大数据平台下的数据挖掘算法设计与实现

摘 要

目前运用单台计算机进行数据挖掘的程序，在处理海量数据时会明显存在计算能力不够，存储空间受限，内存不够大等问题。近年来，伴随着数据库技术的飞速发展，数据库所保存的数据从过去仅管理一些简单数据发展到管理由各种计算机所产生的图形、图像、音频、视频、网页等多种不同的数据结构，在数据量日益增长的情况下，会产生非常庞大的数据信息。庞大的信息量会给数据处理工作带来明显的负面印象，过多无用的信息会产生较大的信息距离和有用知识的丢失。因此一个高可用性，用户友好的系统十分有必要。

本文以建立可用性高的数据挖掘系统为目标，通过常见的数据挖掘算法和对Hadoop平台的研究，实现了基于Spring boot框架的数据挖掘系统。该系统实现了5种Java编写的算法并实现了这集中算法的Map Reduce，使其可以充分利用Hadoop平台的分布式特性，大大提高算法运行效率。利用Layui实现了可交互的操作页面。利用Hadoop的文件系统HDFS的Java api来进行文件操作，通过ssh对远程的Linux主机执行运行算法等命令。

关键词　大数据平台；数据挖掘；Hadoop

Design and Implementation of Data Mining Algorithm Based on Big Data Platform

**Abstract**

At present, there are obvious problems such as insufficient computing power, limited storage space and insufficient memory in the data mining programs using single computer when dealing with massive data. In recent years, along with the rapid development of database technology, database saved data from the past only some simple data to management produced by a variety of computer graphics, images, audio, video, web pages, such as a variety of different data structure, in the case of a growing amount of data, can produce a very large data information. Large amount of information will bring negative impression to data processing, too much useless information will produce large information distance and loss of useful knowledge. Therefore, a highly available, user-friendly system is necessary.

In order to establish a data mining system with high availability, this paper realizes a data mining system based on Spring Boot framework through common data mining algorithms and the study of Hadoop platform. The system implements 5 kinds of Algorithms written in Java and implements Map Reduce of these algorithms, so that it can make full use of the distributed characteristics of Hadoop platform and greatly improve the running efficiency of the algorithm. Use Layui to realize interactive operation page. The Java API of HDFS, the file system of Hadoop, is used to operate files and run algorithms and other commands on remote Linux hosts through SSH.

**Keywords**Big data platform, Data mining, Hadoop

目 录

摘要 I

Abstract II

[第1章 绪论 1](#_Toc104635039)

[1.1 课题来源及研究的背景意义 1](#_Toc104635040)

[1.1.1 课题来源及研究背景 1](#_Toc104635041)

[1.1.2 研究意义 1](#_Toc104635042)

[1.2 国内外研究现状分析 2](#_Toc104635043)

[1.2.1 国外研究现状 2](#_Toc104635044)

[1.2.2 国内研究现状 2](#_Toc104635045)

[1.3 本节研究内容及章节安排 3](#_Toc104635046)

[1.3.1 本文研究工作 3](#_Toc104635047)

[1.3.2 本文章节安排 3](#_Toc104635048)

[第2章 关键技术介绍 4](#_Toc104635049)

[2.1 平台技术架构 4](#_Toc104635050)

[2.2 论文关键技术 5](#_Toc104635051)

[2.2.1 Hadoop平台 5](#_Toc104635052)

[2.2.2 Spring boot框架 7](#_Toc104635053)

[2.2.3 前端相关技术 7](#_Toc104635054)

[2.3 数据挖掘相关算法 7](#_Toc104635055)

[2.3.1 基于物品的协同过滤算法 7](#_Toc104635056)

[2.3.2 Page Rank算法 8](#_Toc104635057)

[2.3.3 K-means聚类算法 8](#_Toc104635058)

[2.4 本章小结 9](#_Toc104635059)

[第3章 系统算法的设计与实现 10](#_Toc104635060)

[3.1 总体功能 10](#_Toc104635061)

[3.2 算法的设计 11](#_Toc104635062)

[3.2.1 基于物品的协同过滤算法 11](#_Toc104635063)

[3.2.2 Page Rank算法 11](#_Toc104635064)

[3.2.3 K-means聚类算法 12](#_Toc104635065)

[3.2.4 键值对排序算法 13](#_Toc104635066)

[3.2.5 Word Counts算法 14](#_Toc104635067)

[3.3 数据结构 15](#_Toc104635068)

[3.3.1 输入数据的数据结构 16](#_Toc104635069)

[3.3.2 输出结构的数据结构 17](#_Toc104635070)

[3.4 相关类设计与实现 17](#_Toc104635071)

[3.4.1 基于物品的协同过滤算法 18](#_Toc104635072)

[3.4.2 Page Rank算法 19](#_Toc104635073)

[3.4.3 K-means聚类算法 20](#_Toc104635074)

[3.4.4 键值对排序算法 21](#_Toc104635075)

[3.4.5 Word Counts算法 21](#_Toc104635076)

[3.5 本章小结 22](#_Toc104635077)

[第4章 系统的设计与实现 23](#_Toc104635078)

[4.1 系统整体架构 23](#_Toc104635079)

[4.2 系统的设计 24](#_Toc104635080)

[4.2.1 系统的主要功能 24](#_Toc104635081)

[4.2.2 系统的前端设计 25](#_Toc104635082)

[4.2.3 系统的后端设计 26](#_Toc104635083)

[4.2.4 数据存储的设计 27](#_Toc104635084)

[4.3 系统功能的实现 28](#_Toc104635085)

[4.3.1 登录注册注册功能 28](#_Toc104635086)

[4.3.2 HDFS文件管理 29](#_Toc104635087)

[4.3.3 运行算法功能 31](#_Toc104635088)

[4.3.4 显示结果功能 32](#_Toc104635089)

[4.4 本章小结 34](#_Toc104635090)

[第5章 测试与分析 35](#_Toc104635091)

[5.1 测试目的 35](#_Toc104635092)

[5.2 测试环境 35](#_Toc104635093)

[5.3 测试过程 35](#_Toc104635094)

[5.3.1 Hadoop平台 35](#_Toc104635095)

[5.3.2 系统的测试 36](#_Toc104635096)

[5.3.3 算法的测试 38](#_Toc104635097)

[5.4 结果分析 40](#_Toc104635098)

[5.5 本章小结 40](#_Toc104635099)

[结论 41](#_Toc104635100)

[致谢 42](#_Toc104635101)

[参考文献 43](#_Toc104635102)

[附录A 44](#_Toc104635103)

[附录B 49](#_Toc104635104)

# 绪论

## 课题来源及研究的背景意义

### 课题来源及研究背景

近年来，数据库的技术伴随着数据存储技术的发展，数据所存储的也不仅仅是简单的文字数据，而是根据多元，更加复杂的数据结构以及文件等，同时每份数据的大小也在不断增长，并且数据量也越来越大。随着互联网和互联网技术的快速发展，每天产生的数据正以指数级的速度增长，对这些大量数据的处理与分析具有巨大的应用价值[1]。因此，目前的企业与用户都迫切希望能对当前环境下的海量数据与信息进行充分利用，对数据进行分析与处理，获取其中隐藏的关键信息，以更好地利用这些数据。

正是在这样的条件下，数据挖掘技术应运而生。数据挖掘通俗来讲，就是从大量的数据之中挖掘出数据蕴含的潜在规律、提炼出具有价值的知识的过程，数据挖掘技术是对在这一过程中运用的方法与技术的通称[2]。数据挖掘的目标是提取有效的、新颖的、潜在的从海量数据中提取有用且最终可理解的模式大量的数据[3]。

### 研究意义

随着互联网的发展，传统的数据分析方法已经不能满足大型企业的需求。大数据平台技术已深入到各个行业，如通信、能源、医疗、零售、游戏、体育、影视、旅游、交通行业等行业[4]，是当下建模领域与仿真领域起到基础支撑作用的技术。社会上各领域都对大数据平台技术有着一定的研究，因为目前大数据技术在各个行业都成为了研究重点。

本课题所要研究的为在大数据平台下的数据挖掘算法的设计与实现，将数据挖掘技术封装成对外的服务，方便用户挖掘出海量的数据中的价值，做出有益的预测。在信息时代的背景下，数字化技术和信息化技术在各行各业的应用，随着互联网技术、信息技术、物联网技术、云计算技术等数字信息技术的高速发展，结合当前高速发展的移动互联网技术以及数字地球技术的发展与应用，全世界范围内数据呈爆炸式增长[5]。所以在各行各业对于数据挖掘技术的需求都很迫切。

## 国内外研究现状分析

### 国外研究现状

对于大数据相关技术，国外有很多知名IT企业都有发展相关的技术。比如传统的IT企业如大家耳熟能详的ORACLE、IBM等公司，传统的IT公司都有长期经营所积累下的技术沉淀，品牌效力，企业的资金链也远超同行业其他公司。这些企业都有着大数据平台的解决方案，其中部分企业通过自研提出了独立的大数据平台解决方案，另一部分则通过收购其他大数据公司，将这些公司已经实现的技术为支撑实现自己的解决方案。国外的高等学校对于大数据相关的技术研究通过学术期刊和会议能有很明显的体现，国际著名期刊《Science》[6]和《Nature》[7]就曾出版了《Dealling with Data》和《Big Data》，这两篇大数据领域的论文，讨论了面临大数据时代，各个领域会遇到的问题以及解决方案。在高校中，大数据平台也有了很多研究成果，比如美国卡耐基梅隆大学针对大数据机器学习特点的 Petuum大数据分布式机器学习平台由计算机学院机器学习系 Eric Xing教授研发的[8]。该平台作为一种开源的分布式框架，为机器学习提供了通用的算法接口。至于新型的互联网企业主要包含耳熟能详的Google、Amazon、Facebook等。这些新型的互联网企业有着良好的市场和用户群体，通过利用大数据不断对企业的商业模式和服务进行改良，构建出了一个良好的互联网大数据应用生态圈。其中比如Google公司在大数据领域就有很高的地位，很多研究成果在学术界都令人赞叹。Google公司自研的Caffeine平台，就是一个对大数据适配很好的平台，可以为分布式数据库加上自订的索引，为网页搜索引擎提供基础的技术支持。

### 国内研究现状

对于大数据相关技术，国内的研究力量主要为高校的科研领域以及新型的互联网企业。国内高校在大数据研究方面也有着不错的成功，如由中国科学院计算技术研究所和中国移动与合作开发的PDMiner平台[9]。该平台以ETL和Hadoop为基础，用于解决通讯领域的云计算与数据挖掘问题。全国各高校也都有各自对于大数据的相关研究成果，比如深圳大学的深圳大学大数据分析平台，就是由该校大数据技术与应用研究所研发的。而对于新型互联网企业，主要即为BAT为首的公司等，阿里巴巴在2011年推出“淘宝指数”大数据分析系统，便是该公司对于解决自身需求所研发的产物。用于卖家通过历史的销售信息和“淘宝指数”进行库存和生产相关的决策，通过这种决策，用户也可以购买到价格合适性价比高的产品。百度在大数据方面并没有采用原生的Hadoop技术架构，综合应用了各种相关技术，比如Map Reduce、DAG等多种技术用于满足各种不同的需求。

## 本节研究内容及章节安排

### 本文研究工作

本次课题主要承担的工作主要是将数据挖掘的算法在大数据平台下实现，并且通过网络编程等技术让算法易于使用。

主要在于将几种数据挖掘算法通过java实现，然后在实现的算法的基础上实现适合Hadoop平台使用的Map Reduce并行化版本。将算法调试好，封装成jar包，导入远程Linux平台中的Hadoop平台中。以便后期调用。

利用网络编程技术，实现用户友好的交互界面，操作简单的大数据平台下的数据挖掘算法平台，实现向Hadoop平台上传文件，调用算法，设置参数，并展示结果等功能。该平台可以让用户零门槛使用数据挖掘算法用以满足日益增长的数据挖掘需求。

### 本文章节安排

本论文共5个章节。

第1章是绪论。介绍了本课题的课题来源及研究的背景和意义、国内外研究现状与展望、以及本文研究内容及章节安排。在本章节安排种概况了本文的组织结构。

第2章是关键技术介绍。对构建平台所用的技术以及平台的技术架构进行介绍，同时对数据挖掘算法进行简要的介绍。

第3章是算法的设计与实现。对本文中实现的5种数据挖掘算法进行介绍，并详细描述对算法的Map Reduce并行化的实现。

第4章是系统的设计与实现。介绍整个平台的设计与实现，所用技术如何应用，如何实现远程调用的整个平台。

第5章是测试与分析，该章节通过对系统的测试，算法运行效率的测试等，分析了系统运行的稳定性，以及算法的运行效率是否满足预期要求。

# 关键技术介绍

## 平台技术架构

大数据平台下的数据挖掘系统基于网络编程环境实现了5种并行算法，这5种算法都是由Java编程实现的，利用Map Reduce编程模型实现了算法基于Hadoop平台的并行化执行版本。算法与用户之间的交互层利用了Spring boot和Layui等网络编程技术，争取为用户带来良好的使用体验。图2-1为本系统的系统结构图。

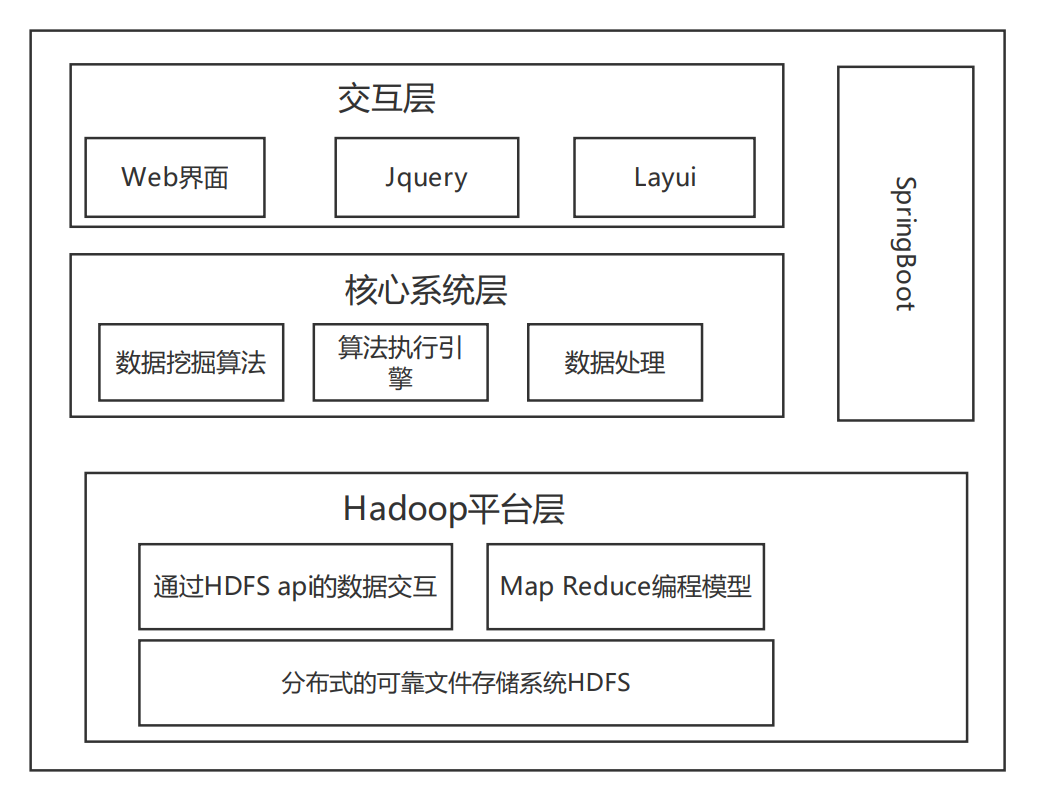


图2-1 系统结构图

本系统整体架构有三层，交互层、Hadoop平台层，系统核心层。交互层提供基于网页的操作页面，主要基于html网页来实现，利用了Layui框架以及Jquery技术等，用户可以在这里上传要挖掘的数据，选择使用的算法，提交算法请求，查看算法运行结果等。Hadoop平台层主要基于Hadoop自带的分布式的可靠文件存储系统（HDFS），HDFS的实现并行算法的基础。Hadoop中还存储了上述提到的编写完成封装成jar包的数据挖掘算法，并且通过Java的HDFS的api来对HDFS中的文件进行操作，上传，读取等。系统核心层主要处理前端交互层发来的请求，并且根据请求的不同提供相应的数据或者调用相应的操作，比如获取HDFS中的文件目录，或者对某一文件调用某种数据挖掘算法。

## 论文关键技术

### Hadoop平台

Hadoop平台起源于2002年，当时Hadoop是包含在Apache Nutch项目中的一部分，该项目为Apache Lucene的子项目，它最初的构想是用于解决 Nutch（开源的爬虫和搜索引擎）中的可扩展性问题[10]。

而Hadoop之所以能实现分布式存储计算等能力与它的特殊的文件存储系统的分不开关系的。Hadoop的文件系统叫做HDFS（Hadoop Distributed File System）。HDFS有着容错性高，可以用来部署在低廉的硬件上等优点，并且在访问应用程序的数据时，对数据具有很高的吞吐量，与超大数据集的适配性很高等特点。通过Hadoop分布式文件系统，可以实现海量数据在廉价设备上的分布式存储，采用主/从模式，包括多个Namenode和多个Datenode[11]。NameNode是一个中心型服务器，负责管理文件系统的元数据和客户端对文件的访问，DataNode 存储实际的数据。SecondaryNameNode 是 NameNode 的备份节点，它会定期对 NameNode 上的系统信息文件进行备份。客户端通过调用HDFS的Java Api接口对文件进行相应的操作。图2-2为HDFS文件系统结构图。

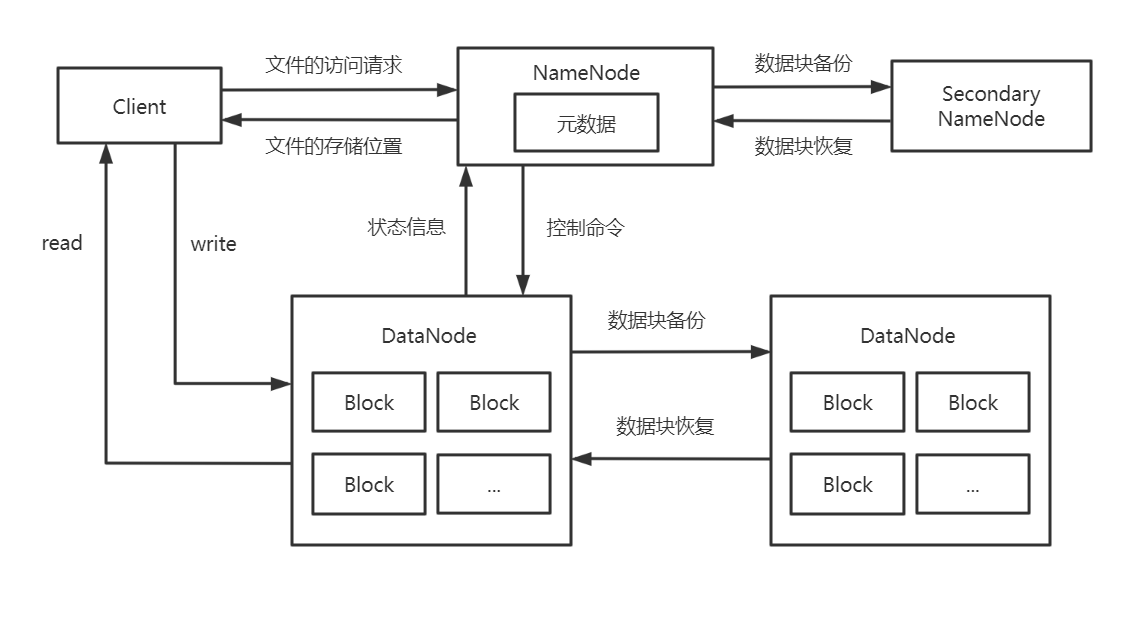


图2-2 HDFS文件系统结构图

Map Reduce是一种计算模型，它可以将海量数据分析处理的任务拆解成一个个可以在服务器集群中并行执行的任务，对这些单个任务并行处理完的结果进行合并计算得到最终结果[12]。Hadoop的Map Reduce模型是在受到Google的启发而实现的。Map Reduce作为Hadoop技术的核心，Map Reduce的原理为通过利用Map（）函数将一个完整的任务分割成多个小任务，Map（）函数在分割任务时，处理的数据格式为key/value数据对的格式，并且在shuffle过程中输出的中间结果也为key/value数据对的格式，通过分布式部署到各个节点，运算完成之后在利用Reduce（）函数将map（）后产生的数据对中key值相同的进行合并。图2-3为Map Reduce流程图。

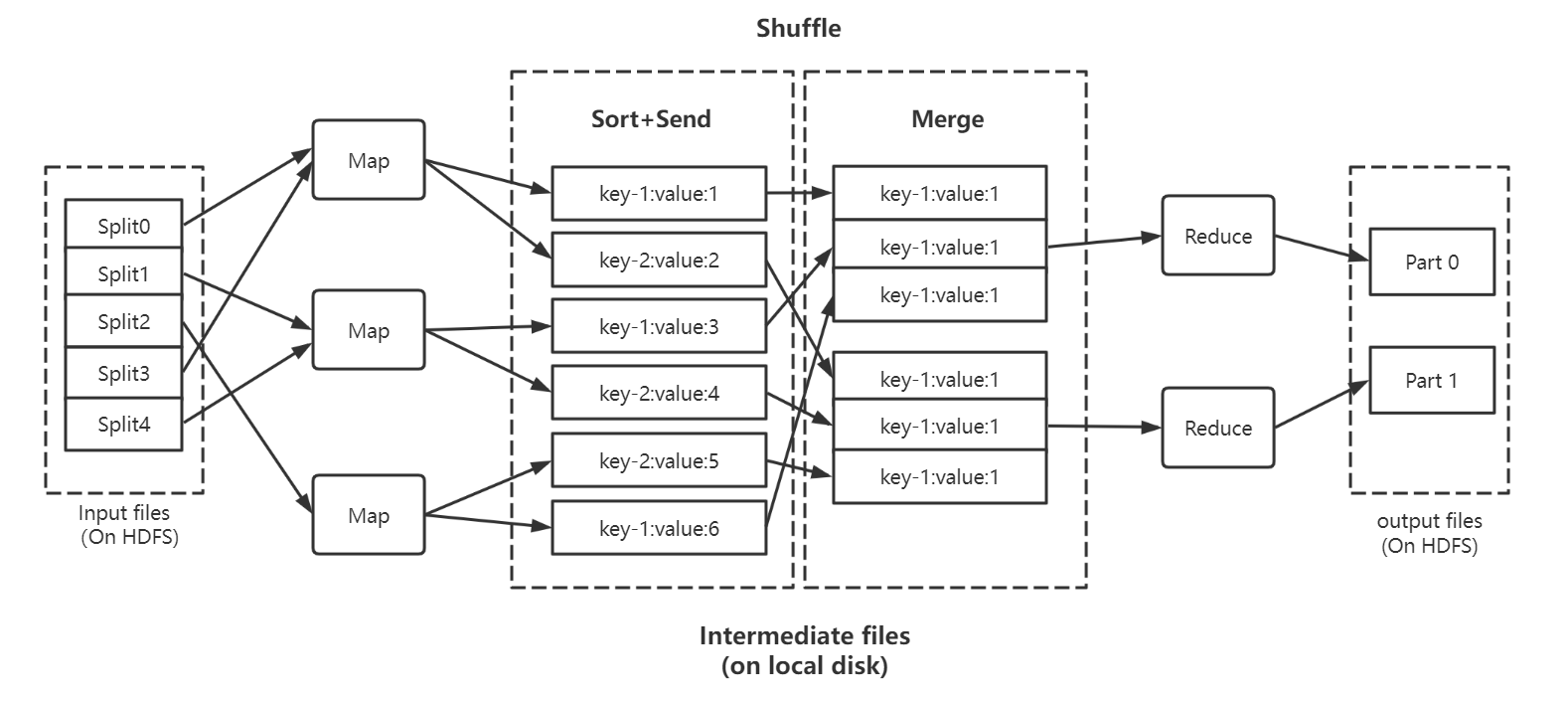


图2-3 Map Reduce流程图

Map Reduce运行框架中包含若干个clent、一个JobTracker和若干个TaskTracker。其中 TaskTracker 又分为 Map Task 和 Reduce Task。JobTracker 作为主控节点，用来调度 MapReduce 任务的运行。Map Reduce作业的运行过程如表2-1所示。

表2-1 MapReduce作业流程运行步骤

|  |  |
| --- | --- |
| 序号 | 流程步骤 |
| 1 | 运行一个 Job 任务。 |
| 2 | 从 jobtracker 发出 get NewJobId()请求，获得新的 JobID。 |
| 3 | 收集作业信息，并将其发送到 HDFS。 |
| 4 | 提交作业，初始化作业 。 |
| 5 | 获取分片信息，创建 map 任务。 |
| 6 | tasktracker 定期给 jobtracker 发送“心跳”，用来充当两者之间的消息通道。 |
| 7 | 作业的 JAR 文件本地化。 |
| 8 | tasktracker 创建本地工作目录，解压 JAR。 |
| 9 | tasktracker 创建一个 TaskRunner 实例。 |
| 10 | 启动一个新的 JVM 来运行每个任务。 |
| 11 | 启动任务，运行 map/reduce。 |

### Spring boot框架

Spring Boot是对Spring的再一次封装，并不提供Spring的核心功能，其包括许多启动器，方便用户快速构建项目，从而达到开箱即用的目的[13]。Spring boot利用了一种叫做控制反转的特性来避免繁琐且固定的配置，控制反转利用了Java的反射机制，实现了对Java对象的统一配置和管理。

### 前端相关技术

本系统的前端是利用Layui框架搭配JQuery技术实现的，利用Layui的table组件实现了文件目录，以及功能按钮的样式等。

Layui是一款基于模块化设计的前端框架，遵循原生 HTML/CSS/JS 的书写与组织形式，提供了丰富的模块，框架中的组件都有可以按需加载，例如layer，table，form等组件，对后端开发很友好。

JQuery是一个快速、简洁的JavaScript框架。有很多的插件，支持跨浏览器，简单易拓展。JQuery可以简化ajax编程，使项目的开发周期更短。

## 数据挖掘相关算法

### 基于物品的协同过滤算法

基于物品的协同过滤算法（Item cf）主要思想为根据用户历史评分计算物品之间的相似度，通过物品相似度和用户历史行为预测用户以往喜欢商品的相似物品[14]。对于物品相似度的计算可以如式4-1所示。

 （4-1）

上式中，表示的为喜欢物品的用户的数量，分子为同时喜欢两种物品的用户的数量。分母不仅有还要有因为如果这两个物品中存在一个热门商品，将会导致热门商品和所有商品的相似度高，因此分母采用了两种物品的用户数量，这样即使其中一种物品的值很大，但是相应的分母也会变大，相当于一种惩罚机制，惩罚了热门物品的权值。在建立了物品相似度矩阵之后，ItemCF算法通过以下公式4-2计算用户对物品的喜爱程度。

 （4-2）

上式中的为用户喜欢的物品的集合，为与物品j最相似的k个物品的集合，是物品j和物品i的相似度，是用户对物品i的评分。该式可以理解为对用户所喜欢的物品列表中的每一个物品i，都根据物品相似度矩阵寻找与其最相似的k个物品，然后用来表示用户对物品的喜好程度。

### Page Rank算法

Page Rank，网页排名算法，是根据网页之间互相的超链接数量，来计算网页的排名的算法。网页中的链接较多，其链接指向的其他网页质量较高，则该网页通常是较重要的网页[15]。

该算法的原理为Page Rank通过让链接来“投票”，通过投票来计算页面的重要性。一个网页的PR值的计算如式4-3所示。

 （4-3）

在上式中为所有链接到网页u的网页集合，网页v是属于集合的一个网页，而则是网页对外的链接数，即为网页的出度。通过多次迭代，收敛PR值，当PR值趋于稳定时即可。

### K-means聚类算法

K-means算法作为一种简单的聚类算法，使用各种编程语言来实现相对容易，并且得到的聚类效果表现良好，得到学术界的一致认可，同时又发展了很多基于 K-means的改进算法，在企业界也被广泛地推广和使用[16]。

该算法计算对象与每个聚类中心的距离的方法如公式4-4所示。

 （4-4）

公式4-4中，表示第i个对象，式中的表示第j个聚类中心的，表示第i个对象的第t个属性，，表示第j个聚类中心的第t个属性。该算法通过依次比较每一个对象计算出的到每一个聚类中心的距离，把对象分配到距离最近的聚类中心的类簇中，得到k个类簇。K-means算法对类簇的定义是类簇中心内所有对象在各个维度的均值，计算公式如公式4-5所示。

 （4-5）

式中，表示第1个聚类的中心，，为第一个类簇中包含的对象数量，表示第一个类簇中的第i个对象，。

## 本章小结

本章主要介绍了系统开发平台的技术框架和平台实现的关键技术，主要包括Hadoop 框架、Springboot 框架以及前端相关技术，这些技术是本系统实现的基础。

# 系统算法的设计与实现

## 总体功能

大数据平台下的数据挖掘算法系统的主要功能需求如图4-1所示。在数据挖掘算法中，并不是所有算法都可以并行实现的，例如常见的Fibonacci 函数：，这种传统意义上用递归去实现的算法因为数据前后项依赖度非常高，所以只能串行计算，并不适合并行运算，即使并行运算，算法的执行效率也不会得到提高。然而实际上并不是所有量级的基本的数据都能通过执行并行算法来提示执行效率，在待处理的数据量较小处理，或者数据的存储结构为普通存储结构的情况下，执行串行算法就足够满足性能的要求，这样的数据集提交到Hadoop平台并行执行算法，对于数据的预处理以及对数据进行的Map Reduce数据拆分和规约的时间要比数据处理的时间消耗要大的多，在这种情况下，对于算法的并行执行，不仅没有提高算法的执行效率，反而会产生很多不必要的数据处理，浪费了时间和资源。但是一旦处理的数据规模够大时，如数据规模较大的情况下，将数据的存储结构选用分布式存储结构，再并行执行算法，此时算法的执行时间远远大于对于数据的预处理和拆分以及规约所需要的时间，在这种情况下并行执行算法能够大大缩短算法执行所花费的时间，提高数据处理的效率。本系统对算法的实现细节进行了封装，实现了5种并行算法，对外提供数据挖掘服务。图3-1为并行算法的详细功能结构。

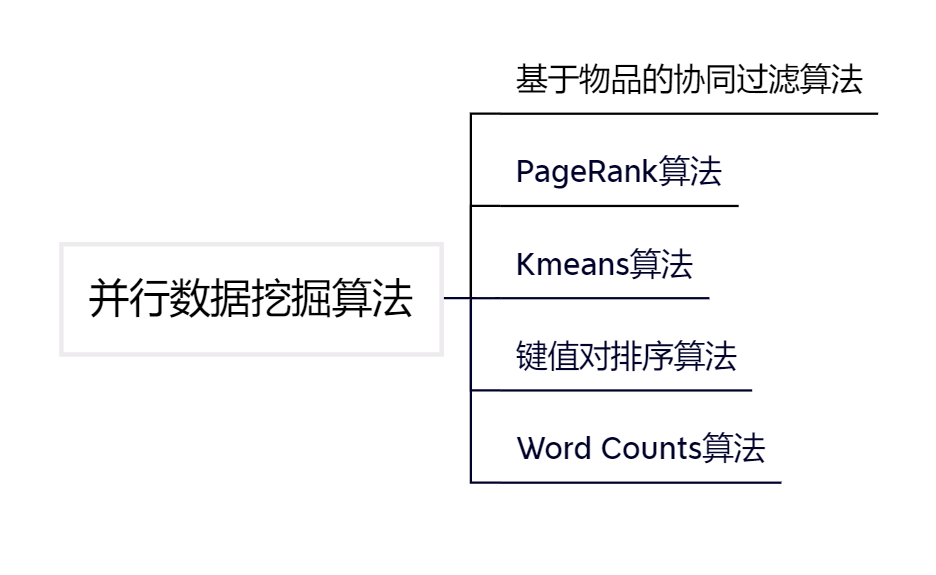


图3-1 并行算法详细功能结构

## 算法的设计

### 基于物品的协同过滤算法

基于物品的协同过滤算法（Item cf）主要可以分为两部分，第一部分为计算物品的相似度，第二部分为根据物品的相似度和用户的历史行为给用户生产推荐列表。该算法的实现利用了三次Map Reduce的过程，第一次Map Reduce用来计算用户的评分矩阵，首先将原始数据进行处理，在Mapper中将输入数据中的UserId作为key，Itemid:Perference的对物品评分作为value进行输出。在Reduce中将上述map的输出聚合作为用户的评分矩阵输出到临时文件中。第二次Map Reduce的作用为计算物品的同现矩阵，Mapper过程中读入数据，将itemid都读取到临时缓存中，全排列作为key输出，对每一个key对应的value都赋值为1将结果输出到临时文件中。第三次Map Reduce中就要将前两次运算的结果读取到缓存中，根据用户的评分记录来查找该用户对应当前物品的喜好度，然后将用户评分过的物品筛除掉，最后将处理后的结果输出到Output文件夹中。图3-2为ItemCF算法流程图。

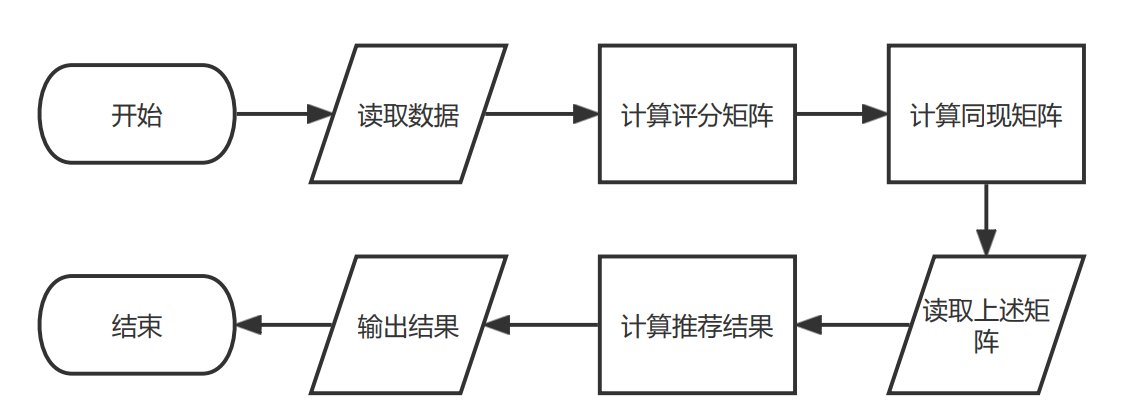


图3-2 ItemCF算法流程图

### Page Rank算法

Page Rank，网页排名算法，该算法的运行步骤可以简单的概况为两步，第一步为给每个网页一个初始的PR值，第二步为通过算法不停的进行迭代，直到达到平稳分布为止。由于该算法需要进行多次迭代运算，所以在该算法的主类中规定了循环次数最大为10次，该算法的实现主要通过自定义的Mapper和Reducer实现，在Mapper中，将网页的Id，网页的PR值，向外链接数等数据进行运算，在Reducer中通过公式将当前一次迭代的PR值计算完成，并输出到临时文件中，之后的每一次迭代所读取的都为上一次迭代所输出的结果，直到结果不在发生变化或者达到最大迭代次数为止。图3-3为Page Rank算法流程图。

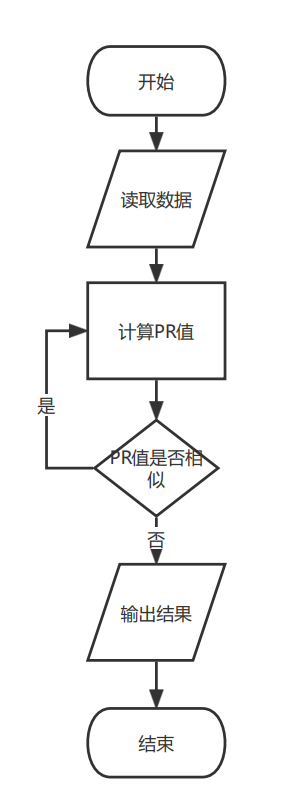


图3-3 Page Rank算法流程图

### K-means聚类算法

该算法的主要思想为在初始给定聚类中类的个数K和这K个类簇的初始中心点的情况下，把每个点分配到与其最近的类簇中，当点都分配完毕后，根据一个类簇内的所有点重新计算该类簇的中心点。进行多次迭代计算，直到类簇中心点的变化与上一次迭代很小，或者达到迭代最大次数。该算法也需要多次迭代，因此在主类中规定最大迭代次数为5次。该算法在实现中也利用了一次Map Reduce的过程，在Mapper的过程中，算法对每一个点进行判断，判断当前的点距离哪一个中心点最近，并将当前点分配给最近的中心点所在的点的集合。在Shuffle过程中，对每一个点的集合将集合内的点进行拼接。在Reducer的过程中，计算每一个集合中，更新中心点之后，所有节点的每一个维度的平均值，在计算完成后将新的中心点输出到临时文件中，每一次迭代读取上一次迭代输出的临时文件，直到满足算法要求，迭代停止。图3-4为K-means聚类算法流程图

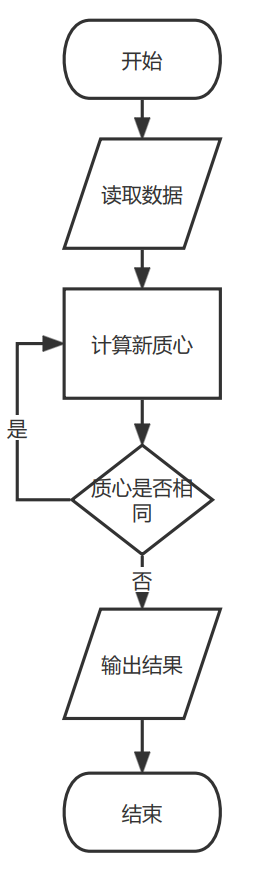


图3-4 K-means聚类算法流程图

### 键值对排序算法

该算法的主要作用为，拼接文档中相同的字符串key所对应的value，并且在拼接的同时将value按升序排列。该算法的实现原理很简单，在该算法的shuffle阶段中，通过将key数值相同的value合并为同一个列表，来进行数据处理，发送到reducer，在reducer中把从mapper接收到的数据进行运算，将key值相同的value进行字符串拼接，在拼接字符串之前进行排序，排序主要通过实现自定义的比较器进行，并且在reduce的时候利用自定义的比较器对value列表进行排序，实现键值对的排序输出。图3-5为键值对排序算法流程图。

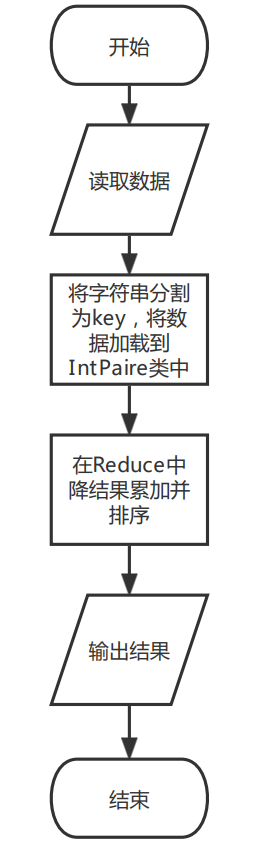


图3-5键值对排序算法流程图

### Word Counts算法

Word Counts，词频统计算法，该算法的主要作用为对目标文档中的语句进行切分，切分为以单词为基本单位，然后将每一个曾出现过的单词的词频记录下来。算法的原理为先通过对文档的分割，将文档分割成以单词为基本单元的形式，然后将单词设置为key，同时把value设置为1，在算法进行shuffle的过程中，将key值相同的value合并为一个列表，然后将这个列表发送给reducer，reducer将从mapper接收结果，然后对相同key的value进行相加，形成最终结果并输出到文件中。图3-6为Word Counts算法流程图。

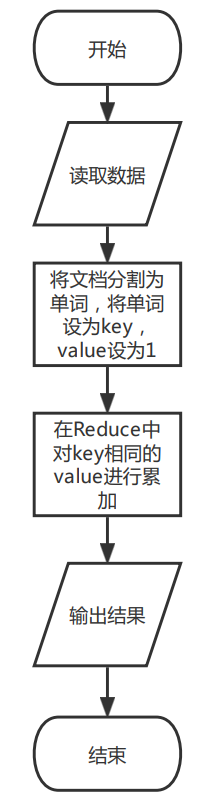


图3-6Word Counts算法流程图

## 数据结构

该系统实现的5种算法，在输入的数据部分以及输出的结果部分所使用的数据结构都完全不同，由于在该系统中后端服务器和Hadoop所在的服务器实际上是完全分离的，对于上传的数据集难以进行修改，在文件提交到后端服务器时就需要格式，数据结构等需要完全正确，否则可能会影响系统对当前数据集的处理。本系统主要接收的数据文件格式以.csv和.txt为主。所以对于这5种算法，需要进行数据格式的规范，并且对上传的文件类型有所要求。

### 输入数据的数据结构

基于物品的协同过滤算法，该算法的输入数据格式如表3-1所示。该算法所需要的文件格式为.csv格式，每个数据之间需要用逗号隔开，一行一组数据。

表3-1 基于物品的协同过滤算法输入数据结构表

|  |  |  |
| --- | --- | --- |
| 字段名 | 说明 | 类型 |
| User\_id | 用户id | Int |
| Item\_id | 物品id | Int |
| Preference | 用户对物品的评分 | Double |

Page Rank算法，该算法的输入数据格式如表3-2所示。

表3-2 Page Rank算法输入数据结构表

|  |  |  |
| --- | --- | --- |
| 字段名 | 说明 | 类型 |
| id | 网页id | Int |
| Rank | 初始PR值 | Double |
| link | 外连接网页id | Varchar |

K-means聚类算法，该算法的输入数据格式如表3-3所示。该算法的输入文件有两个，一个文件为所有点的集合，一个文件为预设的质心的点的集合，两个文件数据结构相同。

表3-3 K-means聚类算法输入数据结构表

|  |  |  |
| --- | --- | --- |
| 字段名 | 说明 | 类型 |
| x | 点的x坐标 | Double |
| y | 点的y坐标 | Double |

键值对排序算法，该算法的输入结构格式如表3-4所示。

表3-4 键值对排序算法输入数据结构表

|  |  |  |
| --- | --- | --- |
| 字段名 | 说明 | 类型 |
| Str | 作为key的字符串 | Double |
| Value | 作为value的值 | Double |

Word Counts算法，该算法的输入没有严格的要求，正常的中英文文档即可。

### 输出结构的数据结构

基于物品的协同过滤算法，该算法的输出数据格式如表3-5所示。

表3-5 基于物品的协同过滤算法输出数据结构表

|  |  |  |
| --- | --- | --- |
| 字段名 | 说明 | 类型 |
| User\_id | 用户id | Int |
| Item\_id | 物品id | Int |
| Preference | 用户对物品的评分 | Double |

Page Rank算法，该算法的输出数据格式如表3-6所示。

表3-6 Page Rank算法输出数据结构表

|  |  |  |
| --- | --- | --- |
| 字段名 | 说明 | 类型 |
| id | 网页id | Int |
| Rank | 最终PR值 | Double |
| link | 外连接网页id | Varchar |

K-means聚类算法，该算法的输入数据格式如表3-7所示。

表3-7 K-means聚类算法输出数据结构表

|  |  |  |
| --- | --- | --- |
| 字段名 | 说明 | 类型 |
| x | 点的x坐标 | Double |
| y | 点的y坐标 | Double |

键值对排序算法，该算法的输入结构格式如表3-8所示。

表3-8 键值对排序算法输出数据结构表

|  |  |  |
| --- | --- | --- |
| 字段名 | 说明 | 类型 |
| Str | 作为key的字符串 | Double |
| Value | Value按序拼接 | Varchar |

Word Counts算法，该算法的输出结构格式如表3-9所示。

表3-9 Word Counts算法输出数据结构表

|  |  |  |
| --- | --- | --- |
| 字段名 | 说明 | 类型 |
| Str | 出现的单词 | Double |
| Counts | 单词的词频 | Varchar |

## 相关类设计与实现

本系统所使用的5种数据挖掘算法均由Java编写，在编写的过程中调用org.apache.hadoop包中的内容辅助完成。其中主要利用了hadoop中的与文件系统相关的类，以及Hadoop中与Map Reduce的job相关的类，和一些配置相关的类，例如conf.Configuration。同时在对算法实现并行化的过程中，让算法具备在Hadoop平台下Map Reduce并行化执行的能力，是通过编写自定义的Mapper类和Reducer类实现的，需要让自定义的Mapper类继承hadoop包下的Mapper类，重写Mapper类中的map方法，实现map的自定义，并且让自定义的Reducer类继承Hadoop包中的Reducer类，重写reduce方法，实现reduce的自定义，并且在算法的主类中，利用job对象，配合Java的反射机制，给job对象分配Mapper类和Reducer类，利用job进行算法的执行。

### 基于物品的协同过滤算法

为了实现该算法的功能，在类的设计上选择了用7个类来实现该算法的Map Reduce并行化版本。其中一个类为主类，剩下6个类中三个类为Mapper类，3个类为Reducer类。该算法的类关系如图3-7所示。

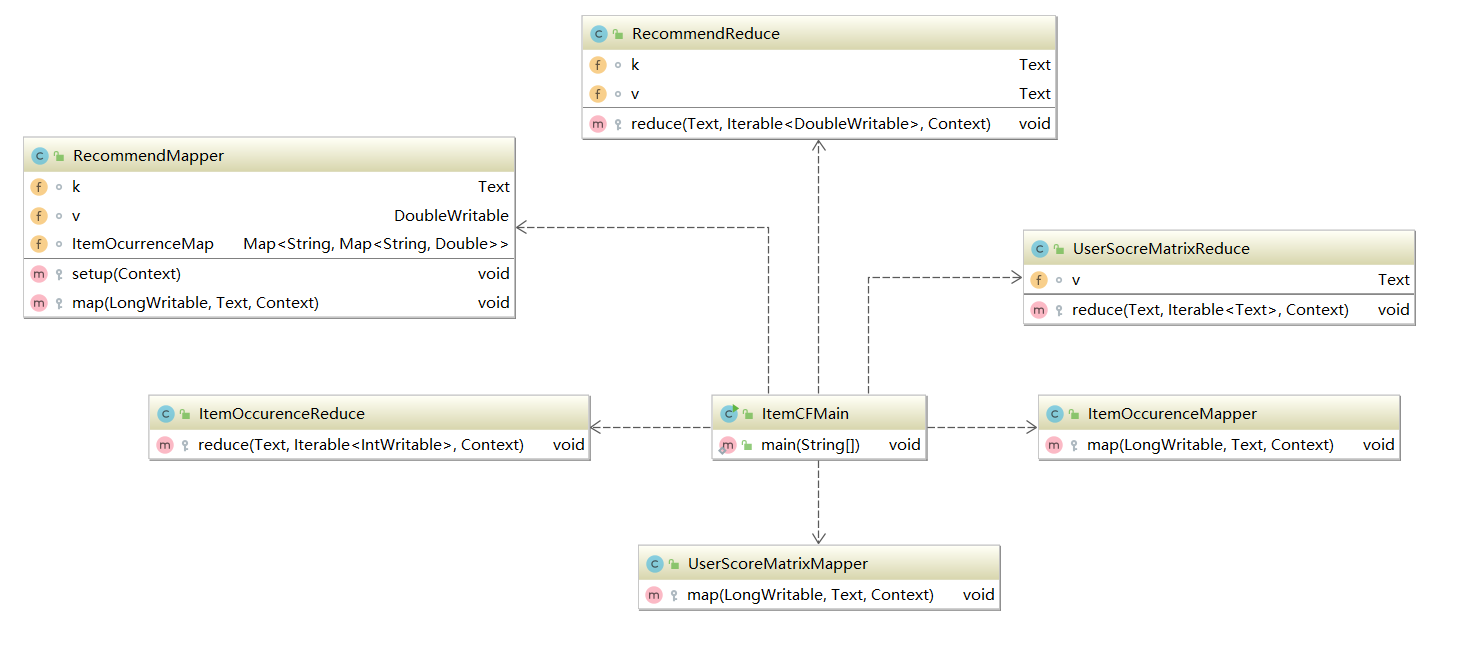


图3-7 基于物品的协同过滤算法类关系图

ItemCFMain类为该算法的主类，作为算法的入口，该类对主方法中运行时获得的参数进行了处理，根据参数设置了读取数据的路径以及结构输出的路径。因为算法在运算过程中需要两个临时文件，所以该算法的输入参数为4个路径，该算法在运行之初会对参数进行自检，如果参数不为4个，该算法的执行会终止。

UserScoreMatrixMapper类为第一次Map时所用到的类，该类的作用为计算用户评分矩阵，将原始数据进行第一次转换。

UserScoreMatrixReducer类的作用为接收UserScoreMatrixMapper传来的数据，并根据key将数据聚合作为用户评分矩阵输出。

ItemOccurenceMapper类为第二次Map时用到的类，作用为计算物品的同现矩阵，在map阶段，将Itemid解析出来，作为key的输出。

ItemOccurenceReducer类接收ItemOccurenceMapper发送的数据，将数据中的value累加，然后输出物品同现矩阵。

RecommendMapper类作为该算法最重要的类，该类从缓存中读取上述的用户评分矩阵以及物品同现矩阵，通过循环物品同现矩阵，一次计算用户对物品的喜好度，并筛选掉用户评分过的物品。

RecommendReducer类为该算法的最终输出类，从RecommendMapper中接收数据，规约出最终结果并输出。

### Page Rank算法

该算法在实现上仅用了3个类，以及一个枚举。主类PageRank，用于map的MyMapper类，用于Reduce的MyReducer类，以及枚举counter。用来记录收敛的个数。该算法的类关系如图3-8所示。

PageRank类作为算法的主类，该类实现了对数据的输入，结果的输出，以及将job对象利用Java的反射机制设定好Mapper和Reducer。并且将迭代次数设置为最大10次，防止程序死循环无法跳出。

Mymapper类，在该类中，通过读取到的网页id以及向外链接数计算出向外部链接的平均贡献值。将网页向外链接的id和“@”符号做组合，作为value存入，并且将网页id和PR值和“#”符号作为另一种组合存入value中。

MyReducer类，该类从MyMapper中接收到id和PR值等数据，通过对符号“@”和“#”进行比对，区分贡献链接值，和链接id，计算出最终PR值。

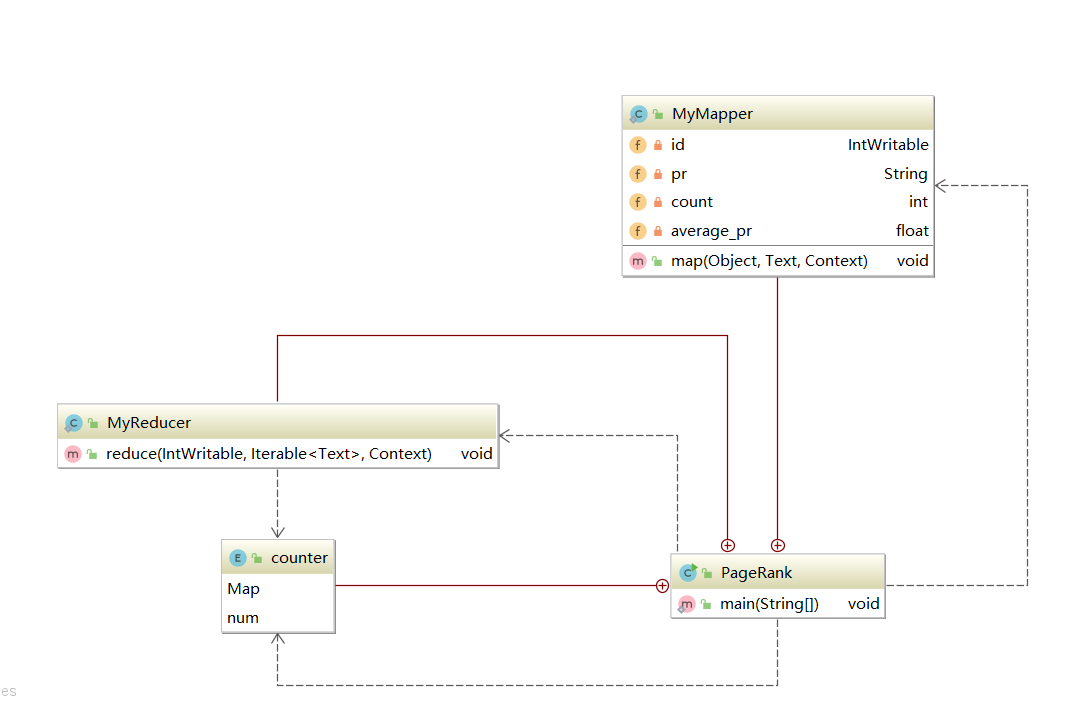


图3-8 Page Rank算法类关系图

### K-means聚类算法

该算法通过4个类进行实现。分别为主类Run，作为质心的center类用于map的TokenizerMapper类，用于Reduce的Intsumreducer类。该算法的类关系如图3-9所示。

Run类，作为该聚类算法的主类，不仅需要读取输入输出参数，还需要初始化质心类，并且向质心类中加载数据。为了防止程序进入死循环，将迭代次数设置为五次。

Center类，主要存放质心的个数，实现了2个从HDFS文件系统上读取质心的方法，这两个方法一个用来读取初始的质心，另一个用来读取每次迭代后生成的新质心。

TokenizerMapper类，该类主要包含3个方法，第一个方法为setup，该方法的作用为将存储在变量中的聚类中心转换为数组的形式，方便后续的数据处理，第二个方法为map方法，该方法通过循环求解对每一个点来说距离最近的质心。第三个distance方法的作用为求两个点之间的距离。

Intsumreducer类，通过在map阶段获取的数据，通过计算对应维度上的值的加和，然后求出各个维度的平均值，将平均值作为新的质心。并且要计算新质心是否与老质心相同。

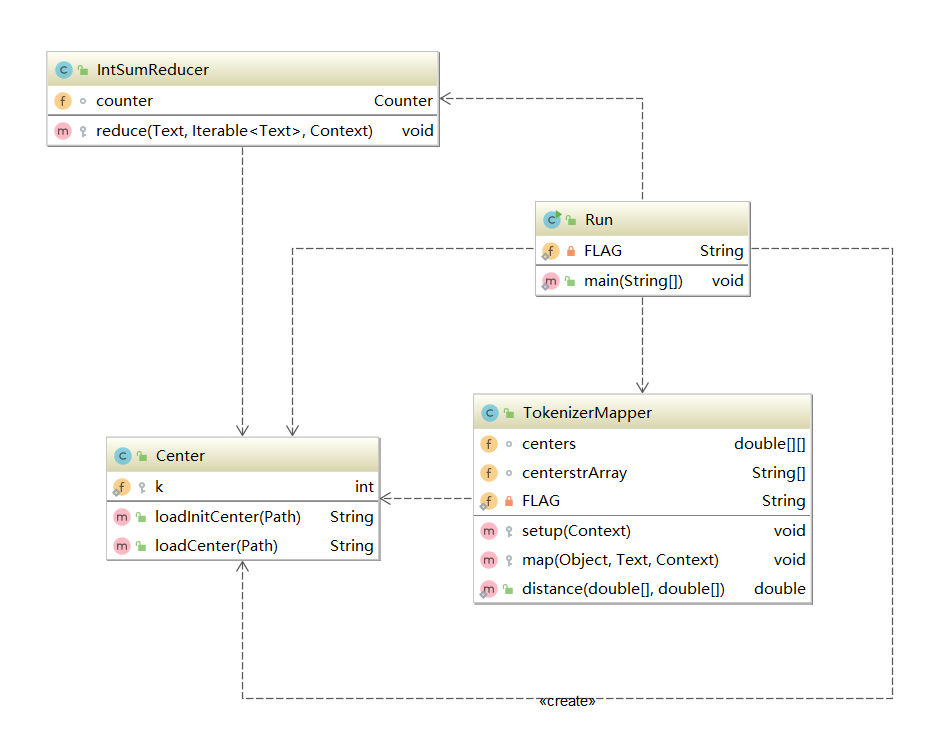


图3-9 K-means聚类算法类关系图

### 键值对排序算法

该算法通过7个类实现，主类SortMain，map类SortMapper，reduce类SortReducer，两个比较器类TextComparator，TextIntComparator，一个自定义的数据结果IntPaire，和一个对自定义数据结构进行操作的类PartitionByText。该算法的类关系如图3-10所示。

SortMain类为该算法的主类，在该类中设置好所用的map，reduce，以及comparator比较器和以及重写的partitioner类。

SortMapper类为该算法所用到的map类，在该类中调用自定义的数据结构Inpaire，并将字符串和字符串对应的value赋值给Intpaire对象。

SortReducer类作为该算法的reduce类，该类在规约的过程中需要对多个value值中的最后一个逗号进行处理。

TextComparator，TextIntComparator两个类为自定义的比较器，需要先对比较器进行注册，然后加上注解。

IntPaire为定义的数据结构，用于存储数据。包含两个变量，String类型的firstKey和int类型的SecondKey。该类需要重写readFields方法，Write方法，以及compareTo方法。

PartitionByText为重写的partitioner类。

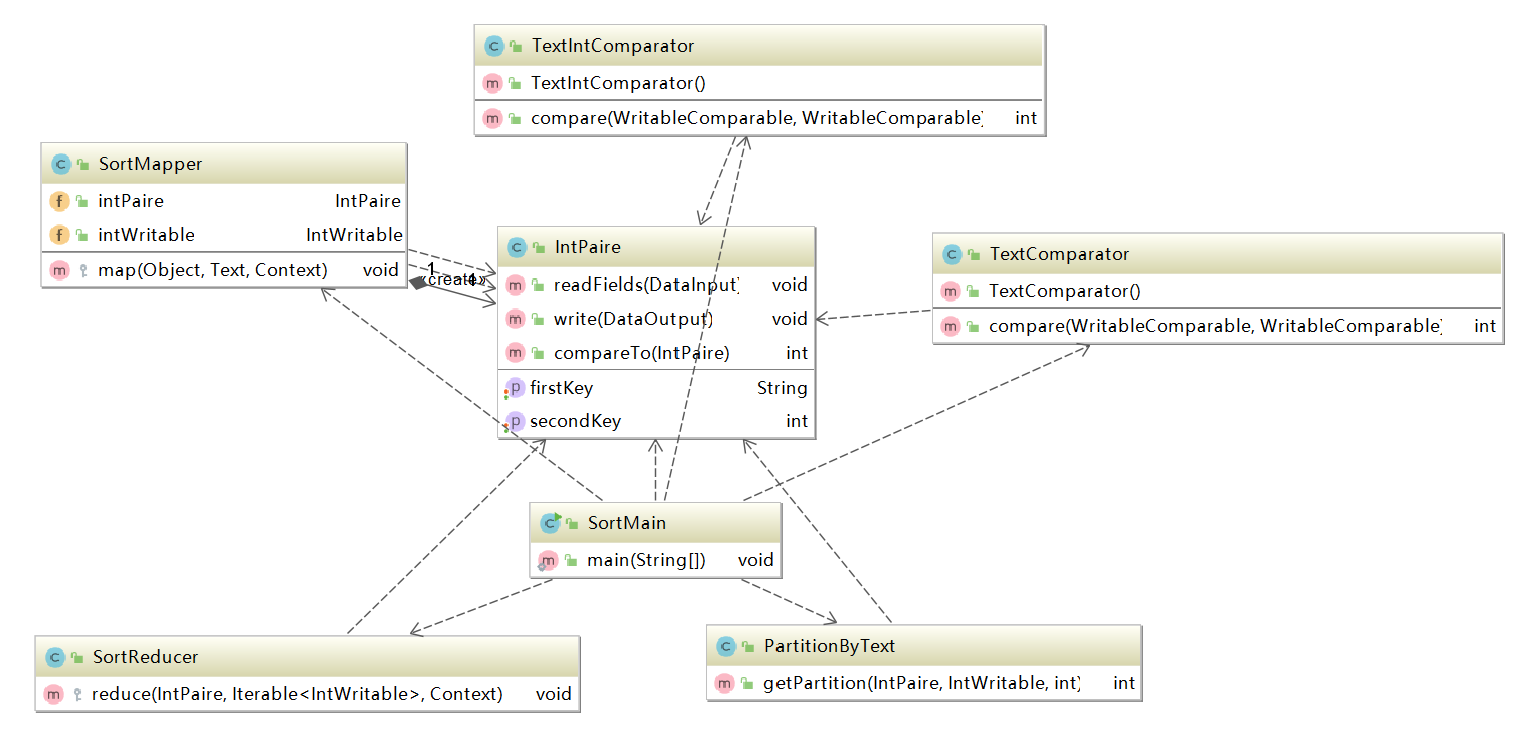


图3-10 键值对排序算法类关系图

### Word Counts算法

该算法通过3个类实现，作为主类的WordCounter，用于map的MapperClass，用于Reduce的ReducerClass。该算法的类关系如图3-11所示。

WordCounter类为该算法的主类，用于输入数据以及设置输入输出路径，同时利用反射将map和reduce对应的类分配给job对象。

MapperClass类，为该算法的map类，在map过程中，利用Java的字符串分割，将文章通过空格拆分为单词，将单词设为key，对应的value设为1，而后交由reduce处理。

ReducerClass类，接收从map过程发送来的数据，因为将单词设为key，所以仅需将key相同的value累加起来，便可得到最终结果。

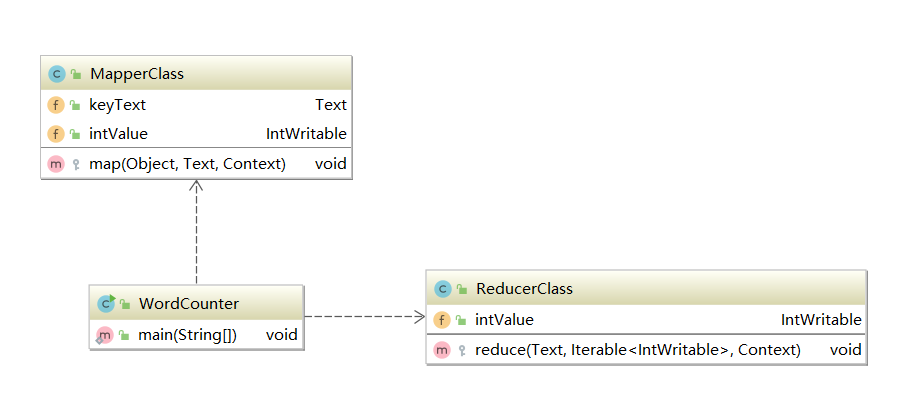


图3-11 Word Counts算法类关系图

## 本章小结

本章节介绍了数据挖掘系统所用到的5种算法的算法设计，数据结构设计，以及算法的类的设计与实现，详细介绍了算法的设计与具体实现。

# 系统的设计与实现

## 系统整体架构

数据挖掘系统整体的设计理念是基于HDFS文件系统而设计的，以Springboot框架为整体架构，编写的MapReduce并行化数据挖掘算法和前端技术等作为支撑，通过互联网为用户提供数据存储，数据挖掘等服务。图4-1为系统架构图。

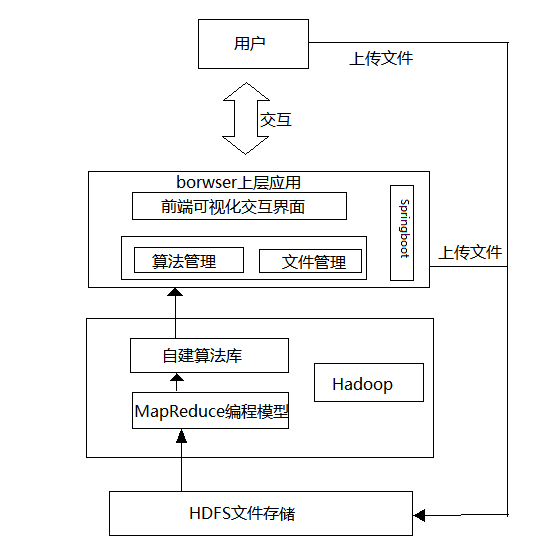


图4-1 系统架构图

该系统整体采用B/S（Browser/Server）结构，在数据存储方面采用直接使用Hadoop中HDFS文件系统的方案，将数据挖掘、数据分析、结果可视化等能力封装为接口，由平台服务器来对外提供。用户使用浏览器登录页面，将数据上传到平台服务器，由服务器接收到数据，再将数据发送到大数据平台所在的服务器，存储到HDFS系统对应的目录下。用户在进行操作请求时，由前端WEB服务器发送AJAX请求到后端的服务器平台，由后端服务器对接收的请求进行解析。如果是对HDFS文件系统的操作，就会调用利用HDFS自带的Java api编写好的对HDFS操作的工具类，进行相应操作。如果为执行算法指令，则会调用编写好的对linux操作的工具类，利用SSH2进行远程执行linux指令调用Hadoop平台对指定的数据并行执行相应并行化后的算法，执行之后的结果可以在浏览器供用户浏览。

本文的并行算法都是要交由Hadoop平台执行的，在核心系统层设计了Linux的远程登录、HDFS文件系统的远程操作、Hadoop平台的远程登录，对执行算法所需要的参数进行了封装，将算法提交到Hadoop平台进行并行执行。并且主要的业务逻辑也均有核心系统层实现。

在算法部分，本文所用的5种并行算法均为Java语言编写，通过实现Hadoop的MapReduce接口实现了算法的并行化。将算法封装为5个jar包，存放到Linux系统下的Hadoop平台的目录下，供Hadoop平台使用。本文通过搭建Hadoop平台，成功的将系统的业务逻辑与算法的实际运行环境分离，这样不仅易于管理，同时对于数据挖掘算法所需要的服务器集群的部署与管理提供了更加便利的条件。

## 系统的设计

### 系统的主要功能

本系统的核心系统层主要用于承接交互层的请求，并根据所接收到的请求执行相应操作。核心系统层承接了几乎所有的业务逻辑的处理，负责几乎所有的指令转发以及处理，包含的具体功能如下。

获取HDFS文件目录功能，在用户完成登录后，根据用户的用户名，获取HDFS文件系统中为该用户创建好的分支，利用HDFS的Java Api读取该用户分支下的文件夹，并判断该文件夹下的文件存储情况，将用户的在文件系统的存储情况，通过entity包中的DIr实现类，封装好，返回给交互层，交互层获取数据渲染网页呈现给用户。

创建文件夹功能，用户在登录后，可以在自己的分支下创建新的文件夹，交互层将用户创建的文件夹的数据封装好，发送给后端，由后端接收请求，调用在hdfstool包下的封装好的工具类，调用创建文件夹函数，通过HDFS api，完成文件夹的创建。

上传文件功能，用户可以向没有Input的文件夹上传文件，交互层将请求和数据封装好发送给和核心系统层，由核心系统层调用在hdfstool包下的封装好的工具类，先将文件保存到本地，然后调用上传文件函数，将文件上传到HDFS系统中的指定位置。

运行算法功能，用户可以对文件系统中，存在Input文件夹的分支执行算法，算法可以在交互层进行选择，交互层将选择的算法种类，以及运行算法的数据的路径等信息封装好，有交互层将请求发送到核心系统层，核心系统层接收到请求之后，根据接收到的算法选择参数，调用linuxtool包下的不同函数，执行对应的算法。

结果显示功能，用户可以对以及运行完算法的文件目录进行结果查看，用户需要在交互层选择算法的种类，因为不同算法的参数是完全不同的，交互层将信息封装好发送到核心系统层，核心系统层向HDFS系统中获取执行完的文件信息，将信息封装到entity包下的实现类里，将数据返回到交互层，由交互层经过渲染后展示给用户。

用户登录功能，用户登录利用账号密码登录自己的账户。由交互层将用户信息发送到核心系统层，核心系统层与Mysql数据库进行交互验证密码是否正确，然后决定用户是否跳转到主页面。

### 系统的前端设计

前端主要包含2部分，分别为静态资源，和html页面。其中静态资源主要包括前端所引用的框架和技术，比如layui的引用，jQuery的引用等。而html页面部分即为前端中所用到的页面的实现。这些html页面的与后端的数据交互均为发送Jquery下的AJAX请求，在数据的显示方面主要利用了Layui的table组件，将后端发送来的数据包，通过参数的对应渲染到table中。包含以下页面。图4-2为前端页面关系图

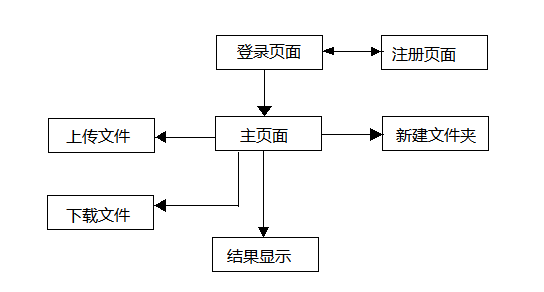


图4-2 前端页面关系图

login.html用户登录的页面，包含用户名和密码的输入，该html文件还利用JavaScript实现了与后端的交互，通过发送Ajax请求，将登录请求和作为参数的用户名和密码发送到后端服务器。

index.html此页面为主页面，包含了文件系统目录的显示，各种功能的按钮，算法的选择等等。利用了Layui框架实现的列表来显示HDFS文件系统的目录，配合选择框和按钮实现了各种功能。该文件通过实现了各种JavaScript函数来实现各种功能，该文件包含了4个JavaScript函数。runal函数就是运行算法的函数，该函数会获取选择的文件路径，以及选择的算法，然后将数据封装好发送给后端。Showans函数为跳转到展示运行结果页面的函数。以及showadddir函数为打卡新建文件夹的子页面。Showuploadfile函数为打卡上传文件子页面。

Showadddir.html页面为创建文件夹的子页面，由index页面作为子页面打开，通过Ajax请求实现了将创建文件夹的相关数据发送给后端服务器。

Showuoloadfile.html页面为上传文件的子页面，利用layui框架实现了快捷的文件上传。

Showitemcfans.html,Showpagerankans.html, showwordcountans.html等页面为展示各自算法的运行结果的页面。

### 系统的后端设计

本系统在后端的设计方面采用了较为规范的springboot常用结构。采用了常见的controller层，server层等结构，将本系统分为了7个包。后端所用到的各种依赖都通过配置Maven的pom文件实现了导入。具体结构如下。图4-3为后端结构关系图。

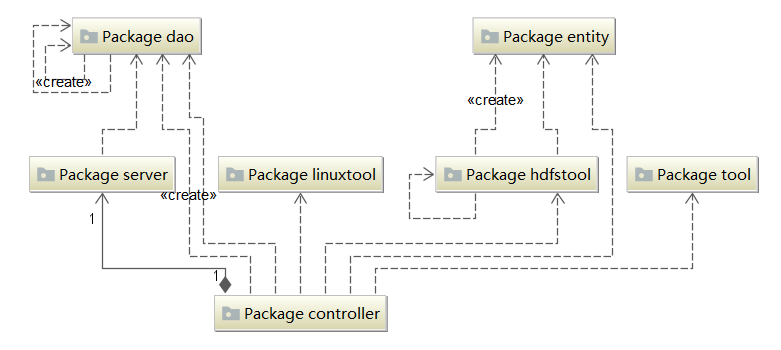


图4-3 后端结构关系图

controller层，该层中包含了springboot所用到的controller文件，controller是springboot中最基础的组件，作用为接收前端的请求，分析其URL，分配给不同的接收器，然后再接收器中进行处理，调用各自的函数或者server。其中所有的页面跳转也都由controller进行分配和实现。

Dao层，该层为对MySQL数据库的操作的封装，本系统中为实现了用户数据再MySQL中的相关操作，利用MySQL的jdbc驱动来实现对数据库的连接。

Entity层，该层为各种数据结构的实现类，包括获取的目录的数据结构Dir，以及展示各种算法的操作结果的数据结构。

Hdfstool层，该层为系统为了实现对HDFS操作所编写的工具类。

Linuxtool层，该层为系统为了实现对linux操作所编写的工具类。

Server层，该层为springboot所用的server，也可以理解为业务层或者服务层，在本系统中主要为封装dao层的操作。

Tool层，为实现的其他工具类的集合，比如Layui所需要的返回值的封装。

### 数据存储的设计

本系统的数据存储基于Hadoop平台的HDFS文件存储系统实现。因为在设计之初，HDFS系统便是为了分布式而设计的，所以和现有的分布式文件系统又有许多相似之处。但同时，区别也很明显，HDFS系统具有很高的容错性，很适合部署在价格低廉的机器上。同时还能指出高吞吐量的数据访问，很适合大规模的数据集去使用。图4-4为HDFS的写入流程。

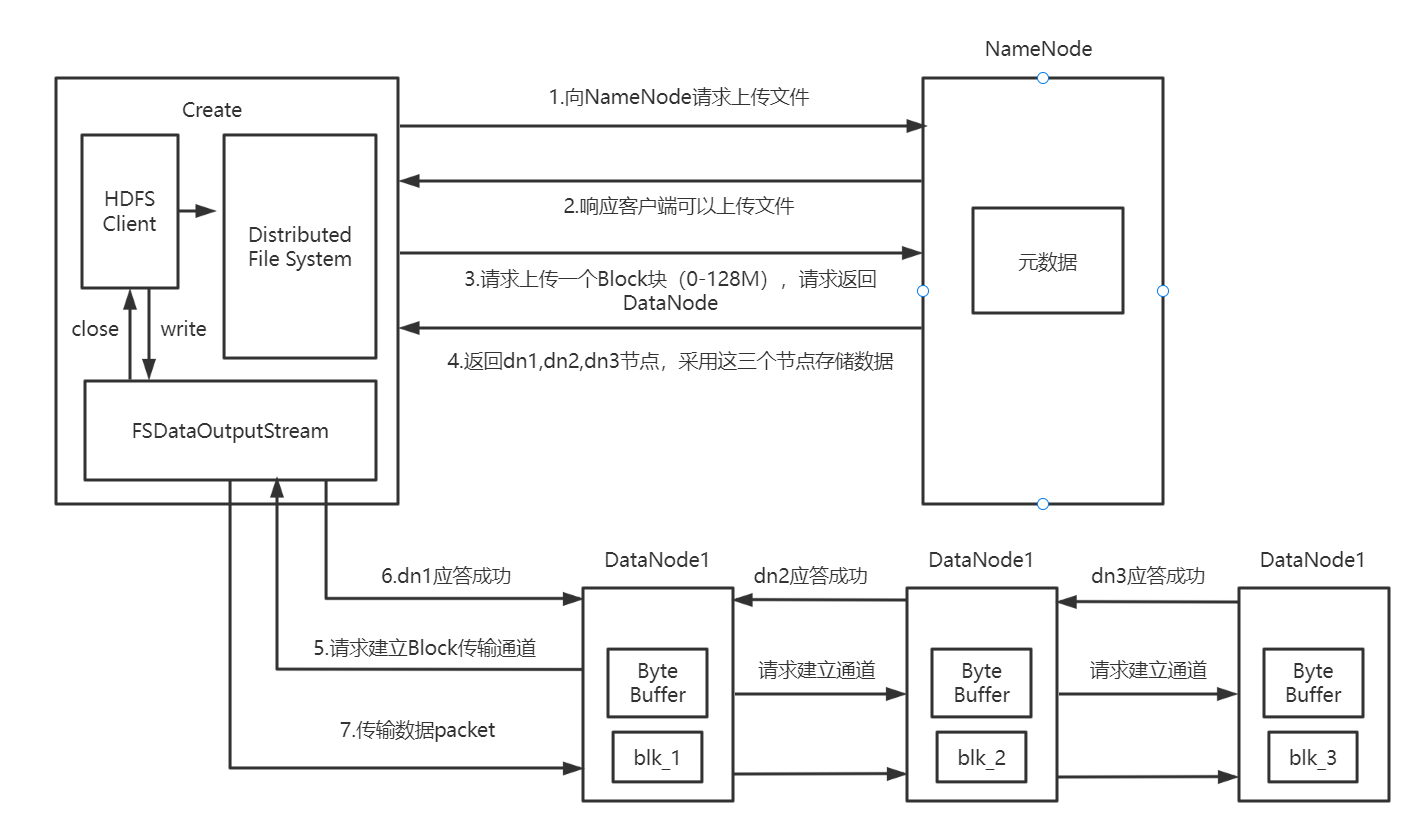


图4-4 HDFS写入流程图

本系统的数据存储依赖于HDFS，实现了一种伪数据结构。即为在HDFS的主分支下，为每个用户创建一个以用户名命名的分支，该用户的所有数据存储在当前分支下。在用户的分支下，每一个文件夹的构成由Input和Outpur文件夹组成，其中Input文件夹存放用户上传的数据，而Output用来存储用户执行完算法的输出结果。

对HDFS的操作主要通过Java api实现，利用org.apache.haoop.fs包下类进行操作。FileSystem提供了很多操作，本系统对HDFS的大部分操作都是基于FileSystem类实现的。利用FileSystem获取HDFS下的文件列表，需要指定好hadoop所在的url，以及对应的服务器用户等参数。包括创建文件夹的函数mkdirs，获取HDFS中文件的指令open，向HDFS上传文件copyFromLocalFile等。对HDFS的操作并指仅仅可以通过Java的api还可以通过shell指令，其中的表4-1为对HDFS文件系统操作的常用指令。

表4-1 HDFS常用shell命令

|  |  |
| --- | --- |
| 命令 | 注释 |
| start-dfs.sh  start-yarn.sh | 启动集群命令 |
| hadoop fs -help rm | 帮助 -help |
| hadoop fs -ls / | 显示目录信息 -ls  递归查看 -lsr |
| hadoop fs -mkdir -p /hdfs/fs | 在HDFS上创建目录 -mkdir -p 递归创建 |
| hadoop fs -moveFromLocal 源文件 目标位置 | 从本地剪切粘贴到HDFS -moveFromLocal |
| hadoop fs -appendToFile 源文件 目标文件 | 追加文件到已存在的文件末尾 -appendToFile |
| hadoop fs -cat 文件 | 显示文件内容 -cat |
| hadoop fs -copyFromLocal 本地文件 HDFS路径 | 从本地文件系统中拷贝到HDFS -copyFromLocal |
| hadoop fs -copyToLocal HDFS文件 本地路径 | 从HDFS拷贝到本地 -copyToLocal |
| hadoop fs -cp 原路径文件 拷贝路径文件 | 从HDFS的路径拷贝到另一个路径 -cp |
| hadoop fs -mv 文件 目标路径 | 在HDFS中移动文件 -mv |

## 系统功能的实现

### 登录注册注册功能

登录页面为本系统的入口页面，只有通过登录才能进入系统的主页面，同时因为数据存储的设计与用户名相关，本系统在登录后的运作需要依赖与正确的用户名。登录页面如图4-5所示。



图4-5 登录界面

登陆时输入的账号密码会通过前端页面发送的Ajax请求传输到后端服务器上，服务器通过查询数据库中的查找出用户的账号和密码，进行对比，如果验证为成功，则成功跳转到主页面。

注册功能通过登录页面的按钮点击进入，本系统的注册功能在使用方面与普通的注册功能相似，但是因为数据存储设计的特殊的原因，本系统的注册功能不仅要在数据库添加新用户的数据，同时还需要在远程的Linux系统中的HDFS系统中添加新的用户分支。图4-6为注册页面。



图4-6 注册界面

### HDFS文件管理

主页面主要显示了在HDFS中当前用户分支下的文件夹，以及文件夹中的文件存储状况。并且提供了新建文件夹，文件上传等功能。登录之后系统会直接向后端发送Ajax请求获取文件目录，并且在后端存储到编写好的数据结构，封装为适合前端Layui使用的数据结构发送回前端，由前端浏览器渲染页面，主页面如图4-7所示。

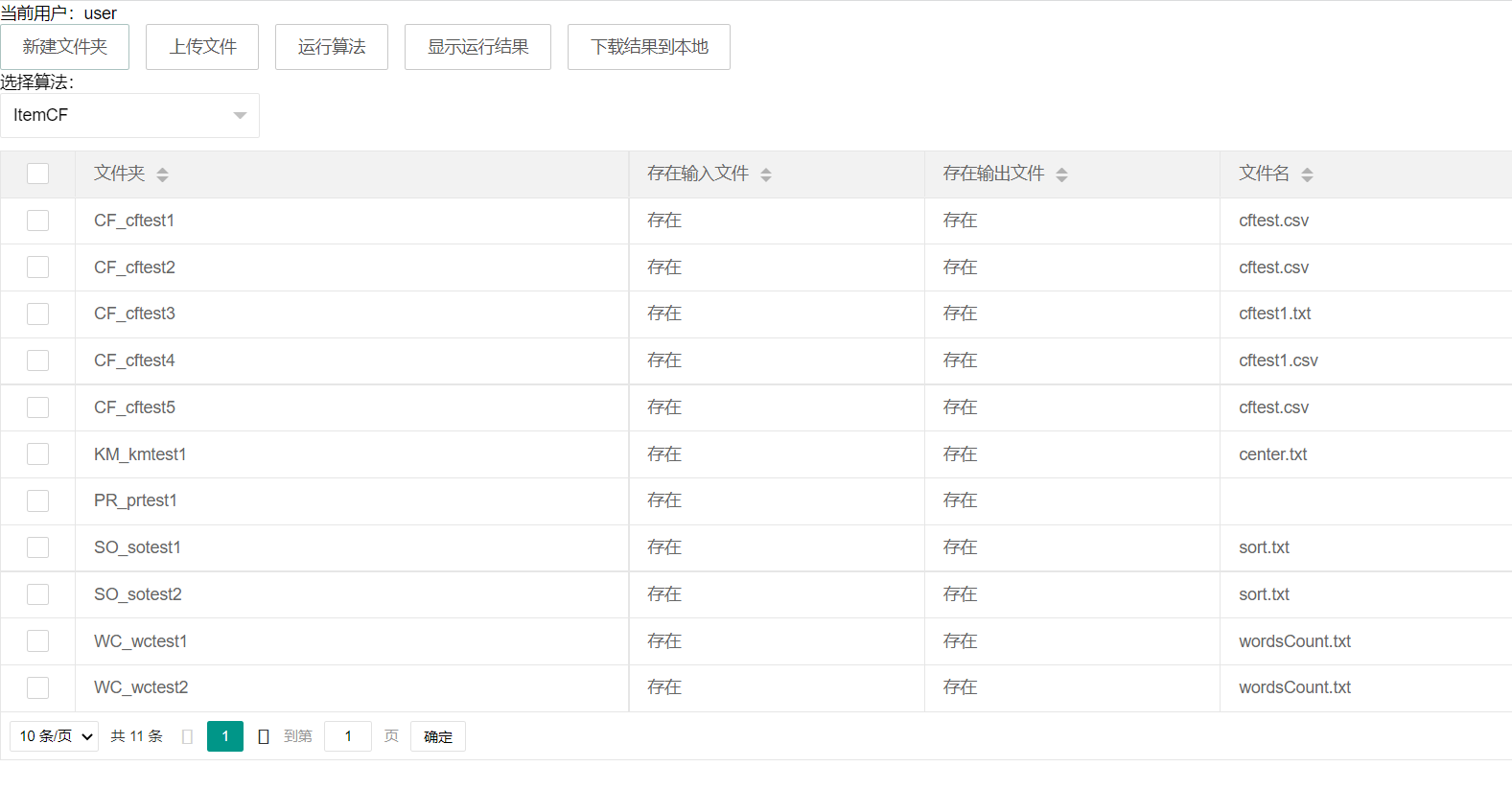


图4-7 系统主界面

新建文件夹功能，直接通过点击对应按钮触发功能，弹出创建文件夹的弹窗，在弹窗中输入文件夹的命名，提交之后前端会像后端服务器发送请求，后端服务器会调用hdfstool中编写好的远程创建文件夹的方法从而在HDFS中创建文件夹。新建文件夹子页面如图4-8所示。



图4-8 新建文件夹页面

上传文件功能，需要先选定上传文件的文件夹，弹出上传文件的弹窗，在弹窗中选择好要上传的文件，提交之后前端会将文件发送到后端服务器，由后端服务器将文件通过调用hdfstool中编写好的上传文件的方法，上传到所指定的路径中。上传文件的子页面如图4-9所示。



图4-9上传文件页面

下载文件功能，需要先选定下载文件的文件夹，弹出下载文件的弹出，在弹窗中输入下载文件的路径，提交之后前端会将下载路径发送到后端服务器中，由后端服务器通过调用hdfstool中编写好的下载文件的方法，将文件下载到指定路径中。下载文件的子页面如图4-10所示。

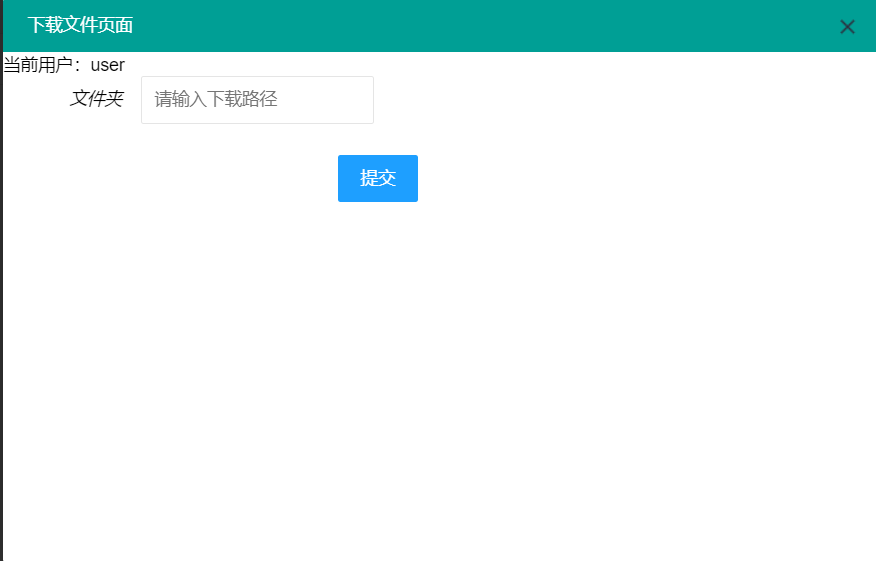


图4-10下载文件页面

### 运行算法功能

运行算法功能，用户在主页面选择文件夹，并且在筛选框中选择好算法通过点击按钮触发前端的JavaScript函数，将运行的算法种类和运行算法的文件路径封装为参数，发送到后端服务器，后端服务器通过接收到的参数，调用linuxtool中编写好的方法，利用ssh2技术远程执行linux指令，调用Hadoop执行相应的算法。Linux系统中便会对选择的文件夹中Input文件夹中的文件，执行选择好的算法，并将结果输出到对应文件夹加的Output文件夹中。也可以在Linux虚拟机中通过shell命名运行算法。运行算法时对算法的选择框图如图4-11所示。

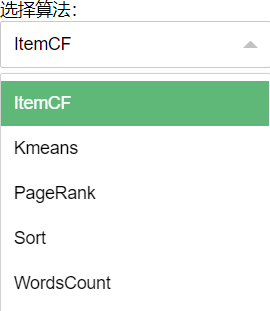


图4-11算法选择框

### 显示结果功能

显示结果功能，因为算法的输出结果格式并不相同，所以不同的算法的结果显示是在不同的html页面实现的。用户在主页面选择好有输出结果的文件夹，并且在筛选框中选择好算法，通过点击按钮触发js函数，像后端服务器发送请求，后端服务器根据接收到的参数，选择对应算法的结果显示前端页面，从HDFS文件系统中获取数据，将数据保存到编写好的数据结构中，再封装到Layui框架适用的类中，跳转到对应的页面，并且获取对应数据渲染到页面。图4-12page rank算法显示页面。

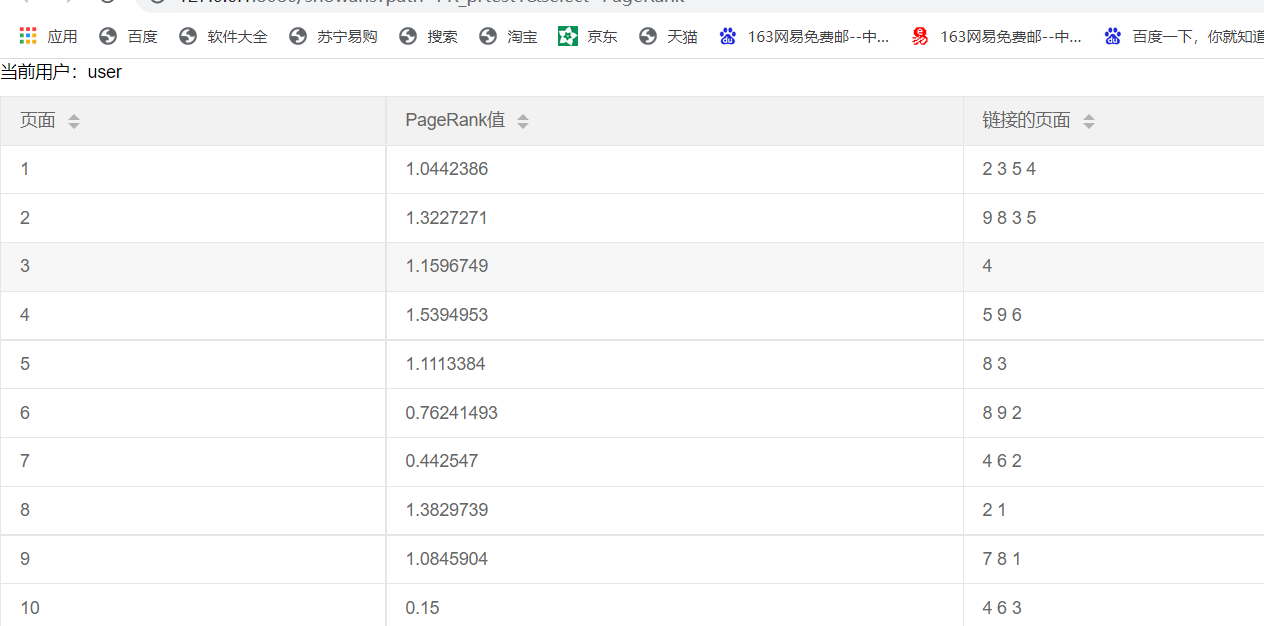


图4-12 page rank算法结果显示页面

图4-13为Itemcf算法的结果显示页面。



图4-13 Itemcf结果显示页面

图4-14为Word Counts算法的结果显示页面。

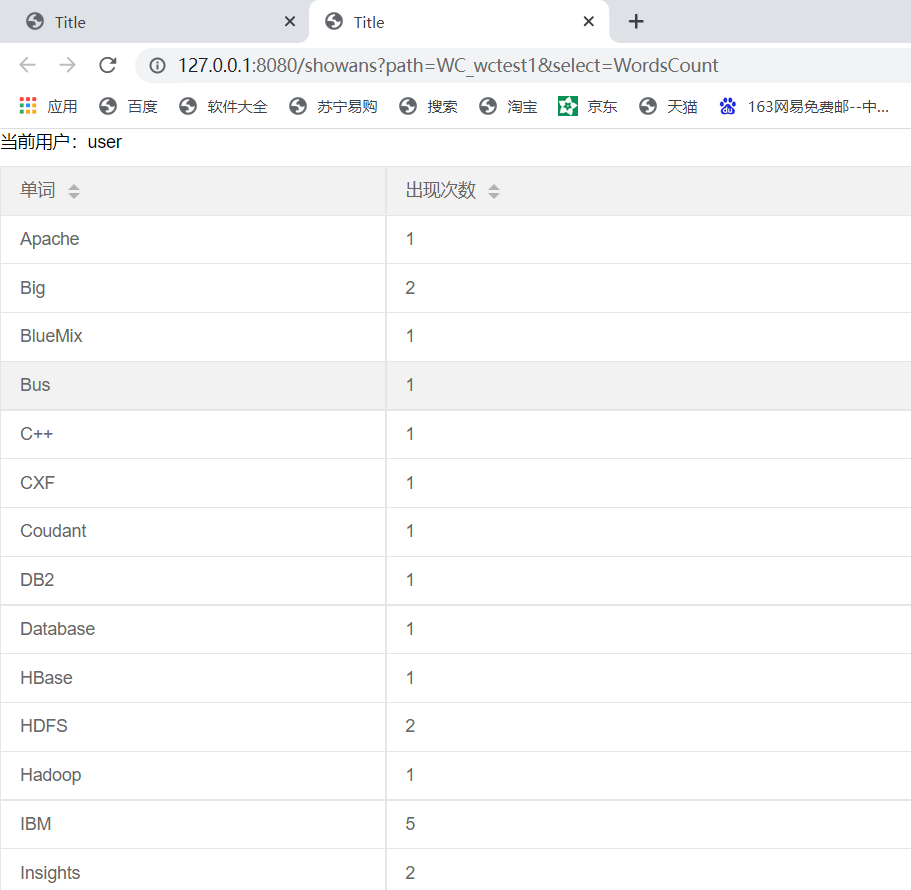


图4-14Word Counts结果显示页面

## 本章小结

本章节介绍了大数据平台下的数据挖掘算法系统的 整体架构，前端的设计，后端的设计，以及数据存储结构的设计。并且对功能的具体实现进行了介绍。

# 测试与分析

## 测试目的

测试的主要目的是对本系统运行的稳定性进行检验与测试，同时验证系统的功能是否与设计之初相符合，发掘系统潜在的漏洞。对算法的运行进行测试，验证算法运行的结果是否与预期相符合。对系统和算法的运行情况进行分析。

## 测试环境

本系统开发和测试都是再个人电脑上进行的。电脑系统为windows10操作系统，Hadoop平台所用到的linux系统为安装在VMware虚拟机中的centos7操作系统。Windows10下的springboot版本为2.6.6，MySQL版本为5.1.47，Hadoop版本为2.7.3。Centos7中安装的Hadoop平台版本为2.10.1。Java版本为jdk11。

## 测试过程

### Hadoop平台

对安装好Hadoop平台的Centos7系统进行测试，图5-1为对Hadoop平台的启动进行测试。在Hadoop平台启动后可以通过Linux系统的IP地址加上端口号50070在网页访问Hadoop平台，查看运行情况，图5-2为Hadoop网页端查看。

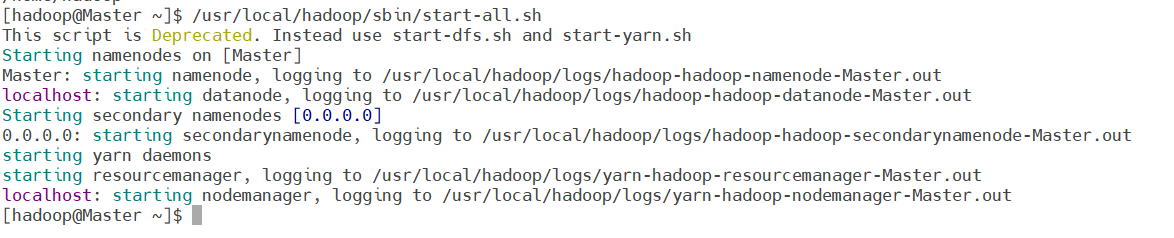


图5-1 Hadoop启动图

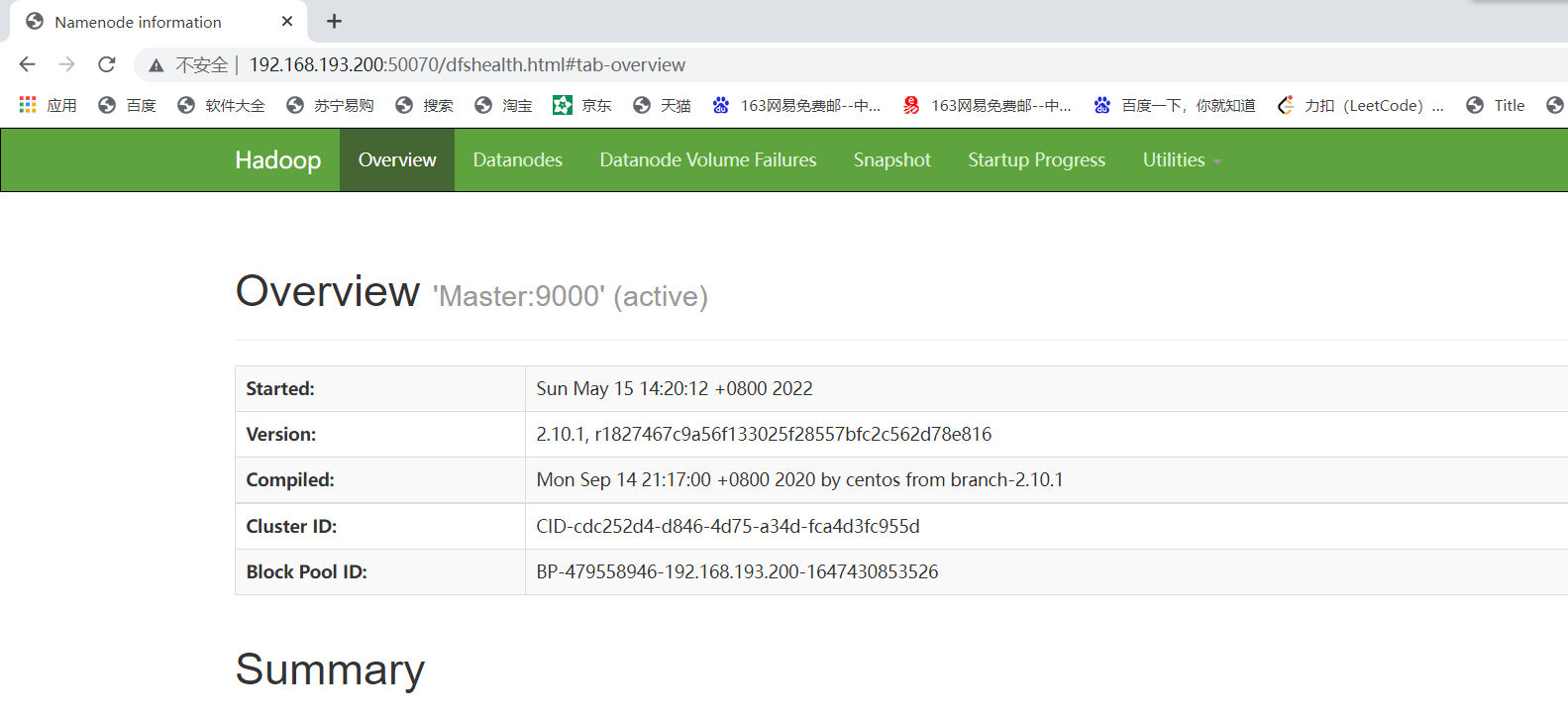


图5-2 Hadoop网页端查看

### 系统的测试

系统的测试过程较多，在此仅列举部分功能的测试。

登录注册功能的测试，登录模块为数据挖掘系统的入口，登录模块的逻辑本身很普通，在此仅验证登录是否可用。图5-3为登录模块图。



图5-3 登录模块

注册功能要更为特殊一些，因为不仅需要注册用户，还需要在远程的Hadoop平台的HDFS文件系统中创建新用户的文件分支。图5-4为注册模块图。



图5-4 注册模块

注册时会对输入的数据进行验证，如果数据库中已经有当前用户名，则会提示“用户名已存在”，两次输入的密码不一致也会有相应提示，如果成功注册，不仅会在MySQL中创建新用户，还会在HDFS中创建新分支。图5-5为注册成功提示。

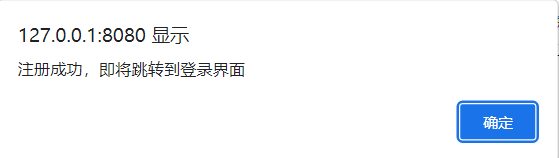


图5-5 注册成功提示

图5-6为HDFS系统中所创建的用户名为“violet”的新分支。

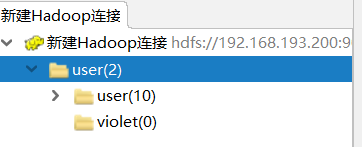


图5-6 HDFS中的新分支

上传文件功能的测试，图5-7为上传文件前的文件目录。



图5-7 上传文件前

图5-8为上传文件页面，在次页面选择上传文件test.txt以及上传路径uploadtest。



图5-8 上传文件页面

图5-9为上传文件后的文件目录。



图5-9 上传文件后

### 算法的测试

五种算法的输入输出数据格式都不同，根据每种算法设计合适的测试样例，进行测试。表5-1展示了测试用例和结果。

表5-1 算法测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 算法名称 | 测试数据 | 测试步骤 | 测试预期 | 测试结果 |
| 基于物品的协同过滤算法 | csv文件/txt文件 | 在网页选择ItemCF算法，选择相应文件夹点击运行算法 | 算法成功执行，在网页端也可以查看结果文件 | 结果正确 |
| Page Rank算法 | Txt文件 | 在网页选择Page Rank算法，选择相应文件夹点击运行算法 | 算法成功执行，在网页端也可以查看结果文件 | 结果正确 |
| K-means聚类算法 | Txt文件 | 在网页选择K-means聚类算法，选择相应文件夹点击运行算法 | 算法成功执行，在网页端也可以查看结果文件 | 结果正确 |
| 键值对排序算法 | Txt文件 | 在网页选择键值对排序算法，选择相应文件夹点击运行算法 | 算法成功执行，在网页端也可以查看结果文件 | 结果正确 |
| Word Counts算法 | Txt文件 | 在网页选择Word Counts算法，选择相应文件夹点击运行算法 | 算法成功执行，在网页端也可以查看结果文件 | 结果正确 |

对于算法的运行情况，在数据集不大的情况下，Itemcf算法的运行时间在一分钟内，图5-10为Item算法运行结果图。

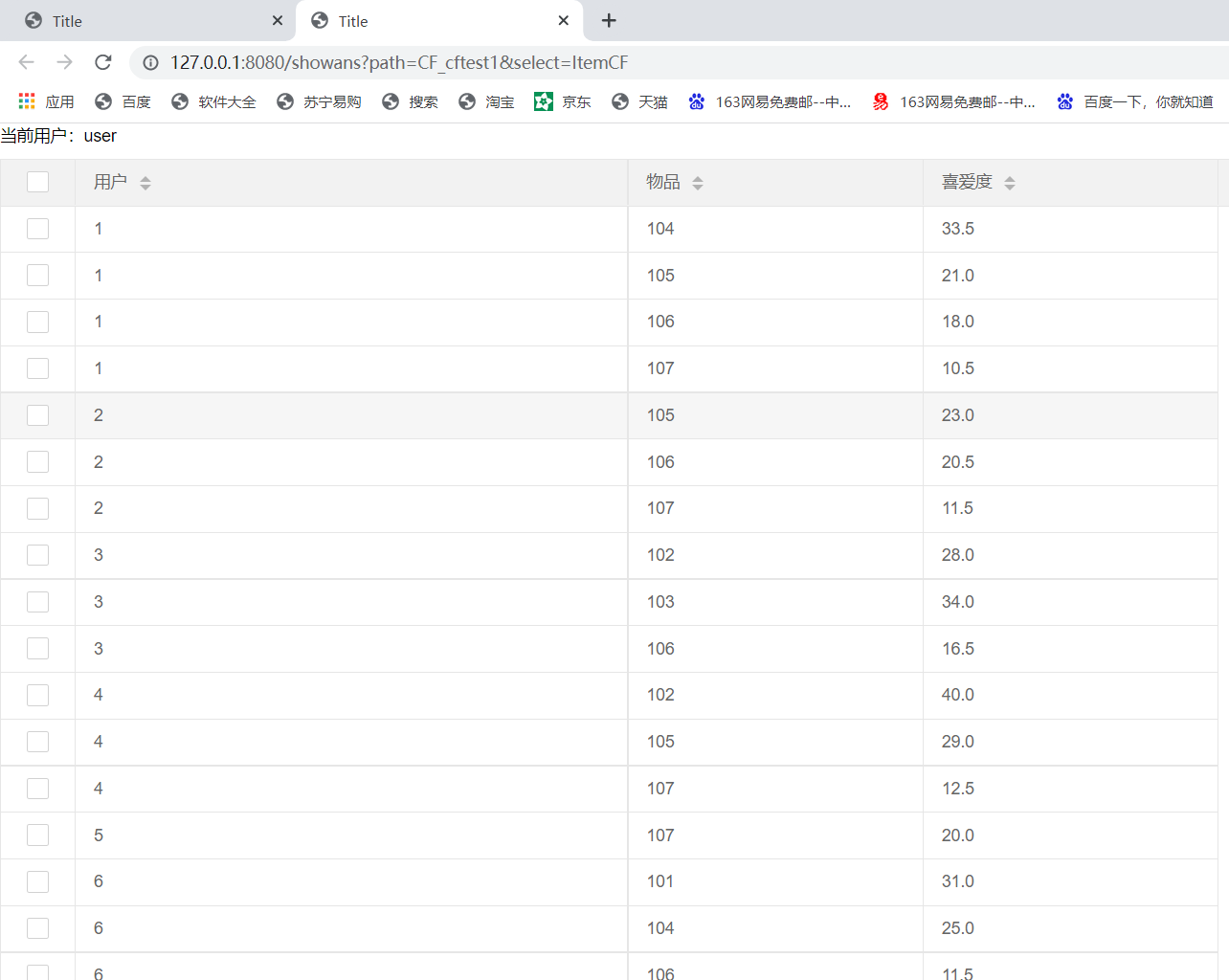


图5-10 ItemCF算法运行结果

但对于从groupslen网站下载的一份10m大小的百万份用户对电影的评分数据进行运算时，运行时间达到了5分钟。图5-11为对10m大小数据集运行的结果。

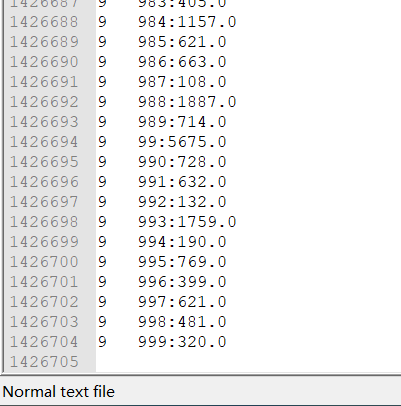


图5-11 ItemCF算法对大数据集运行结果

## 结果分析

本系统在运行方面目前没有发现漏洞，对于目前已有的数据集进行运算也都能得出正确的运行结果，系统整体的健壮性能够得以保证。

但由ItemCF算法运行结果可得出，目前本系统搭配的大数据平台的数据处理能力有限，虚拟机中的运行环境没有充分发挥出Hadoop平台的分布式特性所带来的优势，充分发挥分布式的优势需要在服务器集群中部署平台，通过多机器的协作提高算法的运行效率。

## 本章小结

本章对系统的功能进行了测试，简要介绍了测试环境，测试结果并且对测试的结果进行了分析，得出了目前系统的运行状况良好，但是在算法处理方面需要通过增加集群中的节点来提高运行效率。

结论

本文首先对题目背景进行了论述，对大数据平台技术和数据挖掘相关算法进行了研究，确定了本文对于程序的实现方向和方法。本文利用Hadoop大数据平台，将编写好的数据挖掘算法整合到Hadoop平台中，再通过springboot实现B/S架构的系统，将数据挖掘这一能力封装为服务，将底层调用封装好，用户可以在网页上通过简单的操作，就可以使用系统搭建好的数据处理的能力，对数据进行处理以及分析。

本系统在设计之初就考虑到了数据挖掘这项技术目前应用面很广，各行各业都对此项技术有着不小的需求，使用在该系统的设计时要考虑使用门槛，要重复体现对用户的友好性，并且考虑到应用面会很广泛，使用在设计时也考虑到了扩展性，因为本系统对数据挖掘的实现是通过Hadoop平台调用封装好的jar包，如果需要添加新的算法，仅需将算法封装为jar包，存放到Hadoop集群中对应的目录下即可。

虽然本系统预留出了扩展性，但是当前所实现的算法数量较少，并不能满足所有的需求，通过后续对本系统的持续完善，希望数据挖掘的能力可以对各行各业提供帮助。

致谢

时光匆匆，一转眼大学四年的生涯即将结束，在理工的这几年的经历会成为我人生中最独特的宝贵财富。在此，我想向我大学期间帮助和支持我的老师、同学、家人们表示衷心的感谢！

在我完成毕业设计的过程中，从定题目，开题，到最后完成毕业设计，我首先要感谢的便是我的指导老师李松老师，李老师在我毕业设计的设计与完成工作中给予了我很多帮助，为我指点迷津。还要感谢我的室友和同学们，在我们完成毕业设计的过程中，多少个日日夜夜是我们互相讨论问题，为互相的毕业设计提出宝贵的意见。

通过完成本次毕业设计，让我对大数据方面的知识有了全新的认识和理解。在专业技能方面我得到了很大的提高，对我的能力也是一次很好的锻炼。

在大学期间，我获得的最宝贵的财富就是这段大学时光，在这里我不仅获得的了专业知识，收获了宝贵的技能，还学到了宝贵的人生经验，结交了许多朋友。大学时代的结束，意味着新的开始，在未来我将砥砺前行，有了在大学的收获，我相信今后的什么困难都无法将我击败。

最后，感谢所有教导我的老师，帮助过我的同学们，希望你们在今后的日子里前程似锦！同时祝愿母校的明天更加辉煌！

参考文献

1. 周志阳. 大数据实时计算平台技术综述[J]. 中国新通信， 2017,19(04):47．
2. 王丽丽. 大数据背景下数据挖掘技术的应用[J]. 计算机与网络， 2021,47(20):45-47.
3. Can Altinigneli. The Data Mining Group at University of Vienna[J] Datenbank - SpektrumVolume 20, Issue 4. 2020. 71-79.
4. 卢媛媛. 大数据平台下的数据挖掘算法设计与实现[D].中国石油大学（北京）,2017,2.
5. 杜钢虎. 大数据时代背景下数据挖掘技术刍议 [J].电子技术与软件工程. 2015,(14):221.
6. Science. Dealing with data [J]. Science, 2011, 331(6018): 639-806.
7. Nature. Big data [J]. Nature, 2008, 455(7209): 1-136.
8. 朱紫钰,汤小春,赵全.面向CPU-GPU集群的分布式机器学习资源调度框架研究[J].西北工业大学学报,2021,39(03):529-538.
9. 何清,庄福振,曾立,赵卫中,谭庆.PDMiner:基于云计算的并行分布式数据挖掘工具平台[J].中国科学:信息科学,2014,44(07):871-885.
10. HOLMES A. Hadoop in Practice[M]. Shelter Island: Manning Publications Co. 2012: 4-5.
11. 杨小虎,程锦,朱苍璐.云环境下Hadoop分布式文件系统K-means算法并行处理改进[J].宜宾学院学报,2021,21(12):48-52.
12. 张平.并行计算模型MapReduce的工作原理探究[J].吉林广播电视大学学报,2021(06):154-157.
13. 朱蓉,郑建华.基于SpringBoot的垃圾分类科普及课程平台[J].电脑知识与技术,2022,18(09):22-24.
14. BUCHIN M，DODGE S，SPECKMANN B. Context aware similartity of trajectories［C］. International Conference on Geographic Informa⁃tion Science，2012：43-56.
15. 林婷薇, 莫路锋, 薛晨杰. 基于主题相关与用户历史的网页排序算法[J]. 计算机工程与设计, 2019, 40(8): 2265-2269.
16. Anil K J. Data clustering: 50 years beyond Kmeans [J] .Pattern Recognition Letters, 2010,31(8):651-666.

附录A

**英文原文**

**Big Data Mining Platform**

In typical data mining systems, the mining procedures require computational intensive computing units for data analysis and comparisons. A computing platform is, therefore, needed to have efficient access to, at least, two types of resources: data and computing processors. For small scale data mining tasks, a single desktop computer, which contains hard disk and CPU processors, is sufficient to fulfill the data mining goals. Indeed, many data mining algorithm are designed for this type of problem settings. For medium scale data mining tasks, data are typically large (and possibly distributed) and cannot be fit into the main memory. Common solutions are to rely on parallel computing or collective mining to sample and aggregate data from different sources and then use parallel computing programming (such as the Message Passing Interface) to carry out the mining process. For Big Data mining, because data scale is far beyond the capacity that a single personal computer (PC) can handle, a typical Big Data processing framework will rely on cluster computers with a high-performance computing platform, with a data mining task being deployed by running some parallel programming tools, such as MapReduce or Enterprise Control Language (ECL), on a large number of computing nodes (i.e., clusters). The role of the software component is to make sure that a single data mining task, such as finding the best match of a query from a database with billions of records, is split into many small tasks each of which is running on one or multiple computing nodes. For example, as of this writing, the world most powerful super computer Titan, which is deployed at Oak Ridge National Laboratory in Tennessee, contains 18,688 nodes each with a 16-core CPU.

Such a Big Data system, which blends both hardware and software components, is hardly available without key industrial stockholders’ support. In fact, for decades, companies have been making business decisions based on transactional data stored in relational databases. Big Data mining offers opportunities to go beyond traditional relational databases to rely on less structured data: weblogs, social media, e-mail, sensors, and photographs that can be mined for useful information. Major business intelligence companies, such IBM, Oracle, Teradata, and so on, have all featured their own products to help customers acquire and organize these diverse data sources and coordinate with customers’ existing data to find new insights and capitalize on hidden relationships.

Big Data Mining Algorithms

Local Learning and Model Fusion for Multiple Information Sources：

As Big Data applications are featured with autonomous sources and decentralized controls, aggregating distributed data sources to a centralized site for mining is systematically prohibitive due to the potential transmission cost and privacy concerns. On the other hand, although we can always carry out mining activities at each distributed site, the biased view of the data collected at each site often leads to biased decisions or models, just like the elephant and blind men case. Under such a circumstance, a Big Data mining system has to enable an information exchange and fusion mechanism to ensure that all distributed sites (or information sources) can work together to achieve a global optimization goal. Model mining and correlations are the key steps to ensure that models or patterns discovered from multiple information sources can be consolidated to meet the global mining objective. More specifically, the global mining can be featured with a two-step (local mining and global correlation) process, at data, model, and at knowledge levels. At the data level, each local site can calculate the data statistics based on the local data sources and exchange the statistics between sites to achieve a global data distribution view. At the model or pattern level, each site can carry out local mining activities, with respect to the localized data, to discover local patterns. By exchanging patterns between multiple sources, new global patterns can be synthetized by aggregating patterns across all sites. At the knowledge level, model correlation analysis investigates the relevance between models generated from different data sources to determine how relevant the data sources are correlated with each other, and how to form accurate decisions based on models built from autonomous sources.

Mining from Sparse, Uncertain, and Incomplete：

Data Spare, uncertain, and incomplete data are defining features for Big Data applications. Being sparse, the number of data points is too few for drawing reliable conclusions. This is normally a complication of the data dimensionality issues, where data in a high-dimensional space (such as more than 1,000 dimensions) do not show clear trends or distributions. For most machine learning and data mining algorithms, high-dimensional spare data significantly deteriorate the reliability of the models derived from the data. Common approaches are to employ dimension reduction or feature selection to reduce the data dimensions or to carefully include additional samples to alleviate the data scarcity, such as generic unsupervised learning methods in data mining.

Uncertain data are a special type of data reality where each data field is no longer deterministic but is subject to some random/error distributions. This is mainly linked to domain specific applications with inaccurate data readings and collections. For example, data produced from GPS equipment are inherently uncertain, mainly because the technology barrier of the device limits the precision of the data to certain levels (such as 1 meter). As a result, each recording location is represented by a mean value plus a variance to indicate expected errors. For data privacyrelated applications , users may intentionally inject randomness/errors into the data to remain anonymous.

This is similar to the situation that an individual may not feel comfortable to let you know his/her exact income, but will be fine to provide a rough range like [120k, 160k]. For uncertain data, the major challenge is that each data item is represented as sample distributions but not as a single value, so most existing data mining algorithms cannot be directly applied. Common solutions are to take the data distributions into consideration to estimate model parameters. For example, error aware data mining utilizes the mean and the variance values with respect to each single data item to build a Naı¨ve Bayes model for classification. Similar approaches have also been applied for decision trees or database queries. Incomplete data refer to the missing of data field values for some samples. The missing values can be caused by different realities, such as the malfunction of a sensor node, or some systematic policies to intentionally skip some values (e.g., dropping some sensor node readings to save power for transmission).

While most modern data mining algorithms have in-built solutions to handle missing values (such as ignoring data fields with missing values), data imputation is an established research field that seeks to impute missing values to produce improved models (compared to the ones built from the original data). Many imputation methods exist for this purpose, and the major approaches are to fill most frequently observed values or to build learning models topredict possible values for each data field, based on the observed values of a given in instance.

Mining Complex and Dynamic Data：

The rise of Big Data is driven by the rapid increasing of complex data and their changes in volumes and in nature. Documents posted on WWW servers, Internet backbones, social networks, communication networks, and transportation networks, and so on are all featured with complex data. While complex dependency structures underneath the data raise the difficulty for our learning systems, they also offer exciting opportunities that simple data representations are incapable of achieving. For example, researchers have successfully used Twitter, a well-known social networking site, to detect events such as earthquakes and major social activities, with nearly realtime speed and very high accuracy. In addition, by summarizing the queries users submitted to the search engines, which are all over the world, it is now possible to build an early warning system for detecting fast spreading flu outbreaks. Making use of complex data is a major challenge for Big Data applications, because any two parties in a complex network are potentially interested to each other with a social connection. Such a connection is quadratic with respect to the number of nodes in the network, so a million node network may be subject to one trillion connections. For a large social network site, like Facebook, the number of active users has already reached 1 billion, and analyzing such an enormous network is a big challenge for Big Data mining. If we take daily user actions/interactions into consideration, the scale of difficulty will be even more astonishing.

Inspired by the above challenges, many data mining methods have been developed to find interesting knowledge from Big Data with complex relationships and dynamically changing volumes. For example, finding communities and tracing their dynamically evolving relationships are essential for understanding and managing complex systems. Discovering outliers in a social network is the first step to identify spammers and provide safe networking environments to our society. If only facing with huge amounts of structured data, users can solve the problem simply by purchasing more storage or improving storage efficiency. However, Big Data complexity is represented in many aspects, including complex heterogeneous data types, complex intrinsic semantic associations in data, and complex relationship networks among data. That is to say, the value of Big Data is in its complexity. Complex heterogeneous data types. In Big Data, data types include structured data, unstructured data, and semistructured data, and so on. Specifically, there are tabular data (relational databases), text, hyper-text, image, audio and video data, and so on. The existing data models include key-value stores, bigtable clones, document databases, and graph databases, which are listed in an ascending order of the complexity of these data models. Traditional data models are incapable of handling complex data in the context of Big Data. Currently, there is no acknowledged effective and efficient data model to handle Big Data. Complex intrinsic semantic associations in data. News on the web, comments on Twitter, pictures on Flicker, and clips of video on YouTube may discuss about an academic awardwinning event at the same time. There is no doubt that there are strong semantic associations in these data. Mining complex semantic associations from “text-image-video” data will significantly help improve application system performance such as search engines or recommendation systems. However, in the context of Big Data, it is a great challenge to efficiently describe semantic features and to build semantic association models to bridge the semantic gap of various heterogeneous data sources. Complex relationship networks in data. In the context of Big Data, there exist relationships between individuals. On the Internet, individuals are webpages and the pages linking to each other via hyperlinks form a complex network. There also exist social relationships between individuals forming complex social networks, such as big relationship data from Facebook, Twitter, LinkedIn, and other social media, including call detail records (CDR), devices and sensors information GPS and geocoded map data, massive image files transferred by the Manage File Transfer protocol, web text and click-stream data, scientific information, e-mail, and so on. To deal with complex relationship networks, emerging research efforts have begun to address the issues of structure-and-evolution, crowds-and-interaction, and information-and-communication. The emergence of Big Data has also spawned new computer architectures for real-time data-intensive processing, such as the open source Apache Hadoop project that runs on high-performance clusters. The size or complexity of the Big Data, including transaction and interaction data sets, exceeds a regular technical capability in capturing, managing, and processing these data within reasonable cost and time limits. In the context of Big Data, real-time processing for complex data is a very challenging task.

附录B

**中文译文**

**大数据挖掘平台**

在典型的数据挖掘系统中，挖掘过程需要计算密集型计算单元来进行数据分析和比较。因此，计算平台需要能够有效访问至少两种类型的资源：数据和计算处理器。对于小规模的数据挖掘任务，一台包含硬盘和 CPU 处理器的台式计算机就足以完成数据挖掘的目标。事实上，许多数据挖掘算法都是为这类问题设置而设计的。对于中等规模的数据挖掘任务，数据通常很大（并且可能是分布式的）并且无法放入主内存中。常见的解决方案是依靠并行计算或集体挖掘，对不同来源的数据进行采样和聚合，然后使用并行计算编程（如消息传递接口）来进行挖掘过程。对于大数据挖掘，由于数据规模远远超出单台个人电脑（PC）所能处理的能力，典型的大数据处理框架将依赖于具有高性能计算平台的集群计算机，部署数据挖掘任务通过在大量计算节点（即集群）上运行一些并行编程工具，例如 MapReduce 或企业控制语言 (ECL)。软件组件的作用是确保将单个数据挖掘任务（例如从具有数十亿条记录的数据库中查找查询的最佳匹配）拆分为许多小任务，每个小任务都在一个或多个计算上运行节点。例如，在撰写本文时，部署在田纳西州橡树岭国家实验室的世界上最强大的超级计算机 Titan 包含 18,688 个节点，每个节点具有 16 核 CPU。

没有关键工业股东的支持，这样一个融合了硬件和软件组件的大数据系统几乎是不可能实现的。事实上，几十年来，公司一直在根据存储在关系数据库中的事务数据做出业务决策。大数据挖掘提供了超越传统关系数据库以依赖较少结构化数据的机会：可以挖掘有用信息的博客、社交媒体、电子邮件、传感器和照片。 IBM、Oracle、Teradata 等主要商业智能公司都推出了自己的产品来帮助客户获取和组织这些多样化的数据源，并与客户现有的数据进行协调，以发现新的洞察力并利用隐藏的关系。

大数据挖掘算法

局部学习与模型融合信息来源：

由于大数据应用程序具有自主源和分散控制的特点，因此由于潜在的传输成本和隐私问题，将分布式数据源聚合到集中站点进行挖掘在系统上是禁止的。另一方面，虽然我们总是可以在每个分布式站点进行挖掘活动，但对每个站点收集的数据的偏见视图往往会导致有偏见的决策或模型。在这种情况下，大数据挖掘系统必须启用信息交换和融合机制，以确保所有分布式站点（或信息源）能够协同工作，以实现全局优化目标。模型挖掘和关联是确保从多个信息源发现的模型或模式可以合并以满足全球挖掘目标的关键步骤。更具体地说，全局挖掘可以在数据、模型和知识级别上以两步（局部挖掘和全局关联）过程为特征。在数据层面，每个本地站点可以根据本地数据源计算数据统计，并在站点之间交换统计数据，实现全局数据分布视图。在模型或模式级别，每个站点都可以针对本地化数据进行本地挖掘活动，以发现本地模式。通过在多个来源之间交换模式，可以通过聚合所有站点的模式来合成新的全局模式。在知识层面，模型相关性分析研究从不同数据源生成的模型之间的相关性，以确定数据源之间的相关性，以及如何根据自主源构建的模型形成准确的决策。

从稀疏、不确定和不完整中挖掘：

数据 备用、不确定和不完整的数据是大数据应用程序的定义特征。由于稀疏，数据点的数量太少，无法得出可靠的结论。这通常是数据维度问题的复杂化，其中高维空间（例如超过 1,000 个维度）中的数据没有显示出明显的趋势或分布。对于大多数机器学习和数据挖掘算法，高维备用数据会显着降低从数据中导出的模型的可靠性。常见的方法是采用降维或特征选择[48]来减少数据维度或仔细包含额外的样本以减轻数据稀缺性，例如数据挖掘中的通用无监督学习方法。

不确定数据是一种特殊类型的数据现实，其中每个数据字段不再是确定性的，而是受到一些随机/错误分布的影响。这主要与数据读取和收集不准确的领域特定应用程序有关。例如，GPS设备产生的数据本质上是不确定的，主要是设备的技术壁垒将数据的精度限制在一定的水平（如1米）。因此，每个记录位置都由一个平均值加上一个方差来表示，以指示预期的错误。对于数据隐私相关的应用程序，用户可能会故意将随机性/错误注入数据以保持匿名。

这类似于个人可能不愿意让您知道他/她的确切收入的情况，但可以提供一个粗略的范围，例如 [120k, 160k]。对于不确定的数据，主要挑战是每个数据项都表示为样本分布而不是单个值，因此大多数现有的数据挖掘算法不能直接应用。常见的解决方案是考虑数据分布来估计模型参数。例如，错误感知数据挖掘利用每个单个数据项的均值和方差值来构建用于分类的朴素贝叶斯模型。类似的方法也已应用于决策树或数据库查询。数据不完整是指部分样本数据字段值缺失。缺失值可能是由不同的现实情况引起的，例如传感器节点的故障，或一些有意跳过某些值的系统策略（例如，丢弃一些传感器节点读数以节省传输功率）。

虽然大多数现代数据挖掘算法都有内置的解决方案来处理缺失值（例如忽略具有缺失值的数据字段），但数据插补是一个成熟的研究领域，旨在对缺失值进行插补以生成改进的模型（与从 原始数据）。 为此目的存在许多插补方法，主要方法是填充最常观察到的值或构建学习模型以根据给定实例的观察值预测每个数据字段的可能值。

挖掘复杂动态数据：

大数据的兴起是由复杂数据的快速增长及其数量和性质的变化推动的。 WWW服务器、Internet骨干网、社交网络、通信网络、交通网络等发布的文档都具有复杂的数据特征。虽然数据下的复杂依赖结构增加了我们学习系统的难度，但它们也提供了简单数据表示无法实现的令人兴奋的机会。例如，研究人员已成功使用知名社交网站 Twitter 来检测地震和重大社交活动等事件，速度近乎实时，准确度非常高。此外，通过汇总用户提交给遍布世界各地的搜索引擎的查询，现在可以建立一个早期预警系统来检测快速传播的流感爆发。利用复杂数据是大数据应用程序的主要挑战，因为复杂网络中的任何两方都可能通过社交联系彼此感兴趣。这样的连接相对于网络中节点的数量是二次的，因此一百万个节点的网络可能会受到一万亿个连接的影响。对于像 Facebook 这样的大型社交网站，其活跃用户数量已经达到 10 亿，分析如此庞大的网络对于大数据挖掘来说是一个很大的挑战。如果我们将日常用户操作/交互考虑在内，难度将更加惊人。

受上述挑战的启发，人们开发了许多数据挖掘方法来从具有复杂关系和动态变化量的大数据中找到有趣的知识。例如，寻找社区并追踪其动态演变的关系对于理解和管理复杂系统至关重要。发现社交网络中的异常值是识别垃圾邮件发送者并为我们的社会提供安全网络环境的第一步。如果只是面对海量的结构化数据，用户只需购买更多存储或提高存储效率即可解决问题。然而，大数据的复杂性表现在很多方面，包括复杂的异构数据类型、复杂的数据内在语义关联以及复杂的数据之间的关系网络。也就是说，大数据的价值在于它的复杂性。复杂的异构数据类型。在大数据中，数据类型包括结构化数据、非结构化数据和半结构化数据等。具体有表格数据（关系型数据库）、文本、超文本、图像、音视频数据等。现有的数据模型包括键值存储、大表克隆、文档数据库和图数据库，这些数据模型按照这些数据模型的复杂度升序排列。传统数据模型无法在大数据环境中处理复杂数据。目前，没有公认的有效和高效的数据模型来处理大数据。数据中复杂的内在语义关联。网络上的新闻、Twitter 上的评论、Flicker 上的图片和 YouTube 上的视频片段可能会同时讨论学术获奖事件。毫无疑问，这些数据中有很强的语义关联。从“文本-图像-视频”数据中挖掘复杂的语义关联将显着帮助提高应用系统的性能，例如搜索引擎或推荐系统。然而，在大数据的背景下，如何有效地描述语义特征并构建语义关联模型以弥合各种异构数据源的语义鸿沟是一个巨大的挑战。数据中的复杂关系网络。在大数据的背景下，个体之间存在关系。在互联网上，个人是网页，通过超链接相互链接的网页构成了一个复杂的网络。形成复杂社交网络的个体之间也存在社交关系，例如来自 Facebook、Twitter、LinkedIn 和

其他社交媒体，包括呼叫详细记录 (CDR)、设备和传感器信息 GPS 和地理编码地图数据、通过管理文件传输协议传输的海量图像文件、网络文本和点击流数据、科学信息、电子邮件等在。为了处理复杂的关系网络，新兴的研究工作已经开始解决结构与进化、人群与互动以及信息与交流等问题。大数据的出现还催生了用于实时数据密集型处理的新计算机架构，例如在高性能集群上运行的开源 Apache Hadoop 项目。大数据（包括交易和交互数据集）的规模或复杂性超过了在合理成本和时间限制内捕获、管理和处理这些数据的常规技术能力。在大数据的背景下，复杂数据的实时处理是一项非常具有挑战性的任务。