
    except ValueError:
        continue
    if current_state == state:
        current_count += count
    else:
        if current_state:
            print "%s\t%s" % (current_state, current_count)
            current_count = count
            current_state = state
if state == current_state:
    print "%s\t%s" % (current_state, current_count)
~
~
```

图 4-6 作品制作地区 Reduce 程序

#### 4.4. 编写处理高评价的电影处理程序

编写处理高评价的电影处理程序（评分大于等于 7.0 分评分人数大于 5 万人的电影），如图 4-7 所示：

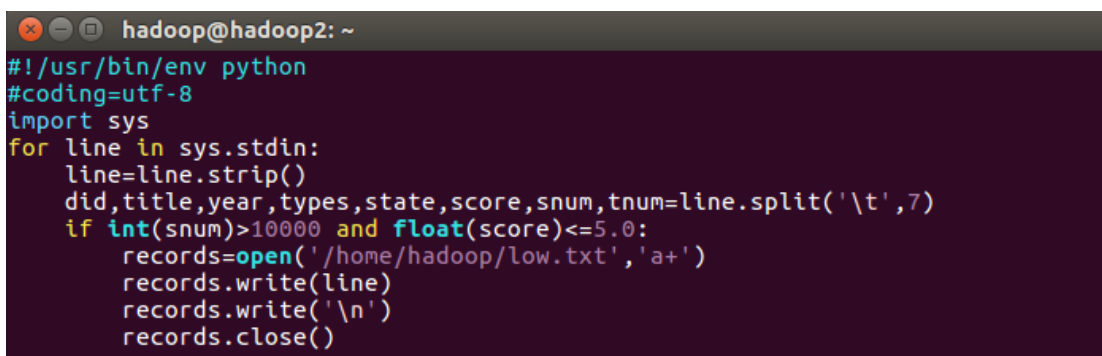


```
hadoop@hadoop2: ~
#!/usr/bin/env python
#coding=utf-8
import sys
for line in sys.stdin:
    line=line.strip()
    did,title,year,types,state,score,snum,tnum=line.split('\t',7)
    if int(snum)>50000 and float(score)>=7.0:
        records=open('/home/hadoop/hot.txt','a+')
        records.write(line)
        records.write('\n')
        records.close()
```

图 4-7 高评价处理程序

#### 4.5. 编写处理低评价的电影处理程序

编写处理低评价的电影处理程序（评分小于等于 5.0 分评分人数大于 1 万人的电影），如图 4-8 所示：

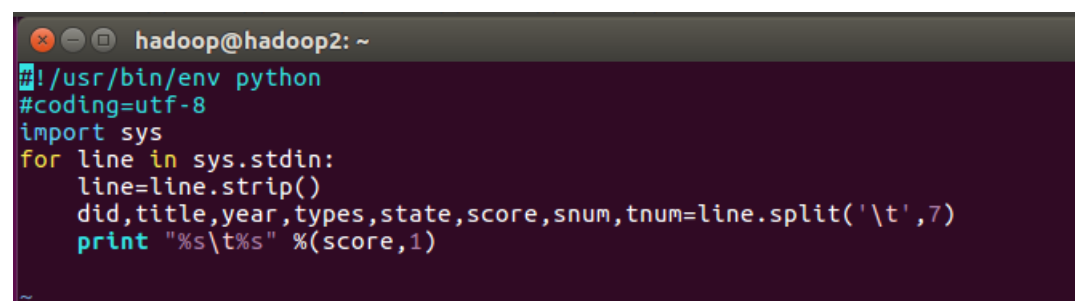


```
hadoop@hadoop2: ~
#!/usr/bin/env python
#coding=utf-8
import sys
for line in sys.stdin:
    line=line.strip()
    did,title,year,types,state,score,snum,tnum=line.split('\t',7)
    if int(snum)>10000 and float(score)<=5.0:
        records=open('/home/hadoop/low.txt','a+')
        records.write(line)
        records.write('\n')
        records.close()
```

图 4-8 低评价处理程序

#### 4.6. 编写作品评分数量的 MapReduce 程序

Map 程序, 如图 4-9 所示：



```
hadoop@hadoop2: ~
#!/usr/bin/env python
#coding=utf-8
import sys
for line in sys.stdin:
    line=line.strip()
    did,title,year,types,state,score,snum,tnum=line.split('\t',7)
    print "%s\t%s" %(score,1)
```

图 4-9 作品评分数量 Map 程序

Reduce 程序，如图 4-10 所示：

```
hadoop@hadoop2: ~
#!/usr/bin/env python
from operator import itemgetter
import sys

current_score = None
current_count = 0
score = None

for line in sys.stdin:
    line = line.strip()
    score, count = line.split('\t', 1)
    try:
        count = int(count)
    except ValueError:
        continue
    if current_score == score:
        current_count += count
    else:
        if current_score:
            print "%s\t%s" % (current_score, current_count)
            current_count = count
            current_score = score
        if score == current_score:
            print "%s\t%s" % (current_score, current_count)
```

图 4-10 作品评分数量 reduce 程序

#### 4.7. 编写作品类型平均评分的程序

编写作品类型平均评分的程序，如图 4-11 所示：

```
hadoop@hadoop2: ~
#!/usr/bin/env python
#coding=utf-8
import sys
for line in sys.stdin:
    line=line.strip()
    types,num,score=line.split('\t',2)
    score=float(score)
    num=int(num)
    avgscore=score/num
    avgscore='%.1f' % avgscore
    record= types+'\t'+avgscore
    records=open('/home/hadoop/output/typesavg.txt','a+')
    records.write(record)
    records.write('\n')
    records.close()
```

图 4-11 不同类型平均评分程序

## 5. 数据存储与处理

数据处理是编写程序完成后使用程序对数据进行一系列的处理。

### 5.1. 在 HDFS 上创建 input 和 output 目录

我们将之前的 Hadoop 服务开启，在 hdfs 上创建一个 input 和一个 output 目录，如图 5-1 所示：

```
hadoop@hadoop2:/usr/local/hadoop$ bin/hadoop fs -mkdir /input
hadoop@hadoop2:/usr/local/hadoop$ bin/hadoop fs -mkdir /output
hadoop@hadoop2:/usr/local/hadoop$ bin/hadoop fs -ls /
Found 2 items
drwxr-xr-x - hadoop supergroup 0 2018-05-01 19:17 /input
drwxr-xr-x - hadoop supergroup 0 2018-05-01 19:18 /output
```

图 5-1 在 HDFS 上创建目录

### 5.2. 将收集处理完的数据上传到 HDFS

将之前我们收集处理完成的数据上传到 hdfs 的 /input 下，如图 5-2 所示：

```
hadoop@hadoop2:/usr/local/hadoop$ bin/hadoop fs -put /home/hadoop/data/input/records.txt /input
18/05/01 19:53:42 WARN hdfs.DFSClient: Slow waitForAcks took 33801ms (threshold=30000ms)
hadoop@hadoop2:/usr/local/hadoop$ bin/hadoop fs -ls /input
Found 1 items
-rw-r--r-- 1 hadoop supergroup 3955538 2018-05-01 19:53 /input/records.txt
```

图 5-2 上传数据到 HDFS

### 5.3. 进行本地测试

在放到 Hadoop 上运行程序之前我们需要在本地进行测试，如图 5-3 所示：

```
hadoop@hadoop2:~$ head -n 10 records.txt > test.txt
hadoop@hadoop2:~$
hadoop@hadoop2:~$
hadoop@hadoop2:~$
hadoop@hadoop2:~$ cat test.txt | python year1.py | sort | python year2.py
2007 10
```

图 5-3 本地测试

### 5.4. 使用 Hadoop 平台处理程序

使用 Hadoop 平台大规模批处理程序，如图 5-4 所示：

```
hadoop@hadoop2:/usr/local/hadoop$ bin/hadoop jar share/hadoop/tools/lib/hadoop-streaming-2.7.3.jar -input /input/* -output /output/year -mapper /home/hadoop/year1.py -reducer /home/hadoop/year2.py
```

图 5.4 hadoop 处理数据

### 5.5. 使用 Python 程序本地处理数据

除了使用 MapperReduce 处理的程序，还有本地使用 Python 程序处理的程序，如图 5-5 所示：

```
hadoop@hadoop2:~$ cat reducer.py | python hot1.py
```

图 5-5 本地处理数据

数据处理完成。

## 6. 数据可视化与分析

### 6.1. 导出 HDFS 上的数据

使用 Python 进行数据可视化之前，我们需要将在 hdfs 上的数据导出到本地，如图 6-1 所示：

```
hadoop@hadoop2: /usr/local/hadoop
hadoop@hadoop2: /usr/local/hadoop$ bin/hadoop fs -get /output/ /home/hadoop/data/
```

图 6-1 导出数据

### 6.2. 安装数据可视化软件 IPython

安装数据可视化软件 iPython，如图 6-2 所示：

```
hadoop@hadoop2:~$ sudo apt-get install ipython
```

图 6-2 安装 ipython

### 6.3. 安装依赖软件包

还需要安装 Python 的数据处理的包 numpy，panda，以及可视化的包 Matplotlib，如图 6-3 所示：

```
hadoop@hadoop2:~$ pip install jupyter numpy panda matplotlib
```

图 6-3 安装依赖包

### 6.4. 打开数据可视化软件 iPython

在终端输入 ipython notebook 如图 6-4 所示：

```
hadoop@hadoop2:~$ ipython notebook
[TerminalIPythonApp] WARNING | Subcommand 'ipython notebook' is deprecated and will be removed in future versions.
[TerminalIPythonApp] WARNING | You likely want to use 'jupyter notebook' in the future
[I 20:50:26.133 NotebookApp] Writing notebook server cookie secret to /run/user/1000/jupyter/notebook_cookie_secret
[I 20:50:26.339 NotebookApp] Serving notebooks from local directory: /home/hadoop
[I 20:50:26.340 NotebookApp] 0 active kernels
[I 20:50:26.340 NotebookApp] The Jupyter Notebook is running at:
[I 20:50:26.340 NotebookApp] http://localhost:8888/?token=1583a190e63f1ec3d53011d1c524b4b2a56d4b027397a604
[I 20:50:26.340 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 20:50:26.341 NotebookApp]

Copy/paste this URL into your browser when you connect for the first time,
to login with a token:
    http://localhost:8888/?token=1583a190e63f1ec3d53011d1c524b4b2a56d4b027397a604
```

图 6-4 启动 ipython

这个时候回弹出一个浏览器，如图 6-5 所示：

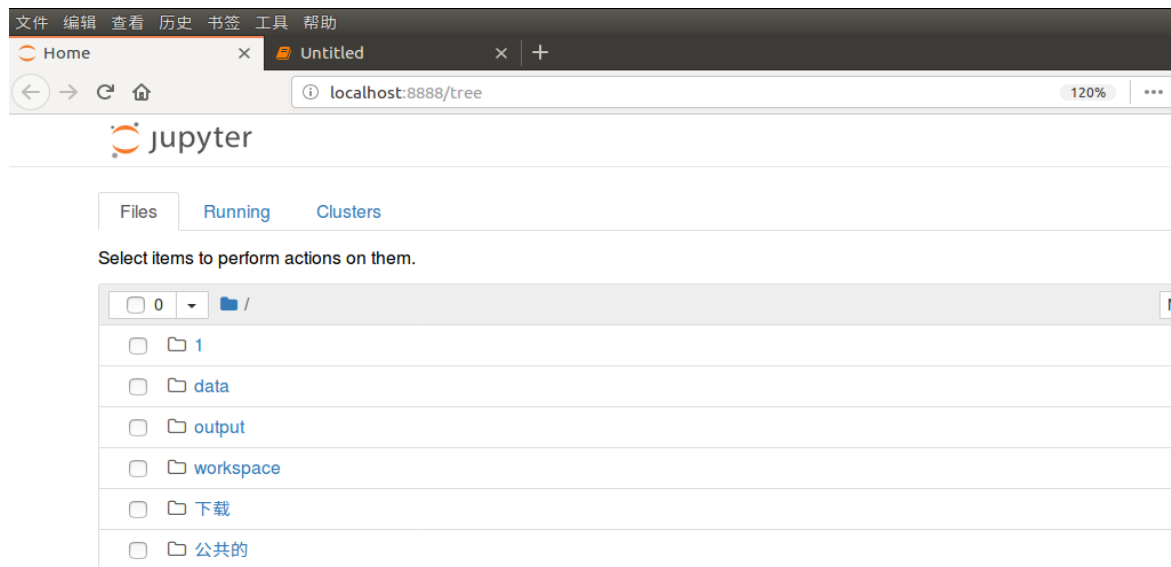


图 6-5 ipython 页面

输入代码生成图片，如图 6-6 所示：

```
In [7]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
types = [u'NULL',u'传记',u'儿童',u'冒险',u'剧情',u'动作',u'动画',u'历史',u'古
df=pd.read_csv('/home/hadoop/output/types//part-00000',delimiter='\t',na
y=np.array(list(df['num']))
x=np.arange(len(df['types']))
plt.bar(x,y,width=0.7,align='center',alpha=1)
plt.xticks(x,types,size=5,rotation=90)
plt.xlabel(u'类别')
plt.ylabel(u'数量')
plt.title(u'电影类型数量图')
for a,b in zip(x,y):
    plt.text(a,b+0.05,'%d' % b ,ha='center',va='bottom',fontsize=5)
plt.savefig('/home/hadoop/f',dpi=2000)
```

图 6-6 数据可视化代码

## 6.5. 所得效果图与分析



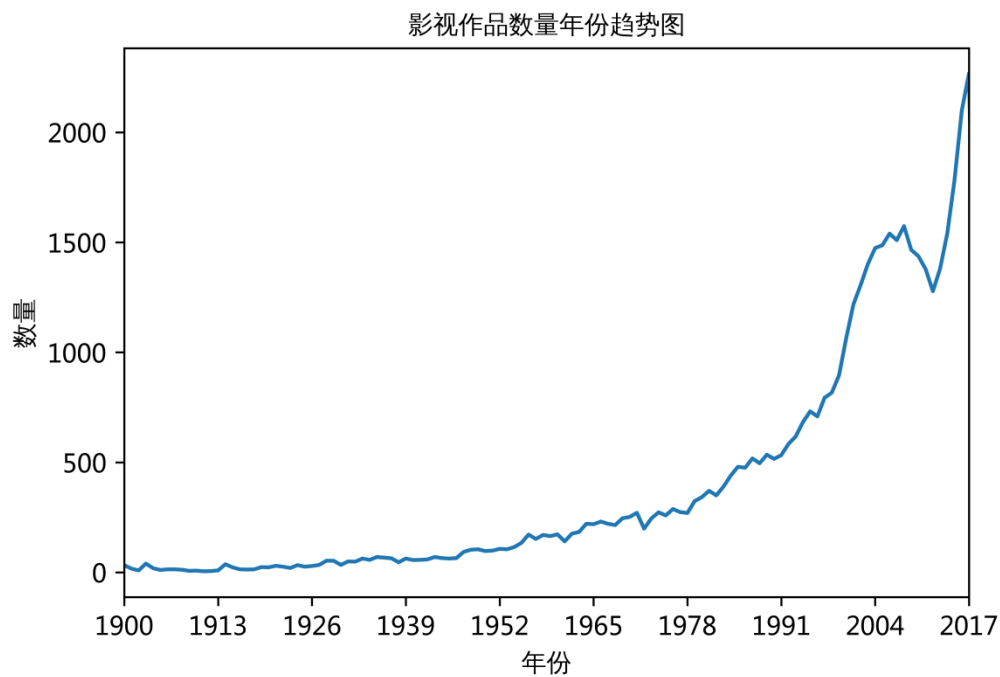


图 6-7 影视作品数量年份趋势图

从 1900 年开始影视作品数量逐年递增，在二十世纪部分年份有所趋缓。

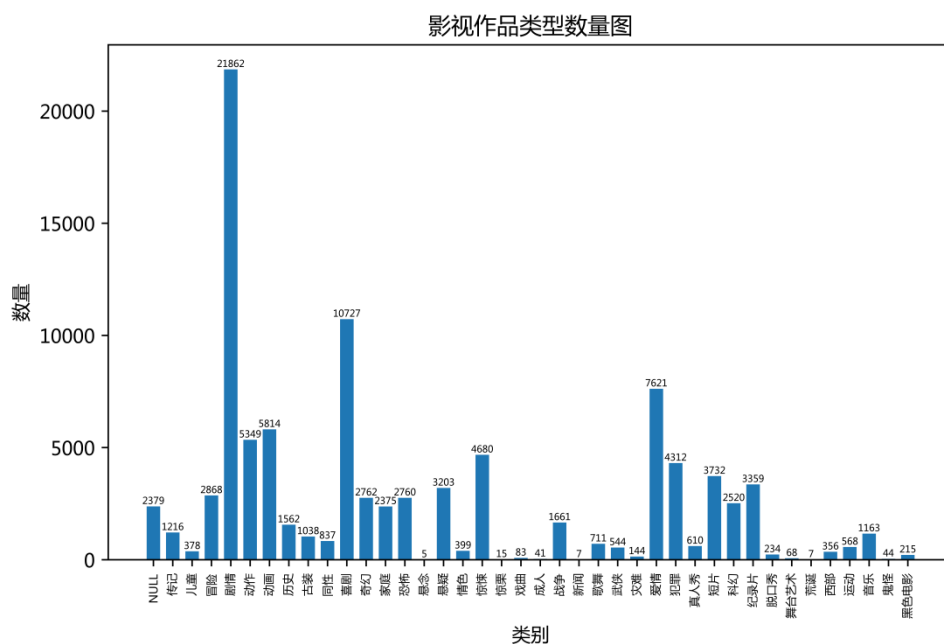


图 6-8 影视作品类型数量图

在所有的电影中剧情、喜剧、爱情类型的电影比较多，其中剧情的作品是爱情的作品的两倍左右。

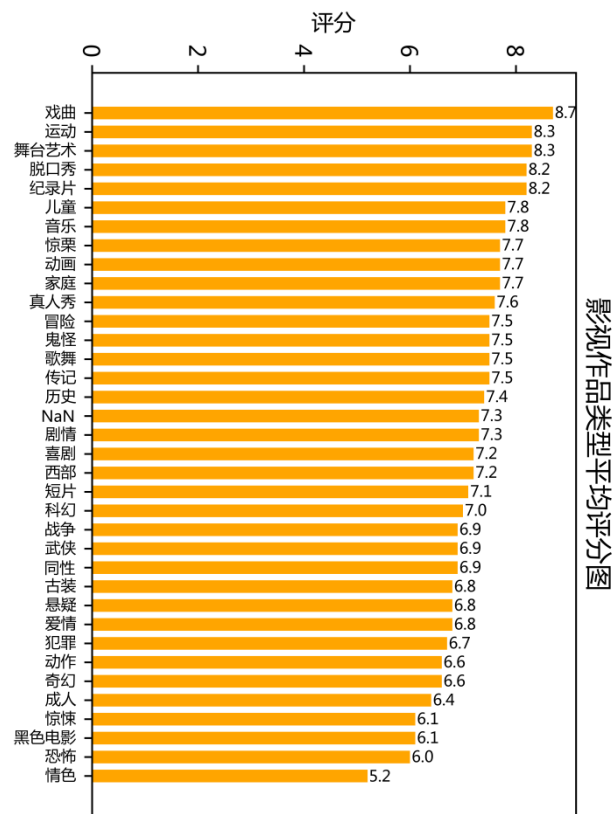


图 6-9 影视作品类型平均评分图

在全部的电影类型评分中，戏剧，运动，舞台艺术，脱口秀，纪录片评分较高，剧情因数量多，良莠不齐，导致评分不高。

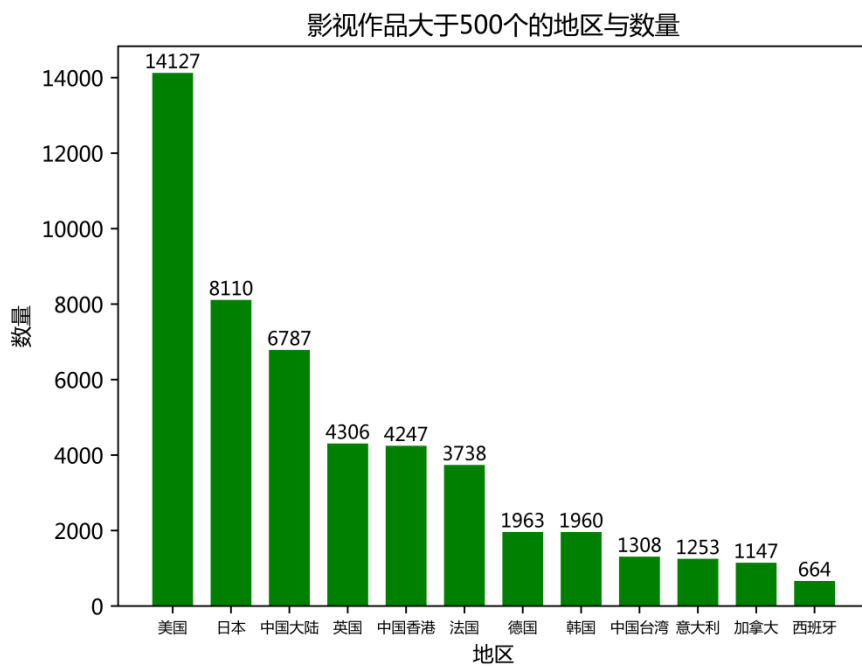


图 6-10 影视作品大于 500 的地区与数量

在所有的影视作品中每个地区的作品数量排行中，美国的作品数量遥遥领先。

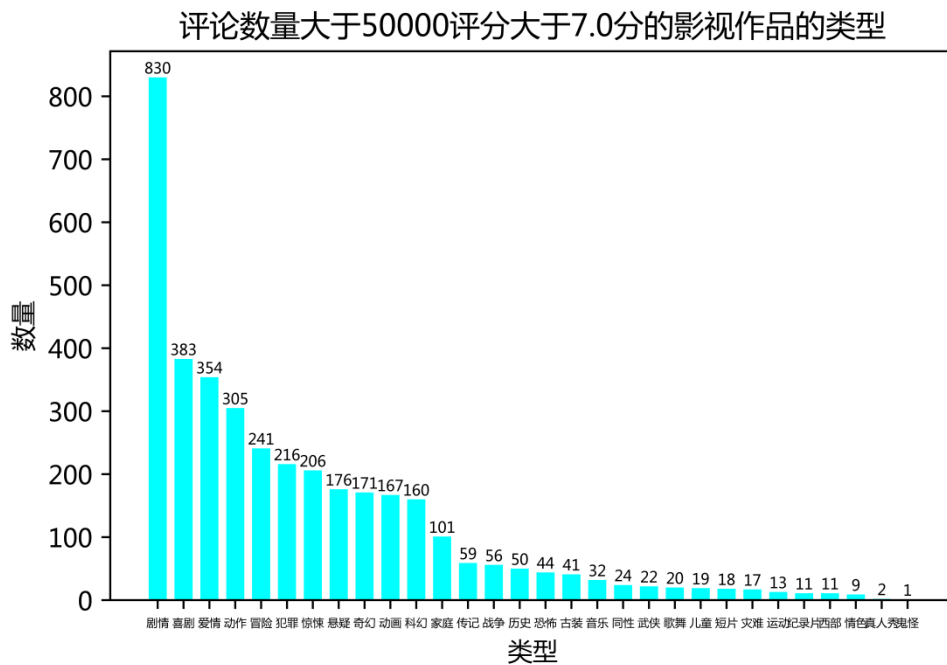


图 6-11 评论数量大于 50000 评分大于 7.0 分的影视作品的类型

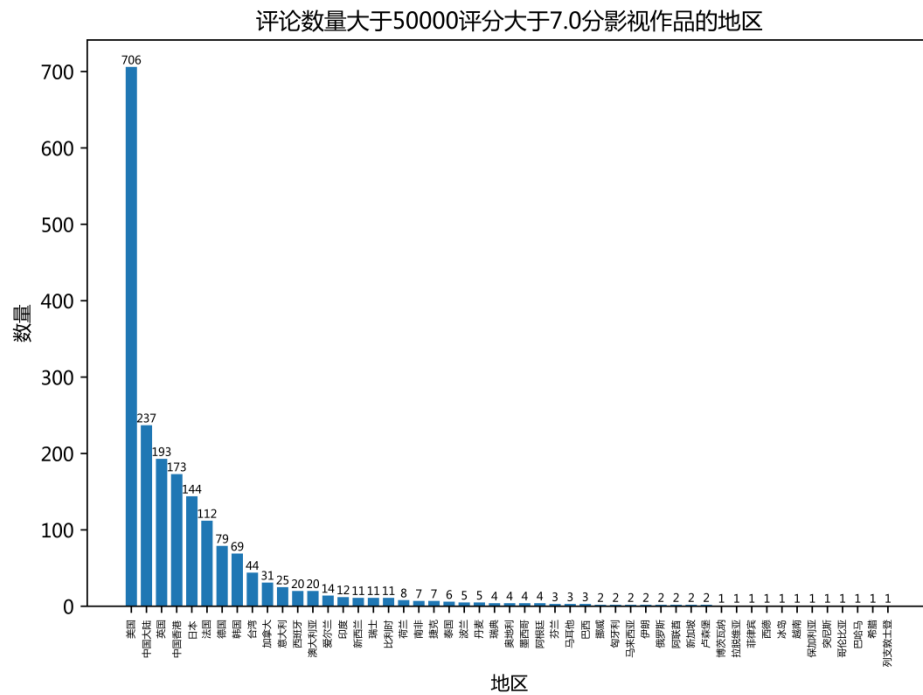


图 6-12 评论数量大于 50000 评分大于 7.0 分的影视作品的地区

评论数量大于50000评分大于7.0分影视作品的地区占全世界比例（部分）

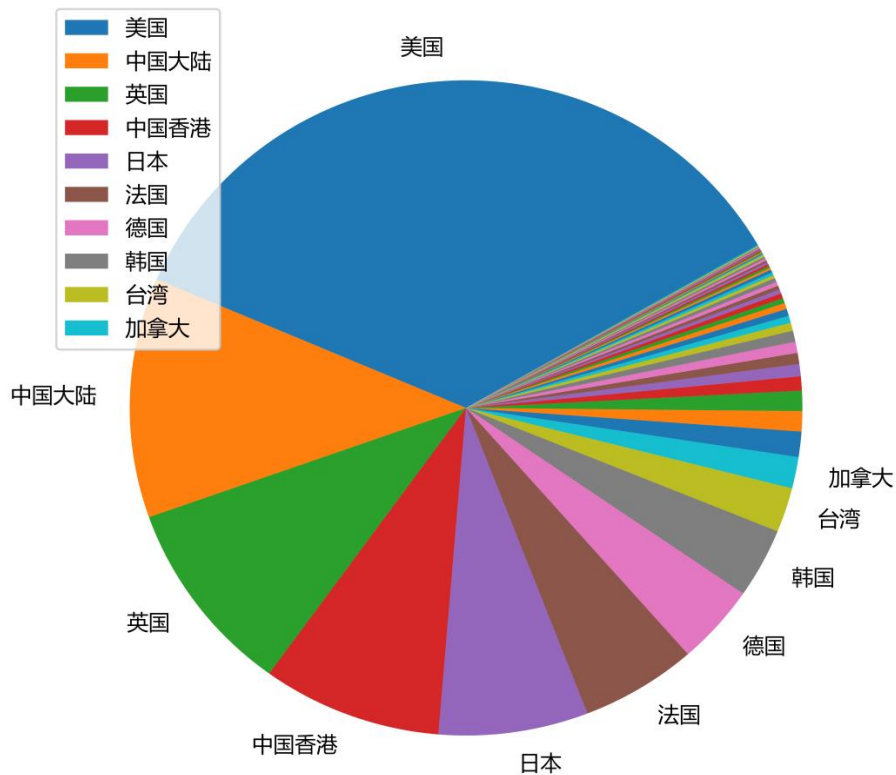


图 6-13 评论数量大于 50000 评分大于 7.0 分的影视作品的地区占全部的比例 (部分)

在所有的作品中评论数量大于五万评分大于 7.0 分的作品中认定为好的作

品 在好的作品中，剧情因为数量多而霸占榜首，喜剧、爱情、动作分列第 2、3、4 名。在评分高的作品制作地区中美国的作品的好评率也是遥遥领先。

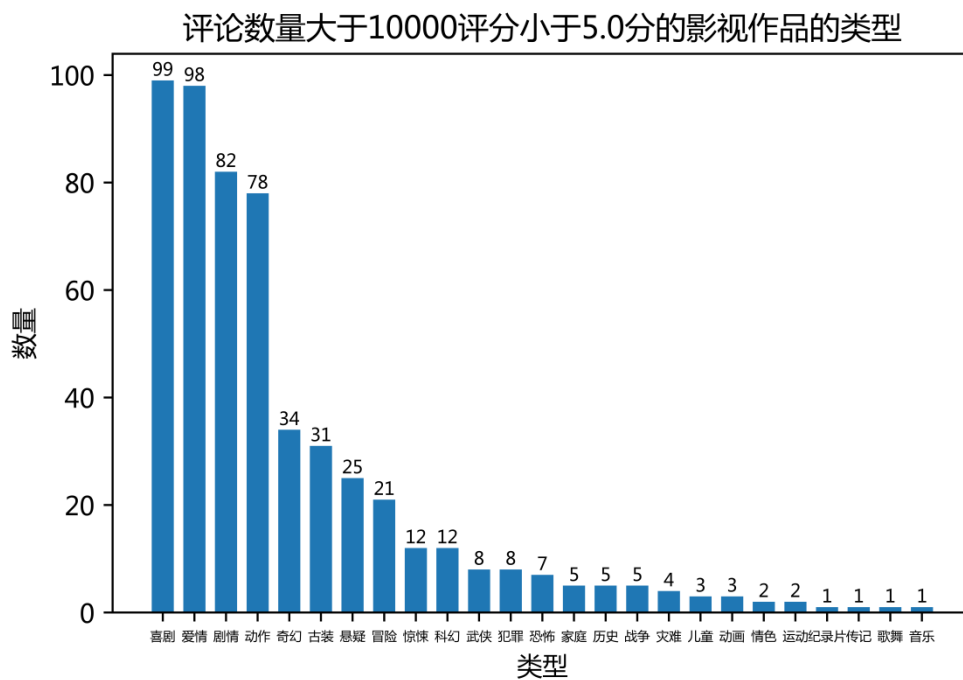


图 6-14 评论数量大于 10000 评分小于 5.0 分的影视作品的类型

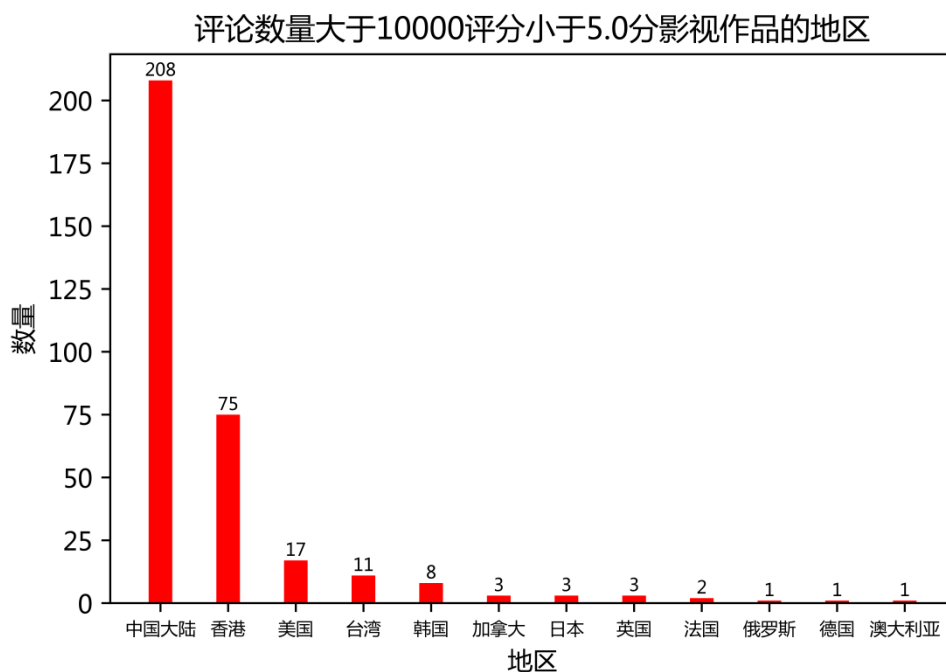


图 6-15 评论数量大于 10000 评分小于 5.0 分的影视作品的地区

评论数量大于10000评分小于5.0分影视作品的地区占全世界的比例

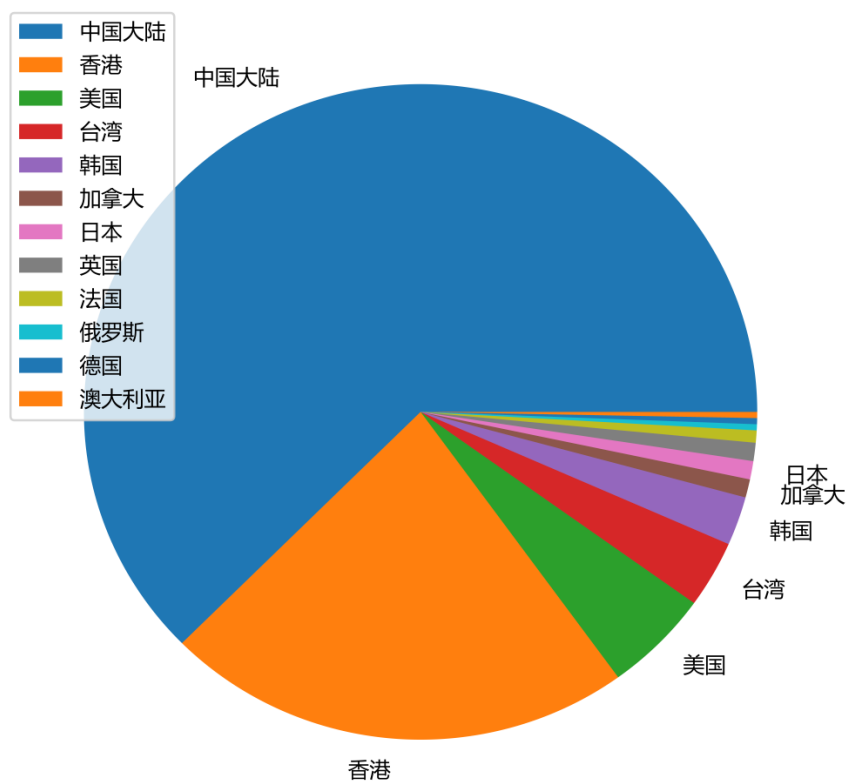


图 6-16 评论数量大于 10000 评分小于 5.0 分的影视作品的地区占全部的比例

在所有的作品中评论数量大于一万，评分小于 5.0 分的作品中认定为差的作品。在的差作品中，喜剧、爱情、剧情、动作位列第 1、2、3、4 位。在评分差的作品制作地区中中国大陆的的作品差评率太高了。

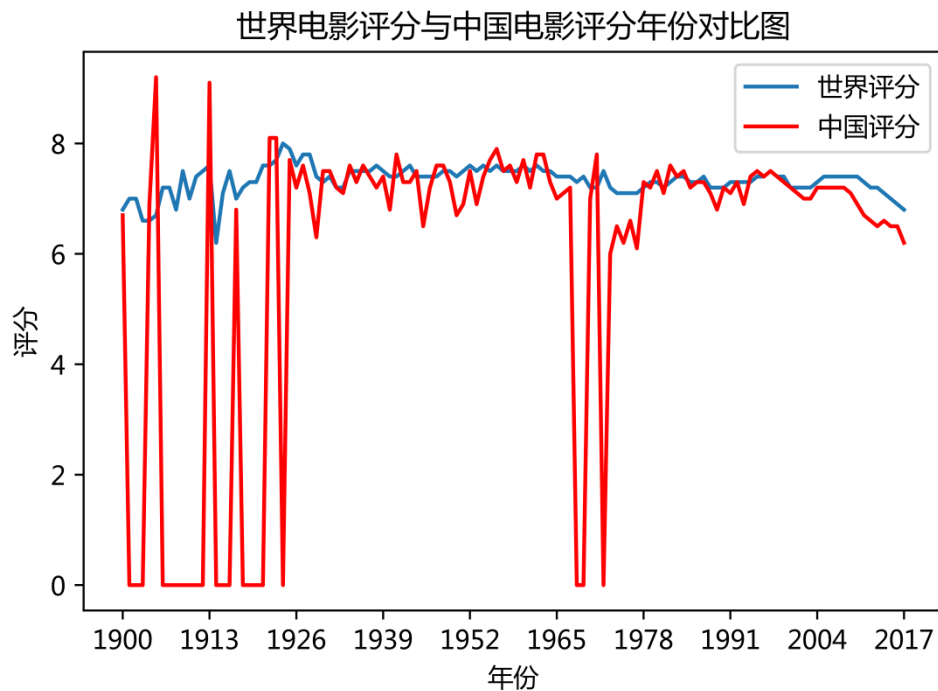


图 6-17 世界影视作品评分与中国影视作品评分对比图

在世界电影史上评分与中国电影史上评分中，中国的评分和世界评分中没有太大差别，但到了二十一世纪评分都有所下滑。

## 6.6. 分析报告

近些年来，电影类别多以剧情、喜剧、爱情、动作等类型呈现，数量颇多。其中，无论从影评数量、电影评分高低来讲都是美国大片居于第一位，中国大陆的电影影评较少，评分较低。近些年，电影类别种类增多，其中以戏曲、运动、舞台艺术等电影类别评分较高，数量不多，受人喜欢，熏陶人的情操。但剧情类电影数量更多，超过同期电影两倍多，甚至更多。从世界电影评分与中国电影评分年份对比中曲线可以看出，中国电影趋势和世界电影的趋势差不多，也是颓废状态，但是分数更加低，大概中位数在 6 分以下，甚至没有评分。从 1900 年，第一部有声电影诞生，电影数量在前期一直呈缓慢递增的速度增长。一直到了 21 世纪前后，电影数量才有明显的增长。

## 7. 总结与展望

### 7.1. 论文总结

上文所讲述的是我们通过 Hadoop 完成电影大数据的分析的过程。

从初步对 Hadoop 的了解,熟悉,到数据平台的搭建,再到电影数据的抓取,以及最后数据的处理和结论的得出,我们每个人都收益颇丰。

日常学习中,对大数据的学习产生的浓厚兴趣和对学习更多 Hadoop 知识的渴望,促使我们更加积极的参与到这次的设计中。在制作的过程中遇到很多不懂的问题、走过很多弯路使设计变复杂、无从下手等多种状况,通过请教老师和与同学探讨、查阅资料等方法最终对问题有了正确的解决方法。此次的毕业设计制作的过程中,我们学到了更多的关于 Hadoop 的知识和技能,使我们对于大数据方面更加的感兴趣、更能深入的理解它,并满足我们所需要的要求。在不断摸索和学习的过程中,我们也是不断磨合,彼此之间的合作也是愈加默契,最后在大家的努力下完成了此次毕业设计。

从最初的茫然,到慢慢的进入状态,再到对思路逐渐的清晰,到今天紧张而又充实的毕业设计终于落下了帷幕。回想这段日子的经历和感受,我们拥有了无数难忘的回忆和收获。

想要出色完成本次设计,首先在硬件上我们应该有一台合适的电脑,能够正常的运行 VMware workstation pro 是必需条件。

由于准备初期,我们对 Hadoop 的了解明显不够,我们百度搜索,向老师请教相关的知识。通过我们的学习,在不挺的尝试下,我们逐步完成 VMware workstation pro 和 Ubuntu 的安装,以及 Hadoop 搭建。做好初步的准备工作。

所谓大数据,最关键的还是数据的获取以及分析工作,相关程序的编写也是我们毕业设计的关键部分,我们从图书馆借阅了相关书籍,通过参考相似的案例,进行了多次的尝试。最终得到实验结果。

整个毕业设计的过程一共持续了三个月的时间,这三个月是我们整个大学生活中收获最大,效率最高的一段时间。它是一段特殊的经历,给大学生活增添了最丰富的色彩。

三月份开学初,确定以“电影数据分析”作为我们的毕业设计课题。带着对于大数据兴趣和对于 Hadoop 和 Ubuntu 的好奇,我们积极的开始了此次毕业



设计。开始之初的我们是迷茫的，面对这个课题空有一番热情却不知道如何下手，经过老师的反复指导和督促，我们才能顺利的完成了这次的毕业设计，在整个过程中我们收获了很多。

资料收集的过程中，每个成员的积极参与是我们更好的完成此次毕业设计的最大动力也是最重要的保障，虽然中间会有意见上的分歧，但是整体目标都是为了更好的完成这次设计，这是最重要的部分。使我们收获更多的是过程中暴露出的一些的问题，比如资料搜集前没有进行细致的划分，所以效率上就不够高。照片重复率很高，相对就浪费了很多的时间。这些问题表现出了我们缺乏经验，也反映出了我们的不够严谨，这些问题在以后的工作生活中回事我们最大的绊脚石，此时问题的暴露对于我们来说是最大的帮助。

毕业设计的制作过程，使我们对于 Hadoop 的操作技巧有了更好的掌握和很大的提高。由于我们知识的欠缺，制作过程中走过不少的弯路，有很多的问题看似很简单，却需要准确考虑，需要不断地完善。考虑不足会走偏方向，走偏之后可能会相对浪费很多时间。通过老师的指导会感觉豁然开朗，更多的学习一项技巧对于之后的设计过程变得相对轻松，节省了更多的时间。当然我们会把这些走弯路的过程当成我们最大的财富，正是这些问题发现和改正的过程给了我们真正的提升。

程序编写过程中会遇到很多自己想不通、解决不了的难题。老师的耐心指导和同学的热情帮助是解决这些问题的重要途径。问题解决的过程，提升了我们的专业技能和总体能力，更为关键的是在这个过程中促进了同学间和同学与老师间的沟通。

论文的整理，是出现问题最多的部分。起初在没有经验的情况下完全走进了死穴，是指导老师的负责态度和认真的指导才能使论文的整理进入正轨。内容整理的部分是每个成员把自己的制作过程做成文档后进行整合，其中会出现很多格式和逻辑方面的错误，需要一点点的进行改正，这方面耗费了较多的时间。整篇论文格式同样在整理的过程中会因为精力不够集中而造成很多错误，通过指导老师的耐心指正和讲解才得以一点点的改正。论文指导的过程中老师给我们指出了很多我们自己没有意识到的问题，以小见大，这些问题恰是我们以后的工作生活中会遇到问题的核心。

整个过程，锻炼了自身的专业能力提升了专业技能，也考验了团队的协作

的能力。至此我们算是经过了所有的考验，呈现出了我们的毕业设计作品，这其中我们道不尽的收获万千。

总结这一次的毕业论文，经历了 Hadoop、Hive、Python 等的数据处理，数据库技术、数据可视化技术，虽然对大神来说非常浅显而没有太多价值，但这也是我这种菜鸡必须需要经历的一步，完完整整，虽有各种波折，所幸全部解决，从中也学到很多，以后编代码思考也会成熟，共勉各位。

## 7.2. 工作展望

大学的三年时光，随着此次毕业设计的完成接近了尾声。这三年中，我们学习了很多专业知识，对自己的专业有了较深层面的了解，对专业知识有了一定的掌握。毕业设计是我们对于这三年的一个交代，也是这三年留给我们的最重要财富。把平时的知识运用于实践中，锻炼了我们的适应能力，巩固了专业技能，并在设计过程中学会了自己发现问题并解决问题。

在有限的时间内完成这次的设计，离不开老师的指导和同学之间的团结协作。这几个月里确实学到了太多的东西，关于人际关系的、关于专业技能的。学习的过程也是自我发现问题的过程，此次的毕业设计使我们更多的发现了自身所存在的问题，能够积极的改正就是对以后学习工作的最好铺垫。此次毕业设计中学习了更多的专业知识，也使我们的就业路更加宽广，有了更好的发展空间。

即将离开学校，但我们并不会放弃学习。在以后的工作学习中我们会保持这份积极，及时的发现自身的问题并加以改正。努力学习更多的技能，成为更好的自己。

毕业设计的顺利完成离不开老师的指导和同学的协助，感谢老师的指导，感谢同学的协助，更加感谢成员们的努力！

## 参考文献、资料索引

文献、资料名称	编著者	出版单位
Hadoop 实战（第二版）	陆嘉恒	机械工业出版社
Python 数据挖掘入门与实践 设计学概论	Robert Layton 【澳】 杜春晓 【译】	人民邮电出版社
大数据（互联网大规模数据 挖掘与分布式处理）	AnandRajaraman 【美】 Jeffrey David Ullman 【美】 王斌 【译】	人民邮电出版社
Python 数据可视化	科斯 拉曼 【印度】 程豪 【译】	机械工业出版社
Python 网络爬虫实践	胡松涛	清华大学出版社
Ubuntu Linux 轻松入门	朱维刚	化学工业出版社
大数据技术原理与应用（第 二版）	林子雨	人民邮电出版社
干净的数据（数据清洗入门 与实践）	Megan Squire 【美】 任政委 【译】	人民邮电出版社
Python 参考手册（第四版）	David M.Beazley 【美】 谢俊 杨越 高伟 【译】	人民邮电出版社
Python 参考手册(第四版 修 订版)	David M.Beazley 【美】 谢俊 杨越 高伟 【译】 宋秉金 【审校】	人民邮电出版社

资料索引：

<http://dblab.xmu.edu.cn/> 厦门大学数据库实验室

<http://Python.jobbole.com/85653/> 数据可视化

## 致 谢

三年的大学生活，随着这次毕业设计的完结接近了尾声。这个时候我们丝毫没有感觉到想象中的如释重负，因为我们知道之后的路会更加艰辛。这次为期三个月的毕业设计中暴露出的我们自身存在的问题，还需要我们更加努力去解决。这三年中收获的百般滋味更值得我们用一生去慢慢品味。

看着自己的毕业设计作品，这些不再陌生的代码，这些并不陌生的文字，不禁让我们感慨万千。对我们来说它不像是作品，它是人生的一次重要的成长经历。而它的呈现更是集结了太多人的心血和深切期望。

对于这次的毕业设计，我们最应该感谢我们的指导老师李晓粉。整个过程的每一步进展都离不开李老师的督促和指导。毕业设计课题的选定得到李老师的肯定，这是我们最大的动力。这段时间的学习将是我们受用一生的财富。在此谨向李晓粉老师致以诚挚的谢意和崇高的敬意。

这次毕业设计的顺利完成，是小组成员共同努力的结果。谢谢共同努力、谢谢相互的帮助。我们有着共同的爱好，我们怀着共同的目标，我们一起向前奔跑，为了这次的毕业设计，为了给自己的大学交出一份最满意的答卷。真心感谢有你们。

感谢父母的养育和培养，才使我拥有上大学的机会，拥有这次难得的经历。在设计的过程中会有很多困惑，很多心情低落的时候，父母给我很多的鼓励，这是我学习生活中最大的动力。

感谢舍友，三年的相处中，你们一直如家人般的陪伴，使彼此的生活从不缺乏温暖。在此次的毕业设计中，你们给了我很多的支持和帮助，谢谢你们！

最后，借此机会向威海职业学院的所有教导过我们的老师们表示衷心的感谢，感谢你们的辛勤培养，教育我们做人，教会我们做事，传授我们知识，未来的路我们也一定不会让你们失望。

整篇论文、整个毕业设计就这样在我们的意犹未尽中进入了尾声，在此诚挚的感谢为这次毕业设计奉献过力量的每一个人，谢谢你们。