学校代号 10532　　学　　号 S1510W0597

分 类 号 TP301　　密　　级 普通

湖南大學
HUNAN UNIVERSITY

融合评分和评论的推荐系统研究

学位申请人姓名 屠晓涵

培 养 单 位 信息科学与工程学院

导师姓名及职称 李仁发 教授 唐涛 高级工程师

学 科 专 业 计算机技术

研 究 方 向 云计算

论 文 提 交 日 期 2017 年 05 月 10 日

学校代号：10532

学　　号：S1510W0597

密　　级：普通

湖南大学工程硕士学位论文

融合评分和评论的推荐系统研究

学位申请人姓名：　屠晓涵

导师姓名及职称：　李仁发 教授　唐涛 高级工程师

培 养 单 位：　信息科学与工程学院

专 业 名 称：　计算机技术

论文提交日期：　2017 年 05 月 10 日

论文答辩日期：　2017 年 05 月 21 日

答辩委员会主席：　邝继顺 教授

A Study on Recommendation Systems of the Fusion of Score and Review

by

TU Xiaohan

B.E.(Nanyang Institute of Technology)2014

A thesis submitted in partial satisfaction of the

Requirements for the degree of

Master of Engineering

in

Computer Technology

in the

Graduate School

of

Hunan University

Supervisor

Professor LI Renfa

Senior engineer TANG Tao

May, 2017

湖 南 大 学

学位论文原创性声明

本人郑重声明：所呈交的论文是本人在导师的指导下独立进行研究所取得的研究成果。除了文中特别加以标注引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写的成果作品。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律后果由本人承担。

作者签名：　　日期：　　年　月　日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权湖南大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本学位论文属于

1、保密□，在_____年解密后适用本授权书。

2、不保密□。

（请在以上相应方框内打“√”）

作者签名：　　日期：　　年　月　日

导师签名：　　日期：　　年　月　日

摘 要

目前，推荐系统面临着可扩展性差、数据稀疏性、冷启动、低效率、忽略评论信息等问题。本文针对这些问题提出融合评分和评论的推荐系统，加入语义分析，优化算法，并设计出相应的管理系统。主要工作如下：

第一，本文在 ALS 推荐算法基础上提出一种新的评分模型 NALS-WR，并借助 Linux 集群上的 Spark 实现 NALS-WR 算法的并行化，由于其分布式计算框架和云计算环境，NALS-WR 具有很强的可扩展性，解决传统的矩阵分解算法的资源分配的瓶颈，提升处理海量数据的速度。在真实的 MovieLens 数据集上进行实验，结果表明，与传统的最小交叉二乘法 ALS 推荐算法和奇异值分解（SVD）等推荐算法相比，NALS-WR 推荐的准确率和运行效率提高了，数据规模越大，NALS-WR 的效率越高。

第二，提出融合评分和评论的模型 FRRM（Fusion of rating and review model），该模型采用 LDA 主题模型挖掘潜在主题信息，构建文档—主题分布，进而基于评分数据与文档—主题分布共同构建一种新的预测评分的策略。利用评论文本增强评分预测是本文的另一大亮点，其克服了传统方法丢弃评论文本的做法。在真实的 Amazon 数据集上进行实验，结果表明，融合评论与评分的模型 FRRM 克服了原始仅利用评分数据进行评分预测的不足，在评分数据稀疏度高的情况下也可以比同类模型 HFT 更准确的预测物品评分。

第三，设计推荐系统的流程管理平台。利用开源项目 Spark-Jobserver 管理推荐系统的作业和程序等。利用大数据基础平台 Hadoop 提供的分布式文件系统存储海量数据。利用 Spark 分布式计算框架进行数据的分析和挖掘。实现数据分析的抽取，转换和加载过程，实现推荐程序的管理，实现推荐系统的执行以及推荐作业的可视化，并提供可扩展的数据分析程序和流程管理接口。然后部署到云端，使在手机、平板等任何联网的设备上都可实现流程管理，达到对推荐系统的控制更加灵活的目的。

关键词：推荐系统；ALS 算法；Spark 平台；LDA 算法；评论数据

Abstract

At present, the recommendation system faces many problems, such as poor scalability, data sparsity, cold start, low efficiency and ignoring the useful comments. In order to solve these problems, we put forward a recommendation system which integrates rating and review data, realize two optimization algorithms, and design the corresponding management system. Our main work is as follows:

Firstly, we propose a new rating model (NALS-WR) based on ALS recommendation algorithm, and implement the parallelization of NALS-WR algorithm with Spark on Linux cluster. Owing to the distributed computing framework and cloud computing environment, our model has strong scalability, especially to solve the bottleneck of resource allocation what traditional matrix decomposition algorithms are facing. Apart from that, the speed of processing massive data is also greatly improved. Then we do experiments on real data set (MovieLens). Compared with other recommendation algorithms based on alternating-least-squares (ALS) and singular value decomposition (SVD) algorithms, the accuracy and operation efficiency of NALS-WR are improved. Much larger the size of data are, much higher the efficiency of NALS-WR is.

Secondly, we construct FRRM (Fusion of rating and review model) model which fuses rating and review. It uses LDA topic model to mine the potential topic information, constructs document-topic distribution, and builds a new forecasting strategy based on the rating data and document-subject distribution. It is another highlight of our paper to use the comment text to enhance rating prediction, which overcomes the traditional method of discarding the review. We do experiments on real data set (Amazon). The results show that, FRRM overcomes the defect of prediction that only uses rating data to predict. In the case of higher sparsity of data, FRRM can be more accurate than the similar model, such as HFT and RMR.

Thirdly, we design a management platform of the process of recommended system. This is a new web application, through the open source project of Spark-Jobserver to manage the work and the context of work of

recommender system. We use a large data platform based Hadoop distributed file system to store massive data, and use Spark (distributed computing framework) for analysing and mining of massive data. The project includs extraction, transformation, loading process for programmers. We achieve the management of recommendation system and visualization of recommended jobs. We provide a scalable program of data analysis and interface of process management. We deploy our platform to the cloud. No matter we use mobile phone, tablet computer or other network devices, we all can control recommendation system, which is more flexible and convenient.

Key Words: Recommendation system; ALS algorithm; Spark platform; LDA algorithm; Review data

目 录

插图索引

附表索引

第1章 绪论

目前几乎大多数广告推荐、电子商务系统搜索引擎，社交网络，等等，都不同程度的使用了各种各样的推荐系统，推荐系统的研究也非常火热。本章首先概述了论文的研究背景及意义，然后明确研究问题，并分析这些问题，进而重点总结本文的研究内容，突出本文的贡献，并列出论文的结构。

1.1 研究背景及意义

随着计算机技术和互联网的快速发展，人们渐渐从信息匮乏的时代走向了信息过载的时代。在日常生活中，无论是信息生产者还是信息消费者都遇到了特别大的挑战：作为信息生产者，怎么样让自己生产的信息快速地脱颖而出，受到大量用户的关注，是一件十分困难的事情。作为信息消费者，怎么样从大量信息中找到自己感兴趣的信息也是一件非常困难的事情。推荐系统就是解决这一矛盾的完美工具。如何联系过载的信息和过多的用户，一方面帮助用户发现对自己有意义的信息，另一方面让有用的信息能够展现在对它们感兴趣的用户面前，这是推荐系统的首要目的。

推荐系统在商业领域中伴随着举足轻重的位置。首个自动推荐系统GroupLens[1]是由美国明尼苏达大学的 Neophytos Iacavou[2]和 MIT[3]的实验室的 Paul Resnick 创建的，这个推荐系统的对象是网络新闻，推荐算法采用的是最近邻协同过滤的方法和 Botter BitBureaus 系统评分服务器[2]，根据用户过去阅读过的文章来寻找相似用户，并参考相似用户的阅读内容来预测用户对新闻的评分。

推荐系统作为一个活跃的研究课题，受到大多数人的关注[4]。完整的推荐系统一般有 3 个参与方：物品提供者、用户和提供推荐系统的网站。但是完整的并不代表好的，一个好的推荐系统一定是一个能够使三方共赢的系统。早期在对推荐系统进行研究时，大部分人将推荐系统的“好”定义为能够作出非常准确预测的推荐系统。然而，最近一些研究表明，精确地做出预测并不代表是好的推荐系统。例如，一个用户早就准备买《深入理解计算机系统》了，无论给这个用户推荐或者不推荐，此用户都会去购买，那么这个推荐结果显然是不好的，因为推荐效果并不一定会使用户购买更多的书，而仅仅会使用户购买一本本来就准备买的《深入理解计算机系统》。除此，对于当前这个用户来说，会觉得这个推荐结果也不是很新颖，因为此用户本来就要购买这本书，不能令人很惊喜。同时，对于《深入理解计算机系统》的出版社来说，这个推荐当然也没能增加这本书的潜在

购买人数和购买欲望。因此，这是一个看上去很好，但其实却是特别失败的推荐。一个更为极端的例子是，某推测系统预测明天太阳将从东方升起，虽然预测准确率是 100%，却是一种没有意义的预测。

因此，一个好的推荐系统不仅能够准确预测用户的行为，而且还能够扩展用户的视线，有利于用户发现那些可能会感兴趣，但并没有发现的东西。除此之外，推荐系统还要能够帮助商家将那些被埋没在万千商品中的好商品推荐给可能会对它们感兴趣的用户。

推荐系统通过向用户分配要使用的产品的建议来为用户提供一些解决方案，减少用户的选择时间。推荐系统以自动化方式向用户提供“正确的”产品来优化长期业务目标，通过算法实现自动化。此外，使用推荐系统，企业可以更好地了解客户的购买行为，并制定有效的营销策略来吸引各种各样的客户。所以推荐系统在实际生活中具有一定的理论意义和实践意义:

1）理论意义

融合评分和评论的推荐系统可以应用到多个学科领域，有数据挖掘技术、信息检索方面、人工智能领域、机器学习及模式识别等，通过深入研究融合评分和评论的推荐技术，综合利用多学科领域的知识也可以在理论上加速各学科的发展。由于机器学习技术的成熟，以及对各种各样复杂特征的利用方式逐步成熟。目前国内较大的公司如今日头条等，都是在使用机器学习算法来构建各种各样的推荐系统。传统的基于内容、基于用户、协同过滤和各种基于规则或者群体智能甚至物理学的传统推荐方法逐渐地被淘汰，尤其在内容推荐上表现明显，由于这些传统的推荐策略都解决不了对长尾内容的热度穿透和精准推荐等问题，而且远远不如机器学习方法的效果。

2）实践意义

随着信息的快速增加，信息过多和信息重复导致用户被动获取的信息过载，将会花费更多的成本通过搜索引擎主动获得高质量的信息，推荐系统是解决这些问题最有潜力的实践方法，推荐系统的意义在于：帮助用户快速发现喜欢的和高质量的信息或其他感兴趣的东西，提升用户的个人体验，增加用户购买产品的机会，减少用户浏览到厌恶或者重复的信息带来的一些不利影响，使推荐信息更为精准。

作为推荐系统的一种，基于用户评论的产品推荐系统通过采集数据，可以挖掘用户的真实兴趣，帮助用户快速找到满足自己要求的产品，然后帮助商家为目标用户推荐与其兴趣最相似的产品。其次，互联网中的产品量之多，用户在做决定是否购买某一个产品时通常依靠不同用户的对该产品的评价内容，但是推荐系统对于结合用户评论的研究少之又少。由于融合评分和评论的推荐系统技术的研究起源比较晚，发展还不成熟，针对产品的评分数据和用户的评论信息进行建

模，对于推荐最合适的产品至关重要。电子商务网站和社交媒体的流行使得用户自发地用评论来描述对产品的态度。这些评论通常是以文本的形式来解释为什么用户喜欢或者不喜欢某件商品。所以加入评论的推荐系统可以刻画用户评论的一些多层次特性，从而为用户构建一个更为精细的偏好模型，这些往往根据整体评分不能推断出来。另外，从营销和消费者行为学的实证研究里，也证明了产品评论对新用户的决策过程的一系列正面影响。

1.2 研究问题

Netflix 推荐系统中一句名言是：搜索的时代已经结束，推荐将永久持续。许多信息系统中的重要任务是推荐内容。诸如淘宝天猫在线购物网站给每个客户提供用户可能感兴趣的产品的推荐，当当网给每个用户提供用户可能感兴趣的书籍的推荐，YouTube 视频门户向客户推荐电影等。通过从多个维度分析用户信息，结合语义分析，并组合推荐技术和无监督聚类语义分析算法，能够充分提升推荐系统的效率和准确率。在面对海量高维稀疏的数据时，更能快速分析定位用户的兴趣。

目前很多研究是在用户提供评分反馈的情况下进行的[5][6]。但是在现实生活中大多数反馈不是评分反馈，而是隐式反馈[7]。传统的方法往往丢弃评论文本，这使得用户和产品潜在的维度难以解释，因为忽略了可证明评分的文本，所以推荐系统的可解释性不强。

由于信息的种类和数量在网络环境下越来越多。而推荐系统是面向用户的系统，在用户对信息的个性化要求日益明显的情况下，推荐系统的关键问题是怎样向用户推荐感兴趣的信息。所以研究和开发先进的推荐系统就越来越被人们所重视。目前推荐系统中存在的主要问题如下图 1.1 所示：

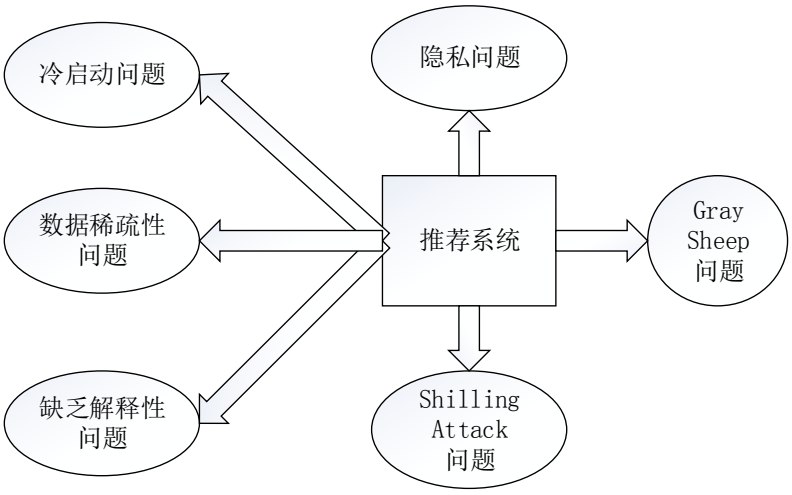

图 1.1 推荐系统面临的问题

推荐系统的表现在：数据稀疏性问题，冷启动问题（缺乏用户和项目的历史

数据），Reduced Coverage 问题和 Neighbour Transitivity 问题。

1）数据稀疏性问题

数据稀疏性问题情况下的挑战是用相对较少的有效评分得到准确的预测。本文归纳总结了三种解决方法。

a）降维技术

降维技术通过奇异值分解（Singular Value Decomposition）来降低稀疏矩阵的维度，为原始矩阵求最好的低维近似，但是存在大数据量运算成本及对效果的影响等问题（因为经过了 SVD 变化之后，一些无意义的 用户或者物品被扔掉了，对这类用户或者物品的推荐效果就要打折扣，小众群体的存在体现不出来）。Funk 在文献[8]中首先将基于 SVD 的矩阵分解应用到推荐算法中，也叫做 SGD 算法。Koren 在文献[9]中提出了 SVD++，使用偏置的 SVD，基于平均评分，用户电影偏置，与矩阵分解组合的基线估计。使用隐式评分扩展了偏置的 SVD，并且在矩阵分解期间仅考虑相关邻域项。Koren 在文献[9]中提出了 time-SVD++，扩展了之前的 SVD++模型，将时间信息考虑在内。这些矩阵分解方法都是基于 SVD 并且使用梯度下降方法来求解这个问题。不同的是 Zhou 等在文献[11]中使用交替最小二乘法与正则化（ALS-WR）算法来求解矩阵分解问题。ALS 算法与 SVD 不同的是，SVD 共享很多的变量，只能应用于较小规模或者大容量的共享内存中求解，而 ALS 算法能够很好地扩展到大规模集群计算，利用分布式并行计算来求解，大大加快了求解速度，并且适应当今大数据的批量计算与云计算的架构模型。

b）填值法

通过对矩阵中已知偏好的研究，对缺少项进行填充，从而降低用户偏好矩阵的稀疏性。如可通过多层次关联规则挖掘方法对用户—项目矩阵进行填充[12]，从而从根本上解决协同过滤算法的数据稀疏性问题。或者采用综合均值优化填充来解决数据稀疏性问题[15]。或者通过 web 数据挖掘，获得隐性数据来填充评分矩阵，然后采用径向基函数进行平滑处理，来解决数据稀疏性问题[16]。再者使用 k 邻近方法对数据填充，后利用支持向量机交叉验证分类，解决数据稀疏性等问题[17]。

c）深度学习

近些年来，深度神经网络成为研究的热点，深度学习方法结合协同过滤方法成为解决数据稀疏性问题的重要方法。深度神经网络可以很大程度上替代某些特征工程的过程，自动化地完成特征的提取，解决推荐系统面临数据不足的问题。YouTube 代表了现有最大规模和最复杂的工业推荐系统之一，但依然注重由深度学习带来的性能改进[18]。文献[19]提出一个基于内容的推荐系统，以解决推荐质量和系统的可扩展性，根据用户的网络浏览历史和搜索查询使用丰富的功能

集来表示用户。并使用深度学习方法将用户和项目映射到潜在空间，其中用户及其首选项目之间的相似性被最大化，通过引入多视图深度学习模型来扩展模型，以共同学习来自不同领域和用户特征的项目的特征，解决了数据稀疏性问题。矩阵因式分解通过简单的固定函数，即作为对于对应的行和列的潜在特征向量的内积来近似矩阵的条目。

文献[20]使用一个任意函数替代内积，从数据中学习在学习潜在特征向量的同时，用多层前馈神经网络替换内积，并通过交替优化固定潜在特征的网络和优化固定网络的潜在特征来学习，稀疏性问题得到解决。文献[21]提出了卷积矩阵分解（ConvMF），一种上下文感知推荐模型，将卷积神经网络（CNN）集成到概率矩阵分解中（PMF），以此来捕获文档的上下文信息，解决了数据稀疏性问题。虽然稀疏输入受到很少的关注，但仍然是神经网络中一个非常具有挑战性的问题。文献[22]介绍了一个神经网络架构，从稀疏评级输入计算非线性矩阵分解。采用基于自动编码器的架构，这是很有前景的一种方法。文献[23]通过使用输入具有缺失值的数据的损失函数，以及通过并入边信息来增强自动编码器架构。其表明边信息略微改善了对所有用户/项目的平均测试误差，但它对冷用户/项目具有更大的影响。

文献[24]提出协同去噪自动编码器（CDAE）方法，使用去噪自动编码器架构，得到 Top-N 推荐。文献[25]提出了 AutoRec，它是一种采用协同过滤（CF）算法的新型自动编码器框架。在线服务在很大程度上依赖于自动个性化来向大量用户推荐相关内容，这要求系统可以迅速扩展以适应第一次访问在线服务的新用户流。AutoRec 是一种紧凑和高效的可训练模型，可以很好的解决数据稀疏性问题。

2）冷启动问题

推荐系统数据稀疏性问题的另一个具体表现是冷启动问题。作为稀疏问题的一个特例，冷启动有新用户和新项目两种情况，当项目新进入系统时，没有或很少用户评价，系统很难推荐这个项目，从而导致恶性循环。当新用户进入系统时，未产生行为数据，系统无法根据其历史行为进行推荐。目前的解决方案研究主要有以下两种：

对于新项目问题，主要使用基于内容和协同过滤混合方法进行推荐。协同过滤推荐算法忽略了项目本身的特征信息，不能推荐没有用户评分的项目也不能对没有评分的新用户推荐项目，如果充分利用各种信息，联合基于内容的协同过滤算法，产生的推荐结果会更加精确[26]。

矩阵分解的潜在因子表示携带用户和项目的情感因子的有价值的信息。从视频情感建模的角度建立有效的推荐系统来研究这些潜在因子。文献[27]提出了一种基于建模用户和项目之间的情感联系的潜在学习因子表示视频的新方法。并提

出了情感建模方法，潜在因子表示视频内容的情感建模的功效，Visual-CLiMF 方法用于隐式反馈，解决冷启动问题[27]。Visual-CLiMF 基于流行的 CLiMF[28] 的改进方法，其表明项目的情感上下文可以用作辅助信息，以提高 MRR 性能。这是一种将辅助内容信息与协同信息结合起来解决视频推荐中的冷启动问题的新方法。

对于新用户问题，一种解决方法是通过用户注册时填写的人口统计学信息给用户提供粗粒度的个性化推荐。另一种方法是在新用户第一次访问推荐系统时，不立即给用户展示推荐结果，而是给用户提供一些物品，让用户反馈其对这些物品的兴趣，然后根据用户的反馈提供推荐[29]。

3）Gray Sheep 问题

Gray Sheep 问题表现为有些人的偏好与任何人都不同，其来源于 Black Sheep，偏好与正常人完全相反，无办法向其推荐，在现实中也很难解决这个问题，一般采用混合推荐的方法解决该问题。

4）Shilling Attack 问题

Shilling Attack 实际上是 AntiSpam 的问题，有些人对自己的东西或者对自己有利的东西打高分，而给竞争对手的东西打低分，这会影响协同过滤算法的正常工作。解决办法可以是采用基于项目的推荐（在 shilling attack 这个问题上，基于项目的推荐效果要比基于用户的推荐的效果好，因为作弊者总是较少数，在计算 Item 相似度的时候影响较小），也可以采用混合推荐的方法（混合推荐方法能够部分的解决偏置的问题）。目前主要的解决办法是采用 AntiSpam 的技术识别，去除作弊者的影响。

5）用户隐私问题

互联网中一直存在的问题是用户隐私问题，推荐系统也不例外，因为需要利用用户的一些记录数据、行为信息、以及用户的人口统计属性信息，其面临着严峻的隐私保护问题。所以推荐系统可能会让用户缺少安全感，使用户不愿提供有利于推荐系统的个人信息，因此影响推荐的效果。Web 技术的大力发展，方便了人们生活，也造成了用户在上网时留下了更多“痕迹”。用户更多不愿透露的隐私可以通过组合来自不同源的用户信息推断出来[30]。用户的隐私保护对于推荐系统的发展显得更加迫切和重要。目前针对用户隐私保护的研究工作主要有分布式协同过滤方法[31]、混合方法[32]和模糊处理方法[33]等。

6）可解释性问题

可解释性问题就是让用户明白推荐系统产生的推荐结果是如何产生的。有研究证明，让用户理解推荐结果的是如何产生的可以使用户更愿意使用推荐系统，明白推荐结果的产生过程可以让用户更加信赖推荐系统，帮助减小使用者的决策风险，提高用户对推荐系统的满意度。这一问题也是该领域学者积极探求的研究

内容之一。

1.3 研究内容与贡献

在大数据环境下，推荐系统面临着许多问题，如可扩展性差、数据稀疏性、冷启动和低效率等。尽管有大量关于建模评分的研究，但是由于网站上的其他形式的反馈—评论，通常被忽略。忽略这种丰富的信息源是现有推荐系统工作的主要缺点。事实上，如果目标是理解（而不仅仅是预测）用户如何评价产品，那么应该依靠评论信息，这样可以为用户解释其评价产品的原因。理解这些评论因素有助于推荐系统预测用户的评分，并可以帮助推荐系统完成其他任务，如类型发现和有意义的评论的识别。

为了解决这些问题，本文首先对数据进行预处理，完成源数据的降维和去噪工作，再将预处理后的数据存储在 Hadoop 分布式文件存储系统中（HDFS），方便操作。其次本文研究的推荐系统加入了用户评分和评论，特别强调用户评论信息，将语义分析融入到协同过滤等方法中，更深入的的分析用户和物品之间的潜在关系，利用评论中的有用信息，同时结合现有的主流推荐方法，作出适当的权衡，得到超出一般推荐方法准确率的推荐算法。将评论中包含的丰富信息融入到用户建模和推荐生成的过程中是一种很有意义的工作。这将使评分得到解释，例如利用评论信息可以发现商品评分是按类型分类的，电影评分是按照电影风格等主题分类的。本文将评分预测与评论语料库联系在一起构建融合评分和评论的模型 FRRM。

结合评分和评论的模型 FRRM 不仅可以使推荐系统通过更好地利用评分维度来“解释”用户的评分数据，而且还能更准确地预测评分。传统模型（仅基于评分）难以应用于新的用户和产品，用来建模隐语义模型的评分数据太少[34]。相比之下，融合评论的模型可仅从一条单一的评论中就可以准确地弥补这些不足，取得更好的效果[35]。本文具体的研究成果可以概括如下：

1）提出一种基于 ALS 的评分模型 NALS-WR，在 Linux 集群利用 Spark 实现这种评分模型，基于内存进行数据处理，实现 NALS-WR 的并行化，与基于 ALS 的推荐算法和奇异值分解（SVD）等推荐算法相比，NALS-WR 推荐的准确率提高了，运行效率提高了，并且数据规模越大，NALS-WR 的效率越高。由于分布式计算框架和云计算环境，本文设计的模型具有很强的可扩展性，特别是解决了传统的矩阵分解算法的资源分配的瓶颈，针对海量数据的处理速度大大提升。实验表明，无论是在可扩展性，或抗稀疏性或效率方面，NALS-WR 算法都更胜一筹。

2）设计融合评分数据和评论信息进行预测评分的模型，在评分模型 NALS-WR 基础上加入评论信息，采用 LDA 主题模型构建文档—主题分布，进而基于

评分数据与文档—主题分布共同构建模型 FRRM，从而依据该模型预测评分。这克服了传统方法丢弃评论文本的做法，使得推荐系统能够识别有用的信息，在评分数据稀疏度高的情况下也可以更准确的预测物品评分。利用评论文本来增强评分预测是本文的另一大亮点。针对传统协同过滤算法存在的数据稀疏性和推荐准确性问题，融合评分和评论的模型 FRRM 结合了评论信息与评分数据，克服了现有的仅利用评分数据进行评分预测的不足。在 Spark 平台上的并行化设计为推荐系统提供了批处理到实时处理的过渡，克服了传统 Hadoop 只有批处理的缺陷。

3）实现相关推荐系统和推荐算法流程管理系统，在任何联网的设备都可实现对推荐程序的管理，对推荐系统的执行，对推荐作业的调度，这种可视化给大数据分析等处理流程带来自动化和直观化的体验。这是一款新型的 web 应用程序，利用大数据基础平台 Hadoop 提供的分布式文件系统（Hadoop Distributed File System，HDFS）存储海量数据，使用 Spark 分布式计算框架，进行海量数据的分析和挖掘。该设计为程序员提供了良好的操作界面，实现了数据分析的抽取，转换和加载（Extract-Transform-Load，ETL）过程，以及业务需求的分析，并提供了可扩展的数据分析程序和流程管理接口，充分体现了 Spark as a service 的设计思想，可以称得上 Another New Web UI for Spark。

1.4 结构安排

本篇论文分为四章，论文组织结构如下：

第 1 章 绪论。主要阐述了课题的研究背景和研究意义，对推荐系统面临的问题及本文要解决的问题进行了具体描述，明确了本文的研究内容。列出了本文的研究贡献。最后组织了本文的章节安排。

第 2 章 相关研究。主要介绍了本文使用的协同过滤算法的研究现状，细致的描述了协同过滤算法的相关改进情况，并对其优缺点进行了分析，详细的介绍了混合推荐算法的研究现状。然后分析了评分和评论它们各自适应的场合，阐述了融合语义分析的推荐算法的研究现状，重点关注了主题模型的相关研究及融合语义分析的推荐算法的改进情况，最后总结了推荐系统的评价指标，为后面章节内容奠定了理论基础和现实依据。

第 3 章 评分模型的建立 NALS-WR。本章设计了融合评分和评论推荐系统的评分模型 NALS-WR，并实现了分布式并行化。与同类推荐算法相比，提高了推荐的准确率和运行效率，然后增大数据规模，实验证明，NALS-WR 的效率与数据规模密切相关。数据集越大，运行效率越高。NALS-WR 位于分布式计算框架和云计算环境下，除了提升海量数据的处理速度以外，可扩展性也很强，特别是解决了传统的矩阵分解算法资源分配的障碍。

第 4 章 评分和评论融合模型 FRRM 的建立。首先介绍了传统语义分析模型在推荐系统领域的缺陷，然后将语义分析模型 LDA 和评分模型结合，利用评论信息加强评分预测，搭建机器学习聚类算法与协调过滤算法的混合框架，既适用于评分数据，也适用于评论信息，可进行语义分析后推荐。弥补了传统方法只利用评分进行预测的缺陷。接下来进行并行化操作，提升推荐的效率，运用大规模数据集实验，实验结果表明 FRRM 在准确率、时间效率等指标上都取得了较好的效果。

第 5 章 推荐系统流程管理平台的设计与实现。本系统采用 JavaEE 架构设计推荐系统的 web 应用程序，通过开源项目 Spark-Jobserver 提供的 REST 接口来提交 Spark 程序到集群执行，管理推荐系统的作业以及作业的上下文，并且在提交作业后，作业的运行信息和执行情况信息可以展示出来，查看作业详细信息，由界面实现程序的管理，之后部署到云端，PC 机和手机等，以便任何联网的设备都可以对推荐系统实现管理。

最后，对本论文主要工作做了总结，并给出了后续的工作展望。

第2章 研究基础及相关进展

2.1 引言

为了解决由数据过载所引起的用户获取其所需信息难等问题，无数商家和工程师提出了很多解决方案，其中有代表意义的是推荐系统。从此不需要用户提供明确的需求，便可以通过分析用户的历史行为给用户的兴趣建模，然后主动给用户推荐能够满足其兴趣和需求的信息。推荐系统可以智能地模拟销售人员为顾客推荐商品的过程，帮助用户决定应该购买什么样的商品，从而减少用户选择商品的时间，提高效率，并可以加强用户的购买行为，提高网上商店的销售利润等。

推荐系统研究的核心之一是评分预测问题。而用户评分数据集是评分预测问题最基本的数据集。此类数据集一般由用户评分记录组成，每一条评分记录是一个包含三个元素的组合（用户 u，项目 i，评分 r），表示用户 u 给项目 i 赋予了评分 r。因为用户不可能对所有项目都评分，所以评分预测问题就是如何通过已知的用户历史评分记录预测未知的用户评分分数。大多推荐系统利用评分数据解决的是如何提高评分的预测精度。另外一种研究是利用评论进行语义分析进行推荐，文献[13]总结了评论信息的具体应用。许多研究都孤立地研究评分和评论文本,将这两个信息来源结合起来的有文献[14]。将评分数据的潜在维度与评论文本中的主题结合在一起的统计模型，比单独考虑两种数据来源的方法更能准确地预测评分。本章对现有的研究评分和评论的推荐系统进行深入研究，挖掘协同过滤推荐算法及语义分析推荐算法的研究现状，为本文设计的模型 NALS-WR 和 FRRM 提供理论指导和现实依据。

2.2 协同过滤推荐算法的相关研究

目前绝大多数推荐系统采用的是协同过滤（Collaborative Filtering）的方法，协同过滤推荐算法是在用户行为中寻找特定模式，来创建用户专属的推荐内容，从历史数据（之前用户对物品的评分）中发现用户和物品的联系，预测用户对未知物品的评分，从而进行各种各样的推荐[12]。基于协同过滤的推荐方法一般分为两大类：基于模型的协同过滤和基于邻域方法的协同过滤。

2.2.1 基于模型的协同过滤

基于模型的协同过滤（Model-based Collaborative Filtering）是先用历史数据训练得到一个模型，再用此模型进行预测。以模型为基础的协同过滤具有代

表性的是隐语义模型（Latent Factor Model）。Latent Factor Model 最早在文本挖掘领域被提出，用于寻找文本的隐含语义，相关的模型常见的有潜在语义分析（Latent Semantic Analysis，LSA）、主题模型（Latent Dirichlet Allocation，LDA），矩阵分解（Matrix Factorization）等。其中实现隐语义模型技术最为广泛的一种方法是矩阵分解，推荐系统的思想也正是来源于此。著名的推荐领域的大神 Yehuda Koren 更是凭借矩阵分解模型勇夺 Netflix Prize 推荐比赛的冠军。

以矩阵分解为基础，Yehuda Koren 在数据挖掘和机器学习相关的国际顶级会议（SIGKDD，RecSys，SIGIR 等）发表了很多文章[5][9][10]，将矩阵分解模型的优势发挥得淋漓尽致。实验结果表明，在推荐系统中使用矩阵分解模型要明显优于传统的基于邻域的协同过滤（又称基于记忆的协同过滤）方法，如 ItemCF、UserCF 等，这也使得矩阵分解成为了目前推荐系统研究领域中的主流模型。

Latent Factor Model 是首先从评分矩阵中发现用户和物品隐因子向量，用户隐因子向量表示用户对物品的兴趣，物品隐因子向量表示物品的特点，然后根据目标用户和未知物品对应的隐因子向量预测未知评分。Koren 总结了隐因子模型，通过最小化均方误差损失函数，寻找一个原始评分矩阵最佳的低秩近似矩阵 R 满足以下条件[5]：

$$R = U^{\mathrm{T}} V \tag{2.1}$$

其中 U 和 V 的列向量分别对应用户的隐因子向量和物品的隐因子向量，二者的内积表示预测评分。以 SVD（奇异值分解）为例，对一个给定的用户行为数据集（数据集包含的是所有的用户（urser），所有的项目（item），以及每个用户有过评分行为的项目列表），使用隐语义模型对其建模后，可以得到如下图 2.1 所示的模型：

	项目1	项目2	项目3	项目4
用户1	R11	R12	R13	R14
用户2	R21	R22	R23	R24
用户3	R31	R32	R33	R34

=

	类别1	类别2	类别3
用户1	U11	U12	U13
用户2	U21	U22	U23
用户3	U31	U32	U33

×

	项目1	项目2	项目3	项目4
类别1	V11	V12	V13	V14
类别2	V21	V22	V23	V24
类别3	V31	V32	V33	V34

R U^{T} V

图 2.1 隐语义模型建模示意图

奇异值分解（SVD）的基本原理是将一个 m×m 的矩阵 R 分解为 3 个矩阵。如下所示公式 2.2 所示[5]：

$$R = P \times S \times Q^{\mathrm{T}} \tag{2.2}$$

其中 P 是 m×m 的正交矩阵，Q 是 n×n 的正交矩阵，S 是 m×m 的对角矩阵，该矩阵对角线以外的元素都为 0，对角线上的每个元素 e_1、e_2、e_3、……、e_n 都满足公式 2.3[5]：

$$e_1 \geq e_2 \geq e_3 \geq ... \geq e_n \geq 0 \tag{2.3}$$

e_i 值按照从大到小的顺序排列，这些值称为奇异值。基于奇异值模型推荐算法的基本思想是通过以上矩阵分解技术，寻找比原矩阵秩低的且与原矩阵最近似的简化矩阵。即经过矩阵分解后，生成 P、S、Q 矩阵，在对角矩阵 S 上，保留矩阵的 k 个最大奇异值，舍去对角线上其他相对较小的值，生成维度为 $k\times k$ 的对角矩阵 S_k，然后按照同样方法处理矩阵 P 和 Q、分别删除矩阵多余的行和列，于是生成 $m\times k$，$k\times n$ 的简化矩阵 P_k 和 Q_k，然后重构原来的矩阵，得到下面维度为 k 的矩阵，如公式 2.4 所示[5]：

$$R_k = P_k \times S_k \times Q_k^{\mathrm{T}} \tag{2.4}$$

R_k 秩为 k，与原矩阵相似，通过矩阵重构，矩阵 R 的数据变得稠密，为了求解评分，现得到一损失函数，如公式 2.5 所示[5]：

$$\min \sum (R - U^{\mathrm{T}}V)^2 + \lambda(\|U\|^2 + \|V\|^2) \tag{2.5}$$

其中 λ 为正则化参数，令更新的步长为 γ，采用梯度下降法求解，如公式 2.6、2.7 所示[5]：

$$U = U + \gamma(e \times V - \lambda \times U) \tag{2.6}$$

$$V = V + \gamma(e \times U - \lambda \times V) \tag{2.7}$$

其中，$e = R - U^{\mathrm{T}}V$，表示实际评分和预测评分的误差。总体上就是得到损失函数，然后用 SGD 求解，对于每个已知用户—项目的评分，都更新一次模型。

隐因子模型的思想在于通过 U、V 得到用户因子和项目因子间的关系，但因项目属性、用户偏好的原因，使得不同项目或用户有不同的评分基准，因而 Koren 考虑了评分基准的差异的情况，为了能更精确的预测评分，给出了一个考虑基准偏差的评分预测模型，如公式 2.8 所示[5]：

$$b_{ui} = \mu + b_i + b_u + U^{\mathrm{T}}V \tag{2.8}$$

然后优化求解的目标得公式 2.9 所示：

$$\min \sum (R - \mu - b_i - b_u - U^{\mathrm{T}}V)^2 + \lambda(\|U\|^2 + \|V\|^2 + \|b_u\|^2 + \|b_i\|^2) \tag{2.9}$$

采用基于项目的协同过滤算法主要有 Amazon.com[36]，它是由 Brent Smith、Gerg Lindent 和 Jeremy York 共同研发的。Amazon 的推荐系统根据用户过去的购买记录及评价的信息，在传统协同过滤算法的基础上做了改进，将离线和在线进行有效结合，找出类似的项目推荐给当前用户，该算法节省了时间与空间的开销，提高了算法的运行效率，并获得了较高的推荐精度。

下面研究另外一类协同过滤推荐算法——基于邻域方法的协同过滤。

2.2.2 基于邻域方法的协同过滤

基于邻域方法的协同过滤包含基于用户的协同过滤和基于项目的协同过滤。基于用户的协同过滤（User—Based CF）首先基于这样一个假设：用户与其邻居用户具有相似的兴趣爱好，用相似统计的方法得到具有相似兴趣爱好的相近用户，这最早由来自美国 Minnesota 大学的 Paul Resnick 等人提出[37]。其优点是在数据集的内容比较完善时，可以避开物品内容上的挖掘进行精准推荐。但是也有一定的缺陷，比如基于用户的协同过滤算法的时间会随着用户数量增多而计算时间变长，会随着新用户加入出现数据稀疏性的问题。于是在 2001 年由 Sarwar 等提出了基于项目的协同过滤推荐算法[38]。这里所依据的一个基本的假设是："能够引起用户喜爱的项目，一定是与其之前评分高的项目相似"，一般来说：基本上喜欢《深入理解计算机系统》的人，都会去看《计算机体系结构》。基于项目的协同过滤推荐主要分以下三步：根据评分矩阵首先计算出项目之间的相似度，然后在目标用户评过分的项目中选择和未知项目最相似的 K 种物品作为相邻项目，最后根据目标用户对相邻项目的评分预测作为其对未知项目的评分。

为了克服冷启动、数据稀疏性问题，通常将最近邻协同过滤推荐算法和关联规则等相结合起来，如 MovineLens[39]。MovineLens 对电影进行多维度评分推荐，用户可以在"没有看过"和"0.5-5"分这几个选项中对电影进行打分，然后网站通过对用户长期观看电影评分的掌握，获取用户对电影的偏好信息，之后对用户进行推荐。核心思想是为每个用户寻找最近邻居，并将最近邻居的喜好推荐给当前用户。

协同过滤是在海量数据中挖掘出结果，尽管常见的协同过滤算法的推荐结果冷人满意，但也有一些缺点，比如数据缺失时，就无法进一步的进行评分预测或 Top-N 预测，表 2.1 对其进行了总结：

表 2.1 协同过滤推荐算法优缺点

优点	缺点
大多情况结果令人满意	冷启动问题
不需要了解用户及物品的特性	稀疏性问题
推荐个性化、自动化程度高	系统延伸性问题
有推荐新资讯的能力	很难提供解释

2.2.3 协同过滤推荐算法的相关改进

由于协同过滤推荐算法存在着缺陷，所以研究者们提出了一些改进策略。比如，Sarwar B M 等人提出利用矩阵降维技术来解决数据的稀疏性问题[40]。Schein A I 等人提出了一种混合算法，这种算法融合基于协同过滤和基于内容的算法为一体，解决了部分冷启动问题[41]。George T 等人提出了基于聚类的协同过滤算法[42]，使计算复杂度降低了，并且提高了推荐系统的可扩展性和实时

性。此外，Koren Y 等人提出引入时间信息的方法来提高推荐系统准确性[10]。下面分别从时间信息等方面介绍协同过滤推荐算法被国内外学者改进的情况。

1）引入时间信息的协同过滤算法

在推荐系统中，用户的兴趣爱好以及产品的流行程度都会随时间的改变而改变。例如，用户在冬天会喜欢购买热饮和热狗等食品，而在夏天对冷饮等更加喜爱，却对热饮的喜爱程度急速下降，更偏向于选择可降温的凉爽饮料，这属于季节效应：一个电视剧在它刚上映时会非常流行，受到众多用户的喜欢，但是当这部电视剧演完一段时间过去之后，这个电视剧的流行程度便会明显的下降。

因此，引入时间信息的协同过滤算法可提高推荐系统的准确率。在之前的推荐算法基础上一些研究学者改进协同过滤推荐算法，其中之一是通过加入时间动态等信息，Koren 曾构建了能够感知时间效应的因子模型，进行时间效应建模的研究工作：首先考虑推荐系统预测的时间效应，假设项目或用户的评分基准随时间而变化，评分基准加入时间效应后如公式 2.10 所示[10]：

$$b_{ui}(t) = m + b_i(t) + b_u(t) \tag{2.10}$$

其中，t 单位是天，$b_u(t)$表示用户 u 在第 t 天的评分基准。对于用户、项目的时间效应则提出了如曲线、线性等形式的时间建模，公式 2.11 和公式 2.12 表示了一种简单线性形式[10]：

$$dev_u(t) = sign(t - t_u)\left|t - t_u\right| \tag{2.11}$$

$$b_{ui}(t) = m + b_i + b_u + a_u \times dev_u(t) + b_{i.Bin(t)} \tag{2.12}$$

$dev_u(t)$表示对用户 u 在第t_u天作出的评分，用户 u 在第 t 天再次评分时候产生的偏差，α_u表示用户 u 评分随时间变化的系数，不同用户的变化率可能不同，$b_{i.Bin(t)}$表示项目 i 在不同时段的评分基准。加入时间效应的因子模型如公式 2.13 所示[10]：

$$b_{ui}(t) = \mu + b_i(t) + b_u(t) + U^T(V(t) + \left|R(u)\right|^{\frac{1}{2}} \sum_{j \in R(u)} y_j) \tag{2.13}$$

其中$b_u(t)$和$b_i(t)$作为程序的基准预测器，y_j表示被当前用户 u 评分过的项目用以描述用户的向量。

2）协同过滤算法的其他改进

在基于模型的协同过滤算法中加入社交网络信息、语义信息来解决缺乏评分数据带来的冷启动等问题是推荐系统的改进重点，Qiang Liu 等人加入上下文信息挖掘项目间的关系，将得到的项目关系加入概率矩阵分解 PMF 中[42]，基本步骤是先构造“项目—上下文”矩阵，各矩阵元素是该上下文中的项目评分，然后矩阵分解得到每个项目的隐因子向量，再计算项目之间的隐因子向量相似度s_{ij}，最后得到一目标函数（含正则化项）。

这种方法可通过很多隐式信息挖掘项目间的相似性，但在计算相似性时由于

所有项目需两两计算，所以时间复杂度较高。

计算方法如式 2.14 所示[43]。

$$F=\frac{1}{2}\sum_{u=1}^{m}\sum_{i=1}^{n}I_{ui}^{R}(R_{ui}-U_u{}^{\mathrm{T}}V_i)^2+\frac{\lambda_U}{2}\sum_{u=1}^{m}\|U_u\|_{F_{ro}}^2+\frac{\lambda_V}{2}\sum_{i=1}^{n}\|V_i\|_{F_{ro}}^2+\frac{\lambda_S}{2}\sum_{i=1}^{n}((V_i-\sum_{j\in N_i}V_jS_{ij})^{\mathrm{T}}(V_i-\sum_{j\in N_i}V_jS_{ij})) \tag{2.14}$$

另外一类改进是在隐因子模型中加入社会信任关系的因子，由 H．Ma 等提出，此类算法基于具有权值的社交网络，基本思想是将“用户一项目”交互中用户的分量分为两部分来考虑，一部分是当前用户所信任的用户对该用户的影响，另一部分是用户自身的偏好习惯，两者间用一权重因子 α 来调整，其形式为式 2.15 所示[44]：

$$L(R,S,U,V)=\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}I_{ij}^{R}(R_{ij}-g(\alpha U_i{}^{\mathrm{T}}V_j+(1-\alpha)\sum_{k\in\tau(\mathrm{i})}S_{ik}U_k{}^{\mathrm{T}}V_j))^2+\frac{\lambda_U}{2}\|U\|_F^2+\frac{\lambda_V}{2}\|V\|_F^2 \tag{2.15}$$

其中，S_{ik} 为当前用户对它所信任的用户的信任度，完全基于信任度的计算如式 2.16 所示：

$$\hat{R}=SR \tag{2.16}$$

这种方法通过观察用户喜好被社交网络中用户之间的信任关系影响，然后在此基础上，对隐因子模型改进，量化信任行为，提高了推荐系统的准确性。除此之外，Yehuda Koren 把隐语义模型的协同过滤推荐方法和基于邻域方法的协同过滤推荐方法集成到一个目标函数中[9]，将二者结合起来克服了各自的缺点，发挥了各自的优点。

2.3 混合推荐算法的相关研究

推荐系统中各种算法都有各自的缺点和优点，为了使推荐效果更加精确，研究者便使用混合推荐算法（Hybrid recommended system）来做推荐。不同推荐算法的混合方式有很多种，目前常用的几种混合推荐方法有以下 6 种[45][46][47]。

1）混合：将多种不同的推荐算法推荐出来的结果混合在一起，其难点是如何重排序。

2）加权融合：加权混合多种推荐算法的结果然后产生推荐，最常见的方式是线性混合，比如将基于项目的推荐算法和基于模型的协同过滤推荐算法赋予相同的权重值，然后比较该推荐系统的预测值与用户对物品的实际评分是否相符，根据比较结果调整基于项目的推荐算法和基于模型的协同过滤推荐算法的权值，这种线性混合基于感知器。

3）切换：推荐方法随着实际情况而变化，不同的背景问题采用不同的推荐方法。比如，使用协同过滤和基于内容推荐算法混合的方式，首先推荐系统使用基于内容的推荐算法，如果推荐结果不准确，然后再尝试使用协同过滤推荐算法。优点是及时采用最优的推荐算法。但由于需要各种情况比较转换标准，因此这种方法使算法的参数化和复杂度比较高。

4）特性结合：首先组合不同推荐数据源的特征，以便其他推荐算法采用。比如作为增加的特征向量可以是协同过滤算法的信息，然后在这增加的数据集上采用基于内容的推荐算法。优点是降低了使用者对项目评分数量的敏感度，允许推荐系统拥有项目的内部相似信息，其对协同系统是不透明的。

5）级联：前一个推荐技术用后一个推荐技术来优化。这是一个分阶段的过程，前一种推荐技术产生一个大概的候选结果时，再使用第二种推荐方法，使推荐更加准确。级联型允许系统对某些项避免在后面低优先级的推荐器中被过滤掉，这些项可能是通过第一种推荐技术被较好的予以区分了的，或者是很少被用户评价从来都不会被推荐的物品。因为级联型的第二步，仅仅是集中在需要另外判断的项上。

6）特征递增：特征递增一个典型的例子是将聚类所提供的类别特征用于关联规则挖掘中。前一个推荐方法的输出作为后一个推荐方法的输入，它与级联型的不同之处在于，上一级产生的目的是为下一级的推荐提供一些有用的特征，并不是直接的推荐结果。表 2.2 对混合推荐算法的优缺点进行了相关总结：

表 2.2 混合推荐算法优缺点

优点	缺点
比单独的某种算法要好	比较难
无冷启动问题	过程繁琐
不存在流行度偏颇	平衡时会出现很多问题
可以实现多样化	技术要求高

2.4 融合语义分析的推荐算法相关研究

针对推荐系统中数据的稀疏性问题，一个有效的解决办法是运用主题模型处理评论信息。主题模型就是用来发现大量评论内容的主题的算法。借助这些算法可以对文档集合进行归类，其适用于大规模数据场景。目前可以做到分析流数据。

早期的工作中，结合评论文本和评分的有文献[48]等。G.Ganu 等人发现评论信息的多重维度，并且用户的评分将取决于每个特征的重要性。这些维度可以从评论文本中恢复，恢复出的信息可以用于评分预测[48]。个体用户可以在确定其总体得分时向这些特征分配不同的权重，例如一家酒店的酒店评论员可能会给“价格”赋予较低的权重，而对“服务”赋予较高的权重，从而解释为什么这些用户的整体评分和对价格感兴趣的评论者不同[48]。这项研究针对自由文本识别

信息，并使用这些信息来改进用户查看评论时的体验。具体来说，这些工作专注于提高餐厅评估的推荐准确性。利用分类工作，在此过程中去获得用户评论行为。然后应用用户评论的文本部分，提出新的基于特征和回归的推荐方法。文献[48]的研究结果表明，使用基于文本信息结果的一般或个性的评论信息比用户给出的数字星级评分得到的效果要好。但是，此研究必须予以提供“特征”（例如，使用领域知识来确定餐馆的“特征代表”是价格或服务等），其中每个情感分类器必须被训练，缺点是比较繁琐。

Blei D M 等人在文献[49]中提出了潜在狄利克雷分配模型（Latent Dirichlet Allocation，LDA）。在 LDA 中，所有的评论共有同样的话题集，但是每个评论以不同的比例展示对应的主题。LDA 的主要目标是自动发现一个评论集合中的主题。这些评论本身是可以观测到的，而主题的结构是隐藏的。主题建模的核心计算问题就是使用观测到的评论来推断隐藏主题结构。这也可以看作是生成过程的逆过程—什么样的隐藏结构可以产生观测到的评论集合。LDA 的分布函数如公式 2.17 所示[49]：

$$Dir(\theta|\alpha)=\frac{\Gamma(\sum_{k=1}^{K}\alpha_k)}{\prod_{k=1}^{K}\Gamma(\alpha_k)}\prod_{k=1}^{K}\theta_k^{\alpha_k-1} \tag{2.17}$$

其中$0\le\theta_k\le1$，Γ是伽马函数。每条评论或每个文档 d 都有一个与其相对应的主题分布θ，表示这条评论或这个文档包含不同主题的概率，或不同主题在这条评论或文档中出现的概率。向量θ是从狄利克雷分布公式 2.17 抽样产生的，由狄利克雷分布的超参数α生成。

通过主题模型获得的这些主题可比喻成望远镜不同的放大倍数。根据特定的情形设置可以看到的关注对象的精度，并且可以研究主题随着时间变化产生的相关变化。这样的思考方式对于很多场景都是有效的，例如在推荐时，可以把单纯使用关键词的相关性推进到结合主题的结果整合从而给用户更好的体验。

主题模型实际上是一种语义上的理解，相比其他一些语法分析模型（例如 NLP 中的 HMM 模型）是一种更加宏观的文档描述。因为语法模型只能描述单独句子中的元素之间的关系，这对于真正了解作者的目的是帮助不大的。有了主题模型，可获知用户的偏好是哪些主题，这些主题是关键词相关的，最简单的方式就是拿这些关键词去匹配项目物品，获得的项目物品作为推荐的种子，然后根据用户的历史记录来过滤已经观看过或读过并喜欢的项目或物品。

推荐系统最近的研究考虑了自动识别评价维度的功能，并尝试以无监督的方法进行推荐[50][51]，但是这些复杂的方法仅限于只有几千个评论信息的数据集。此外，文献[52]总结了评论信息的具体应用。对自由文本的维度建模也是一个研究重点。主题模型（LDA）目的是发现文本中的隐藏维度[49]。这样的模型已经

应用于推荐系统。例如，推荐引文网络中的科学文章[53]。随后 LDA 的变体被提出，其维度与输出变量（例如评分）有关，例如监督主题模型[54]。关于情感分析的广泛类型的研究有文献[55]，其改进 LDA 算法为 sLDA，这是一个标记文件的统计模型，之后借助于参数估计的近似极大似然法，依赖于变分方法去处理比较棘手的后期望，使用拟合模型来预测新文件的响应值，该模型可容纳多种响应类型。文献[56]提出了一种新的概率模型框架 JST，其基于 LDA，从文本中同时检测情感和话题。不像其他的机器学习的情感分类方法，往往需要标记的语料库训练分类器，JST 是一个有效的四层模型，模型是完全无监督的。如图 2.3 所示[56]：

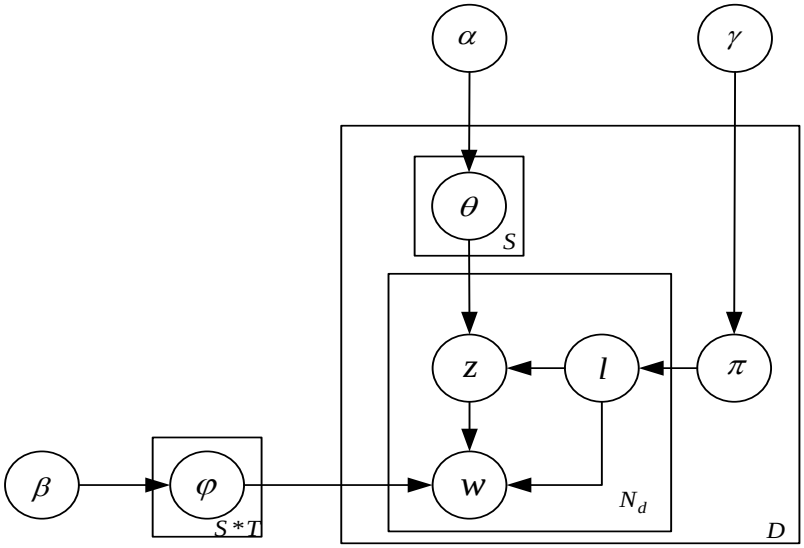

图 2.2 JST 模型示意图

其中超参数 α 表示与情感标签 l 相连的主题 j 的取样次数，β 表示从与情感标签 l 相连的主题 j 中单词的取样次数，超参数 γ 表示从文档取样情绪标签 l 的次数，有三个需要推断的潜变量：联合情感/主题—文档分布 θ，联合情感/主题—词语分布 φ，以及情感—文档分布 π。情感—文档分布 π 可以检测文档的属性。z 表示在集合中的所有单词的主题的分配向量，l 表示在集合中的所有单词的标签的分配向量，ω 表示生成的单词。

文档 D 的集合记为$\{d_1,d_2,d_3,\ldots,d_D\}$，每个文档是一堆单词 N_d 的集合，可表示为 $d=\{w_1,w_2,w_3,\ldots,w_{Nd}\}$，每个单词都是由词汇索引中的一个项组成的，词汇索引中包含 V 个不同的项，可记为$\{1,2,3,\ldots,V\}$，S 表示不同的情感标签的数目，T 为主题的总数目。在文档 d 中生成单词 w_i 的过程有 3 个过程：首先从文档特定情感分布 π_d 中选择一个情感标签 l，然后从主题分布 $\theta_{l,d}$ 中随机选择一个主题，$\theta_{l,d}$ 是基于情感标签 l 选择的。值得注意的是，在这一点上，JST 的主题分布不同于 LDA，LDA 对于每个独立的文档仅有一个主题—文档分布 θ。相反，JST 中每个文档有 S 个主题—文档分布，每个都和情感标签 l 相关联。此功能主

要为 JST 模型衡量主题的情绪提供了方法。最后，由主题和情感标签定义的单词分发词中划出一个单词，这与 LDA 再次不同，LDA 从仅由主题定义的词分布中抽取单词。对应上图 2.2 的生成过程为[56]：

1）对于每个文档 d，选择一个分布 $\pi_d \sim Dir(\gamma)$；

2）对于每个基于文档 d 的情感标签 l，选择一个分布 $\theta_{d,l} \sim Dir(\alpha)$；

3）对于文档 d 中的每个单词 w_i：

a）选择一个情感标签 $l_i \sim \pi_d$，

b）选择一个主题 $z_i \sim \theta_{d,l}$，

c）从主题 z_i 定义的词分布、情感标签 l_i 和 φ 中选择一个单词 w_i。

以上分析都是采用 LDA 的变体进行文本分析，其目标是对来自文本的数字分数进行建模，综合利用文本的信息和其他显式反馈进行情感分析，预测文本的情绪。

2.5 推荐系统的评价指标

1）平均误差 MAE（Mean Absolute Error）

平均绝对误差 MAE 是评价推荐算法质量的标准之一，它通过计算预测评分与真实评价数 据上的差别来衡量推荐结果的准确性。MAE 的值越小，推荐准确性越高。假设预测的用户 i 评分表示为 p_i，对应的实际用户评分集合为 q_i，则具体的 MAE 计算公式为[29]：

$$MAE = \frac{\sum_{i=1}^{N} |p_i - q_i|}{N} \tag{2.18}$$

2）准确率与召回率（Precision & Recall）

推荐系统中广泛使用的两个度量值是准确率和召回率，评价推荐效果的好坏。其中准确率是推荐出的正确项目条数与提取出的项目条数的比率，衡量的是推荐系统的查准率。召回率是指推荐出的正确项目条数与所有的项目条数的比率，衡量的是推荐系统的查全率。通俗来说，准确率就是推荐出来的条目中（比如：物品、电影等）准确的有多少，召回率就是准确的项目中被推荐出来的有多少。为了避免单方面的缺陷，引入另外一种评价指标 F1 值，即正确率和召回率的调和平均值。这些指标的定义如 a）、b）、c）所示[29]：

a）正确率=提取出的正确信息条数/提取出的信息条数；

b）召回率=提取出的正确信息条数/样本中的信息条数；

c）F1 值=正确率×召回率×2/（正确率+召回率）。

其中，正确率和召回率取值在 0 和 1 之间，正确率取值结果越接近 1，正确率就越高，召回率取值结果越接近 1，召回率就越高。本文用 $R(u)$表示向用户 u

推荐的 N 个物品集合。$T(u)$表示数据集合中用户 u 真实评分过的物品集合。则如公式 2.19 所示计算正确率[29]：

$$Precision = \frac{\sum_{u} |R(u) \cap T(u)|}{\sum_{u} |R(u)|} \tag{2.19}$$

3）综合评价指标（F-Measure）

准确率和召回率指标有时候会出现矛盾的情况，这样就需要综合考虑，最常见的方法就是 F-Measure（又称为 F-Score）。F-Measure 是正确率和召回率加权调和平均[29]：

$$F = \frac{(a^2+1)P \times R}{a^2(P+R)} \tag{2.20}$$

当参数 α=1 时，就是最常见的 *F1* 评价指标，如公式 2.21 所示：

$$F1 = \frac{2 \times P \times R}{P+R} \tag{2.21}$$

可知 *F1* 综合了准确率和召回率的结果，当 *F1* 较高时则能说明试验方法比较有效。

4）流行度

一般来说，流行度即为评价某项目的用户数，如公式 2.22 所示[29]：

$$Popularity = \frac{\log_e(1+P)}{N} \tag{2.22}$$

其中 P 为平均流行度，N 为推荐的项目数。

5）新颖性

新颖的推荐是给用户推荐以前没有听说过或购买过的物品。在一个网站中实现新颖性的最简单办法是，把那些用户之前在网站有过购买行为的物品从推荐列表里过滤掉，当然最好是把在其它网站有过购买行为的物品也过滤掉[29]。评测新颖度的最简单方法是利用推荐结果的平均流行度，因为越不热门的物品越可能让用户觉得新颖。所以，如果推荐结果中物品的平均热门程度较低，那么推荐结果就越有可能有比较高的新颖性。但是，用推荐结果的平均度度量新颖性比较粗略，因为不同用户不知道的东西是不同的。因此，要准确地统计新颖性需要做用户调查。通过牺牲精度来提高多样性和新颖性是比较容易的，而困难的是如何在不牺牲精度的情况下提高多样性和新颖性[29]。

6）惊喜度

如果推荐的结果和用户的历史兴趣不相似，但却让用户满意，那么推荐结果的惊喜度就很高，而推荐的新颖性仅取决于用户是否听说过这个推荐结果[29]。

2.6 小结

本章围绕推荐系统和语义分析，系统论述了推荐系统中协同过滤推荐算法的相关研究，具体到协同过滤算法的改进有哪些，之后对混合推荐算法进行了研究，总结了融合语义分析的推荐算法的相关研究，重点关注了主题模型及融合语义分析的推荐算法改进情况，最后明确了推荐系统的评价指标。

第 3 章 基于矩阵分解的评分模型 NALS-WR

本章介绍 ALS 矩阵分解算法，及优化和并行化的实现过程，布署本节的实验框架，比较本节模型 NALS-WR 和经典算法 ALS-WR 和 SVD 的优劣，并分析了评分模型 NALS-WR 的可扩展性、抗稀疏性和实现效率。

3.1 基于最小二乘法的协同过滤推荐算法

推荐系统中，多维稀疏矩阵通过对矩阵进行降维达到降低数据集稀疏性的目的。矩阵分解技术，不仅能够对矩阵降维，而且能够过滤数据中的噪音，因此，矩阵分解方法经常被采用来解决数据集稀疏性问题。矩阵分解推荐算法的一种 ALS（alternating-least-squares）是通过模型来预测一个用户对一个项目的评分，最基本的数据是用户—物品的评分矩阵，例如表 3.1 所示：

表 3.1 用户-项目评分矩阵

用户|项目	V_1	V_2	V_3	V_4	V_5
U_1	4	4	5	?	?
U_2	5	3	1	?	?
U_3	2	?	3	5	4
U_4	1	2	?	?	3
U_5	?	?	1	?	1

这里U_1，U_2，U_3，U_4，U_5表示的是 5 个不同的用户，V_1，V_2，V_3，V_4，V_5表示的是 5 个不同的商品，这样便构成了用户—商品矩阵，在该矩阵中，有用户对每一件商品的打分，其中“？”表示的是用户未对该商品进行打分。现在目的是把没有评分的都预测出来，然后按预测的分数高低，给用户进行推荐。对于缺失的评分，可以转化为基于机器学习的回归问题，也就是连续值的预测，对于矩阵分解来说，其目标是把用户 user 和项目 item 映射到同一个维数为 d 的隐因子空间中，然后可通过隐因子空间中的内积来建模 user 与 item 之间的关系，如下式子 3.1 所示[11]：

$$X \approx UV^T,\ U \in C^{m\times d},\ V \in C^{n\times d} \tag{3.1}$$

其中 X 是一个 $m\times d$ 的矩阵（m 表示 user 的数量，也是 X 中横向量的个数，d 表示因子个数，即低维矩阵的维数），V 是一个 $n\times d$ 的矩阵（n 表示 item 的数量）。用户 user 因子矩阵为 U，项目 item 因子矩阵为 V，则每个 user 对应的向量为 U_i，每个 item 对应的向量为 V_j。计算出每个 user 的因子向量 U_i 和每个 item 的因子向量 V_j，使得预测出的用户 U 对项目 V 喜好程度的值 $\hat{X} = U_i V_j^T$ 能够

尽可能地反映出真实值 X，当给定的数据比较稀疏时，这个矩阵分解模型可能会过拟合。所以，目前很多研究都只建模存在的评分项，然后加入正则化参数避免过拟合现象，得到一损失函数，如式 3.2 所示[11]：

$$L(U,V)=\sum_{ij}(X_{ij}-U_{i.}V_{j.}^{\mathrm{T}})^2+\lambda(\left\|U_{i.}\right\|_F^2+\left\|V_{j.}\right\|_F^2) \tag{3.2}$$

其中：λ参数用来正则化模型，通常称为正则化系数，求解上面的模型，采用交替最小二乘迭代法（alternating-least-squares）。其思想和线性回归做预测类似，大概过程如下 4 个步骤[11]：

1）定义一个预测模型，如下公式 3.3：

$$X\approx U_{i.}V_{j.}^{T},\ U\in C^{m\times d},\ V\in C^{n\times d} \tag{3.3}$$

2）确定一个损失函数，如下公式 3.4：

$$L(U,V)=\sum_{ij}(X_{ij}-U_{i.}V_{j.}^{\mathrm{T}})^2+\lambda(\left\|U_{i.}\right\|_F^2+\left\|V_{j.}\right\|_F^2) \tag{3.4}$$

3）将已有数据作为训练集，不断迭代来最小化损失函数的值（其实 ALS 就是上面损失函数最小化的一个求解方法），每次迭代必须满足以下 a）和 b）条件：

a）固定U，逐个更新每个项目的特征V（对V求偏导，令偏导为 0 求解），

b）固定V，逐个更新每个用户的特征U（对U求偏导，令偏导为 0 求解）。

4）最终确定参数，把参数套到预测模型中做预测。

3.2 评分模型 NALS-WR 的设计与实现

本节介绍 NALS-WR 的具体优化细节，以及在 Spark 上的并行化实现，具有理论意义和现实意义。

3.2.1 评分模型的设计

由于 SVD（sigular value decomposition）要同时对整个矩阵 P 进行矩阵分解，得到三个矩阵，SVD 共享很多的变量，只能应用于较小规模或者大容量的共享内存中求解，因此难以并行化。ALS（alternating-least-squares）算法与 SVD 不同的是，其可以很好地扩展到大规模集群中计算，能利用分布式并行计算来求解，可以加快求解速度，适应当今大数据的批量计算与云计算的架构模型，并且经过优化改进，可成为准确率较高的推荐模型。

本文针对推荐系统中的海量数据，以及现有传统的矩阵分解算法的资源分配

的问题，在 ALS 基础上进行改进，设计评分模型 NALS-WR（Normalized Alternating-least-squares with Weighted-λ-regularization）。然后借助 Spark 基于内存进行数据处理，由于矩阵是非常大的且稀疏的，所以选择坐标矩阵。通过弹性分布式数据集（Resilient Distributed Dataset，RDD）的[MatrixEntry]实例创建坐标矩阵，便于进行分布式并行计算。基本思想是通过随机化矩阵 A，然后通过目标函数求得 B，再对 B 进行处理，根据 B 去求 A，在迭代过程中并行化操作来同时更新多个用户隐因子或者项目隐因子。不断迭代下去，直到 A×B 满足一定的收敛条件为止。

本节使用 $P_{i.}$表示表示用户 i 评过的项目的评分组成的向量，V_{ui} 表示用户 i 评过的项目的特征向量组成的特征矩阵。假设 n_{ui} 表示用户 i 评过的项目的数量。$P_{.j}$ 表示评过项目 j 的用户的评分组成的向量，U_{mj} 表示评过项目 j 的用户的特征向量组成的特征矩阵。n_{mj} 表示评过项目 j 的用户的数量。本文的评分模型如下公式 3.5 所示：

$$P \approx UV^{\mathrm{T}},\ U \in C^{m\times d},\ V \in C^{n\times d} \tag{3.5}$$

为了找到一个低秩矩阵来最大程度地逼近矩阵评分矩阵 P。需要最小化下面（公式 3.6）的损失 Frobenius 函数。

$$L(U,V)=\sum_{ij}(P_{ij}-U_{i.}V_{j.}^{\mathrm{T}})^2 \tag{3.6}$$

为了防止过拟合，给上述公式 3.6 加上正则化项，则上述公式 3.9 可改写如公式 3.7 所示：

$$L(U,V)=\sum_{ij}(P_{ij}-U_{i.}V_{j.}^{\mathrm{T}})^2+\lambda\left(\left\|U_{i.}\right\|_F^2+\left\|V_{j.}\right\|_F^2\right) \tag{3.7}$$

这样就把一个协同推荐的问题通过低秩假设成功转变成了一个优化问题。下面要进行的内容很显然：这个优化问题怎么求解。本文采用交替最小二乘法求解。ALS 的目标函数不是凸的，而且变量互相耦合在一起，所以并不容易求解。但如果把用户特征矩阵 U_{mj} 和项目特征矩阵 V_{ui} 固定其一，这个问题立刻变成了一个凸的可拆分的问题。现在固定 $V_{j.}$，对 $U_{i.}$求导，让其导数等于 0，可得到以下公式 3.8：

$$\frac{\partial L(U,V)}{\partial U_{i.}}=0 \tag{3.8}$$

得到下面求解 $U_{i.}$的公式 3.9：

$$U_{i.}=P_{i.}V_{ui}(V_{ui}^{\mathrm{T}}V_{ui}+\lambda n_{ui}I)^{-1},\quad i\in[1,m]. \tag{3.9}$$

同理，固定 $U_{i.}$，得到 $V_{j.}$的求解公式 3.10：

$$V_{j.}=P_{.j}^{T}U_{mj}(U_{mj}^{\mathrm{T}}U_{mj}+\lambda n_{mj}I)^{-1},\ j\in[1,n]. \tag{3.10}$$

之后进行迭代处理，因为每步迭代都会降低重构误差，并且误差是有下界

的，所以本算法一定会收敛。评分预测部分如下公式 3.11 所示：

$$\hat{p}_{ij} = \frac{(U_{i.}{}^{\mathrm{T}}V_{j.} - (U_{i.}{}^{\mathrm{T}}V_{j.})_{\min})}{(U_{i.}{}^{\mathrm{T}}V_{j.})_{\max} - (U_{i.}{}^{\mathrm{T}}V_{j.})_{\min}} \times s + t \tag{3.11}$$

在评分过程中，进行归一化操作，如果把评分映射到区间[0.5，5.0]，那么 s=4.5，t=0.5 可以取。通过归一化方法，克服了原始 ALS 方法并没有考虑将评分 $\hat{p}_{ij}$ 映射到预定的评分区间[r_{min}，r_{max}]的缺点，实验表明引入边界使其求解结果更加准确，由以上分析本文的 NALS-WR 的步骤如下：

1）首先用 0 均值，偏差为 0.01 的高斯随机数初始化矩阵 V；

2）然后通过目标函数 $U_{i.} = P_{i.}V_{ui}(V_{ui}{}^{\mathrm{T}}V_{ui} + \lambda n_{ui}I)^{-1}$ 求得 U；

3）同理再去求 V，不断地迭代下去；

4）直到算法计算出的 RMSE 值收敛或迭代次数足够多再结束迭代；

5）根据 $P=UV^{\mathrm{T}}$，返回矩阵 P；

6）对 P 进行归一化处理。

该算法的关键是反复迭代运用公式 $U_{i.} = P_{i.}V_{ui}(V_{ui}{}^{\mathrm{T}}V_{ui} + \lambda n_{ui}I)^{-1}$ 和 $V_{j.} = P_{.j}{}^{T}U_{mj}(U_{mj}{}^{\mathrm{T}}U_{mj} + \lambda n_{mj}I)^{-1}$ 更新 U 和 V，由于每次公式 3.9 和公式 3.10 更新计算的是矩阵 U 和 V 的一行值。因此能够进行并行运算，把矩阵 U、V 进行分割成几个子矩阵（等列长）来并行。现在采用 Spark 框架实现 NALS-WR 的并行化运算，并比较并行化之后的 NALS-WR 与 SVD 和 ALS 算法。

3.2.2 评分模型 NALS-WR 的并行化实现

由以上可知 NALS-WR 每步迭代中优化问题的目标函数可以拆分成互相独立的最小二乘子问题，故从计算的角度来看 NALS-WR 是适合分布式求解的。本文采用 Apache Spark 进行并行化设计。Spark 是一个开源的通用并行计算框架，将中间值缓存到内存中，基于内存进行集群计算，减少数据的读写次数。

本文借助 Spark 的弹性分布式数据集（Resilient Distributed Dataset，RDD）进行优化算法的执行。弹性分布式数据集的本质是可以并行的数据容器，不同的数据集格式拥有不同类型的 RDD。若一个弹性分布式数据集分片丢失，Spark 可以根据日志信息重它，所以 RDD 具有弹性。此外，可以用操作本地集合的方式来操作分布式数据集，所以 RDD 还具有分布式的性质。由于，RDD 可以被缓存在内存中，在之后的计算过程中可以直接从内存中读入，省去了大量的磁盘存取开销，正适合本文迭代计算的矩阵分解算法 NALS-WR。

设计本节的实验框架，如图 3.1 显示了本文 NALS-WR 的实验架构，共有 5 个节点的 Spark 执行器，每个执行器都有各自的 Executor 进程，整个 Application 的生命周期和每个进程的生命周期一致，而且各个进程内部维持着许多线程去执行（并行）被分配的 Task。由此实现了各个 Application 之间的调度隔离和资源隔离。

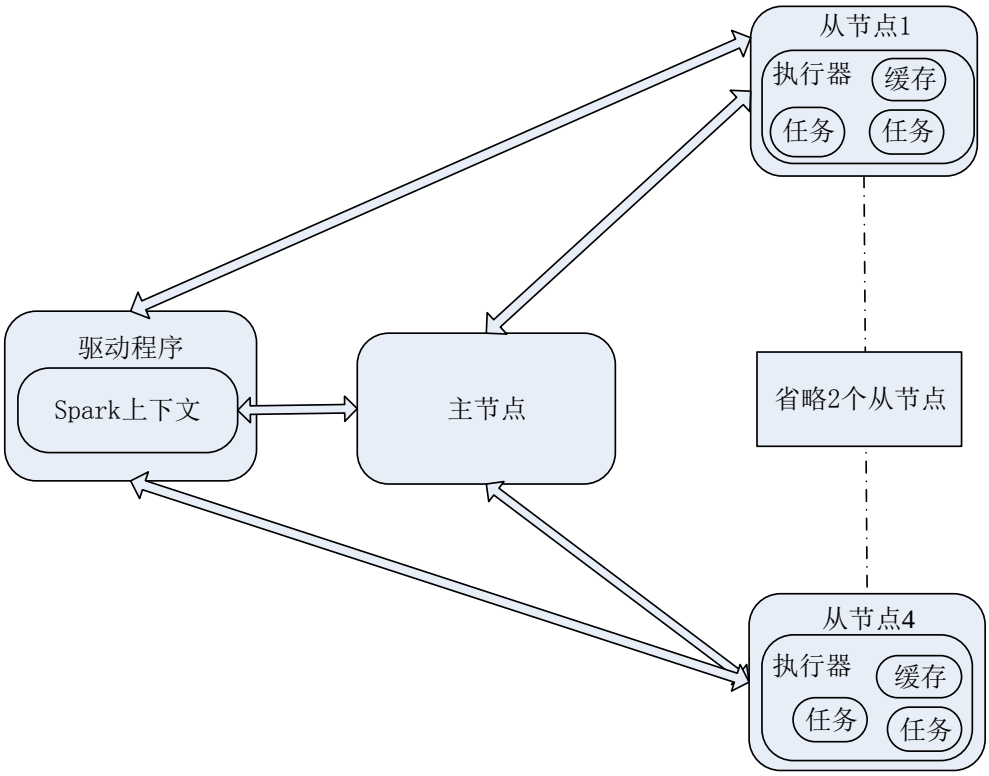

图 3.1 实验平台工作流程图

为了使 NALS-WR 在 Spark 上高效地实现，需要合理的设计 RDD 缓存和数据分区来减少数据交换。在同一个分区上计算向量V_1和V_2，之后把向量U_1和U_2一次发给这个分区，然后在计算向量V_1和V_2的时候在本机内存中直接读取向量U_1和U_2即可。这样就省掉了不必要的数据传输。下图 3.2 举例说明如何在分区的情况下通过向量U_i来求解V_j，来达到减少节点之间的数据交换量的目的。使用这种分区结构，需要在原始打分数据的基础上额外保存一些信息。在A_1模块，把向量U_1发给B_1和B_2，把向量U_2发给B_1。通过查看向量U_1和U_2相关联的所有项目来确定需要分别把向量U_1和U_2发给哪个计算节点，但每次迭代都扫一遍数据比较浪费时间。所以在 NALS-WR 实现中只计算一次这个信息，然后把结果通过 RDD 缓存起来重复使用。这部分数据本文在代码里称作 OutBlock。

构建最小二乘法问题的求解过程中，需要知道B_1模块的向量V_1和哪些用户向量有关联及其对应的评分值。这部分数据不仅包含原始评分数据，还包含从每个用户分区收到的向量排序信息，本文在代码里称作 InBlock。所以，需要通过用户的 OutBlock 信息把用户向量发给项目分区以便根据向量U_i求解V_j，然后通过项目的 InBlock 信息构建最小二乘法问题并求解。根据向量V_j求解U_i，需要项目的 OutBlock 信息和用户的 InBlock 信息。所有的 InBlock 和 OutBlock 信息在迭代过程中都通过 RDD 缓存。这样原始的评分数据其实在用户的 InBlock 和项目的 InBlock 各存了一份，但分区方式不同，这就避免了在迭代过程中对原始数据的交换所带来的 I/O 开销。下图 3.2 举例表示上述过程的分解过程：

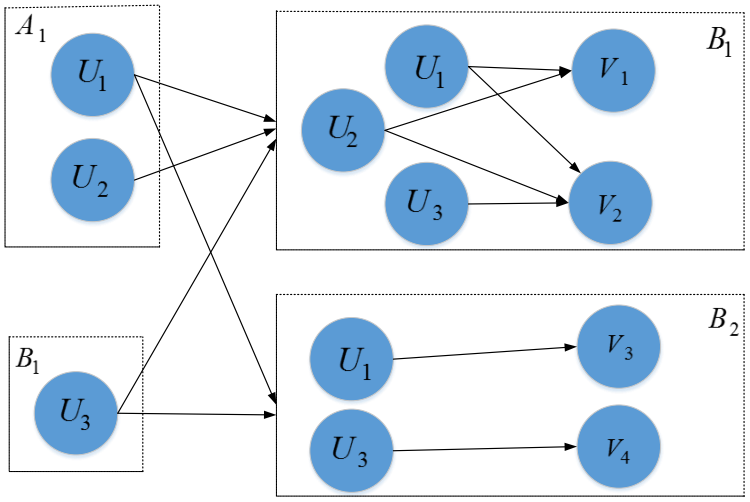

图 3.2 数据分区设计示意图

NALS-WR 通过能对其进行并行操作的 RDD 实现并行化。并行化时，在 RDD 各分区封装着指定的分片个数，之后进行并行数据的相互转化（map，union 和 filter 等）操作。通过设定 InBlock 和 OutBlock 的分区进行数据的并行处理。将 InBlock 和 OutBlock 信息缓存在内存中，避免重复计算和重复读写操作，然后以各自 OutBlock 信息初始化用户和项目的因子矩阵，其中需要将更新信息发给目标因子矩阵。最后本文用实验来表明并行算法 NALS-WR 的高效性及其准确率。

3.3 实验分析

本节介绍实验环境，数据集，之后分析本文设计的模型的性能，对比算法为经典的 ALS-WR 和 SVD，评价指标为均方根误差和运行时间。

3.3.1 实验环境及数据集

首先布署 5 个节点的 Spark 集群和 Hadoop 集群，采用本实验室的机器，其中 Master 节点配置为 55G 内存，12 核 CPU，Worker 节点的配置均为 4G 内存，4 核 CPU，实验数据由 Hadoop HDFS 文件系统存储。实验环境结构图如下图 3.3 所示，主节点和从节点均采用 Ubuntu 系统配置：

主节点 0	从节点 1	从节点 2	从节点 3	从节点 4
Ubuntu	Ubuntu	Ubuntu	Ubuntu	Ubuntu

图 3.3 实验环境示意图

本节采用 MovieLens 数据集，它是由美国 Minnesota 大学的 GroupLens 研究小组创建并维护的[57]。其中 MovieLens 20m 是本评分模型的重点实验对象。该数据集包含 20000263 个评分，27278 部电影和 138493 个用户。用户编号是

1-138493，电影编号是 1-131262。根据统计，单个用户最多参与了 9254 部电影的评分。而实际总共只有 26744 部电影评分数据显示，评分密度为 0.54%。

为了更好地比较 NALS-WR 并行化前后的运行效率，在 MovieLens 数据集运行该算法，对比 NALS-WR 算法和传统的矩阵分解算法 ALS、SVD 在各个数据稀疏度下的好坏。本章采用均方根误差 RMSE 作为推荐效果的评价指标，通过计算用户评分的预测值和实际评分值的偏差来衡量推荐的准确性[27]。如果 RMSE 越小，则算法的效果越好。假设数据集的项目有 N 个，NLAS-WR 预测得出的评分向量为 $\{p_1, p_2, p_3.....,p_n\}$，而用户评分集合实际为 $\{r_1, r_2, r_3.....,r_n\}$，那么 NALS-WR 算法的 RMSE 表示为公式 3.12 所示：

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(p_i - r_i)^2}{N}} \tag{3.12}$$

3.3.2 NALS-WR 和 ALS-WR 的 RMSE 对比分析

本节将 NALS-WR 模型和几个经典的协同过滤（CF）算法比较性能。参与比较的算法有经典的 SVD 和 ALS-WR。图 3.4 显示的是 NALS-WR 算法和 ALS-WR 算法在采用 RMSE 作为性能评价指标时的性能对比（数据集为 MovieLens）。各个算法均在训练集上训练，在测试集上测试得到 RMSE 值，迭代次数为 18 次。横轴表示特征矩阵的特征个数，纵轴表示 RMSE 值。横轴从 2 依次加 3 变化到 17，NALS-WR 和 ALS-WR 算法在特征数均为 8 时，RMSE 均取得最小值，此时训练的效果最好。特征数为 2、5 时，两个算法处于欠拟合状态，所以 RMSE 均比特征数为 8 的时小。特征数为 11、14、17 时，两个算法处于过拟合的状态中，当特征数为 17 时，过拟合现象最严重，训练的效果过差于特征数为 8 时。如图 3.4 所示，但无论特征数为多少，NALS-WR 的 RMSE 值一直低于 ALS-WR 的 RMSE 值，表示其性能要优于 ALS-WR。

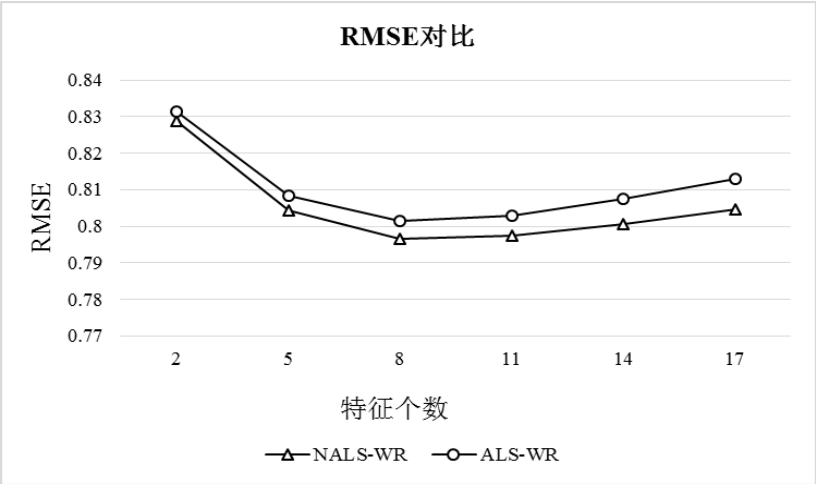

图 3.4 NALS-WR 和 ALS-WR 算法的 RMSE 比较示意图

3.3.3 NALS-WR 和 SVD 的 RMSE 对比分析

下面比较 NALS-WR 和 SVD 算法采用 RMSE 作为性能评价指标时的性能。

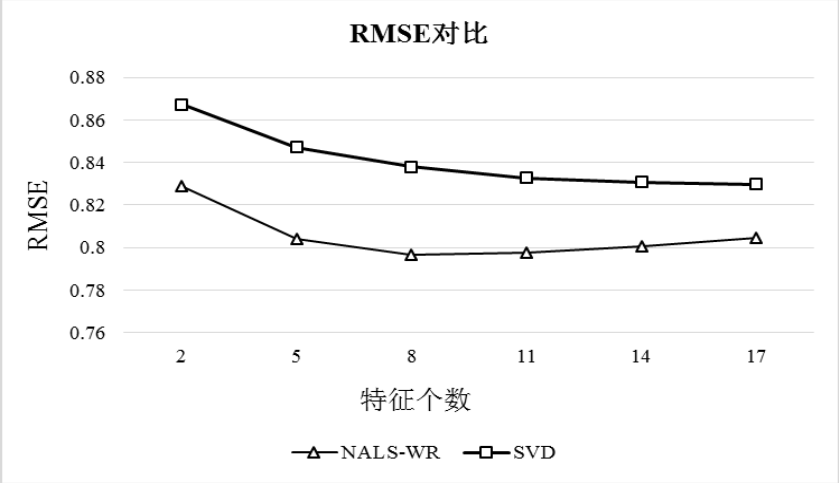

图 3.5 NALS-WR 和 SVD 算法的 RMSE 比较示意图

如上图 3.5 所示，各个算法在训练集上训练，在测试集上测试得到均方根误差值，迭代次数为 18 次。横轴表示特征矩阵的特征个数，从 2 依次加 3 变化到 17。纵轴表示的是均方根误差值。如图 NALS-WR 的均方根误差一直低于 SVD 的均方根误差，特征数为 8 时，NALS-WR 算法均方根误差最小，为 0.7966。当特征数小于 8 时，两个算法均出现欠拟合现象。当特征数大于 8 时，两个算法均出现过拟合现象。实验充分表明 NALS-WR 的性能也优越于 SVD。

3.3.4 NALS-WR 和 SVD、ALS-WR 的效率对比分析

本节用数据集 MovieLens-1m，10m 和 20m 来证明 NALS-WR 处理数据的速度。评价指标为时间（ms）。

表 3.2 三种算法（MovieLens-1m）的运行时间对比（迭代次数固定）

特征数	2	5	8	11	14	17
NALS-WR (Spark) 运行时间（ms）	18000	19640	20176	20203	21691	22719
SVD 运行时间（ms）	5034	6563	8382	10379	12191	14146
ALS-WR (Hadoop) 运行时间（ms）	474548	482602	485783	486582	487137	491830

表 3.3 三种算法（MovieLens-10m）的运行时间对比（迭代次数固定）

特征数	2	5	8	11	14	17
NALS-WR (Spark)运行时间（ms）	46161	55467	73185	75082	83838	99143
SVD 运行时间（ms）	50416	70196	88665	107783	127000	146134
ALS-WR (Hadoop)运行时间（ms）	619625	623885	626749	639887	660833	693666

表 3.4 三种算法在（MovieLens-20m）的运行时间对比（迭代次数固定）

特征数	2	5	8	11	14	17
NALS-WR (Spark)运行时间（ms）	72000	90000	102000	126000	150000	168000
SVD 运行时间（ms）	101889	140637	179521	217848	256726	295984
ALS-WR (Hadoop)运行时间（ms）	751387	769453	781549	800101	865819	937248

SVD 单机版串行，不存在多个节点的通信开销，在数据少的情况下比较快。NALS-WR 基于内存，并行分布式计算，存在多个节点的通信开销，故在数据少的情况下，可能比 SVD 耗时。ALS-WR 运行是采用 Hadoop 框架下的 Mapreduce 对数据进行处理。MapReduce 在每次执行时都要从磁盘读取数据，计算完毕后都要把数据存放到磁盘上，不断的迭代数据会消耗大量的 I/O 开销，比较耗时，如表 3.2 所示。

实验表明数据量增大，NALS-WR 的优势增强，如表 3.3 和表 3.4 所示。数据量很大的情况下，本算法执行速度会远远的超过传统单节点的 SVD 算法和基于磁盘的 ALS-WR 算法。该算法可以提高协同过滤推荐算法在大数据规模下的推荐效率，解决大数据情况下矩阵分解推荐算法时间代价过高的问题。

图 3.6 显示了 NALS-WR 和 ALS-WR、SVD 算法在采用运行时间（ms）作为执行效率评价指标时的时间对比。

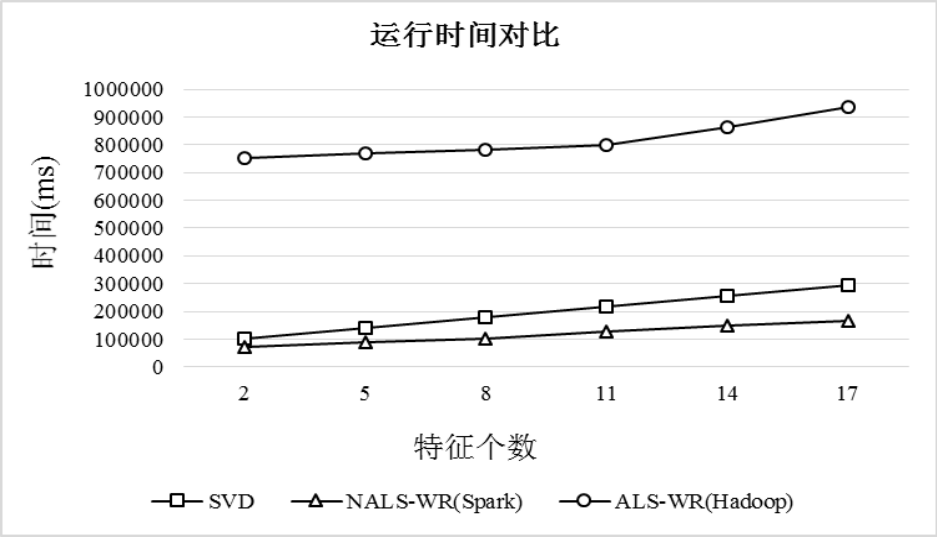

图 3.6 迭代次数为 18 时三种算法的运行时间对比（MovieLens-20m）

各个算法还是在训练集上训练，在测试集上测试得到执行时间值，迭代次数为 18 次。横轴表示特征矩阵的特征个数，从 2 依次加 3 变化到 17。纵轴表示运行时间（ms）值。从下往上，三根不同颜色的线条分别表示在不同特征个数下并行算法 NALS-WR 运行时间，SVD 串行运行时间，Hadoop 下 ALS-WR 运行的时间。

由于 Hadoop 下的 ALS-WR 运行是采用 Mapreduce 对数据进行处理的，每次从集群中读取数据，进行一次处理，将结果写到集群，从集群中读取更新后的数据，进行下一次的处理，再将结果写到集群。而 NALS-WR 会在内存中以接近“实时”的时间完成所有的数据分析：从集群中读取数据，完成所有必须的分析处理，将结果写回集群。实验表明，基于 Spark 的并行矩阵分解推荐算法 NALS-WR 运行时间远短于 Hadoop 下运行 ALS-WR 的时间，也短于 SVD 串行运行时间，NALS-WR 花费的时间随着特征数的增加缓慢的线性增加，NALS-WR 执行效率能带来最高的加速比。

3.4 小结

本章采用 ALS 矩阵分解优化算法建立本文的评分模型，并分析了其可扩展性、抗稀疏性和实现效率。实验表明，NASL-WR 算法无论是在可扩展性、稀疏性或效率，都更胜一筹，并且数据规模越大，效率越高。NASL-WR 提高了协同过滤推荐算法在大数据量的执行效率，并发挥了矩阵分解算法进行推荐系统的优点。从而解决了传统的基于矩阵分解的协同过滤算法难以并行化、可扩展性差的问题。由于评分模型 NASL-WR 的分布式计算框架和云计算环境，本文设计的模型可扩展性更强，后续还可来做实时处理（批处理过渡到实时处理，Hadoop 无法做到这一点）。在之后的工作中，本文将结合评论信息进行语义分析，建立融合评分和评论的模型。

第 4 章 融合评分和评论的 FRRM 模型

本章通过建立融合评分和评论的模型，弥补了传统方法只利用评分进行预测的缺陷。针对传统语义分析模型在推荐系统领域的缺陷，本章将语义分析模型 LDA 和第三章评分模型 NALS-WR 结合，利用评论信息加强评分预测，搭建机器学习聚类算法与协调过滤算法的混合框架，使其既适用于评分数据，也适用于评论信息，达到语义分析后进行推荐的效果，再进行并行化操作，提升推荐的效率，最后在大规模数据集上进行实验，发现实验结果在准确率、时间效率等指标上都取得了较好的效果。

4.1 模型建立的基础

主题模型中最经典的一种挖掘模型是 LDA。它是一种非监督的机器学习技术，可以用来识别大规模文档集或语料库中潜在隐藏的主题信息。该方法假设每个词都是由背后的一个潜在隐藏的主题中抽取出来的。对于语料库中的每篇文档，LDA 定义了以下生成过程[49]：

1）对每一篇文档，从主题分布中各抽取一个主题

2）从上述被抽到的主题所对应的词语分布中抽取一个单词

3）重复第一步骤和第二步骤，直至遍历文档中的每个单词

所以一篇文档中，每个单词出现的概率可以表示为公式 4.1 所示：

$$p(word \mid doc)=\sum_{topic} p(word \mid topic)p(topic \mid doc) \tag{4.1}$$

以上公式表明，经过 LDA 模型的训练之后，得到两个分布，即文档在主题上的概率分布和主题在单词上的概率分布。当本文把每条评论当作是一篇文档之后，这个概率公式可以用矩阵表示，具体如下图 4.1 所示[49]：

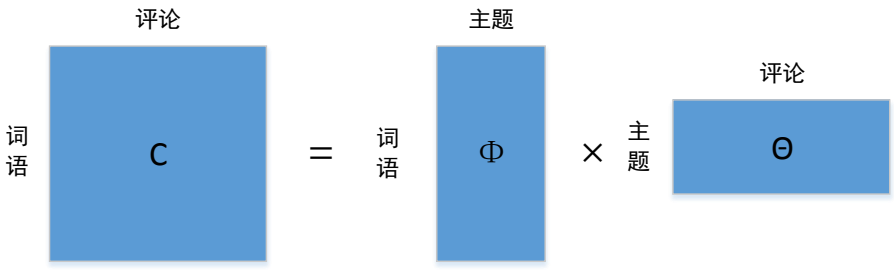

图 4.1 文档、主题、词语关系图

其中“评论-词语”矩阵表示一条评论中一个词语的出现的概率，“主题-词语”矩阵表示一个主题中一个词语出现的概率，“评论-主题”矩阵表示一条评论中一

个主题出现的概率。通过对每条评论分词，计算每条评论中每个词语的词频，得到左边“评论-词语”矩阵。通过左边“评论-词语”这个矩阵进行训练，学习出右边“主题-词语”矩阵和“评论-主题”矩阵，这便是主题模型 LDA。其主要目标是自动发现一个文档集合中的主题。这些文档本身是可以观测到的，而主题的结构—主题、每个文档的主题分布和每个文档的每个词的话题赋值—是隐藏的。主题建模的核心计算问题就是使用观测到的文档来推断隐藏主题结构。LDA 模型的生成过程可以用概率图描述，如图 4.2 所示[49]：

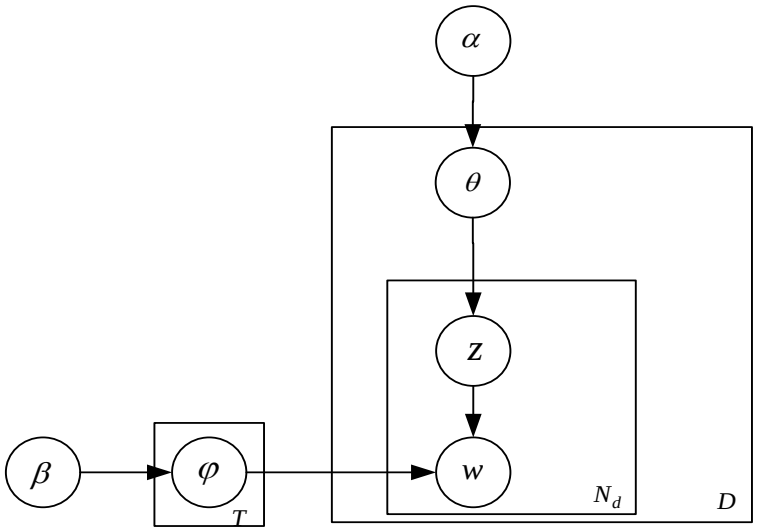

图 4.2 LDA 的概率图模型

其中 w 圆圈表示文档中的单词，其他圆圈表示参数或隐变量，黑色箭头表示两个变量之间的相互依赖关系。假设整个语料库中有 D 篇文档，每篇文档有 T 个主题。α、β 为狄利克雷分布的超参数，α 用于生成一篇文档的主题向量表示为 θ，β 用于生成各个主题对应的单词概率分布 φ。对于每篇文档中的单词，首先从文档—主题分布 θ 中抽取一个主题 z，之后从主题—单词分布中抽取一个单词，如此重复 N 次（文档 d 中的单词总数），可以生成一个完整的文档 d。运用 LDA 主题模型生成一篇文档时的过程如下表 4.1 所示[49]：

表 4.1 LDA 主题模型建模过程

LDA 文档建模过程：
选择文档 d 中的单词个数 N，N 服从泊松分布
采样向量 θ，θ 的每个元素代表每个主题在 d 中的概率，θ 服从 Dirichlet(α)
采样 φ，φ 服从 Dirichlet(β)分布
对文档中的每个单词 w
选择隐藏主题 z，z 服从 Multinomial(θ)多项分布
从多项式分布 φ 中抽取 w，w 服从 Multinomial(w)

借助 LDA 算法可以得到主题的结构，需要指出的是，算法本身并不需要用到这些主题的信息，文档本身也没有使用主题或者关键字进行标注。这个隐藏结构最有可能产生现在可以观测到的文档集合。主题模型方便之处是可以通过推断的隐藏结构来组成文档的主题结构。这样的信息对于语料研究提供了有力的支

撑，更适合于本文的研究。

4.2 评分模型和语义分析算法的融合

由于推荐系统中有大量的评论信息，利用这些评论内容进行语义分析可以提高推荐系统的质量。本文把一条评论看作是一篇文档。传统语义分析是计算文档之间的相似性，这往往是基于两个文档中同时出现的词语的个数进行的，但是在一些情况下，采用这种方法得到的文档相似性并不准确，因为这种方法没有考虑词语背后的语义关联[58]。由于不同词语可能表达了相同的语义，因此比较两篇文档的相似性不再是简单的比较两篇文档中同时出现的词语个数，而是比较它们所隐含的潜在语义之间的联系。本文采用 LDA 主题模型这种有效的语义分析方法，通过挖掘评论信息所隐含的潜在主题之间的相似性来衡量评论之间的实用价值，进行融合评分和评论模型的设计。

首先，定义每条评论作为一个文档文件 $g_{i,j}$（i 代表用户，j 代表项目），文档 g_j 为一项目 j 的所有评论的集合，g_i 为一用户 i 的所有评论的集合。对于每个项目 j 学习成一个主题分布 ϑ_j，ϑ_j 是一个随机向量，如果物品项目评分因子的数量与评论主题的数量相同，ϑ_j 可编码在该产品 j 中所有评论中涉及到的主题中的每一个主题。对于每个用户 i 学习成一个主题分布 ϑ_i，ϑ_i 也是一个随机向量，如果用户评分因子的数量与评论主题的数量相同，ϑ_i 可编码在该用户 i 中所有评论中涉及到的主题中的每一个主题。

为了增加效率和准确率，本文使用户的评分参数 $U_{i.}$ 和每个用户的评论参数 ϑ_i 对应起来，一个高的评分将对应一个活跃的话题。所以本文提出了一个单调变换，使评分参数的最大值 $U_{i.}$ 和评论参数 ϑ_i 的最大值对应起来，公式如下 4.2 所示：

$$\vartheta_{i,\lambda} = \frac{\lambda U_{i,\lambda}}{\sum_{\lambda'} \lambda U_{i,\lambda'}} \times a + b \tag{4.2}$$

$\vartheta_{i,\lambda}$ 经过运算可能会有负数，所以文本采用了归一化处理，系数 a，b 可以使 $\vartheta_{i,\lambda}$ 的值映射到相应的区间，参数 λ 可以控制上述公式转换的峰值，使转化出的结果尽可能地接近于真实值。可增大 λ，λ 越大代表着用户讨论的话题越重要，λ 越小代表着用户讨论的话题越一般。

由于公式 3.5 的因子能准确地建模用户的评分，但是，这些因子应该可以转化为主题，方便本文联合利用评分数据和评论信息。为了实现这些，现增加公式 3.7 的功能，定义语料库 ϖ 的主题（评分和评论）为公式 4.3：

$$L(\varpi \mid Q, F, \lambda, z) = \sum_{ij} (P_{ij} - U_{i.} V_{j.}^{\mathrm{T}})^2 + of(\varpi \mid \vartheta, \phi, \mathrm{z}) \tag{4.3}$$

其中 Q 和 F 分别是评分参数，话题参数，参数 λ 控制变化，使其更加合理。

等号右边的求和平方是公式 3.7 中的评分误差。参数 o 可以使评分误差和 $of(\varpi|\vartheta,\phi,\mathrm{z})$（评论语料库的概率的表达式）的关系得到权衡，z 表示主题模型中每个单词的主题分配，即使最终目标仅仅是预测评分。$of(\varpi|\vartheta,\phi,\mathrm{z})$ 对本章的工作有很大帮助，其可以作为正则化参数，防止过拟合。

公式 3.7 的缺陷是当评分数据很少时，正则化参数会使用户对物品的喜好因子和物品属于该类中的概率致为 0，公式 4.3 克服了这种缺陷。并且在没有评论的情况下，也可以预测评分。

由于最终目标是优化模型，因此要找到损失函数 $L(\varpi|Q,F,\lambda,z)$ 的最小值，本章设计一个通过最小交叉二乘法和吉布斯采样法的迭代交替的程序来训练模型。第一步的公式如下 4.4 所示：

$$\text{NALS-WR}(Q^{(t)},F^{(t)},\lambda^{(t)})=\arg\min L(\varpi|Q,F,\lambda,z^{(t-1)}) \tag{4.4}$$

公式 4.4 是采用 NALS-WR 更新评分部分，主题模型中每个单词的主题分配 z 是固定的，通过 NALS-WR 方法来求出 Q, F和λ。

对于评论部分，LDA 将每个文档 $d\in D$ 与 K 维主题分布 ϑ_d 相关联，运用 LDA 可以得出文档 d 中的单词讨论的主题 k 的概率 $\vartheta_{d,k}$。每个主题 k 还具有相关联的词分布 ϕ_k，ϕ_k 编码一主题中特殊词的概率。类似 LDA，现把每个单词随机分配给一个主题（1 和 k 之间的整数），与单词对应的话题的概率成比例，由于每个用户 i 有 k 维的主题分布 ϑ_i。假设 $z_{i,j,t}=k$，那么对每一个字 $\omega_{i,j,t}$（用户 i 对项目 j 的评论）的概率比例是用户 i 的主题 k 的概率 $\vartheta_{i,\mathrm{k}}$，乘以话题 k 的特定词 $\omega_{i,k,t}$的概率 $\phi_{k,\omega_{i,k,t}}$，即 $\vartheta_{i,\mathrm{k}}\phi_{k,\omega_{i,k,t}}$。

遍历所有文档 d 和所有的单词位置 j，以及更新用户的主题作业，根据更新的参数计算两个分布向量 ϑ 与 ϕ，对每个词的主题分配进行重新采样，如下公式 4.5 所示：

$$\text{sample } \mathrm{z}_{d,t}{}^{(t)}\text{chance p }(\mathrm{z}_{d,t}{}^{(t)}=k)=\phi_{k,\omega_{d,t}}{}^{(t)} \tag{4.5}$$

公式中 sample 是取样的意思，为了保证（话题 k 的词分布）ϕ_k 是随机变量，本章假设附加变量为ψ，并有以下定义（公式 4.6）：

$$\phi_{k,w}=\frac{\psi_{k,w}}{\sum_{w'}\psi_{k,w'}}\times s+t \tag{4.6}$$

正如公式 4.2 一样，引入参数 s 和 t，引入变量ψ_k，进行归一化处理使$\phi_{k,w}$映射到相应的区间。方程 4.4 和 4.5 不断迭代，直到收敛，与 LDA 更新主题作业不同的地方是，主题概率ϑ不从狄利克雷分布取样，相反是根据之前公式 4.2 的计算出的θ值决定的，θ被方程 4.4 更新后，取样新的主题作业，通过多次交叉迭代，Q和F会趋于稳定值，最终可以根据评论内容预测评分。

4.3 FRRM 模型的并行化实现

由于 FRRM 模型的运行时间复杂度比较高，当训练文本语料数据或主题数过多时，很有必要进行并行化设计，而单台机器内存有限，FRRM 模型的字和主题数量过多时，所以基于 Spark 将数据分成若干个部分，各个部分由一个独立的处理机来并行计算，通过并行计算集群完成对数据的处理。

首先将原始训练数据进行切分后分散到各个机器上执行，不同的是进一步考虑了列切分，由于主题模型的采样中可以不考虑同一条评论中单词出现的顺序，所以只要没有统计量更新依赖，就可以把文章像切蛋糕一样切分开并行执行。这个算法的关键思想在于训练语料文本矩阵中同一行（同一条评论）或同一列（同一个单词）的 word 都会产生统计量更新依赖，所以同一行或同一列的数据不能被同时采样。先将原始训练评论转换为 wordid：

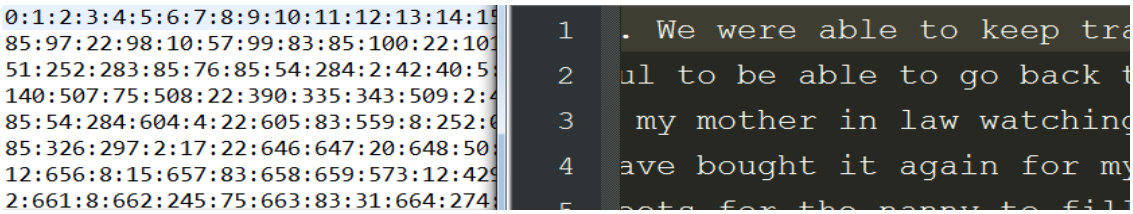

图 4.3 Wordid 和原始训练评论（wordid 以：分割，评论单词以空格分割）

获得每条评论 wordid 的原训练语料矩阵，即将原始语料转换为矩阵，同一行和同一列的 wordid 在采样时会产生统计量依赖。若一条评论没有这个词则被对齐留空，在此基础上进行并行化设计。

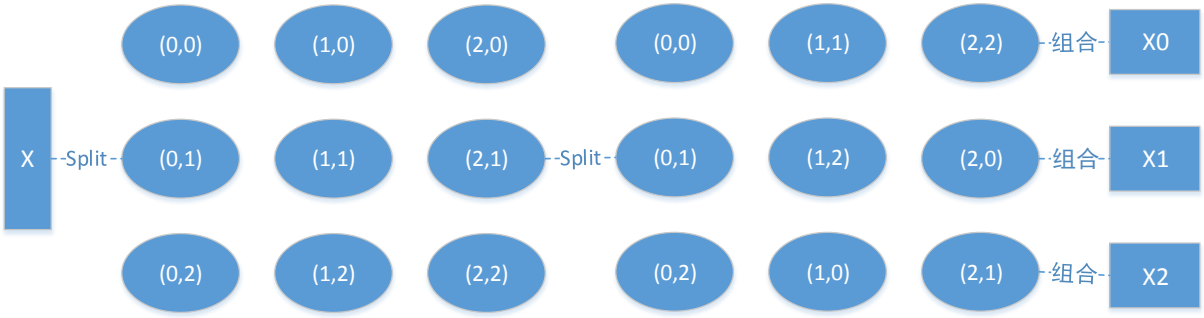

图 4.4 Spark-FRRM 切分流程（并发度 3）

整个并行化过程是：原始数据 X 划分为 $P \times P$ 份数，这里 P 是设置的并发数，如图 4.4 所示，P=3，每一份数据都相应地标注了坐标，注意到由于有的行列不可以被同时执行，因此图 4.5 的执行分为几个阶段执行，对角线上的（0，0）、（1，1）、（2，2）这 3 块数据先被同时执行，而每一块内部执行单机版的 Gibbs Sampling，这三块并行执行结束后，再执行另外不在十字线上的 3 块（0，1）、（1，2）、（2，0），这三块执行结束后最后执行（0，2）、（1，0）、（2，1）三块，如下图 4.5 所示，三行行内并行计算，行间串行计算。

切分策略如下：

（1）首先将一数据集 X 均等切分（按照列），得到 P 份，如果它们包含的

词语数目不均等，为了保证均等，就要随机交换列再进行切分。

（2）在第一步基础上，将 P 份列，切成 P 份行，如果不等，就多次随机交换行，在各个块之间差距的最大值中，挑选出最小的那一次。

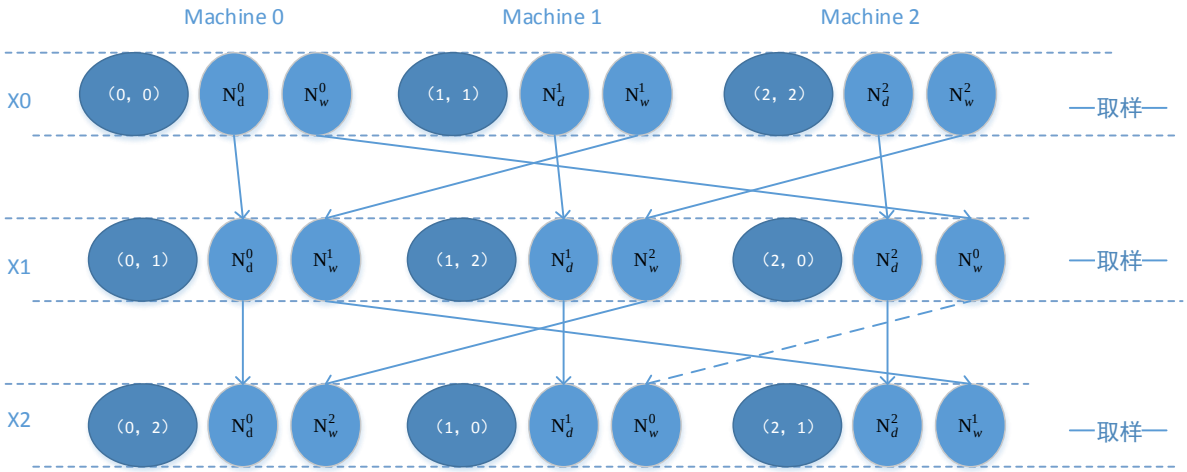

图 4.5 同一个单词的 *nw* 和同一条评论的 *nd* 统计量被合并的过程

切分成 $P \times P$ 块完成后，挑选出哪些无冲突的 P 个块可以同时并发执行（组内并行，组件串行）。切分之后进行计算和合并，由于上述算法过程中的方法是：组内并行，组件串行，因此小组并行执行的一次迭代完毕后，小组的 *nd*（评论中主题个数）、*nw*（评论中单词个数）等统计量需要同步到下一个小组内。而组内各个块内采样计算过程与单机版的吉布斯采样完全一致（将统计量切换）：*nd*、*nw* 和 *ndsum* 切分成 P 份，例如第 j 份是 *nd_j*（*nd* 的第 j 部分）和 *nw_j*（*nw* 的第 j 部分），由于 *nwsum* 总会冲突，所以这个统计量决定整个算法的误差。因此 *nwsum*（主题 k 中被指定的单词总数）被修改为第 P 台机器上的 *nwsum_p* 后（并行计算完结后）都需要全局更新合并。除了 *nwsum* 需要合并之外，其余几个统计量，*nd_j* 和 *nw_j* 都需要合并到下个小组的评论 d 和单词 w（这种合并过程可以用 Spark 的广播变量）。计算后合并，同一个单词的 *nw* 和同一条评论的 *nd* 统计量被合并的过程示意图如上图 4.5 所示：

实验证明，这种算法产生的混淆度与单机版的吉布斯采样一致，因此误差很小，而且分布式计算中的内存可以接受，但是实现复杂度稍高，总结如下 6 个步骤：

1）整个训练评论集不仅以行切分（行指评论），也以列切分；

2）各个小块被切分尽可能数据量均等；

3）统计量 *nw* 和 *nd* 因此也被切分到各个机器进程上；

4）执行流程为：每轮迭代组内并行，组间串行；

5）同一行或同一列的数据不能在一个小组内的不同块上同时并行执行；

6）各个独立进程运行完毕后，误差只在 *nwsum* 上面，通过全局更新合并 *nwsum*。

由于以上算法会消耗更多的计算资源，因此减少数据块大小来增加分区个数。增大 Spark 分片数，使处理任务时同时使用的处理器核数增多。然后根据 hdfs 中 block 的分块情况或者自定义分片数，将数据划分为若干个部分，各个部分在从节点中并行执行。采用 Spark 的广播变量，对集群中的每隔一个进程发送一个副本，并且在内存中也保存一个副本。进程中任务数目较多时，使用广播变量可节省网络开销。调用缓存 cache 指令 Spark 框架在 RDD 计算好之后将其暂时存储在集群的内存中，避免每次使用 RDD 都要从原始数据中重新计算，通过分布式并行计算完成数据处理。

4.4 实验分析

本节搭建实验环境，选取数据集并进行预处理，验证模型 FRRM 的效果，对比模型为经典的 HFT[59]和 RMR[60]模型。

4.4.1 实验环境及数据集

首先部署 5 个节点的 Spark 集群和 Hadoop 集群，采用本实验室的机器，其中 Master 节点配置为 55G 内存，12 核 CPU，Worker 节点的配置均为 4G 内存，4 核 CPU，实验数据由 Hadoop HDFS 文件系统存储，实验环境和第三章完全一样。

数据集 Amazon product data（http://jmcauley.ucsd.edu/data/amazon/），此数据集包含 1996 年 5 月至 2014 年 7 月的 14280 万条评论，包含亚马逊的商品评论（文字，有用的投票），评分数据，亚马逊的产品元数据（描述，类别信息，价格，品牌和图像功能）和链接。本章实验是从 Amazon product data 选择的，包含评论信息的 json 格式数据，其涉及的种类有 24 种，分别是：Books，Electronics，Movies and TV，CDs and Vinyl，Clothing，Shoes and Jewelry，Home and Kitchen，Kindle Store，Sports and Outdoors，Cell Phones and Accessories 等。

4.4.2 数据预处理

首先将原始的 json 数据格式，转换为 arff 格式。原始数据格式是 json 格式，包含的字段 reviewerID，asin，helpful，reviewText，overall，summary，unixReviewTime，reviewTime。实验提取的有用字段是 reviewerID，asin，reviewText，overall，分别是用户 ID，项目 ID，评论内容，评分。现转换为 arff 格式的数据，第一列表示用户 ID，第二列表示项目 ID，第三列表示评分分数（小数点后取一位），每列之间用逗号间隔，所有的评论词语用数字 1、2、3、4、5 等来标识，分号间隔评论词语，arff 格式如图 4.6 所示：

```
@RELATION user-music

@ATTRIBUTE user NUMERIC
@ATTRIBUTE item NUMERIC
@ATTRIBUTE rating NUMERIC
@ATTRIBUTE review STRING

@DATA
0,0,3.0,0:1:2:3:4:5:6:7:8:9:10:11:12:13:14:15:16:17:3:4:
1,0,5.0,31:32:33:13:34:0:11:35:4:21:36:37:16:38:39:40:41
2,0,5.0,62:63:11:56:12:64:65:66:33:67:21:68:2:56:1:69:21
3,0,5.0,33:91:92:93:94:95:9:11:96:97:24:97:98:43:21:99:2
4,0,5.0,111:82:112:80:113:42:1:2:1:96:114:115:116:117:10
5,0,5.0,11:21:186:3:28:24:12:185:35:16:13:73:33:187:188:
6,0,5.0,62:193:33:185:34:194:195:12:196:24:43:21:197:9:1
7,0,5.0,11:35:21:210:24:201:13:73:207:211:1:2:212:70:13:
8,0,5.0,207:207:207:11:35:62:225:137:226:80:227:16:33:22
9,0,5.0,100:28:76:237:42:46:9:238:212:239:240:241:242:1:
```

图 4.6 处理后的 arff 数据集示意图

当数据列均大于四，arff 数据格式存储数据方便执行程序。最上面的一行代表数据集的名字，user-music 是 Amazon 数据集中的一个类别。之后下面的几行分别表示当前数据集的列名（用户、项目、评分和评论）和数据类型（数值类型和字符串类型）。最后是将用户 ID，项目 ID 和评论内容的单词转换为数字的形式。

统计数据集，统计其用户数（users）、项目数（items）、总评分数（ratings）、总词数（words）和评分的平均值（avg.rating）。如表 4.2 所示：

表 4.2 数据集信息统计表

数据集	users	items	ratings	words	avg.rating
Patio_Lawn_and_Garden	1686	962	13272	29173	4.1865
Musical_Instruments	1429	900	10261	20599	4.4887
Automotive	2928	1835	20473	27861	4.4718
Amazon_Instant_Video	4520	906	18460	41320	4.3189
Apps_for_Android	12137	157	15018	17323	4.0668
Baby	8728	547	15994	20852	4.1414
Beauty	10277	1150	17451	23871	4.2054
Books	16601	368	19544	52347	4.2352
CDs_and_Vinyl	10462	990	16674	56554	4.4339
Cell_Phones_and_Accessories	10337	1092	17248	27171	3.9814
Clothing_Shoes_and_Jewelry	12458	967	17339	20332	4.3043
Digital_Music	3631	890	17071	55094	4.4817
Electronics	16304	765	19931	34726	4.2928
Grocery_and_Gourmet_Food	9081	1322	19748	28506	4.3348
Health_and_Personal_Care	12943	945	17810	29551	4.3175
Home_and_Kitchen	13743	599	18712	25953	4.3931
Kindle_Store	10553	1465	18549	37548	4.0755
Movies_and_TV	12891	296	18712	57735	4.2707
Office_Products	4608	1133	18627	29621	4.4301
Pet_Supplies	10019	731	17245	23719	4.3050
Sports_and_Outdoors	11347	862	15508	26274	4.4253
Tools_and_Home_Improvement	7909	1265	16341	27564	4.3943
Toys_and_Games	8731	990	16956	25590	4.3839
Video_Games	5455	1229	18923	54538	4.2185

4.4.3 对比依据

本节说明实验选用同类经典算法 HFT、RMR 对比的依据。HFT 算法是斯坦福大学的 Julian McAuley 设计的，其将评分模型与评论模型相结合，评分模型选用 BiasedMF 算法，主题模型选用 LDA，将每个 item 所有的评论表示为一个 document。如何将 rating model 与 topic model 关联在一起，是构建模型的关键问题。HFT 使用的关联方式是，定义一个转换函数将 BiasedMF 中的项目特征向量 γ 与 LDA 中每个 document 对应的主题分布 θ 联系在一起（HFT 的另一种实现方式是将用户特征向量与主题分布关联）。这种关联方式正体现了 HFT 的基本思想是“Hidden Vectors as Topics”[59]。

更具体来说，HFT 的项目特征向量与此项目对应的主题分布中各项具有关联关系，且项目的某种属性程度越高，此属性对应的主题在评论中出现的概率也越大，即特征向量中某一项的值越大，其在主题分布中对应项的值也越大。为满足 HFT 的基本思想，其转换函数需要满足以下两点要求[59]：

1）允许 γ 中各项可取任意实数值的同时，保证 θ 是一个概率分布向量，即 θ 中各项非负且小于等于 1，各项累加和为 1；

2）保证 γ 与 θ 各对应项在各自向量中的大小位序相同。例如 γ 中最大值项对应的 topic，在 θ 中也具有最大值。

定义一转换函数后，HFT 设计目标函数，由评分模型 BiasedMF 和 topic model 两部分的目标函数构成，两种模型结合成一整体模型，最优化过程交替的完成这两个训练过程，BiasedMF 采用梯度下降法求解，主题模型采用吉布斯采样法求解，HFT 模型可以很准确的预测评分[59]。

另一具有代表性的算法是 RMR，其利用评分数据和评论信息，联合基于内容的推荐算法和协同过滤算法提高预测精度。在数据集非常稀疏的情况下，与基本推荐算法相比，RMR 达到了只有评分模型不能做出的效果，特别是在数据及其稀疏时，开发吉布斯采样器来学习模型参数，利用先验知识来缓解冷启动问题，取得了很好的效果[60]。

本节选用融合评分和评论中经典有代表性的算法 HFT 和 RMR 作对比。

4.4.4 FRRM 和 HFT 算法的 RMSE 对比分析

本章实验采用 RMSE，MSE，MAE 作为评价标准。根据公式 3.15，采用 RMSE 计算 FRRM 模型预测的用户评分与实际的用户评分之间的偏差来度量预测的准确性[30]。

本文设置主题因子 K=10，迭代次数为 2，比较 22 个亚马逊产品类别（不同类别对应不同数据集）的 RMSE 结果，FRRM 模型的均方根误差在 22 类数据集中均比同类模型 HFT[59]的均方根误差小。在相同的数据集，相同的实验环境，

相同的迭代次数情况下，计算 FRRM 模型和 HFT 模型的均方根误差。对于每个数据集来说，模型的均方根误差值越少，该模型预测的准确率就越高。

实验结果如下表 4.2 所示，在数据集为 Patio_Lawn_and_Garden 时，模型 FRRM 的均方根误差比 HFT 的均方根误差降低了 5.5%，在数据集为 Musical_Instruments 时，模型 FRRM 的均方根误差比 HFT 的均方根误差降低了 8.5%，在数据集为 Automotive 时，模型 FRRM 的均方根误差比 HFT 的均方根误差降低了 5.9%。当数据集是 Amazon_Instant_Video 时，模型 FRRM 的均方根误差比 HFT 的均方根误差降低了 5.9%，当数据集是 Apps_for_Android 时，模型 FRRM 的均方根误差比 HFT 的均方根误差降低了 3.1%。在这些数据集中，模型 FRRM 提高倍数和数据稀疏性有很大关系。当数据集为 Grocery_and_Gourmet_Food 时，用户评分矩阵较稀疏，本文的抗稀疏性比模型 HFT 强，本模型的 RMSE 降低最多，为 14.8%。

这些实际数据集的实验证明了 FRRM 的预测准确率确实提高了，结果如表 4.3 所示：

表 4.3 FRRM 和 HFT 算法的 RMSE 比较

数据集	RMSE(FRRM)	RMSE(HFT)	对比(↓)
Patio_Lawn_and_Garden	1.1066	1.170641	5.5%
Musical_Instruments	0.8577	0.937177	8.5%
Automotive	0.9726	1.033392	5.9%
Amazon_Instant_Video	1.0411	1.105893	5.9%
Apps_for_Android	1.2688	1.309275	3.1%
Baby	1.2081	1.260912	4.2%
Beauty	1.2326	1.321968	6.8%
Books	1.1347	1.21573	6.7%
CDs_and_Vinyl	0.9753	1.027911	5.1%
Cell_Phones_and_Accessories	1.3483	1.349111	0.6%
Clothing_Shoes_and_Jewelry	1.0597	1.196495	11.4%
Digital_Music	0.8674	0.927631	6.5%
Grocery_and_Gourmet_Food	1.1046	1.252637	14.8%
Health_and_Personal_Care	1.2206	1.238184	1.8%
Home_and_Kitchen	1.0857	1.104038	1.8%
Kindle_Store	1.0953	1.105441	1.0%
Movies_and_TV	1.0819	1.150782	6.9%
Office_Products	0.9233	0.987877	6.5%
Pet_Supplies	1.1474	1.243141	9.6%
Tools_and_Home_Improvement	1.0086	1.108287	10.0%
Toys_and_Games	1.0235	1.079444	5.6%
Video_Games	1.1403	1.190042	5.0%

4.4.5 FRRM 和 HFT 算法的 MAE 对比分析

为了进一步验证融合评分和评论的模型 FRRM 的准确率，现采用平均绝对误差 MAE 作为评价指标，假设算法对 N 个项目预测的评分向量为 $\{p_1,p_2,p_3,.....,p_n\}$，对应的实际用户评分集合为 $\{r_1,r_2,r_3.....,r_n\}$，则平均绝对误差

MAE 评价指标如公式 4.7 所示[29]：

$$MAE=\frac{\sum_{i=1}^{N}|p_i-r_i|}{N} \tag{4.7}$$

同样主题因子 K=10，迭代 2 次，计算 24 个亚马逊产品类别的 MAE。如表 4.4 所示，部分 FRRM 的 MAE 值比同类算法 HFT[59]的小，MAE 的值越小，推荐准确性越高。缺陷是本模型对于每个数据集的 MAE 值并不是一直比 HFT 的小，但 MAE 评价指标没有 RMSE 权威，本模型 FRRM 有可取之处。

表 4.4 FRRM 和 HFT 算法的 MAE 比较

数据集	MAE(FRRM)	MAE(HFT)
Patio_Lawn_and_Garden	0.8747	0.8598
Musical_Instruments	0.6740	0.6867
Automotive	0.7361	0.7210
Amazon_Instant_Video	0.8201	0.7870
Apps_for_Android	1.0041	1.0107
Baby	0.9633	0.9764
Beauty	0.9838	0.9722
Books	0.8940	0.9328
CDs_and_Vinyl	0.7524	0.7378
Cell_Phones_and_Accessories	1.0789	1.0479
Clothing_Shoes_and_Jewelry	0.8240	0.8793
Digital_Music	0.6835	0.6531
Electronics	0.8896	0.8327
Grocery_and_Gourmet_Food	0.8647	0.9106
Health_and_Personal_Care	0.9165	0.9076
Home_and_Kitchen	0.8274	0.7950
Kindle_Store	0.8384	0.8362
Movies_and_TV	0.8552	0.8477
Office_Products	0.7219	0.7181
Pet_Supplies	0.9011	0.9068
Sports_and_Outdoors	0.8786	0.7665
Tools_and_Home_Improvement	0.7771	0.7728
Toys_and_Games	0.7998	0.7809
Video_Games	0.9037	0.8682

4.4.6 FRRM 和 HFT 算法的 MSE 对比分析

最后采用 MSE 衡量模型 FRRM 的评分和用户实际评分的贴近程度。对每个绝对误差首先做平方运算，假设模型对 N 个项目预测的评分向量为 $\{p_1,p_2,p_3.....,p_n\}$，对应的用户实际评分集合为 $\{r_1,r_2,r_3.....,r_n\}$，则 FRRM 模型的 MSE 表示如公式 4.7 所示：

$$MSE=\frac{\sum_{i=1}^{N}(p_i-r_i)^2}{N} \tag{4.7}$$

数据集采用以上 Amazon 的 23 种类别，迭代 2 次，主题因子 K=10，计算

FRRM 和经典模型 HFT[59]的 MSE 值，结果如表 4.5 所示：

表 4.5 FRRM 和 HFT 算法的 MSE 比较

数据集	MSE(FRRM)	MSE(HFT)	对比(↓)
Patio_Lawn_and_Garden	1.2246	1.3704	10.6%
Musical_Instruments	0.7357	0.8783	16.2%
Automotive	0.9459	1.0679	11.4%
Amazon_Instant_Video	1.0839	1.223	11.3%
Apps_for_Android	1.6099	1.7142	6.1%
Baby	1.4594	1.5899	8.2%
Beauty	1.5192	1.7476	13.1%
Books	1.2876	1.478	12.9%
CDs_and_Vinyl	0.9511	1.0566	10.0%
Cell_Phones_and_Accessories	1.8179	1.8201	0.1%
Clothing_Shoes_and_Jewelry	1.1231	1.4316	21.5%
Digital_Music	0.7524	0.8605	12.6%
Electronics	1.3662	1.2507	9.2%
Grocery_and_Gourmet_Food	1.2203	1.5691	22.2%
Health_and_Personal_Care	1.4898	1.5331	2.8%
Home_and_Kitchen	1.1788	1.2189	3.3%
Kindle_Store	1.1997	1.222	1.8%
Movies_and_TV	1.1704	1.3243	11.6%
Office_Products	0.8525	0.9759	12.6%
Pet_Supplies	1.3165	1.5454	14.8%
Tools_and_Home_Improvement	1.0173	1.2283	17.2%
Toys_and_Games	1.0475	1.1652	10.1%
Video_Games	1.3003	1.4162	8.2%

表 4.5 里的各大数据集中，FRRM 模型的 MSE 值也小于 HFT 的 MSE 值，证明 FRRM 的性能比 HFT 强。

4.4.7 FRRM 和 RMR 算法的 MSE 对比分析

现对比 FRRM 和 RMR[60]的均方误差 MSE，选取数据集 Toys_and_Games 、Tools_and_Home_Improvement 、Pet_Supplies 、Sports_and_Outdoors、Office_Products、Movies_and_TV、Kindle_Store 等。当数据集为 Toys_and_Games 时，RMR 的均方误差是 1.372，FRRM 的均方误差是 1.047，当数据集为 Tools_and_Home_Improvement 时，RMR 的均方误差是 1.491，FRRM 的均方误差是 1.017，当数据集为 Pet_Supplies 时，RMR 的均方误差是 1.562，FRRM 的均方误差是 1.316，当数据集为 Musical_Instruments 时，RMR 的均方误差是 1.374，FRRM 的均方误差是 0.735。

图 4.7 可看出模型 FRRM 在以上各数据集中的 MSE 值均有所提高，当数据据集为 Musical_Instruments 时，提高最大，为 46.51%。由于 Musical_Instruments 数据集的评分数据较多，本模型评分预测比 RMR 模型更加准确，故提高的百分点多。事实证明，在数据稀疏的情况下，融合语义分析的

模型 FRRM 对推荐系统的准确率有所提高。

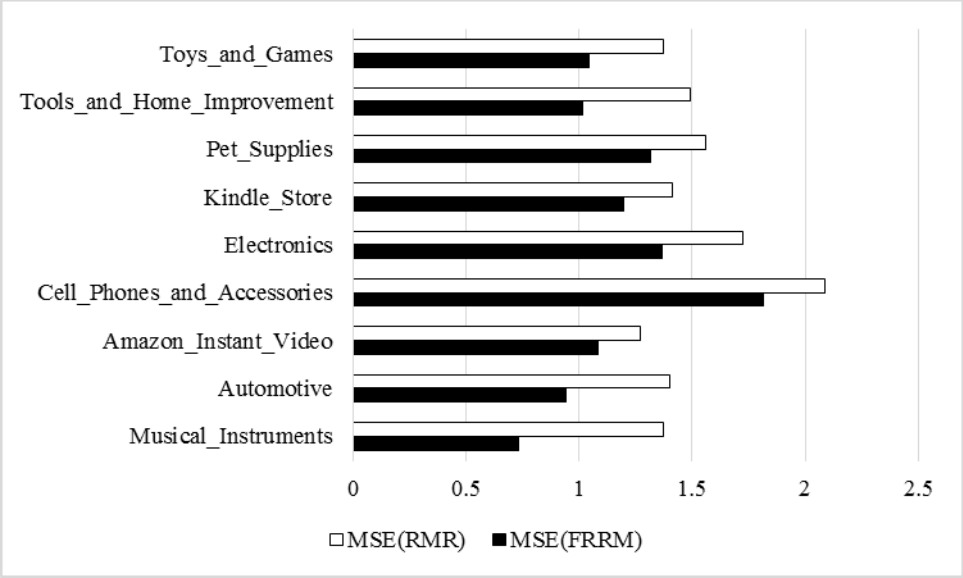

图 4.7 FRRM 和 RMR 的 MSE 对比

4.5 小结

本章研究了语义分析的基本原理，以及融合评分数据和评论信息的推荐模型，之后进行并行化设计，并搭建真实的实验环境，采用 Amazon 的 24 种数据集上进行测试，相比同类先进的方法 HFT，融合评分数据和评论信息的推荐模型 FRRM 抗稀疏性效果更好，准确率有所提升。事实证明，采用评论文本来增强评分预测的模型不仅可以进行更加精确的预测评分，还能利用来自评论的信息使推荐具有良好的解释性，进而在一定程度上缓解冷启动问题。进行语义分析后推荐，弥补了传统方法只利用评分进行预测的缺陷。尤其是并行化操作，大大提升了推荐的效率，并行化结果类似第三章，这里不再累赘。

第 5 章 融合评分和评论的推荐系统管理平台

本章设计推荐系统的流程管理系统，管理融合评分和评论推荐系统的作业、程序以及作业的上下文。实现数据分析的抽取，转换和加载过程，并提供了可扩展的数据分析程序和流程管理接口。布署到云端，实现在手机、平板等联网的设备上上传推荐程序，查看推荐程序，删除推荐程序，提交作业，查看作业执行的详细信息等，达到对推荐系统的控制更加灵活方便的目的。

5.1 管理平台设计原理

本章进行融合评分和评论的推荐系统管理平台的设计。利用大数据基础平台 Hadoop 提供的分布式文件系统 HDFS（Hadoop Distributed FileSystem）存储海量数据，采用 JavaEE 架构设计推荐系统的 web 应用程序，通过开源项目 Spark-Jobserver 提供的 REST 接口来提交 Spark 程序到集群执行，管理推荐系统的作业以及作业的上下文，部署到云端，使用户在 PC 机，手机以及任何联网的设备上实现对推荐系统的灵活控制。该系统基于 Spark 集群和 Spring MVC 框架，实现数据分析的抽取，转换和加载过程，并提供可扩展的数据分析程序。

用户通过 PC 端的浏览器或智能终端（如手机或平板）访问部署到云端的融合评分和评论的推荐系统管理平台。通过 HTTP 协议请求 Tomcat 服务器，Tomcat 服务器响应用户的请求，返回响应结果，展示给用户。云端是带防火墙功能的，起到安全隔离的作用，根据业务规模和需求，弹性伸缩配置 Tomcat 服务器、MySQL 数据库和 Spark 集群的规模。本节所依据的原理如图 5.1 所示：

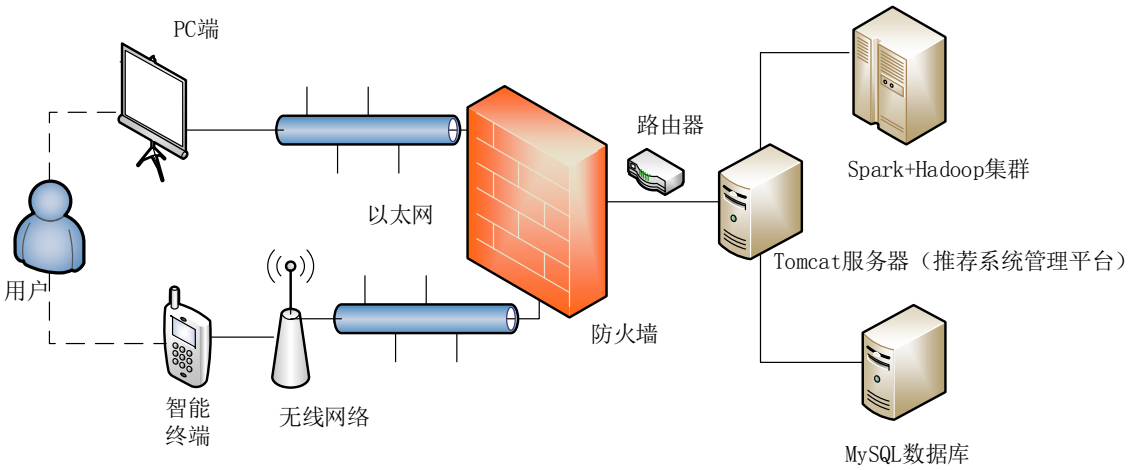

图 5.1 推荐系统管理平台原理图

本项目基于 JavaEE，SpringMVC+MyBaits 框架和 MySQL 数据库，使用 Java 语言进行编写。依据 Hadoop 集群，Spark 集群以及 Spark-Jobserver 的接

口进行程序的管理和作业的执行，依据 MySQL 数据库，提供数据存储服务，然后定义 Tomcat 作为 Web 服务器，把管理平台部署到 Tomcat 上。设计本平台所依据的基本架构如图 5.2 所示：

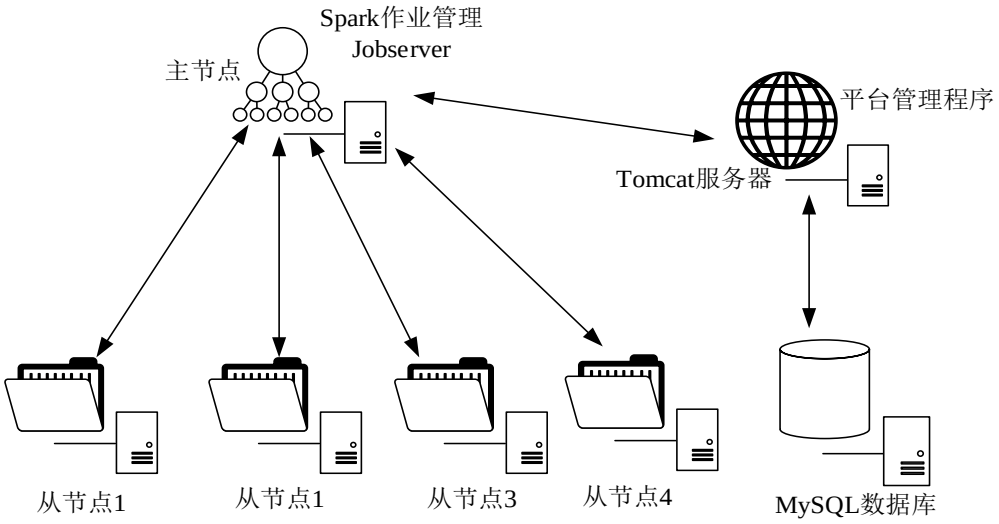

图 5.2 实验平台的架构图

5.2 管理平台的设计

本平台采用 Eclipse 新建 Maven 项目，在配置文件 pom.xml 中配置 Spring 的版本为 3.2.3，MyBatis 的版本为 3.2.4。使用 Log4j 作为日志接口，版本为 1.2.12，为了方便调试，使用日志来输出信息，本文通过使用 Apache 的一个开放源代码项目（Log4j），来控制日志信息的输送。Log4j 可通过定义每一条日志信息的级别来控制每一条日志的输出格式，这样能够更加细致地控制日志的生成过程。

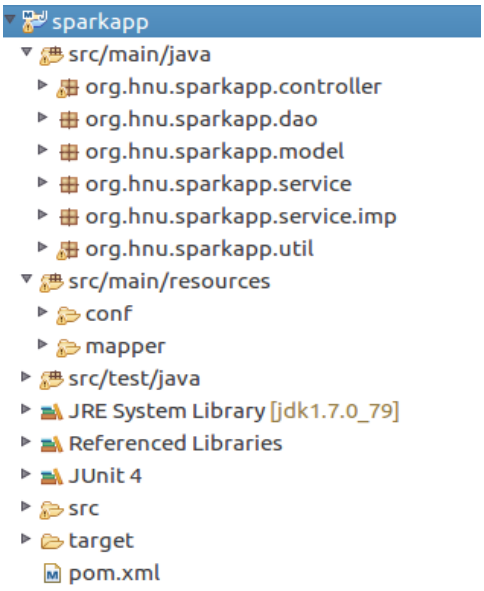

图 5.3 系统的目录结构

采用 JUnit 工具进行单元测试工具，版本为 4.8.2。采用的 web 服务器是 Tomcat7。运用 MVC 模型视图控制器，将代码逻辑分成 controller 层，dao 层，model 层，service 层以及 util 工具类。依次设计其他的模块，系统的目录结构如上图 5.3 所示。

设计数据库，新建 SparkApp 表保存 Spark 程序的信息，字段有主键 id，程序名称，上传时间，程序说明和程序参数说明。如表 5.2 所示：

表 5.1 SparkApp 表

字段	类型	是否为空	关键字	说明
id	int(11)	否	主键，自增	唯一标识
appname	varchar(100)	否		程序名称
uploadtime	varchar(100)	是		上传时间
app	varchar(100)	是		程序说明
parameter	varchar(100)	是		程序参数说明

新建 SparkJob 表保存 Spark 作业的信息，字段有主键 id，作业 ID，程序名称，Main 类，作业运行状态，长下文，开始运行时间，作业运行时间和结果。如表 5.3 所示：

表 5.2 SparkJob 表

字段	类型	是否为空	关键字	说明
id	int(11)	否	主键，自增	唯一标识
jobid	varchar(100)	否		作业 ID
appname	varchar(100)	是		程序名称
classpath	varchar(100)	是		Main 类
status	varchar(100)	是		作业运行状态
context	varchar(100)	是		长下文
duration	varchar(100)	是		作业运行时间
startTime	varchar(100)	是		开始运行时间
result	varchar(100)	是		结果

之后依据设计原理规划本平台的功能模块，如图 5.4，主要有 3 大模块，每个模块实现 2 个功能，分别为：

1）文件系统管理，管理 HDFS 文件系统。实现的功能有：

a）HDFS 文件列表；

b）HDFS 目录树。

2）推荐作业管理，管理 Spark 作业的提交和状态信息。实现的功能有

a）作业执行列表；

b）提交 Spark 作业。

3）推荐程序管理，管理 Spark 程序，以 Jar 包的形式保存。实现的功能

a）程序列表；

b）上传 Spark 程序。

融合评分和评论的推荐系统管理平台的功能如图 5.4 所示，按此进行设计，配置环境，布署平台，实现管理平台的各项功能。

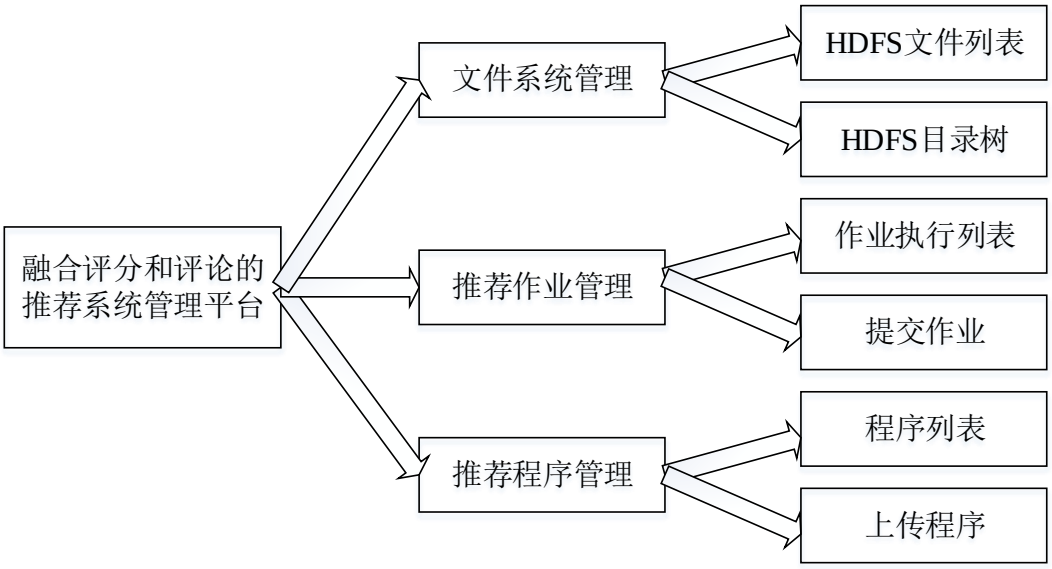

图 5.4 融合评分和评论的推荐系统管理平台功能图

5.3 管理平台的配置与布署

本设计首先安装 5 个节点（1 个 maste 节点，4 个 slaver 节点）的 Hadoop 集群，Spark 集群以及 Spark-Jobserver，安装过程如下：

1）下载 spark-1.6.1-bin-hadoop2.4.tgz，命令如下：

tar zxvf spark-1.6.1-bin-hadoop2.4.tgz -C /home/hadoop/

2）编辑 spark-defuatls.xml，添加 spark 的配置信息；

3）编辑 spark-env.sh，添加全局变量信息；

4）配置完成后，拷贝 master 节点下的 spark-1.6.1-bin-hadoop2.4 目录到 slave1，slave2，slave3，slave4 节点：

a）将 lib 目录下的 spark-assembly-1.6.1-hadoop2.4.0.jar 上传到 HDFS 中，

b）进行如下命令，$hdfs dfs -put spark-assembly-1.6.1-hadoop2.4.0.jar /data/spark_jars，

c）进行 spark-shell 和 spark-submit 的命令测试。

5）启动 spark-shell，连接到集群：

```
$spark-shell --master yarn-client
```

6）查看 spark-shell 管理界面 http://localhost:4040/；

7）提交 Spark 作业到集群：

```
$spark-submit \
  --class org.hnu.sparkapp.WordCount \
  --master yarn-client \
  /home/hadoop/SparkApp/WordCount.jar /data/input/wc.txt /data/ouput
```

8）编译安装 Spark-Jobserver:

a）解压 spark-jobserver 目录，通过 sbt 编译后得到 Spark-Jobserver 部署目录，

b）Spark-Jobserver 安装目录如下图 5.5 所示：

```
[hadoop@master job-server]$ pwd
/home/hadoop/job-server
[hadoop@master job-server]$ ll
total 23768
drwxr-xr-x. 3 hadoop hadoop     4096 May 31 11:16 filedao
-rw-r--r--. 1 hadoop hadoop      147 Jun 22 16:19 gc.out
drwxr-xr-x. 2 hadoop hadoop     4096 May 31 11:16 jars
-rwxr-xr-x. 1 hadoop hadoop     1626 May 20 18:37 kill-process-tree.sh
-rw-rw-r--. 1 hadoop hadoop     2842 May 31 15:07 local.conf
drwxr-xr-x. 2 hadoop hadoop     4096 May 31 11:33 log
-rw-rw-r--. 1 hadoop hadoop      710 May 20 18:37 log4j-server.properties
-rwxr-xr-x. 1 hadoop hadoop     1480 May 20 18:37 manager_start.sh
-rwxr-xr-x. 1 hadoop hadoop     1934 May 31 15:09 server_start.sh
-rwxr-xr-x. 1 hadoop hadoop      607 May 20 18:37 server_stop.sh
-rwxrw-r--. 1 hadoop hadoop     1516 May 20 18:37 setenv.sh
-rwxrw-r--. 1 hadoop hadoop     1011 May 31 14:56 settings.sh
-rw-rw-r--. 1 hadoop hadoop 24281648 May 20 18:37 spark-job-server.jar
-rw-rw-r--. 1 hadoop hadoop        5 Jun 22 16:18 spark-jobserver.pid
```

图 5.5 Spark-Jobserver 安装目录

c）编辑 settings.sh 文件，配置相关信息，

d）编辑 local.conf 文件，配置 spark jar 包在 HDFS 中的存放位置。

9）通过 server_start.sh 脚本启动 Spark-Jobserver，然后访问 http://localhost:8090，可以看到 Spark-Jobserver 的 Web UI。

5.4 管理平台的实现

5.4.1 文件系统管理的实现

文件系统管理和普通的资源管理器类似，可以实现简单的文件树查看、创建文件、删除文件等操作。文件系统管理栏目包括 HDFS 文件列表和文件目录树功能。

表 5.3 管理平台获取 HDFS 目录树结构伪代码

通过 HDFS 提供的 API，获取 HDFS 执行目录下的文件和目录信息，展示到页面
输入：目录名 输出：目录下的文件和目录名 1、通过 HDFS API 获取 HDFS 文件系统目录信息 2、获取目录下的文件和目录，保存类型信息和文件名 3、保存在目录列表 dirlist 中 4、根据“/”解析目录，得到目录的列表信息

HDFS 目录树实现 HDFS 文件系统以树状形式展现文件和目录结构，方便用户浏览，方便用户查看 HDFS 文件，进行相关操作。HDFS 目录树结构如图 5.6

所示：

图 5.6 HDFS 目录树

5.4.2 推荐系统程序管理的实现

推荐系统程序管理包括展示程序列表和上传程序的功能。上传程序是将编写好的 Spark 程序，上传到项目中，由项目来进行管理。首先将程序 jar 包保存到项目的 appfiles 目录下，通过 http 协议提交程序到 jobserver，指定程序名，然后获得 http 相应信息，之后解析 json 格式的响应信息，依次进行如下表 5.4 所示的操作：

表 5.4 管理平台上传程序伪代码

通过 http 请求指定程序名和上传程序 jar 包到 jobserver，并将程序信息保存到 Sparkapp 表，返回程序列表界面
输入：程序名，程序参数，程序 jar 包 输出：程序信息 1、将程序 jar 包保存到项目的 appfiles 目录下 2、通过 http 协议提交程序到 jobserver，指定程序名 3、获得 http 相应信息 4、解析 json 格式的响应信息 5、如果返回“OK” 6、根据程序名在 Sparkapp 表查询程序信息 7、如果存在程序信息，则更新程序信息 8、如果不存在程序信息，则添加程序信息到 Sparkapp 表 9、返回程序列表页面

输入程序名称、程序的参数等信息，选择文件，即可上传程序。如图 5.8 所

示：

图 5.7 上传程序示意图

程序列表是查看程序的详细信息，如 ID、程序名、上传时间、程序说明、参数说明等具体信息。也可以对程序进行删除等操作。如果程序过多，可以输入程序名称，进行查找。程序列表分页显示程序信息。也可以管理程序的相关信息，如程序名称，程序说明和程序参数说明。程序列表模块的伪代码如表 5.5 所示：

表 5.5 管理平台程序列表伪代码

获取所有 Spark 程序的信息
输入：分页的长度 length、起始序号 start 输出：当前页所有程序的信息 1、将长度和起始页信息保存到 map 中，作为分页参数 2、根据分页参数，查询 sparkapp 表对应的程序列表 3、计算获取程序列表的长度 4、将获得的程序信息保存为 json 格式 5、返回程序列表页面，显示程序信息

程序列表展示如图 5.9 所示：

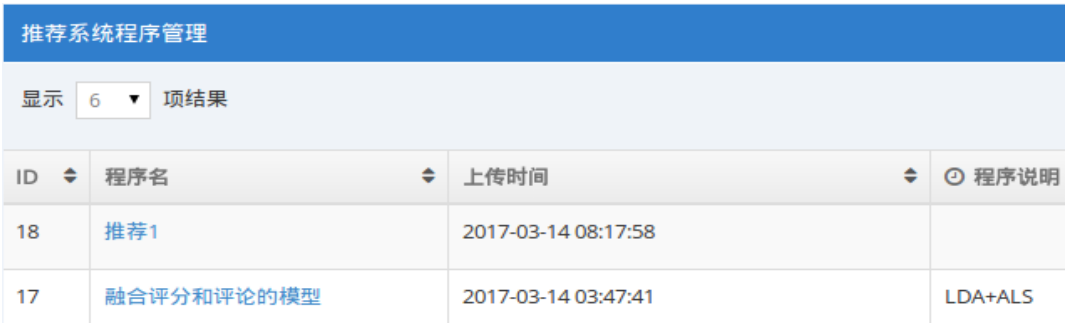

图 5.8 程序列表示意图

5.4.3 推荐系统作业管理的实现

本平台还设计了定时执行作业，作业执行完后报警（发邮件等），从人机交互角度看：本平台是人与机器（计算机）之间传递和交换信息的媒介，是程序员和系统进行双向信息交互的快捷方式。

1） 提交作业

管理平台提交 Spark 作业伪代码设计如下表 5.1 所示：

表 5.6 管理平台提交 Spark 作业伪代码

通过 Jobserver 的 REST 接口提交 Spark job 执行，并将 job 信息插入数据库，返回作业列表界面
输入：程序输入路径，程序输出路径，程序名，类名 输出：作业信息 1、根据程序名 appname 在数据库 app 表的字段中查询程序信息 2、找到程序信息对象，包括程序名，类名，参数信息 3、通过 http 协议提交作业到 jobserver，包含程序名，类名和参数信息 4、获得 http 相应信息 5、解析 json 格式的响应信息 6、获得作业的执行信息，包括状态，作业 ID，结果，上下文信息 7、创建作业信息对象 8、插入到数据库 9、设置页面的作业信息 10、返回页面 11、列出作业的信息

提交作业功能是提交 Spark 程序到集群执行，用户提交程序，Main 类，输入参数，输出参数等信息，即可提交作业。如果输入有误，还可以进行重置操作，重新输入数据等。指定提交作业的参数，如程序名称，classPath 名称（运行 Main 类），以及作业的输入、输出参数。提交作业如图 5.10 所示：

图 5.9 提交作业示意图

2）作业执行列表

作业执行列表可查看作业的执行信息，如作业 ID、程序名称、classPath、

状态、开始时间，并且可以进行删除操作。作业列表分页显示作业信息。

表 5.7 管理平台获取作业列表伪代码

通过管理平台获取 Spark 执行作业列表
输入：分页的长度 length、起始序号 start 输出：当前页所有作业的信息 1、将长度和起始页信息保存到 map 中，作为分页参数 2、根据分页参数，查询 sparkjob 表对应的作业列表 3、计算获取作业列表的长度 4、将获得的作业信息保存为 json 格式 5、返回作业列表页面，显示作业信息

图 5.10 作业执行列表

本平台在提交作业后，作业的运行信息和执行情况信息可以展示出来。如图 5.11 所示。

5.5 小结

本章采用 JavaEE 架构来设计推荐系统的流程管理系统，通过开源项目 Spark-Jobserver 提供的 REST 接口来管理融合评分和评论推荐系统的作业、程序以及作业的上下文。实现了数据分析的抽取，转换和加载（Extract-Transform-Load，ETL）过程，以及业务需求的分析，实现了推荐程序的管理，推荐系统的执行以及推荐作业的可视化，给大数据分析等处理流程带来自动化和直观化的体验，并提供了可扩展的数据分析程序和流程管理接口，不仅让 Spark 变得有个性，还让相关操作变得简单和自由，并将 Spark 作业本身的信息更加顺畅的传递给使用者，为使用者提供了良好的操作界面，充分体现了 Spark as a service 的设计思想。之后把此平台布署到云端，无论在哪里，都可以在手机、平板等联网的设备上上传推荐程序，查看推荐程序，删除推荐程序，提交作业，查看作业执行的详细信息等，达到了对推荐系统的控制更加灵活方便的目的，取得了良好的效果。

结 论

本文融合评分数据和评论信息，利用评论文本获得的主题信息增强评分预测，解释评分数据的变化。建立评分和评论的聚合框架，评论部分采用机器学习无监督聚类算法，进行语义分析，评分部分采用协调过滤算法，进行评分预测，这样既可以处理评分数据，也可以处理评论信息。通过评论中的信息提供关于用户偏好的额外信息来解决数据稀疏性的问题。当推荐系统有很少的评分或没有评分可以利用时，本文还可以利用评论来推断协同过滤算法所需的评分。对于推荐系统中通常存在的新用户，本文将评论信息（如特征观点或者评论主题）和数值评分相匹配，来为评分较少的用户建模。如果用户完全是新用户，用户的评分只有在其使用推荐系统时才能被抽取出来。所以本文会更关注评论元素，使用评论信息帮助用户填充偏好矩阵或者富化产品模型。如果数据集不稀疏，那么本文的模型可用来决定用户评分的质量，另外还可以推断用户的偏好是否与周围环境相关，从评论中提到的观点的特征来学习出用户的隐式偏好。

工作总结

本文研究的内容是融合评分数据和评论信息的推荐系统，主要对矩阵分解算法中的协同过滤推荐算法进行了深入的研究。其中有基于 ALS 的协同过滤算法，基于主题模型的无监督机器学习算法。本文针对现有这些算法存在的缺点和不足，提出一些改进模型和算法。本文完成的工作可以概括为以下三个部分：

1）提出了高效率的评分模型 NALS-WR

传统的评分模型是在协同过滤算法基础上利用用户的评分数据进行推荐的。然而，一般的算法的运行效率会随着数据规模的增大下降很多，数据规模越大，效率越低。本文的评分模型 NALS-WR 是在 Linux 集群上进行的，基于内存进行数据处理，由于分布式计算框架和云计算环境，NALS-WR 具有很强的可扩展性，特别是针对传统的矩阵分解算法的资源分配的障碍，本模型脱颖而出。并且数据规模越大，NALS-WR 的效率越高。实验表明，不仅仅 NALS-WR 推荐的准确率提高了一点，无论是在可扩展性，或抗稀疏性或效率方面，NALS-WR 算法都非常优越。

2）提出了高准确率的评分评论融合模型 FRRM

由于推荐系统中系统中可用的用户评分记录极为有限，而用户和项目的评论可能会隐藏一些潜在的主题信息，评分数据的极端稀疏性将导致得到的预测评分不够精确，为了进一步提高推荐系统的准确性，在评分模型 NALS-WR 的基础

上加入主题模型挖掘评论中隐含的潜在主题信息，从而得到文档-主题概率分布，进而基于评分数据和文档-主题概率分布来共同构建模型 FRRM，可解释推荐的原因。将 FRRM 算法应用于 Amazon 数据集，实验结果表明该算法有效解决了数据稀疏性问题，同时也显著提高了推荐系统的推荐准确性。基于评分数据与文档—主题分布的模型 FRRM，大大提高了推荐系统的推荐质量。并且在 Spark 平台上的并行化设计增加了效率，为推荐系统提供了批处理到实时处理的过渡，克服了传统 Hadoop 只有批处理的缺陷。

3）设计了融合评分和评论的推荐系统管理平台

现在的推荐系统管理不够灵活，控制不太方便，为了使在任何联网的设备上能够很便捷的管理推荐系统，本文设计了一款流程管理平台，也是一款新的 web 应用程序，采用 JavaEE 架构，通过开源项目 Spark-Jobserver 提供的 REST 接口来管理融合评分和评论推荐系统的作业、程序以及作业的上下文。利用大数据基础平台 Hadoop 提供的分布式文件系统 HDFS（Hadoop Distributed File System）存储海量数据，使用 Spark 分布式计算框架，进行海量数据的分析和挖掘。该设计为程序员提供了良好的操作界面，实现了数据分析的抽取，转换和加载过程，实现了推荐程序的管理，推荐系统的执行以及推荐作业的可视化，给大数据分析等处理流程带来自动化和直观化的体验，并提供了可扩展的数据分析程序和流程管理接口，本文的实验把此平台部署到云端，无论在哪里，都可以在手机、平板等联网的设备上管理推荐程序，查看推荐程序，提交作业，查看作业执行的详细信息等，使对推荐系统的控制更加灵活自如。

工作展望

由于时间和水平的限制，本改进系统的一些不足之处还需要进一步的完成。本文只是矩阵分解推荐算法和主题模型做了初步改进，在未来的工作中需从一些几个方面进行展开和深入：

1）本文提出的结合评分和评论的 FRRM 模型只是部分解决了推荐算法的数据稀疏性，推荐准确性和效率问题，对推荐算法存在的其他问题，如实时性，用户隐私等问题，本文并没有给出有效的解决办法。因此，今后将在本文工作的基础上继续研究和改进，以解决推荐系统面临的更多问题。

2）本文只对评论信息做了语义分析，并没有对物品内容信息作出相应的操作，基于内容和基于协同过滤的混合推荐算法应该考虑进来。因此，今后将研究如何更好的利用物品项目内容信息来做混合推荐，利用语义分析的丰富性、语义表达性、易获取性来提高推荐系统的质量。

参考文献

[1] Mobasher B, Dai H, Luo T, et al. Discovery of Aggregate Usage Profiles for Web Personalization. Proceedings of the Webkdd Workshop, 2000.

[2] 邓爱林, 朱扬勇, 施伯乐. 基于项目评分预测的协同过滤推荐算法. 软件学报, 2003, 14(9): 1621-1628.

[3] Cooley R, Mobasher B, Srivastava J. Grouping Web Page References into Transactions for Mining World Wide Web Browsing Patterns. In: Proceedings of Knowledge and Data Engineering Exchange Workshop. Newport Beach: IEEE, 1997, 2-9.

[4] Ricci F, Rokach L, Shapira B. Introduction to recommender systems handbook. New York: Springer US, 2011, 1-35.

[5] Koren Y, Bell R, Volin sky C. Matrix factorization techniques for recommender systems. Computer, 2009, 42(8): 30-37.

[6] Oard D W, Kim J. Implicit feedback for recommender systems. In: Proceedings of the AAAI workshop on recommender systems. Madison: AAAI, 1998, 81-83.

[7] Pan R, Zhou Y, Cao B, et al. One-class collaborative filtering. In: Proceedings of the 8th IEEE International Conference on Data Mining. Pisa: IEEE, 2008, 502-511.

[8] Funk S. Stochastic gradient descent. http://sifter.org/simon/journal/2006-1211.html, 2012-6-6

[9] Koren Y. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. Vegas: ACM, 2008, 426-434.

[10] Koren Y. Collaborative filtering with temporal dynamics. Communications of the ACM, 2010, 53(4): 89-97.

[11] Zhou Y, Wilkinson D, Schreiber R, et al. Large-scale parallel collaborative filtering for the netflix prize. In: International Conference on Algorithmic Applications in Management. Shanghai: Springer, 2008, 337-348.

[12] Kim C, Kim J. A recommendation algorithm using multi-level association

rules. In: Proceedings IEEE/WIC International Conference on Web Intelligence. Madrid: IEEE, 2003, 524-527.

[13] Lu Y, Zhai C X, Sundaresan N. Rated aspect summarization of short comments. In: Proceedings of the 18th international conference on World Wide Web. Madrid: ACM, 2009, 131-140.

[14] Titov, Ivan, McDonald, et al. A Joint Model of Text and Aspect Ratings for Sentiment Summarization. PROC. ACL-08: HLT, 2008, 308--316.

[15] 刘庆鹏, 陈明锐. 优化稀疏数据集提高协同过滤推荐系统质量的方法. 计算机应用, 2012, 32(4): 1082-1085.

[16] 薛福亮, 张慧颖. 应用 WUM 和 RBFN 补值的协同过滤推荐研究. 计算机工程与应用, 2012, 48(9): 22-26.

[17] 吕成戍, 王维国, 丁永健. 基于 KNN-SVM 的混合协同过滤推荐算法. 计算机应用研究, 2012, 29(5): 1707-1709.

[18] Covington P, Adams J, Sargin E. Deep neural networks for youtube recommendations. In: Proceedings of the 10th ACM Conference on Recommender Systems. Boston: ACM, 2016, 191-198.

[19] Elkahky A M, Song Y, He X. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In: Proceedings of the 24th International Conference on World Wide Web. Florence: ACM, 2015, 278-288.

[20] Dziugaite G K, Roy D M. Neural Network Matrix Factorization. Computer Science, 2015, 1511.06443, 85-93.

[21] Kim D, Park C, Oh J, et al. Convolutional Matrix Factorization for Document Context-Aware Recommendation. In: Proceedings of the 10th ACM Conference on Recommender Systems. Boston: ACM, 2016, 233-240.

[22] Strub F, Mary J. Collaborative Filtering with Stacked Denoising AutoEncoders and Sparse Inputs. In: NIPS Workshop on Machine Learning for eCommerce. 2015.

[23] Strub F, Gaudel R, Mary J. Hybrid Recommender System based on Autoencoders. In: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems. Boston: ACM, 2016, 11-16.

[24] Wu Y, DuBois C, Zheng A X, et al. Collaborative denoising auto-encoders for top-n recommender systems. In: Proceedings of the Ninth ACM International Conference on Web Search and Data Mining. San Francisco:

ACM, 2016, 153-162.

[25] Sedhain S, Menon A K, Sanner S, et al. Autorec: Autoencoders meet collaborative filtering. In: Proceedings of the 24th International Conference on World Wide Web. Florence: ACM, 2015, 111-112.

[26] 曹毅. 基于内容和协同过滤的混合模式推荐技术研究: [中南大学硕士学位论文]. 长沙: 中南大学, 2007, 11-22.

[27] Roy S, Guntuku S C. Latent Factor Representations for Cold-Start Video Recommendation. In: Proceedings of the 10th ACM Conference on Recommender Systems. Boston: ACM, 2016, 99-106.

[28] Shi Y, Karatzoglou A, Baltrunas L, et al. CLiMF: learning to maximize reciprocal rank with collaborative less-is-more filtering. In: Proceedings of the sixth ACM conference on Recommender systems. Dublin: ACM, 2012, 139-146.

[29] 项亮. 推荐系统实践. 第 1 版. 北京: 人民邮电出版社, 2012, 3-40.

[30] Ramakrishnan N, Keller B J, Mirza B J, et al. Privacy risks in recommender systems. IEEE Internet Computing, 2001, 5(6): 54-63.

[31] Aïmeur E, Brassard G, Fernandez J M, et al. Alambic: a privacy-preserving recommender system for electronic commerce. International Journal of Information Security, 2008, 7(5): 307-334.

[32] Jiawei Han, Micheline Kamber, 数据挖掘概念与技术. 范明, 孟小峰. 第 1 版. 机械工业出版社, 2007, 223-261.

[33] Shyong K, Frankowski D, Riedl J. Do you trust your recommendations? An exploration of security and privacy issues in recommender systems. Emerging Trends in Information and Communication Security. Berlin: Springer Berlin Heidelberg, 2006, 14-29.

[34] Levi A, Mokryn O, Diot C, et al. Finding a needle in a haystack of reviews: cold start context-based hotel recommender system. In: Proceedings of the sixth ACM conference on Recommender systems. Dublin: ACM, 2012, 115-122.

[35] Wang H, Lu Y, Zhai C. Latent aspect rating analysis on review text data: a rating regression approach. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington: DBLP, 2010, 783-792.

[36] 毛国君, 段立娟. 数据挖掘原理与算法. 第三版. 北京: 清华大学出版社, 2005, 10-100.

[37] Miller B N. GroupLens: An Open Architecture for Collaborative Filtering. 1995.

[38] Sarwar B, Karypis G, Konstan J, et al. Item-based collaborative filtering recommendation algorithms. In: International Conference on World Wide Web. New York: ACM, 2001, 285-295.

[39] Polat H, Du W. SVD-based collaborative filtering with privacy. In: Proceedings of the 2005 ACM symposium on Applied computing. Santa Fe: ACM, 2005: 791-795.

[40] Sarwar B, Karypis G, Konstan J, et al. Application of dimensionality reduction in recommender system-a case study[R]. Minnesota Univ Minneapolis Dept of Computer Science, 2000.

[41] Schein A I，Popescul A，Ungar L H，et a1．Methods and metrics for cold Start recommendations. In: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Tampere: ACM, 2002, 253-260.

[42] George T, Merugu S. A scalable collaborative filtering framework based on co-clustering. In: Proceedings of the 5th IEEE International Conference on Data Mining. Houston: IEEE, 2005, 1-4.

[43] Liu Q, Wang C, Xu C. A Modified PMF Model Incorporating Implicit Item Associations. In: IEEE International Conference on TOOLS with Artificial Intelligence. Herndon: IEEE, 2013, 1041-1046.

[44] Ma H, King I, Lyu M R. Learning to recommend with social trust ensemble. In: International ACM SIGIR Conference on Research and Development in Information Retrieval. Boston: ACM, 2009, 203-210.

[45] 王正武. 基于用户喜好类型的协同过滤推荐算法研究: [华东师范大学硕士学位论文]. 上海: 华东师范大学, 2011, 21-30.

[46] 刘智超. 基于混合模型的学术论文推荐方法研究: [北京邮电大学硕士学位论文]. 北京: 北京邮电大学, 2015, 21-24.

[47] Burke R. Hybrid recommender systems: Survey and experiments. User modeling and user-adapted interaction, 2002, 12(4): 331-370.

[48] Ganu G, Elhadad N, Marian A. Beyond the Stars: Improving Rating Predictions using Review Text Content. In: International Workshop on the Web and Databases(WEBDB), Providence: DBLP, 2009, 9: 1-6.

[49] Blei D M, Ng A Y, Jordan M I. Latent dirichlet allocation. Journal of machine learning research, 2003, 3: 993-1022.

[50] Titov I, Mcdonald R. Modeling Online Reviews with Multi-grain Topic Models. In: International Conference on World Wide Web. Beijing: ACM, 2008, 111-120.

[51] Zhao W X, Jiang J, Yan H, et al. Jointly modeling aspects and opinions with a MaxEnt-LDA hybrid. In: Conference on Empirical Methods in Natural Language Processing. Massachusetts: ACL, 2010, 56-65.

[52] Lerman K, Blair-Goldensohn S, Mcdonald R. Sentiment summarization: evaluating and learning user preferences. In: Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics. Athens: Association for Computational Linguistics, 2009, 514-522.

[53] Wang C, Blei D M. Collaborative topic modeling for recommending scientific articles. In: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. San Diego: ACM, 2011, 448-456.

[54] Mcauliffe J D, Blei D M. Supervised topic models. In: Advances in neural information processing systems. Vancouver: Curran Associates, 2008, 121-128.

[55] Kim S M, Hovy E. Determining the sentiment of opinions. In: Proceedings of the 20th international conference on Computational Linguistics. Geneva: Association for Computational Linguistics, 2004, 1367.

[56] Lin C, He Y. Joint sentiment/topic model for sentiment analysis. In: ACM Conference on Information and Knowledge Management. Hong Kong: ACM, 2009, 375-384.

[57] Harper F M, Konstan J A. The movielens datasets: History and context. In: ACM Transactions on Interactive Intelligent Systems (TiiS), 2016, 5-19.

[58] 党齐民, 吕冬煜. 基于词关联语义的文本分类研究. 计算机应用, 2004, 24(4): 62-63.

[59] Mcauley J, Leskovec J. Hidden factors and hidden topics: understanding rating dimensions with review text. In: ACM Conference on Recommender Systems. Hong Kong:ACM, 2013, 65-172.

[60] Ling G, Lyu M R, King I. Ratings meet reviews, a combined approach to recommend. In: ACM Conference on Recommender Systems. Foster City: ACM, 2014, 105-112.

致 谢

光阴似箭，岁月如梭，弹指一挥间，硕士研究生生涯即将结束。站在毕业的门栏上回首求学历程，我感慨万千。在此论文完成之际，我想对所有帮助过我的人表达深深的感谢！

首先，十分感激我的导师李仁发教授和唐涛高级工程师对本人的谆谆教诲和悉心栽培。李老师为人热情，治学严谨，对待学生慈祥和蔼，处处为学生考虑，宽容待人，像父亲一般耐心教导，及时给予引导。这两年的研究生生活使我不仅掌握了扎实的理论知识，还锻炼了较强的科研攻关能力、项目实践经历和创新能力，让我少走了很多弯路。能够成为李老师的学生，是我一辈子的荣幸。我要向李老师表示真挚的谢意，祝愿老师身体健康，桃李满天下!

感谢唐涛老师对本人的指导，他以丰富的实践能力和渊博的项目经验指导我们做完一个又一个有意义的项目。唐老师敏锐的学术思维、精益求精的工作态度以及侮人不倦的师者风范是我终生学习的楷模。

衷心感谢徐成老师，徐老师是嵌入式领域的专家。他认真对待学生的所有事情，尤其是他宽广的专业知识、严谨的治学态度，让我敬佩。这种专研的态度和平易近人的作风是我终身学习的榜样。

特别感谢邝继顺老师、谢鲲老师、彭蔓蔓老师、杨磊老师、肖玲老师和叶柏龙老师。他们在论文完成过程中对我的指导，对我完成学业帮助很大。在本人毕业之际，他们倾注了大量的精力，在此谨向我的邝老师、谢老师、彭老师、杨老师、肖老师和叶老师表达我最衷心的感谢。

感谢付斌和李蕊老师，付老师无私的帮助过我，李老师在我实习时耐心的指导我，他们严谨的学术作风和踏实的钻研精神，是我今后工作不断追求的目标。

感谢博士师兄师姐，和各位同门，黄晶师兄在我学习和生活中遇到问题时不断的鼓励和引导，使我勇敢面对困难，朱立民师兄耐心地给我修改论文，感激他们。

感谢父母总是无时无刻地资助我，教导我做人做事的道理。

感谢各位专家和老师百忙之中抽出宝贵的时间对本论文的评阅。

最后，衷心感谢母校信息科学与工程学院对我的悉心培养！感谢我的家人朋友对我的照顾和支持，这是我战胜无数困难的精神动力与物质基础。

屠晓涵

2017年5月于长沙

附录 A 攻读硕士学位期间发表的学术论文

论文发表：

[1]屠晓涵, 刘四平, 李仁发. Improving Matrix Factorization Recommendations for Problems in Big Data. ICBDA, 2017.（已录用）

[2]刘四平, 屠晓涵, 李仁发. Unifying explicit and implicit feedback for Top-N recommendation. ICBDA, 2017.（已录用）