2020 届硕士专业学位研究生学位论文（全日制研究生）

分类号：_____ 学校代码: 10269

密　级：_____ 学　　号: 51174500139

# East China Normal University

# 硕士专业学位论文

**MASTRER'S DISSERTATION**

## 论文题目：基于分布式框架的电影推荐系统的设计与实现

| | |
|---|---|
| 院 系 名 称： | **软件工程学院** |
| 专业学位类别: | **工程硕士** |
| 专业学位领域: | **软件工程** |
| 指 导 教 师： | **王江涛 高级工程师** |
| 学 位 申 请 人: | **徐翊鑫** |

2019 年 11 月 25 日

Dissertation for professional master's degree in 2019

University Code: 10269

Student ID: 51174500139

# East China Normal University

**Title:** **Design and Implementation of Film Recommendation System Based on Distributed Framework**

| | |
|---|---|
| Department: | **School of Software Engineering** |
| Professional degree category: | **Master of Engineering** |
| Professional degree field: | **Software Engineering** |
| Supervisor: | **Wang Jiangtao.Senior Engineer** |
| Candidate: | **Xu Yixin** |

2019.11.25

## 华东师范大学学位论文原创性声明

郑重声明：本人呈交的学位论文《基于分布式框架的电影推荐系统的设计与实现》，是在华东师范大学攻读硕士/博士（请勾选）学位期间，在导师的指导下进行的研究工作及取得的研究成果。除文中已经注明引用的内容外，本论文不包含其他个人已经发表或撰写过的研究成果。对本文的研究做出重要贡献的个人和集体，均已在文中作了明确说明并表示谢意。

作者签名：张翔鑫　　日期：2019 年 11 月 25 日

## 华东师范大学学位论文著作权使用声明

《基于分布式框架的电影推荐系统的设计与实现》系本人在华东师范大学攻读学位期间在导师指导下完成的硕士/博士（请勾选）学位论文，本论文的著作权归本人所有。本人同意华东师范大学根据相关规定保留和使用此学位论文，并向主管部门和学校指定的相关机构送交学位论文的印刷版和电子版；允许学位论文进入华东师范大学图书馆及数据库被查阅、借阅；同意学校将学位论文加入全国博士、硕士学位论文共建单位数据库进行检索，将学位论文的标题和摘要汇编出版，采用影印、缩印或者其它方式合理复制学位论文。

本学位论文属于（请勾选）

（　）1.经华东师范大学相关部门审查核定的“内部”或“涉密”学位论文*，于　　年　　月　　日解密，解密后适用上述授权。

（√）2.不保密，适用上述授权。

导师签名　　本人签名 张翔鑫

2019 年 11 月 25 日

*“涉密”学位论文应是已经华东师范大学学位评定委员会办公室或保密委员会审定过的学位论文（需附获批的《华东师范大学研究生申请学位论文“涉密”审批表》方为有效），未经上述部门审定的学位论文均为公开学位论文。此声明栏不填写的，默认为公开学位论文，均适用上述授权）。

# 徐翊鑫 硕士学位论文答辩委员会成员名单

| 姓名 | 职称 | 单位 | 备注 |
| --- | --- | --- | --- |
| 宋晖 | 教授 | 东华大学 | 主席 |
| 章玥 | 副教授 | 华东师范大学 | |
| 陈伟婷 | 副教授 | 华东师范大学 | |
| | | | |
| | | | |

# 摘 要

大数据时代人们可以获得越来越多的信息，但是在如此庞杂的信息海洋中真正对用户个人实用的信息可能只占很小的比例。推荐系统的出现，就是为了帮助人们寻找个性化和准确的信息。在传统的推荐系统中，对推荐算法的研究已经非常深入了，并且可以基本实现推荐功能，然后却普遍存在以下的问题：1.初次启动数据过少或者没有 2.有一些推荐算法需要计算用户或者物品的相似度，常用的做法是使用聚类算法先获得同一类别的用户或物品，然后计算在此类别中的最近邻，在这个过程中聚类算法的性能还有待提升 3.推荐的准确性和多样性无法兼得，如果一味地使用某种推荐算法，那么很可能会发生过度推荐的情况。

针对以上问题，并结合新时代的背景，本文提出了以下的改进思路：

1.引入用户的特征属性，利用很小的代价收集用户相关信息，将这些信息用在后续的推荐算法中，以解决冷启动的问题。并将电影按照内容分解为不同类型，使用电影类型偏好作为用户的属性。引入时间因子来动态调整用户特征属性和电影偏好属性的比例。

2.在使用聚类寻找最近邻的时候，为了得到 K-means 聚类的必要参数 K 和 k 个初始聚类中心，本文使用 canopy 算法进行预处理，然后再更加细化地聚类。大数据时代出现了很多针对海量数据的分析处理工具，Hadoop 是其中之一，其特点有框架化、并行化、硬件要求低等等，可以大大减少算法的运行时间。本文使用 MapReduce 框架对聚类算法进行了并行化的实现。

3.基于混合推荐算法的思想，在系统中实现了混合推荐算法，既保证了多样性，又保证了准确性。

本文提出了上述 3 个改进点，并实现了一个简单的电影推荐系统，经过实验和数据的证明，本文提出的对于推荐算法的优化方法，有效地提高了推荐系统的效率，可以让用户找到更适合自己的信息，具有很大的实用性。

**关键词：推荐系统；大数据；Hadoop 平台；协同过滤；K-means**

# ABSTRACT

In the era of big data, people can get more and more information, but in such a huge information ocean, the information that is really useful to users may only account for a small proportion. The recommendation system can help people find personalized and accurate information. In the traditional recommendation system, the research on recommendation algorithm has great depth, and can basically realize the recommendation function. But there are always some following problems: 1. The initial start-up data is too little. 2. Some recommendation algorithms need to calculate the similarity of users or items. The common method is to use clustering algorithm to get the same category of users or items first, and then to use clustering algorithm to get the same category of users or items. Computing the nearest neighbors in this category, the performance of clustering algorithm needs to be improved in this process. 3. The accuracy and diversity of recommendation cannot be achieved simultaneously. If a recommendation algorithm is used blindly, the situation of over-recommendation is likely to occur.

In view of the above problems and the background of the new era, this thesis puts forward the following ideas for improvement:

1. Introduce user's characteristic attributes, collect user's relevant information at a very small cost, and use these information in subsequent recommendation algorithms to solve the problem of cold start. The movies are decomposed into different types, and the preferences of movie types are used as the attributes of user's vector. Time factor is introduced to dynamically adjust the proportion of user characteristic attributes and movie preference attributes.

2. When using clustering to find the nearest neighbor, in order to get the necessary parameters K and K initial clustering centers of K-means clustering, this thesis uses canopy to preprocess, and then clustering in more detail. In the era of big data, there are many analysis and processing tools for massive data. Hadoop is one of them. Its characteristics are framing, parallelization, low hardware requirements and so on,

which can greatly reduce the running time of the algorithm. This paper uses MapReduce framework to realize the parallel clustering algorithm.

3. Based on the idea of hybrid recommendation, hybrid recommendation algorithm is implemented in the system designed in chapter 3, which guarantees both diversity and accuracy.

This thesis puts forward the above three improvement points, and implements a simple film recommendation system. The experiment and results prove that the proposed optimization method for recommendation algorithm effectively improves the efficiency of recommendation system, and enables users to find more suitable information for themselves, which has great practicability.

**Keywords:** **recommendation system; big data; Hadoop platform; collaborative filtering; K-means**

# 目录

# 图目录

# 表目录

# 第一章 绪论

## 1.1 选题的背景和意义

日常生活中，每个人都不可避免会有推荐或者被推荐的经历，互联网出现之前的推荐主要来源于人们的口口相传、广告和传单，对于一个具体用户来说能获取信息的渠道很少，获取到的信息数量也有限。随着互联网近几年的飞速发展，我国的网民越来越多[1]。人们获取信息的渠道变得丰富而便捷，同时由于每个人产生的数据也急剧增长，所以人们可获得的信息也越来越多，面对如此规模浩大的数据量，用户就面临信息过载的问题[2]，即对某个用户来说，真正对其自身有用的信息的占比是很少的，如何在众多数据中寻找到自己想要的或者需要的数据越来越困难了。在这种情况下，人们需要一种方法来解决上述问题，推荐系统就是为了解决这样的问题出现的[3]。一个高效的推荐系统会给目标用户带去他最想要的信息，有些信息甚至是用户自己也没有想到的。

传统的推荐系统面对的数据量的挑战很小，因为数据量小所以计算量也小，没有对软硬件提出很高的要求。进入了互联网时代，人们足不出户，就能获取茫茫多的数据，使用传统的推荐算法就无法满足信息需求了，因为此时的推荐系统将要面对的是 PB 级别的数据量[4]，如果还是采用传统的推荐系统架构，那么数据的处理会面临内存不足、计算时间太长等问题，所以大数据时代推荐系统应该要结合大数据处理框架，与时俱进，这样才能应对新挑战。Hadoop 就是针对大数据处理的框架，其一经问世就引来了数据科学界的广泛关注，目前已经发展到了 Hadoop2.0。Hadoop 包括三大核心：HDFS、MapReduce、YARN，配合 Hadoop 生态圈的其他组件，提供了一个完整的存储、计算、资源调度的框架。Hadoop 能够将一个任务切分成很多子任务，并使用计算集群并行计算，相对于传统的串行处理，这样能很显著地提高效率[5]。

推荐算法是整个推荐系统最重要的部分，根据用户的信息，使用数学的方法，计算出该用户可能喜欢的其他物品，这可以为用户节省很多时间，如果使用在电商系统，更是可以为商家带来更多的利润。在众多推荐算法中，协同过滤使用得

最为广泛，从“协同”两个字可以看出来该算法主要是利用群体中其他单位的信息，对特定的物品进行过滤，从而得到预期物品。该算法认为，如果一些用户有相似的偏好或者类似的行为，那么这些用户在将来也会具有相仿的偏好和行为。这是符合常理的，而且使用协同过滤推荐算法计算出来的推荐列表具有很好的解释性，让用户更信服，比如给一个大学生推荐计算机方面的图书，很可能是因为他在浏览图书网站的时候，查看了部分关于计算机的书籍，不会让用户觉得莫名其妙而感到厌烦。在协同过滤算法中，往往要对用户或者物品进行聚类操作[6]，最常用的方法是 K-means。K-means 的好处是算法简单易实现，但是缺点也很明显，就是对于 K 值以及初始聚类中心的选取会直接影响聚类的效率和效果。本文对 K-means 聚类提出改进，并通过实现一个电影推荐系来说明改进后算法的优势。

随着生活水平的提高，人们的文化需求也越来越趋向于精准化、个性化。体现在电影这方面，人们希望能更方便地寻找到自己喜欢的电影，不单单是新上映的电影，还包括已经放映过的老电影。当今市面上，人们对于音乐、服装、图书等的推荐系统早已司空见惯，但却很少有关于电影的个性化推荐，目前国内用户量比较大的视频网站有腾讯视频、爱奇艺、优酷、豆瓣电影等等[7]，这些网站的主要内容是把电影按照不同的类型罗列出来，而且由于面对的用户很广，所以功能多且杂，内容也很多，用户的思路更多的是跟着视频网站在走，自己缺少主动权，还有一个最大的问题就是它们无法提供实时推荐的效果。而像猫眼、美团、淘票票等网站提供的是电影票的购买服务，针对的更多的是当前的电影，而且并无推荐功能，只是按照评分排序。以上两种情况用户都需要自己去阅读介绍、评论并加以选择，浪费了大量的时间精力，过多的信息和无关的内容使得用户体验不佳。

考虑到用户有时候只是很单纯地想获得自己喜欢的电影，所以本文基于改进的推荐算法，结合电影的物品特殊性，将推荐系统和大数据处理技术相结合，设计并开发了一个基于分布式数据处理框架的电影推荐系统，并且证明是可实施的、有效的。该推荐系统可以有效地减少用户的寻找时间，满足了人们的文化活动需求。并且本系统具有很高的拓展性，可以经由很小的改动而运用到其他领域中。

## 1.2 国内外研究现状

### 1.2.1 推荐系统的研究现状

推荐系统诞生于 20 世纪 90 年代，由于人们获取信息的需要，推荐系统一经诞生就受到了广泛的关注和研究，并且经过二十多年的积累和沉淀，推荐系统不仅成为互联网时代的高效检索工具，也逐渐发展成了一门独立的学科，可以说在学术研究和工商业应用上都取得了很多成功。

国外对于推荐系统的研究比较早，1994 年明尼苏达大学 GroupLens 研究组推出第一个自动化推荐系统 GroupLens[8]，这个系统使用的就是协同过滤算法，由于出现得很早而且效果很好，很快就被人们熟知。正是这个推荐系统体现出的巨大的使用价值，使得推荐系统开始受到广泛的研究。

21 世纪后，推荐算法的研究更加深入，对其划分也越来越细。2005 年 Adomavicius 等人的综述论文将推荐系统分为基于内容的推荐、基于协同过滤的推荐和混合推荐[9]。

由于商业界的关注和使用，推荐系统的研究进展越来越快。2006 年 10 月，由 Marc Randolph 和 Reed Hastings Netflix 两位创始人在美国硅谷创建的 Netflix 公司组织的提升本公司推荐算法的竞赛，吸引了来自世界各地的参赛队伍[10]，他们对 Netflix 公司现有的推荐算法进行了努力的研究和改进，这次比赛的成果显著，而且由于受众面很广、关注的很高，在很大程度上推动了推荐系统的发展。

2015 年，随着推荐算法的广泛使用，人们越来越在意其推荐准确性如何，Alan Said 等人开始对推荐系统的效果进行评估，并在 RecSys 会议上发表了相关论文[11]。

2016 年，RecSys 会议开始举办有关推荐系统和深度学习的相关研讨会，目的在于将当下最热门的深度学习技术融入到传统的推荐系统中去。同年，YouTube 视频网站对外宣布，他们成功地在其视频推荐系统中使用了深度神经网络的技术[12]，使得用户可以在大量数据中找到最想要的信息。

2017 年 AlexandrosKaratzoglou 等人在论文中更加详细地阐述了基于深度学习的内容推荐算法和协同过滤算法[13]，可见推荐系统在新技术的支持下正在经

历着前所未有的改革创新。

从推荐系统诞生至今已经有 25 年左右了，经过诸多学者的研究、工商业的广泛关注，推荐系统应用到了许多领域，如视频、社交网络、音乐、电子商务、广告等等。随着移动互联网的普及，推荐系统更是渗透到了人们生活的方方面面，在大数据的时代背景和深度学习的理论指导下[14]，推荐系统焕发了新的活力，开始不断尝试新的领域，如移动社交数据、地理数据等等。

不过推荐系统也带来了不少负面影响，其中之一就是“过度推荐”。过度推荐是指不断地给用户推荐某一方面的物品，导致其接触面越来越窄，这是精确推荐带来的坏处。更加让人感到不安的是隐私泄露问题，有些商家为了获取用户的信息，可能会从各个角度收集用户信息，甚至侵犯了用户的隐私，对于商家来说，这样的做法是不可行的，对于用户来说，应该更加注重保护自己的个人隐私。

### 1.2.2 Hadoop 的研究现状

Hadoop 是当今最流行的分布式计算系统，并且是开源的，个人使用者或者商家都可以根据自己的需求对源码做修改，这也促生了许多的第三方类库的诞生。然而由于不是组织运营，所以开源框架最突出的问题就是没有统一的规范，这会导致兼容性的降低。国外的分布式计算应用已经非常广泛，但是对于大多数国内 IT 工程师来说，分布式计算仍是触及较少的领域，这也是为什么国内大数据发展速度并不快的原因之一。

国外对于 Hadoop 的应用已经很普遍了，Yahoo 是最先将 Hadoop 投入到应用中的企业，早在 2008 年，Yahoo 已经在 2000 多个节点上执行超过 1 万个 Hadoop 虚拟机来处理超过 5PB 的数据了[15]。

Facebook 主要使用 Hadoop 进行日志和数据的存储，然后会利用这些数据作为推荐系统的输入。 此外 Facebook 为了更高效地解决存储遇到的问题，还建立了 Hive 数据库框架，并且发展成为基于 Hadoop 的 Apache 一级项目[16]。

其他的公司诸如 Amazon、Adobe、ebay 等等都搭建了自己的分布式集群，或为商品提供搜索索引，或为顾客提供推荐计算。

国内对于 Hadoop 的应用主要集中在几个大的互联网厂商。百度作为国内最老牌的提供搜索服务的公司，对 Hadoop 的关注从 2006 就开始了，截止到 2012

年，百度已经有近十个 Hadoop 集群，单集群的主机数量达到 3000 台左右。百度也对 Hadoop 源码进行了改写，开发了 HadoopC++扩展 HCE 系统。阿里巴巴为了给淘宝、天猫、 聚划算、支付宝等应用提供存储和计算服务，也用到了 Hadoop 集群，整个集群内有几百个用户组，每个组内有 5000 个主机节点。腾讯对 Hadoop 的使用也很早，截至 2012 年，腾讯最大的一个集群内约有 2000 多个主机节点，由于建立了多个集群，其总的主机数量已经达到了 6000 台左右，规模庞大的分布式集群为腾讯各个产品线提供了基础的计算和存储服务。其他公司诸如华为、奇虎 360、中国移动等等都在自己的业务或者内部系统中使用到了 Hadoop 框架。

由此可见随着时代的发展，大数据是必然的趋势，对大数据和分布式的研究是非常有意义和价值的。

## 1.3 论文的主要内容

本文的主要研究内容是利用 canopy 算法和 Hadoop 框架对 K-means 聚类做出了优化，然后应用到基于用户的协同过滤算法中去，并在此基础上开发了一个电影推荐系统，该系统遵循系统开发的要求，各个模块的耦合性较小，可扩展性强，可独立部署。本文对当前主流的系统开发技术进行对比，选择了 Spring Boot 作为项目的主要框架。论文的主要贡献包括以下几点：

1） 引入了用户特征属性解决新用户进入系统所遇到的冷启动问题，引入电影偏好属性来重新定义了用户-评分矩阵。引入时间因子来协调用户特征属性和用户电影偏好属性的比例。该系统能够高效地在用户使用的各个阶段发挥作用。

2）在聚类过程中，利用 canopy 的粗聚类对 K-means 的输入参数进行优化，对初始 K 值进行优质选取，降低聚类的计算量，提高准确性。

3）为了避免了单一推荐算法存在缺点，本文基于混合推荐的思想进行算法改，既保证了推荐的多样性也保证了准确性，避免了对用户在某一领域的过度推荐。

## 1.4 论文结构

本论文共分为六章，具体的章节内容概括如下：

第一章：绪论，主要介绍了推荐系统的作用、现阶段人们对于电影推荐的需

求程度，随后对研究的课题所涉及的框架、算法等等做了国内外的调研，主要是推荐系统和 Hadoop 平台这两部分，最后总结了本文的创新点。

第二章：相关技术介绍，本章分为三个部分，第一部分介绍了推荐算法的分类以及各个算法的优缺点比较，第二部分介绍了 Hadoop 平台的由来和发展应用，以及它在大数据处理中的优势，第三部分介绍了 Spring Boot 开发框架。

第三章：基于 MapReduce 的推荐算法的改进与实现，本章是本文的主要创新点，提出了对聚类算法的优化和分布式实现，并且应用到系统中去。引入用户特征属性解决冷启动问题，并提出时间因子来协调用户特征属性的权重。在基于用户的协同过滤推荐算法下混合基于内容的协同过滤推荐算法[17]，既保证了多样性又保证了准确性。

第四章：系统需求分析和系统设计，主要分析了系统的主要需求和功能，并根据需求进行系统的架构设计。

第五章：系统实现与测试，主要描述了系统开发的整个过程，根据设计对各个模块功能进行了编码实现。开发完成以后对系统进行测试，检测系统的各个功能是否按照预期的功能需求进行实现。

第六章：总结与展望，对本文的研究内容、过程和结果的工作总结，并且指出本文研究需要再多加改进的地方，以待未来的进一步研究。

# 第二章 相关技术介绍

## 2.1 推荐系统算法

前面介绍过，推荐系统的核心是推荐算法，推荐算法使用用户的信息，经过一系列的计算、筛选，得到用户潜在的喜好物品，不同的算法需要的数据不同，计算方法不同，所得到的结果也是不同的。面对不同的场景，需要使用适合的算法，这样才能发挥出推荐算法的最大效用。接下来对主流的推荐算法做一个详细的介绍，并通过对比其优缺点来阐述应该在什么情况下使用哪种算法。

### 2.1.1 协同过滤 (Collaborative Filtering Recommendations)

协同过滤推荐算法是推荐算法中出现较早，使用时间较长的算法之一，是一个典型的利用群体中其他单位的信息来做决策的方法，利用群体的大多数个体的行为推导某一个个体的行为。该算法认为，如果一些用户有相似的偏好或者类似的行为，那么这些用户在将来也会具有相仿的偏好和行为。简单来说是利用拥有共同经验的群体来获得推荐列表，然后展示给目标用户。协同过滤分为两种，第一种是基于用户的，另一种是基于物品的[18]。

基于用户的协同过滤算法的思想是同一类中的不同的人可能对不同的物品有过评价信息，评价不仅仅局限于特别感兴趣的，有时候不感兴趣的评价或者较低的评分也是非常重要的，记录下这些评价，然后对目标用户还没有消费过的物品做推荐的预测计算，以预测的分数为基准，分数越高，说明用户越有可能喜欢这个物品，分数越低，说明用户对这个物品不感兴趣，也可以将分数最高的前 N 个物品汇总、筛选，得到一个推荐列表。使用数学模型来说就是将用户和商品之间建立一个向量的关系，向量的各项是用户的属性。首先计算用户之间的相似程度，选择最接近的几个邻居后，根据邻居的相似度权重和对物品的评分，计算出当前用户没有消费过的物品的预测分数，得到一个商品列表作为推荐，如图 2-1 所示：

| 用户/物品 | 物品A | 物品B | 物品C | 物品D |
|---|---|---|---|---|
| 用户A | √ | | √ | 推荐 |
| 用户B | | √ | | |
| 用户C | √ | | √ | √ |

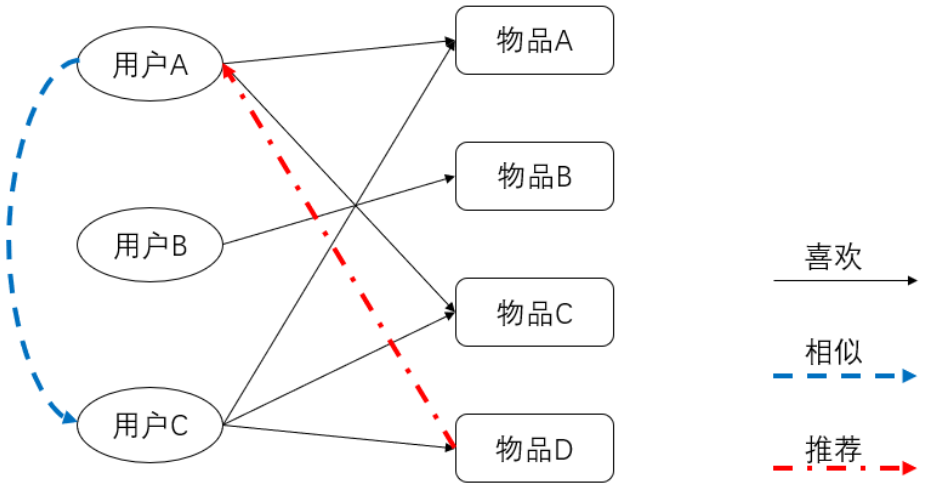


图 2- 1 基于用户的协同过滤算法示意图

从图中可以看出，用户 A 喜欢物品 A 和 C，用户 C 也喜欢物品 A 和 C，那么就认为二者是近邻，用户 C 喜欢物品 D，因此向用户 A 推荐物品 D。这只是一个非常简单的例子，如果数据量很大，用户向量的属性太多，那么寻找最近邻会需要很大的计算量，这也是该算法优化的重点。

基于物品的协同过滤算法的原理和上述算法类似，区别在于前者是将所有用户对某个物品的偏好作为一个向量来计算物品之间的相似度，得到物品的最近邻后，预测当前用户还没有偏好的物品，具体做法是根据用户历史的偏好对最近邻计算推荐度得到一个倒序排列的物品推荐列表，如图 2-2 所示：

| 用户/物品 | 物品A | 物品B | 物品C | 物品D |
|---|---|---|---|---|
| 用户A | √ | | | 推荐 |
| 用户B | √ | √ | | √ |
| 用户C | √ | | √ | √ |

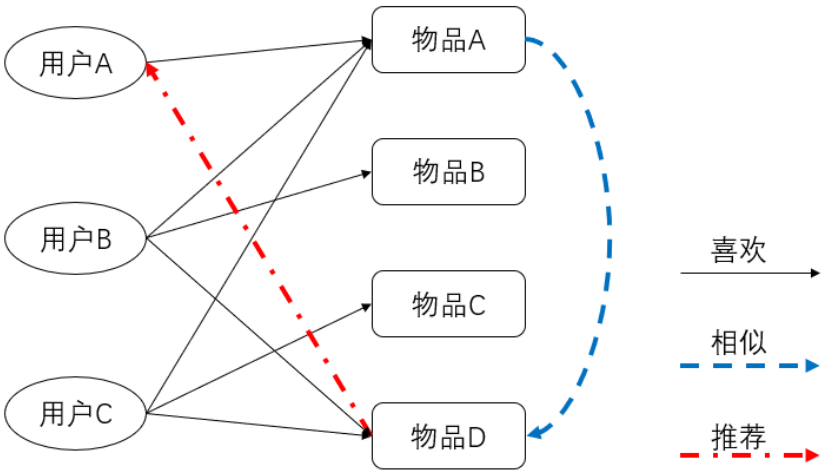


图 2- 2 基于物品的协同过滤算法示意图

从图中可以看出，如果一个用户购买了物品 A，那么他也会购买物品 D，由此得出物品 A 和物品 D 就是近邻。现在用户 A 喜欢物品 A，所以为用户 A 推荐物品 D。该方法在大量数据的情况下同样面临着相似度计算量大的问题。

协同过滤算法的优点是可以利用其他人的经验做决策，不需要对物品内容进行非常完全和准确的分析，因为群体的经验能够对于一些难以表述的概念如个人品味、情感倾向等进行过滤。由于需要群体信息，所以该算法存在的缺点就是必须有足够多的数据，如果是刚刚上线的系统，很可能会出现推荐不准确或者无法推荐的情况。

### 2.1.2 基于内容的推荐(Content-Based Recommendations)

基于内容的推荐算法顾名思义，根据用户过去消费过的或者有过评分的物品可以得出该用户对哪一类的物品比较偏爱，然后为其推荐这一类的其他物品，往往需要对物品进行相似度的计算[19]。当用户 A 喜欢 A 电影的时候，推荐系统会计算和 A 电影类似属性的其他电影，将结果降序排列得到推荐列表，推荐给用户 A，这显然是合理的。由于推荐的物品是用户之前消费过的物品，所以基于这种推荐

算法的物品很有说服力，如图 2-3 所示：

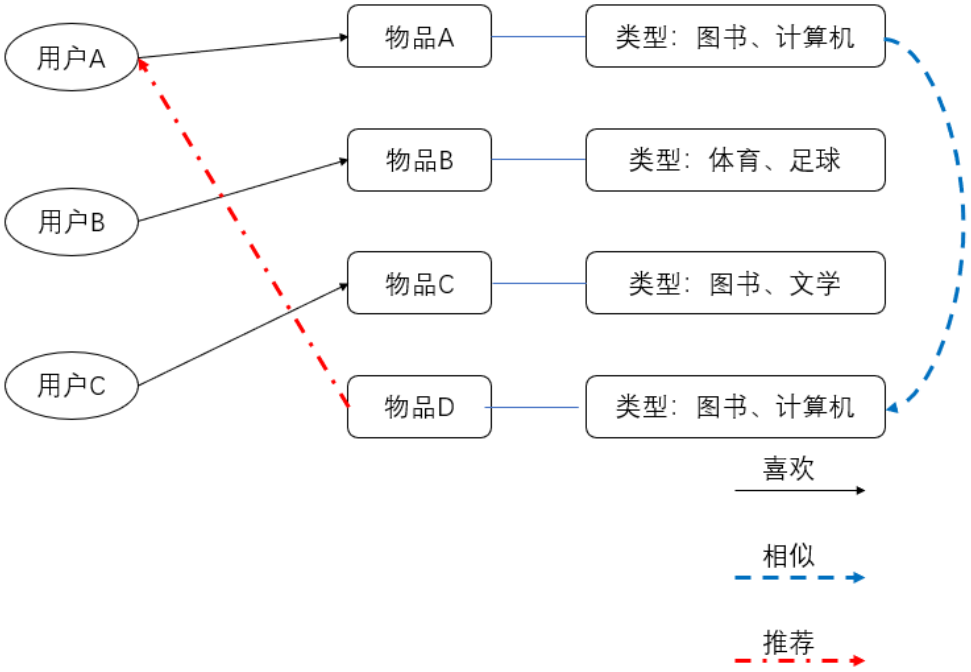


图 2- 3 基于内容的推荐算法示意图

基于内容过滤推荐的优点是，它不需要大量的用户，可以解决协同过滤推荐系统的冷启动问题[20]，而协同过滤推荐算法需要大量的用户，所以当用户数量不足的时候，可以使用基于内容的推荐。基于内容推荐的缺点就是，由于推荐的物品基本上都与用户喜欢的物品相类似，容易让用户陷入过度推荐的窘境，只能接触到某一类的物品，欠缺多样性。还有一点就是内容的相似度计算，对于电影、图书、音乐来说，单单用比如名字、国家、作者等作为属性可能会对内容反映得不够精确，所以基于内容的推荐可以辅助其他推荐算法一起使用。

### 2.1.3 混合推荐（Hybrid Recommendations）

在实际的生产开发中，通常都是几种甚至更多的推荐算法一起使用，这就是混合推荐。使用混合推荐的原因是，它可以消除某一种单一推荐算法的缺点。通过多种推荐算法构成的系统，推荐效果会大大增加。混合推荐算法的存在形式通常是以下几种：

1）使用多种推荐算法进行独立的计算，得到多个独立的推荐列表，然后对得

到的推荐列表取并集并且进行筛选已经推荐过的物品，得到最终的推荐列表。

2）使用多种推荐算法进行独立的计算，得到多个独立的推荐列表，然后给每一个列表设置不同的权重，进行加权、降序、筛选等操作，得到最终的推荐列表。

3）使用若干种算法互相辅助合作，如在使用矩阵分解算法时候，可能会遇到填充缺失值，这时候可用基于内容的推荐算法，对一些缺失值进行补充，以此解决稀疏矩阵带来的问题[21]。

4）使用多种推荐算法组合，即前一个算法的输出是后一个算法的输入。

## 2.2 Hadoop 框架

Hadoop 平台包含很多组件，目的是在处理海量数据的时候能有一套成熟的工具，包括 HDFS、MapReduce、Hbase、Hadoop 内核、Hive 和 pig 等，如图 2-4 所示：

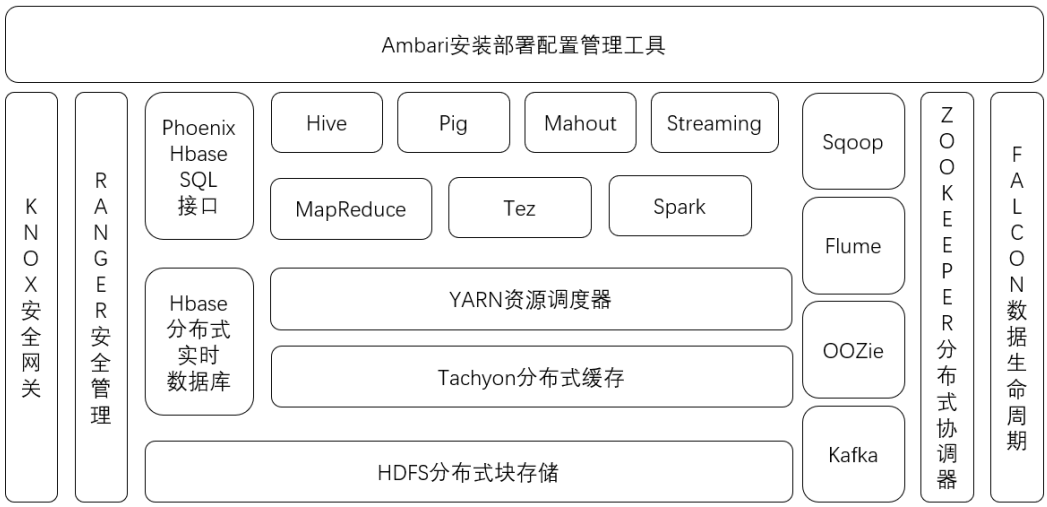


图 2- 4 Hadoop 架构图

其核心的三大组件分别是 HDFS、MapReduce 和 YARN，接下来对这三个主要的组件做一下介绍。

### 2.2.1 HDFS 分布式文件系统

HDFS 的全称是 Hadoop Distributed File System，即谷歌分布式文件系统，

是 Hadoop 的一个非常重要的子项目，为分布式计算提供数据存储和管理的服务，目的是为了满足流数据访问模式和超大文件的处理需求。HDFS 所具有的高容错、高可靠性、高可扩展性、高获得性、高吞吐率等特征为海量数据提供了不怕故障的存储，为超大数据集的应用处理带来了很多便利[22]。HDFS 的设计理念是将大文件分成很多小文件，然后分别存放在服务器上，而且为了提高安全性，可能每一个小文件会有若干个副本。各类分布式运算框架如 MapReduce，Spark 等可以使用 HDFS 来实现数据的存储。

HDFS 虽然具有高容错性、可运行在廉价机器上、适合大数据处理等等优点，这些优点也为 Hadoop 的高效运行提供了支持，但是 HDFS 也有一定的缺点，如不适合对小文件的操作、不支持并发写入、不支持修改，所以在使用 HDFS 的时候一定要趋利避害。

HDFS 是一个主/从（Mater/Slave）体系结构，HDFS 集群拥有一个 NameNode 和一些 DataNode，NameNode 负责管理文件系统的元数据，DataNode 存储实际的数据[23]。HDFS 的架构如图 2-5 所示：

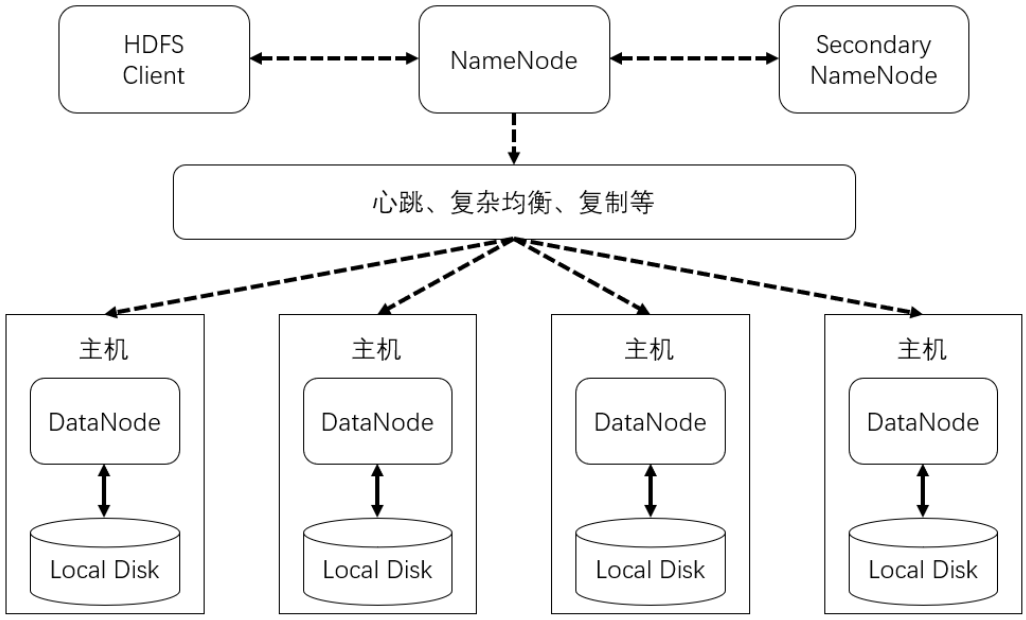


**图 2- 5 HDFS 架构图**

HDFS Client 是整个架构的用户交互入口，用户可以通过在 Client 中输入命令来管理、访问 HDFS。最常用的两个命令就是读文件和写文件。在进行数据读取时，Client 向 NameNode 发送请求，获取文件的位置信息，然后再从指定的节

点读取数据，如果是一个数据被分成多个块，还会有一个自动打包的过程。在进行数据写入时，Client 会先判断文件大小，如果文件超过预定值，就会把它切分成若干个子块，分别存储在不同的机器节点上。NameNode 负责管理 HDFS 的名称空间、管理数据块的映射信息以及配置副本的个数和大小等等[24]。DataNode 则是在接收到 NameNode 的指令后，真正存储数据和执行读写操作的角色。由于 NameNode 的作用非常重要，所以为了安全起见，整个架构还会有一个 SecondaryNameNode，从名字可以看出它是对 NameNode 的辅助，保存了最新的元数据检查点 fsimage 和自最新检查点后的命名空间的变化 edits，会定期合并、推送给 NameNode，过程如图 2-6 所示：

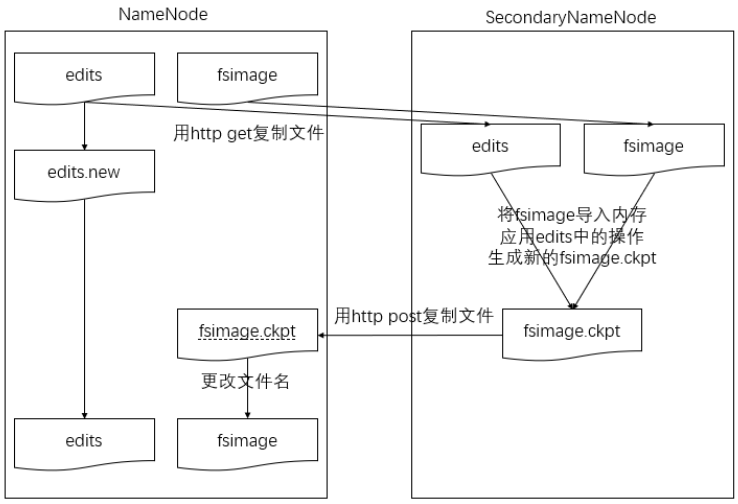


图 2- 6 SecondaryNameNode 合并 fsimage 和 edits 过程图

### 2.2.2 MapReduce 分布式计算框架

MapReduce 是 Hadoop 的分布式计算框架，能将海量数据处理的过程拆分为 map 和 reduce 两个阶段，其设计思想如图 2-7 所示：

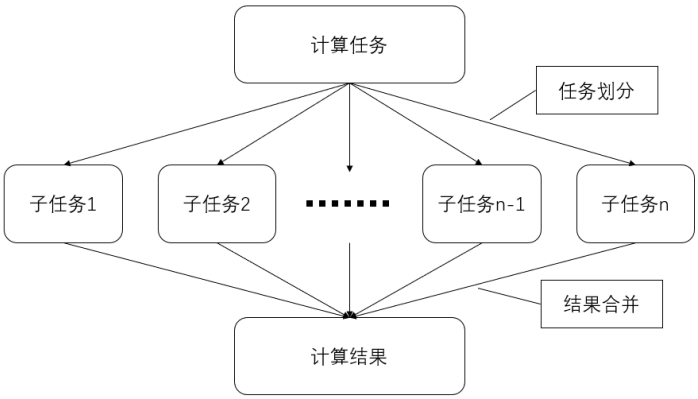


图 2- 7 MapReduce 设计思想图

Google 于 2004 年发表了的一篇论文《MapReduce:Simplefied Data Processing on Large Clusters》，里面写的是 MapReduce 的主要设计思想[25]。Map 阶段主要对杂乱无章的原始数据进行处理，每个节点读取自己的文件块，经过一系列的处理得到一组一组的 key 和 value，接着经过 Shuffle 的排序，到了 Reduce 阶段数据已经按照预定的规则归纳好，在此基础上可以做进一步的处理。MapReduce 的推出对大数据处理来说影响巨大，并且几乎成为了大数据处理的工业标准，也是目前为止最成功、使用范围最广的大数据并行处理技术。

对 MapReduce 一词，可以有 3 种理解，首先它是一个基于集群的高性能并行计算平台。这里的集群是指市面上可以广泛购买到的普遍服务器构成的计算集群，往往一个集群里包含着成百上千个节点，也正是通过这一点，Hadoop 才能在普通条件下拥有很强的计算力。MapReduce 的第二个含义是一个并行计算与运行软件框架。这个框架可以自动划分计算数据和计算任务，然后将任务按照规则自动分配到集群的节点上，计算完成后自动收集结果并返回。正是由于这一系列设计精良的过程，大大简化了开发人员的工作，让他们能更加专注于业务逻辑。MapReduce 的第三个含义是并行程序设计模型与方法，提供了并行编程接口，开发人员只需要实现接口的相关方法就可以了。MapReduce 计算框架的详细流程如图 2-8 所示：

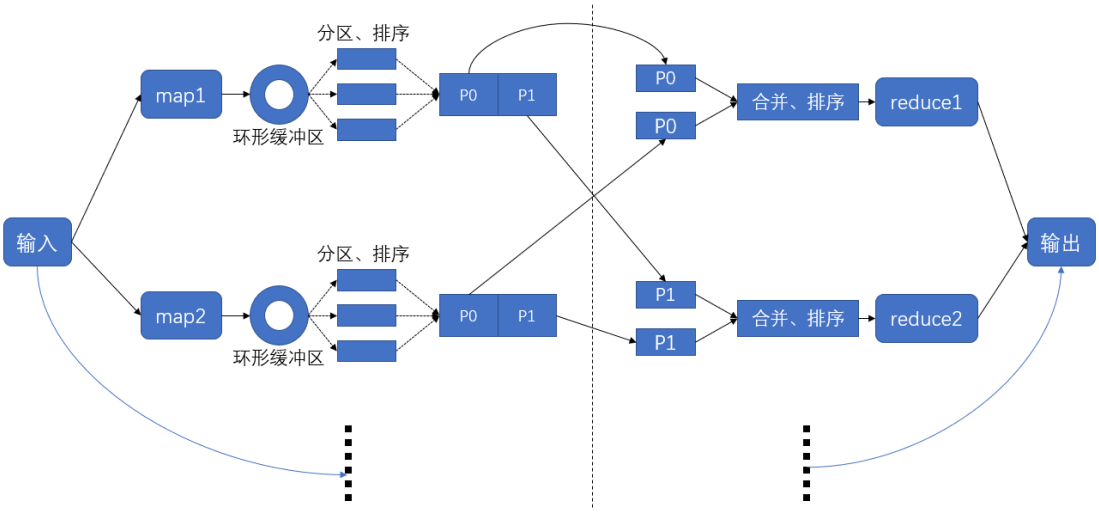


图 2- 8 MapReduce 计算框架示意图

第一个阶段是输入切片，如果输入的文件很大，就会将其切分为若干个切片，每个切片进入一个指定的 map()方法。第二个阶段 Map，接收切片后的文件，并循环处理一组输入，获得键值对，产生的结果作为输出。第三个阶段 Combiner，这是可选的阶段，对 map 函数计算出的中间文件进行一个简单的合并操作，需要注意的是，Combiner 的使用前提是必须不能破坏程序的逻辑，比如计算平均值就不适合使用 Combiner。第四阶段是 shuffle，是衔接 map 和 reduce 的一个过程，也是 MapReduce 优化的重点，该过程包括向环形缓冲区写文件、溢写文件到磁盘，然后 Partition 操作会找到对应的 map 输出文件进行复制操作，作为下一阶段的输入[26]。第五阶段 Reduce，对输入的数据进行最终的合并操作，将得到结果写入磁盘。

### 2.2.3 YARN 资源管理

YARN 是一个资源调度平台，如果把 Hadoop 比作一台分布式的计算机，MapReduce 就是一个可运行的程序，而 YARN 就是操作系统。YARN 是在 Hadoop2.0 后出现的，它出现的目的就是为了把资源管理和任务调度功能分开。YARN 并不知道程序是如何运行的，只是单纯地负责提供程序运行所必需的资源[27]，它的运行流程如图 2-9 所示：

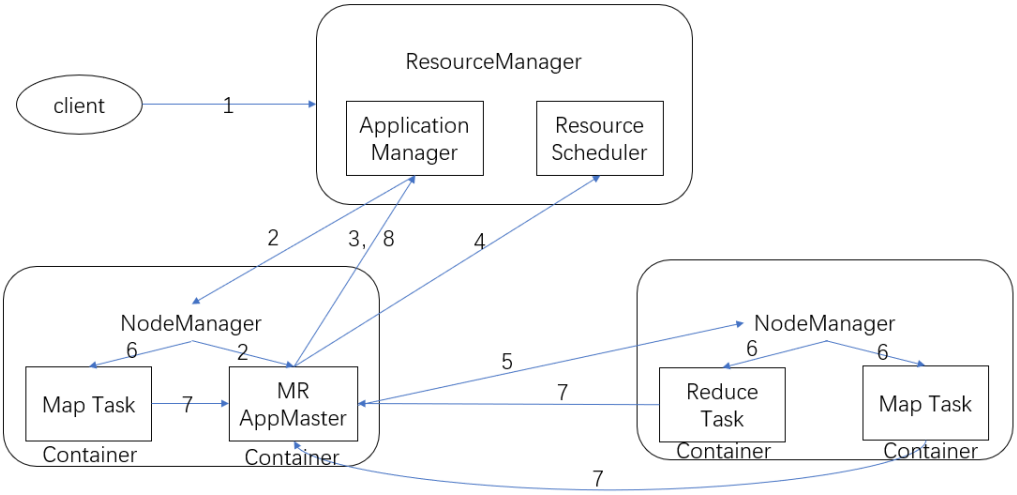


图 2- 9 YARN 运行流程示意图

图中有四个主要的角色，第一个是 ResourceManager（以下称为 RM），负责协调和管理整个 Hadoop 集群，扮演着主控节点的角色，当用户提交一个计算任务后，RM 就会启动一个分散在各个 NodeManager 节点的 MRAppMaster。第二个是 MRAppMaster，负责文件的分块、程序运行所需资源的申请以及任务的监控和容错处理。第三个是 NodeManager，是真正提供资源的角色，它们会处理 RM 和 MRAppMaster 的命令。第四个是 Containiner，它是一个资源抽象，里面封装了如内存、CPU、磁盘、网络等资源。

在进行作业调度的时候，目前主要有 FIFO、Capacity Scheduler 和 Fair Scheduler 三种调度方式，默认的是 Capacity Scheduler 容量调度器，策略是使用多个队列，每个队列配置一定的资源类并且使用 FIFO 的策略，这保证了任务的并发性。通过计算每个队列中已经运行的任务个数与该队列的资源的比值，比值越小，则该队列越空闲，应该优先接收任务，反之则反，如图 2-10 所示。

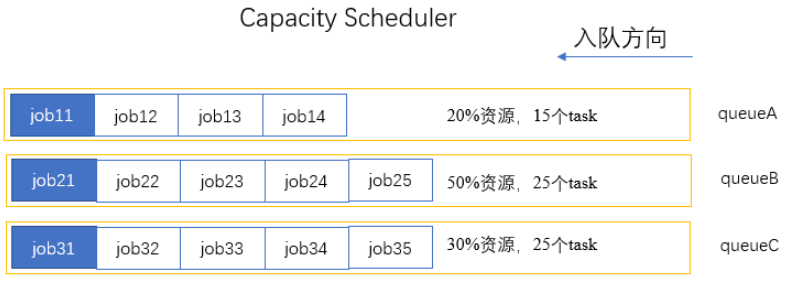


图 2- 10 Capability Scheduler 示意图

## 2.3 Spring Boot 框架

在 21 世纪初，java 企业级应用使用最多的开发方案是 J2EE，其中应用的业务逻辑会由 EJB 抽取、封装形成一个组件，这些组件会运行在一个独立的服务器上。2002 年，Rod Johnson 编写了 interface21 框架以解决传统 J2EE 中开发、测试和部署难的问题。两年后，他推出了《Expert One on one J2EE without EJB》[28]，并且在 interface21 的基础上编写了 spring 框架。随着互联网的兴起，企业应用开发越来越普遍，Spring 很快就占据了 Java 开发的头号地位，并且在 Java 不断推出新版本的同时，也能与时俱进不断地推陈出新，Pivotal 团队更是在 2014 年基于 Spring4.0 开发了 Spring Boot。Spring Boot 的初衷就是为了简化 spring 的配置，使得开发中集成新功能时更快，简化或减少相关的配置[29]。

在使用传统的 Spring 去做 Java EE 开发中，项目往往存在着大量的 XML 文件，导致 J2EE 项目变得越来越臃肿，为了能够整合第三方的框架，需要十分繁琐的配置，这些不可避免地会导致开发和部署效率的降低。Spring Boot 的目的并不是用来替代 Spring 框架，而是在其基础上提高开发者的体验，更像是框架的框架。它集成了大量无需配置的常用第三方库包，由于配置代码的减少、配置过程的简化，这使得开发者能够更加专注于编写业务的逻辑代码，极大地提升了开发效率。Spring Boot 继承了 Spring 的优良特性，在此基础上还进一步增加了简化编码、配置、部署、监控的功能[30]，主要特点有以下几个：

简化编码：基于“习惯优于配置”的理念，让开发者不用或者只需要很少的 Spring 配置就能创建一个独立运行的、准生产级别的 Spring 项目。

简化配置：以前使用 Maven 进行依赖的管理时，依赖都是单独配置和使用，而 Spring Boot 以功能模块为单位，将某个常用功能所需要的全部依赖包装成了一个 starter，常用的 starter 如表 2-1 所示：

表 2- 1 常用 starter 表

| | |
|---|---|
| spring-boot-starter-web | 支持全栈 web 开发 |
| spring-boot-starter-data-jpa | 使用基于 Hibernate 的 SpringData JPA |
| spring-boot-starter-thymeleaf | 在应用中使用 thymeleaf 渲染视图 |

| | |
|---|---|
| spring-boot-starter-test | 提供对常用测试框架的支持，如 Junit |
| spring-boot-starter- log4j | 引入默认的 log 框架——logback |
| spring-boot-starter-jdbc | 支持 Spring JDBC |

简化部署：在使用 Spring 开发完项目后，部署往往是比较繁琐的，要先在服务器上部署 tomcat，然后将项目打包放到 tomcat 里。有了 Spring Boot 以后，不需要在服务器上去部署 tomcat，因为 Spring Boot 内嵌了 tomcat，只需要将项目打成 jar 包，使用 java -jar xxx.jar 就能一键式启动项目，给部署带来了极大的便捷。

应用监控：Spring Boot 提供了基于 http、SSH、telnet 对正在运行的项目进行监控。

## 2.3 本章小结

本章主要介绍了本文涉及到的主要技术，一共分为两个小节。

第一节主要介绍了推荐算法的定义以及算法分类，详细说明了基于内容的推荐、协同推进和混合推荐的设计思想和优缺点。

第二节主要介绍了 Hadoop 框架的主要组件。

第三节主要介绍了 Spring Boot 框架的来源、设计思想、三大组件和相较于 Spring 给开发者带来的诸多便捷。

下一章将介绍本文对于推荐算法的改进，也是本文的主要创新点。

# 第三章 基于 MapReduce 的推荐算法的改进与实现

本文的推荐算法主体是基于用户的协同过滤算法，并在其基础上加以改进，下面结合本文的应用场景，详细地阐述该算法的流程。

假设有 n 个电影，m 个用户，每个用户会对自己观看过的电影有一个评分，把电影评分作为一个用户向量的属性值，只有部分属性是有评分数据的，其它部分是空白，这样我们就得到了用户-项目评分矩阵，如表所示：

**表 3- 1 用户-评分矩阵表**

| | M1 | M2 | M3 | … | Mn |
|---|---|---|---|---|---|
| U1 | R11 | Null | R13 | … | R1n |
| U2 | R21 | R22 | R23 | … | Null |
| U3 | R31 | R32 | Null | … | R3n |
| … | … | … | … | … | … |
| Um | Rm1 | Null | Rm3 | … | Rmn |

我们要做的就是利用已有的电影评分来预测某些用户和未观看电影之间的评分关系，找到其最有可能打高分的评分的电影，然后推荐给该用户。接下来要做的就是计算相似度，选择最近邻，根据近邻和相似度计算出推荐列表，流程如图 3-1 所示：

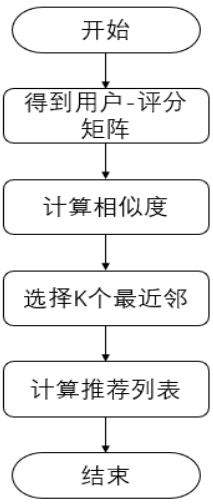


**图 3- 1 推荐功能流程图**

得到用户-评分矩阵后，第二步就是计算用户向量的相似度。计算相似度常用的方法有以下 3 种：

1.欧氏距离：欧氏距离是计算两个向量间距离最简单的方法之一[31]，适合用于用户向量稠密，且属性值大小十分重要的情况，计算公式如下：

$$d(x,y)=\sqrt{\sum_{k=1}^{n}(x_k-y_k)^2} \qquad \text{（公式 2.1）}$$

其中 $x_k$ 表示用户 x 的第 k 个评分，$y_k$ 表示用户 y 的第 k 个评分。欧氏距离越小表示两个向量的相似度越高，反之则越低，有时候为了能更直接地体现出相似度，可以将上面的公式修改成为：

$$sim(x,y)=\frac{1}{1+d(x,y)} \qquad \text{（公式 2.2）}$$

sim(x,y)的区间是(0,1]，越接近于 1 代表着两个用户向量相似度越高，反之则越低。

2.余弦相似度：计算两个用户向量夹角的余弦值，值越小说明两个向量的相似度越高，反之则说明两个向量的相似度越低[32]，计算公式如下：

$$\cos(x,y)=\frac{\sum_{k=1}^{n}(x_k\times y_k)}{\sqrt{\sum_{k=1}^{n}x^2}\times\sqrt{\sum_{k=1}^{n}y^2}} \qquad \text{（公式 2.3）}$$

其中 cos(x,y)表示了用户 x 和用户 y 的相似度，$x_k$ 表示 x 用户向量的第 k 个评分，$y_k$ 表示 y 用户的第 k 个评分。

3.Pearson 相关系数：用来衡量两个向量的相关程度，即两个向量存在同时增长或下降的趋势，它的结果区间是[-1,1]，当 r＞0 表明两个变量是正相关，一个变量的值越大，另一个变量的值也会越大；r＜0 表明两个变量是负相关，一个变量的值越大另一个变量的值反而会越小[33]。Pearson 相关系数计算公式如下：

$$r=\frac{\sum_{k=1}^{n}(x_k-\bar{x})(y_k-\bar{y})}{\sqrt{\sum_{k=1}^{n}(x_k-\bar{x})^2}\sqrt{\sum_{k=1}^{n}(y_k-\bar{y})^2}} \qquad \text{（公式 2.4）}$$

其中 $x_k$ 表示 x 用户向量的第 k 个评分，$y_k$ 表示 y 用户的第 k 个评分。 $\bar{x}$ 表示 x 用户的平均评分， $\bar{y}$ 表示 y 用户的平均评分。

本文采用的是欧氏距离进行相似度的计算。得到其他用户对于目标用户的相似度系数后，根据此系数将用户进行降序排列，取前 k 个得到相似邻居集合 U={$u_1$，$u_2$ … $u_k$}，预测目标用户 u 对于电影 i 的评分的计算公式如下：

$$P_{u,i}=\overline{r_u}+\frac{\sum_{u'\in U} S(u,u')(r_{u',i}-\overline{r_{u'}})}{\sum_{u'\in U}|S(u,u')|} \quad \text{（公式 2.5）}$$

其中 U 是 u 的近邻集合，S(u,u′)是 u 和 u′ 的相似度，$\overline{r}_{u'}$ 表示用户 u′ 的平均评分。这是计算单个评分的预测值，可以对用户 u 没有观看过的电影都进行计算，进而得到推荐列表。

上述基于用户的协同过滤算法只是理论上的过程，真正运用到实际中，有许多方面需要进行改进，本文主要对以下 3 个问题进行探讨：

1）如果用户是第一次进入系统，没有历史数据，那么系统无法使用协同过滤算法为其推荐。现实生活中电影的数量非常多，无法一一列举，所以用户-评分矩阵要根据的应用场景重新定义。

2）如果是少量的数据，那么在计算相似度的时候可以使用遍历的方法，把当前用户和每一个用户都比较一次，然后再排序，选择最接近的 k 个邻居。然而这种情况却不适合大数据背景下的应用。由于大数据背景的数据量太大，而且有的用户之间差别很大，根本不需要做相似计算就可以排除，如果还是使用遍历的方法逐个计算，那么时间会非常长，也造成了计算资源的浪费。所以在寻找近邻之前，可以对用户进行聚类，通过聚类把具有相似特征的用户分到一起，这样再在其中寻找最近的 k 个邻居，然后再进行后续的计算。既然使用了聚类算法，那么聚类的效率是可以提升的。

3）基于用户的协同过滤算法是有前提的，那就是相似用户在未来的喜好也会基本相同，虽然这个算法能保证推荐的多样性，但是与此同时也要在一定程度上保证推荐的准确性。

接下来就以上 3 点提出改进的方法。

## 3.1 对用户-评分矩阵的创新设计

本节将重新设计用户-评分矩阵，将特征属性和电影类型偏好属性作为用户的属性值，代替之前的用户对某个单独电影的评分。

### 3.1.1 引入用户特征属性

初次进入系统的用户，由于没有任何和推荐相关的数据，所以无法为其做针对性的推荐，只能给他推荐当下最热门的电影，这不符合本文所研究的个性化推荐系统的初衷。所以本文提出，引入用户特征属性，采集用户信息，以此来解决用户第一次进入系统的问题。收集用户信息的流程如图 3-2 所示：

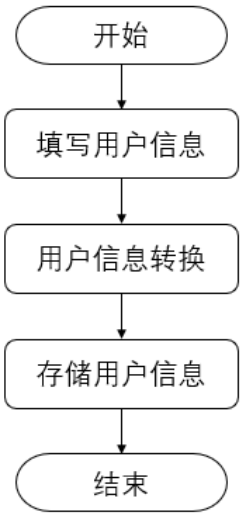


图 3- 2 用户信息收集流程图

研究发现，如果两个人的特征越相似，那么他们的行为爱好也很大程度是相似的，比如男生喜欢战斗类的电影，女生则更倾向于偶像剧，青少年喜欢休闲运动服，中年男女更偏好于商务或者正式服装。引入用户特征属性后，在聚类的时候也更加有效。引入的特征属性不宜过多，太多会导致用户的厌烦，不利于信息的正确收集，本文引入的特征属性有{年龄，性别，性格}，所以用户-评分矩阵可以扩展如表 3-2 所示：

表 3- 2 拓展的用户-评分矩阵表

| | Age | Gender | Char | M1 | M2 | M3 | … | Mn |
|---|---|---|---|---|---|---|---|---|
| U1 | | | | R11 | Null | R13 | … | R1n |
| U2 | | | | R21 | R22 | R23 | … | Null |
| U3 | | | | R31 | R32 | Null | … | R3n |
| … | | | | … | … | … | … | … |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Um | | | | Rm1 | Null | Rm3 | … | Rmn |

### 3.1.2 使用电影偏好代替电影评分

考虑到现实情况下电影的数量非常多，而且每个人看过的电影类型不尽相同，在这种情况下构造成用户-项目评分矩阵，会有矩阵庞大并且稀疏性极高的问题，本文将电影进行了分解，使用电影类型贡献值来构成用户电影偏好属性。

本文将电影类型分为{剧情，喜剧，爱情，传记，动作，历史，科幻，奇幻，悬疑，国外，港台，大陆，真人，动画，其他}，这些同时也是用户电影偏好属性的组成。通过让用户选择最近看过或者以前看过的电影，并且打分。假设一部电影的类型构成是{喜剧，动作，科幻}，用户对其评分为 r，那么该电影对用户的电影偏好贡献值就是 r{喜剧，动作，科幻}。通过不断地累积贡献值，得到一个用户的完整的电影偏好向量。当用户给一部电影评分的时候，需要有个转化的过程，电影评分到增益值的转化规则如表 3-3 所示：

**表 3- 3 转化规则表**

| 评分 | 类型增益 |
|---|---|
| 0-3 | 1 |
| 4-7 | 2 |
| 8-10 | 3 |

通过这样的转化，也能在一定程度上弱化不同用户的不同评分标准，比如有的用户觉得 8 分是一个比较好的评价，也有人觉得 10 分是比较好的评价，虽然分数不同，但是最后该用户的电影偏好的贡献值是一致的。于是用户-评分矩阵如表 3-4 所示：

**表 3- 4 加入电影偏好属性的用户-评分矩阵表**

| | Age | Gender | Char | 剧情值 | 喜剧值 | 科幻值 | … |
|---|---|---|---|---|---|---|---|
| U1 | | | | | | | |
| U2 | | | | | | | |
| U3 | | | | | | | |
| … | | | | | | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Um | | | | | | | |

### 3.1.3 引入时间因子

综合上述两个观点，目前用户向量的组成是{用户特征属性，电影偏好属性}，具体的属性名称是{年龄，性别，性格，剧情，喜剧，爱情，传记，动作，历史，科幻，奇幻，悬疑，国外，港台，大陆，真人，动画，其他}。随着用户观影数量的增加，那么{年龄，性别，性格}对于计算相似度的计算就显得没有那么重要，有时候甚至是有误导作用的，比如一个年轻人也会喜欢看历史片或者纪录片，一个年长的人也会喜欢看动作片或者动画片。所以本文创新性地引入了时间因子，来协调用户特征属性和电影偏好的比例。

用户的相似度由两部分构成，分别是用户特征相似度 $S_1$ 和电影偏好相似度 $S_2$，假设用户访问本系统网站的次数为 x，随着访问网站次数的增多，用户的电影偏好向量就越完整，那么用户的特征属性就应该越弱，最后总的相似度的计算公式如下：

$$S = S_1 \frac{1}{1+x} + S_2 \frac{x}{1+x} \qquad \text{(公式 3.1)}$$

## 3.2 Canopy-Kmeans 聚类算法的设计与实现

由于在基于用户的近邻推荐算法中要使用聚类，对聚类算法的优化也是一个主要的问题，是否能高效聚类也决定了推荐算法的性能。最常用的聚类算法是 K-means，这是一个简单实用的聚类算法，流程如图 3-3 所示。

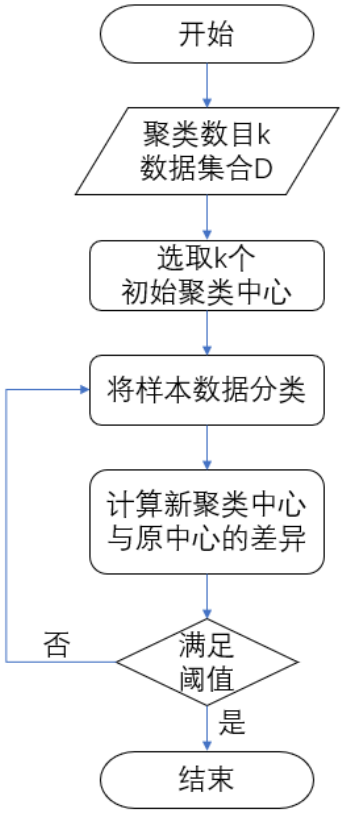


**图** 3- 3 K-means **聚类算法流程图**

K-means 的思想是这样的：假设有一个数据集合为{$x_1$，$x_2$，$x_3$，…}，并且各个数据对象是同维的，然后人为确定聚类的数量，并从原始的数据集中，随机选择 k 个向量，作为聚类的中心。在聚类数量 k 以及初始聚类中心确定后，算法就开始对整个数据集剩下的向量与初始聚类中心进行距离的计算，计算距离的方法主要是欧氏距离，根据距离来对剩下的向量进行划分。遍历结束之后，根据各个类别中的数据对象重新计算类的中心，目的是为了进行下一次迭代的聚类中心。经过反复迭代多次，直到达到了指定的迭代次数或者聚类中心的变化小于指定的阈值才停止[34]。算法的伪代码如下：

**Algorithm 1** K-means 算法流程

**输入:** 数据集 $D = x_1, x_2, ..., x_m$, 聚类簇数 $k$ , 迭代次数 $maxIter$

**输出:** 簇划分 $C = C_1, C_2, C_3, ..., C_k$

**过程:** 函数 K-means$(D, k, maxIter)$

```
repeat
    令 C_i = φ(1 ≤ i ≤ k)
    for j = 1, 2, ..., m do
        计算样本 x_j 与各 "簇中心" 向量 μ_i(1 ≤ i ≤ k) 的欧式距离
        根据距离最近的 "簇中心" 向量确定 x_j 的簇标记
        将样本 x_j 划入相应的簇
    end for
    for i = 1, 2, ..., k do
        计算新的 "簇中心" 向量 μ_i'
        if μ_i' == μ_i then
            将当前 "簇中心" 向量 μ_i 更新为 μ_i'
        else
            保持当前 "簇中心" 向量不变
        end if
    end for
until 当前 "簇中心" 向量未更新或者迭代次数达到 maxIter
```

K-means 的主要优点有：原理比较简单，实现很容易，收敛速度快；算法的可解释度比较强；主要需要调整的参数仅仅是簇数 k。然而他的缺点也是很明显的，主要缺点有：K 值的选取不好把握；采用迭代方法，得到的结果可能会是局部最优而非全局最优[35]；噪音或者异常点对结果的影响很大；由于取值的随机性，可能会存在算法的时间复杂度很高的情况；同样的样本，由于 K 值不同，会得到不同的结果，如图 3-4 所示：

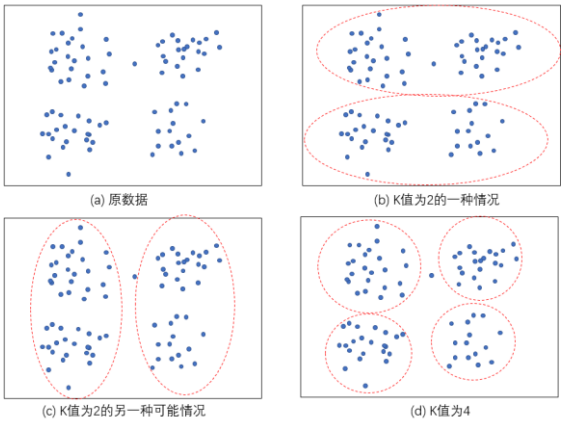


图 3- 4 不同 K 值结果对比图

K 均值的改进方法有很多，下面是两个比较常见的改进方法：

1．K-means++，与 K-means 算法的迭代部分内容几乎相同,改进的地方主要在于 k 个初始聚类中心的优质选取[36]。改变了之前从整个数据集中随机选取 k 个数据对象作为初始聚类中心的做法,而是认为优质的初始聚类中心之间的距离应该越远越好，并以此原则进行选取。在选取第一个聚类中心时因为没有对比，所以同样是使用随机的方法,但是当已经选好了 m 个聚类中心(0<m<k)后,在选取第 m+1 个时，计算与当前 m 个聚类中心的距离，距离越远，则被选中的概率越高。

2．ISODATA 算法，之前所说的 K-means 和 K-means++的聚类 K 值一旦确定后就不会改变了，这样算法的灵活度就降低了。而 ISODATA 算法在运算过程中，会对各个类别进行判断，并且用分裂和合并这两个操作来改变聚类的个数。分裂意味着聚类中心数增加，合并意味着聚类中心减少。为了判断执行哪种操作，ISODATA 算法需要额外设置几个主要的参数，分别是预期的聚类中心数目 K、每个类所要求的最少样本数目 N、最大方差 Sigma 以及两个类别对应聚类中心之间所允许最小距离 d。

K-means++的思想浅显易懂，而且不需要额外指定参数。ISODATA 算法虽然可以得到更加灵活的聚类个数，不过由于需要额外指定的参数比较多，因此 ISODATA 算法并没有 K-means++受欢迎。

另一种优化思路就是，尽量消除 K 值和 k 个初始聚类中心的随机性，本文引入 Canopy 算法进行 K 值和 k 个初始聚类中心的获取。Canopy 是一种非精确聚类算法，目的是可以用非常小的代价获得粗聚类的结果，得到聚类中心点的大概位置，然后再进行更细化的聚类[37]，虽然表面上步骤增加了，但是由于初始 K 值的优质选取，会使得后续的细聚类极大地提高效率，最后提高整体的聚类效果，很大的实际应用价值。Canopy 的整体算法流程如下：

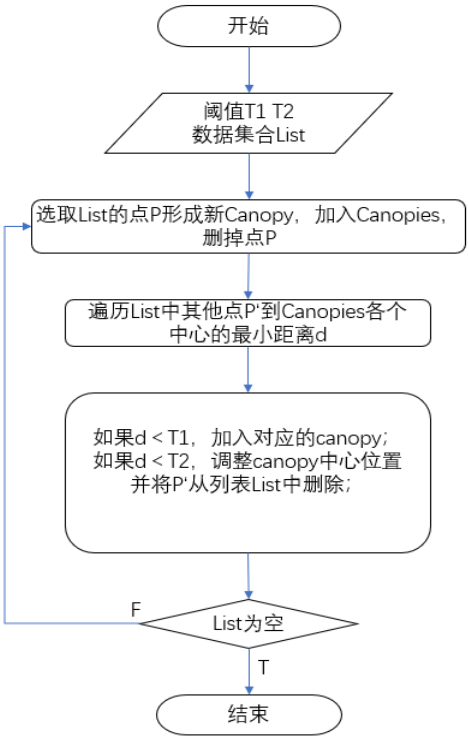


**图** 3- 5 Canopy **聚类算法流程图**

（1）将原始样本集随机排列成样本列表 List=[$p_1, p_2, \ldots, p_n$]，排列好后不再更改，设定初始距离阈值 T1、T2，且 T1>T2 。

（2）从列表 List 中随机选取一个样本 P 作为第一个 canopy 的中心，并将 P 从列表中删除。

（3）从列表 List 中随机选取一个样本 P′，计算 P′ 到所有质心的距离，考察其中最小的距离 d：

如果 d＜T1，则给 P′ 一个弱标记，表示 P′ 属于该 canopy，并将 P′ 加入其中；

如果 d＜T2，则给 P′ 一个强标记，表示 P′ 属于该 canopy，且和质心非常接近，随后将该 canopy 的中心设为所有强标记样本的中心位置，并将 P′ 从列表 List 中删除；

如果 d>T1，则 P′ 将会在迭代的过程中形成一个新的聚簇[38]。

（4）重复第三步直到列表 List 中元素个数为零，聚类结果示意如图 3-6：

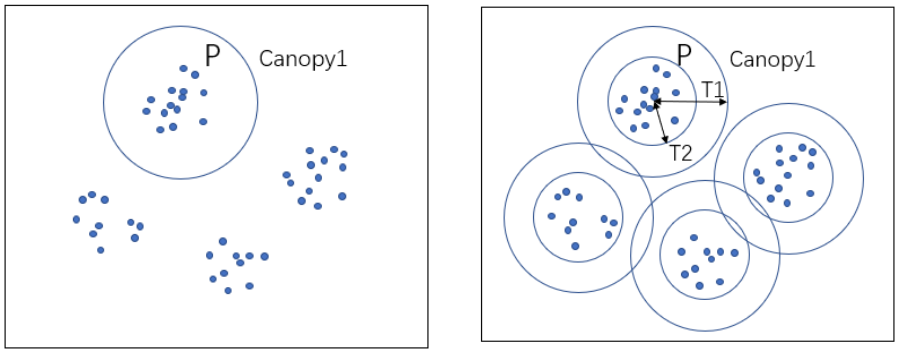


图 3- 6 Canopy 聚类示意图

算法的伪代码如下：

**Algorithm 2** Canopy 算法流程

**输入:** 数据集 $D = x_1, x_2, ..., x_m$, 距离阈值 $T_1$、$T_2$, 迭代次数 $maxIter$

**输出:** 簇划分 $C = C_1, C_2, C_3, ..., C_k$

**过程:** 函数 Canopy$(D, T_1, T_2, maxIter)$

初始化： $canopies = null, datalist = D$
在 $datalist$ 中随机选取 $x_1$
在 $datalist$ 中删除 $x_1$
**repeat**
  **for** $x_i$ in $datalist$ **do**
    **for** $canopy$ in $canopies$ **do**
      **if** distance($x_i$,$canopy$) < $T_1$ **then**
        canopy.add($x_i$)
        **if** distance($x_i$, $canopy$) < $T_2$ **then**
          在 $datalist$ 中删除 $x_i$
        **end if**
      **end if**
    **end for**
  **end for**
  调整 $canopies$ 的各个 $canopy$ 的中心点的值
**until** $datalist$ 为空或者迭代次数达到 $maxIter$

在使用 canopy 的过程中，距离计算的选择对 canopy 的分布非常重要[39]，本文使用的是最为简单的欧式距离。当 T2＜d＜T1 时，点 P′不会从列表中被删除，因为 P′ 很可能离另外一个簇类更近（即距离小于 T2），甚至自己会成为一个新的聚类中心。参数 T1、T2 的取值会影响 canopy 的重叠率及粒度，如果 T1 过大，会使某个样本属于多个类，聚类效果很差。由于孤立点或者干扰点会很大程度形

成新簇类，所以应该把包含样本点数目较少的族类删除掉，这个数目也是可以人为控制的。图 3-7 展示的是 canopy 自动获得 K 值的正确率。横坐标是类别个数，由此构成数据，并进行 30 次 canopy 聚类计算，得到每一次聚类 K 值然后计算出平均值，从图 3-7 可以看出，canopy 自动获取的 K 值和真实值非常接近。

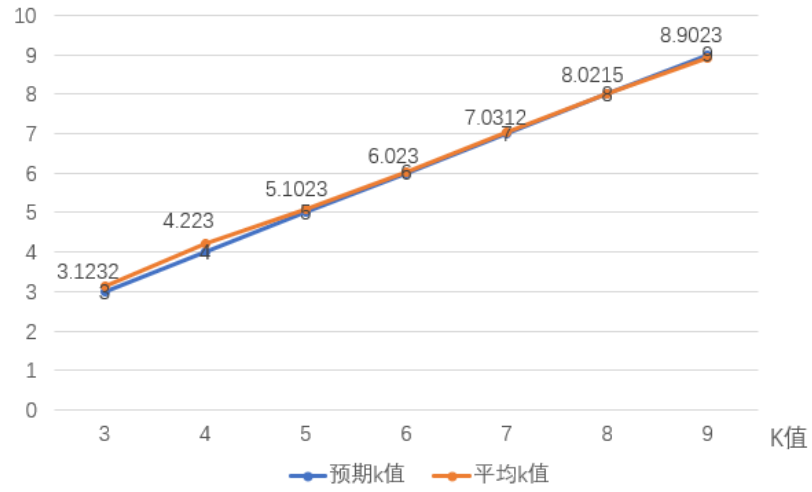


图 3- 7 Canopy 获取 K 值与真实 K 值对比图

获取到 K 值和 k 个初始聚类中心后，就可以进行 K-means 聚类了，为了使得效果更加明显，样本是通过 sklearn 的 make_blobs 方法生成的，该方法是专门用来产生聚类数据的，主要参数有样本点个数 n_samples、数据的维度 n_features、数据的中心点 centers、数据集的标准差 cluster_std。实验结果如图 3-8 所示：

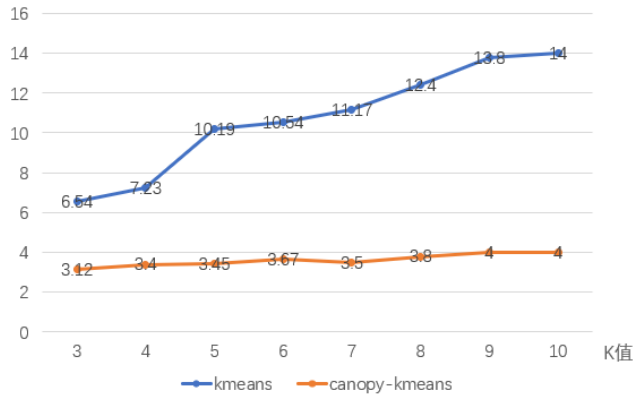


图 3- 8 canopy-kmeans 与 K-means 迭代次数对比图

图中显示的每个 K 值的数据集大小是 1000 个样本由于 canopy-kmeans 的初始聚类中心相比于 K-means 的随机选取更加合理，所以迭代的次数显著降低了。

接下来是验证聚类的准确率，依然是使用 make_blobs 方法提前生成数据，数据集如表 3-5 所示：

**表 3- 5 验证准确率的数据集表**

| 数据集 | 数据量 | 类别数 |
|---|---|---|
| DataSet1 | 127 | 4 |
| DataSet2 | 548 | 5 |
| DataSet3 | 923 | 6 |
| DataSet4 | 1228 | 6 |
| DataSet5 | 1593 | 7 |

分别对上述各个数据集合进行两种聚类算法操作，取 10 次结果的平均值。本实验给准确度做的定义如下：提前将数据点的类别标好，准确度的值是聚类算法运行后被正确分类的数据点与总数据点的比值，得到的准确度对比如图 3-9 所示：

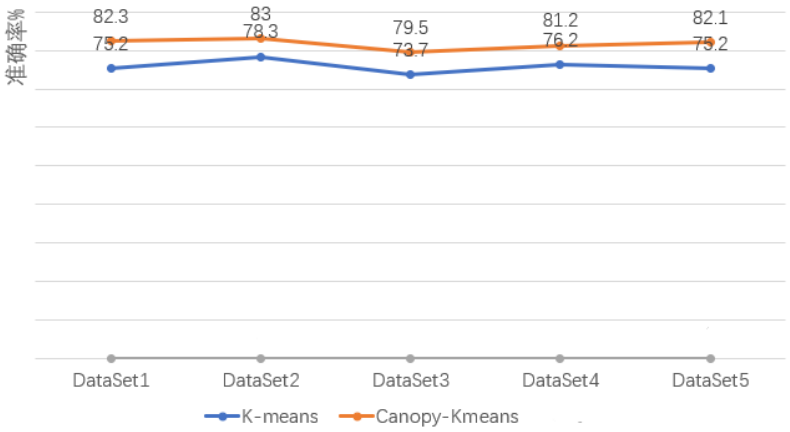


**图 3- 9 两种聚类算法的效果对比图**

可以看出改进后的聚类算法在各个数据集上都比原来的聚类算法的准确率更高，更高的准确率带来的结果是聚类的效果更好，那么将 Canopy-Kmeans 算法运用到电影推荐系统中以后，推荐的准确率也会相应提高。

虽然在聚类之前使用 canopy，会使得 K-means 聚类的效率提升，但是考虑到大数据时代的背景下，串行执行显然是不能满足需求了，而且就算是使用多线程，那么会对硬件提出很高的条件，也不利于实施，所以就需要利用大数据的技术，使得算法可以在集群上运行[40]，这样可以使用廉价的机器、多个节点运行算法，

使得算法的效率进一步提高。并行化的思路如图 3-10 所示：

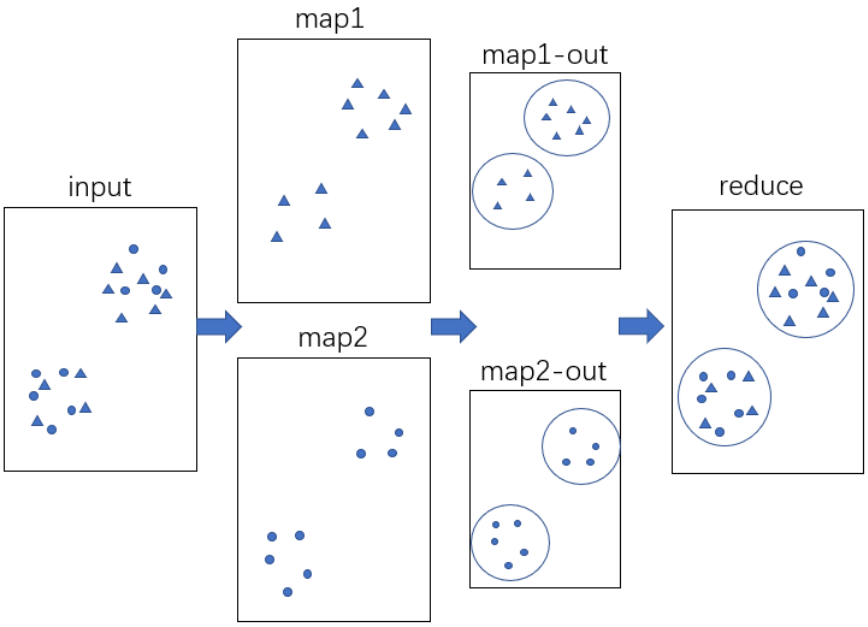


图 3- 10 算法并行化示意图

之前提到了 Hadoop 中 MapReduce 的架构和流程，所以一开始要将数据块切分，然后在每一个 map 上并行地进行 K-means 聚类。第一次 map 的输出，作为第一次 reduce 的输入，在 reduce 上归并 map 的聚类结果，这样就可以到得到全局数据的中心。在这个过程中，第一次 map 是对当前 node 的数据执行 K-means 算法，第一次 reduce 是对生成的群组进行筛选得到全局中心。筛选的依据是设定好的数据点的数量，如果 reduce 后的群组的数据点小于阈值，则该群组的孤立点过多，不能成为聚类中心，应该删除该群组。算法的并行化设计与未并行化的运行时间对比如图 3-10 所示：

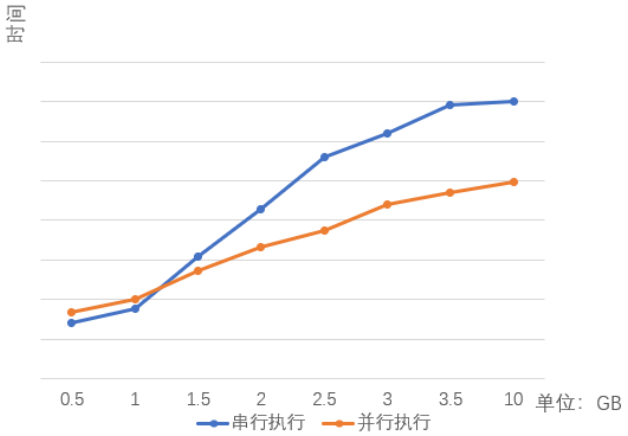


图 3- 11 并行化与非并行化时间对比图

从图中可以看出，一开始并行计算的时间会比串行计算要长，因为多节点计算需要在集群间进行通信，而且在数据量非常小的情况下，并行计算没有优势。但是随着数据量的增加，串行计算的时间增加的很快，而并行计算增加的缓慢，而且使用的时间更少。根据图中的趋势可以看出，当数据量非常大的时候，使用并行计算能够非常有效地提高算法的效率。

## 3.3 混合算法的设计

在基于用户的协同过滤算法中，用户得到的推荐是与他们类似的用户所看过的电影，存在一种情况：虽然有些电影不属于同一类的电影，但是这类的用户都喜欢。比如用户 A 和用户 B 有共同的电影偏好，用户 B 还喜欢历史类的电影，但是现在用户 A 并没有这类电影的表现，由于 A 和 B 属于近邻，可以基于用户的协同过滤，推荐给用户 A 历史类的电影，这样就体现了推荐的多样性。但是这样的推荐也是有风险的，因为很有可能用户 A 并没有喜欢历史类的电影，那么这样的推荐就是无效的。为了防止以上的问题，本文引入了混合推荐算法的思想，将基于内容的推荐算法也引用进来，并且由于电影内容的属性具有固定性，关于电影的聚类、相似度计算等可以在线下进行。在推荐的列表中，偶尔也加入基于内容的推荐算法推荐的电影。混合推荐算法使得推荐既有多样性，又有准确性，极大地提升了系统的用户体验。

基于内容的推荐算法需要对电影创建特征，所以对特征定义的准确性就提出了要求。有效的、合理的特征定义，可以更好地反映出用户的内容，而不准确的特征定义很可能会使得算法出错。

基于内容的推荐算法的过程是这样的：首先根据电影类型的不同，提取出电影的特征，而一部电影通常是带有很多属性的，比如科幻类的电影也会有喜剧的色彩，动画片中也会有爱情元素等等。考虑到每一部电影的类型占比不一样，可能在某部电影中科幻占了大多数，而喜剧只占了一小部分，所以不能简单地用{0，1}作为属性值，应该让用户根据自己的观看体验对属性值进行打分，最后把所有观看这部电影的人对该属性的评分的平均值作为该属性的最终得分，这是一个动态的过程。

对每一部电影做好定位以后，就可以进行聚类了。把一个电影看成一个向量，

计算相似度也同样是主流的 3 种方法：欧氏距离、余弦相似度和 Pearson 相似度，本文选用的是欧氏距离，计算公式如下：

$$d(a,b)=\sqrt{\sum_{k=1}^{n}(a_k-b_k)^2} \quad \text{（公式 2.6）}$$

其中 $a_k$ 表示电影 a 的第 k 个属性值，$b_k$ 表示电影 b 的第 k 个属性值。欧氏距离越小表示两个向量的相似度越高，反之则越低，有时候为了能更直接地体现出相似度，同样可以将上面的公式修改成为：

$$sim(a,b)=\frac{1}{1+d(a,b)} \quad \text{（公式 2.7）}$$

sim(a,b)的区间是(0,1]值越接近于 1，两个电影向量相似度越高，反之则越低。

考虑到电影数量的庞大，逐个计算相似度的复杂度太高，所以在计算相似度之前，也可以使用 Canopy-kmeans 的算法进行聚类，将同类型的电影确定，在此中计算相似度。

当用户觉得最近推荐的电影不太符合胃口，就可以选择“精确推荐”，这时候根据离线计算好的电影相似度，和自己评分最高的 K 个电影进行聚类，排序得到前 M 个，根据用户的电影偏好*电影属性，得到电影的预测评分。如果存在评分相同的电影，就按照热度降序排列。去掉用户已经观看过的电影，进而得到推荐列表。

## 3.4 本章小结

本章在开头介绍了基于用户的协同过滤算法的基本流程，然后将流程和具体应用场景结合，并提出了 3 个问题，分别是新用户进入系统没有历史信息、K-means 聚类效率待提高和单一推荐算法无法把多样性与准确性统一的问题，接下来针对 3 个问题逐个提出了改进思路。

# 第四章 系统需求分析和系统设计

本章将主要对基于 MapReduce 的电影推荐系统做需求分析，搞清楚系统的功能和预期的页面效果。在开发一个系统的时候，首先要对系统进行功能分析，根据功能得到若干个子系统，每个子系统中还会包含若干个模块。系统需求分析是对后续的系统开发的重要保证，按照需求分析实现的系统才是符合要求的。

## 4.1 系统需求概述

目前市面上的电影网站，大多是提供下载和评论以及评分的功能，很少有实时推荐的功能。本系统通过收集用户信息，将推荐算法应用其中，在已有的电影库中为用户寻找到最喜欢的电影。本文将系统分为三个模块，第一个是管理员模块，第二个是普通用户模块，第三个是系统独立功能模块。

因为要个性化推荐，就需要知道个人信息，而且需要分别出用户的身份，所以需要登陆注册的功能。用户登录成功后，如果是初次进入，为了解决冷启动的问题，需要填写一份微型的调查问卷，以获取用户特征属性。用户可以搜索电影，查看电影详细信息并对其进行评分、收藏、评论等操作。最关键的一项，也是系统的核心功能，用户可以点击查看推荐列表。

管理员登录成功后，可以对系统自动录入的电影进行审核，此时的电影只能管理员可见，审核的主要内容是电影的名称、类型是否正确，如果有错误就要进行修正。当管理员审核通过后，就可以将该电影发布到系统中，用户就可以看到新录入的电影。管理员还可以对所有的电影进行基本信息的管理。最后管理员可以对用户的信息进行管理，如果用户存在某些违规行为，就要对其进行一定的惩罚。

除了用户和管理员模块以外，系统还需要一些额外的独立功能。为了获取完整和最新的电影信息，自动录入电影，本系统还需要一个爬取电影网站信息的功能。由于自动录入会存在一定的误差，尤其是对电影类型的录入，涉及到后面推荐算法的数据输入，所以在自动录入后还需要管理员的审核的环节。由于用户-评分矩阵和电影的属性得分是在变化的，所以需要系统定时对用户和电影进行聚类操作，将数据提前计算出来，方便更快地相应用户的页面请求。接下来是对每

个功能的具体描述。

## 4.2 系统需求

### 4.2.1 功能性需求

1. 用户的用例图如图 4-1 所示：

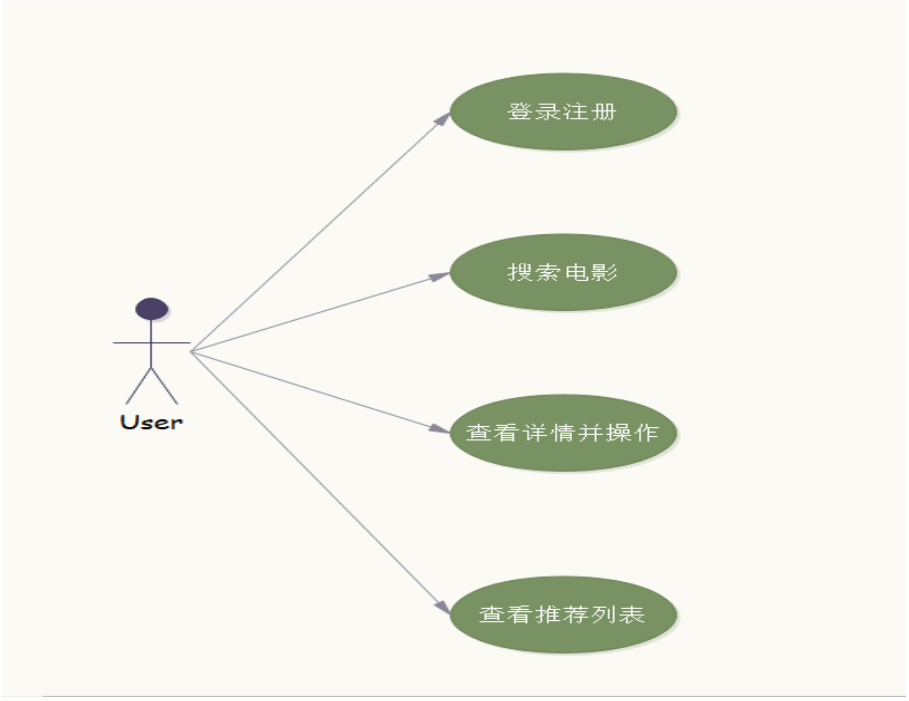


图 4- 1 用户的用例图

1）登陆注册

系统最基本的功能，用户从这里进入系统，为了提供个性化的推荐服务，必须要知道当前的用户，所以这个功能不可或缺。问卷的内容分两部分，一部分是用户特征属性，另一个部分是最近看过的电影和自己喜欢的电影的评分，流程如图 4-2 所示。

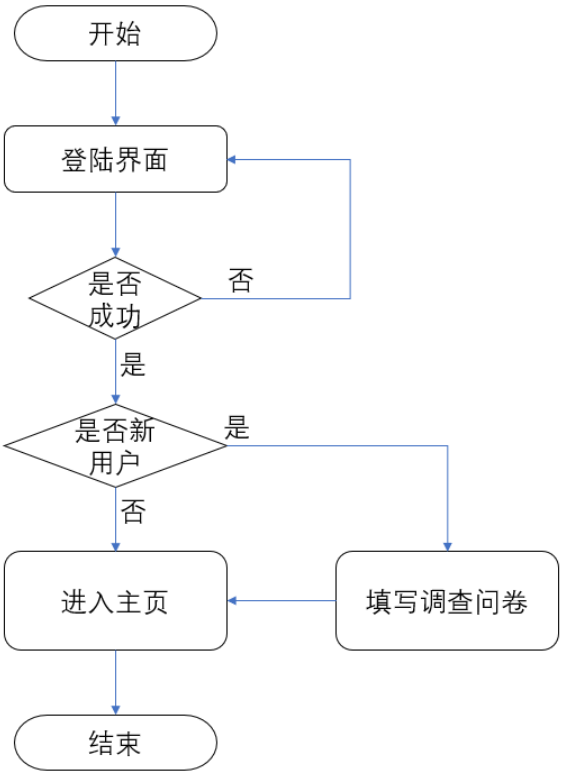


**图 4- 2 登录流程图**

2）搜索电影

用户登陆成功以后，就可以在搜索框搜索自己喜欢的电影，用户的搜索信息也作为日志记录下来，这些信息以后可以用作对用户的更深层次数据挖掘使用。

3）查看电影详情并操作

用户对感兴趣的电影可以进行收藏，评分和评价，这些操作将决定了用户对于电影的偏好，同时对电影的评分会在后续作为协同过滤的输入数据，非常重要。

4）查看推荐列表

用户点击按钮之后，系统通过用户的登录信息，去数据库获取用户的个人其他信息，并开始推荐算法的执行，流程如图 4-3 所示。

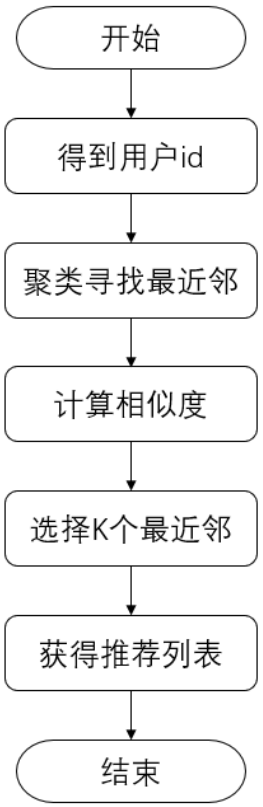


图 4- 3 推荐功能流程图

2. 管理员的用例图如图 4-4 所示：

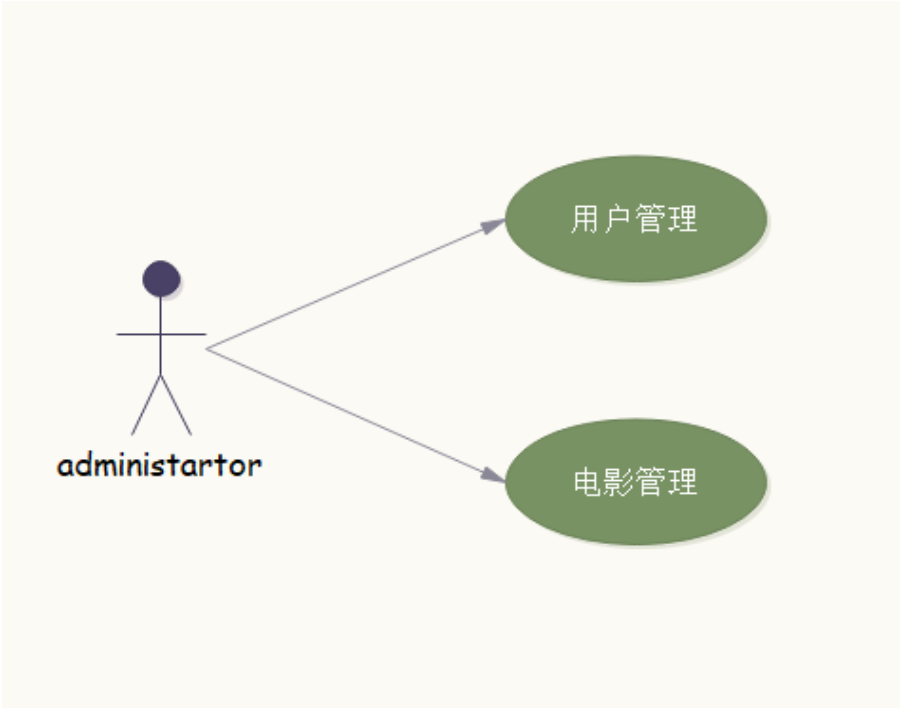


图 4- 4 管理员的用例图

1）电影信息管理

管理员负责对自动录入的电影进行审核，每一部电影都不是只有一个单一类型的，往往是好几个类型，要求管理员仔细审核，因为这也是推荐算法的数据来源，信息错误会直接导致算法的有效性降低。

2）用户信息管理

对用户信息的审核以及操作，如果用户存在违规行为，要对其执行相对应的惩罚。

### 4.2.2 非功能需求

一个良好的系统除了要实现上述的功能性需求之外，还要考虑到系统本身的运行是否稳定，性能是否良好，是否会容易遭到不法分子的恶意攻击，是否需要扩展，进行版本更新等等。

1. 稳定性

系统应该充分地考虑到运行的稳定性，不能经常出现宕机、无法回应的等情况，要对各个功能模块做测试，在通过测试的前提下才能上线给用户使用。

2. 准确性和多样性

由于推荐系统的特殊性，所完成的推荐系统一定要既有推荐的准确性，又有多样性，只有这样用户才能得到自己预期的资源，才能实现资源合理分配。

3. 安全性

应该考虑到网站的安全，尤其是用户的隐私，不能被黑客或者网络爬虫窃取走，要对安全性做一系列的安全测试。

4. 可拓展性

考虑到推荐算法的种类很多样，而且由于人工智能的发展，推荐算法也在发生着很大的变化，不排除以后会有更高效的算法出现，那么就要求系统能无耦合地添加一个其他的推荐模块进来，而不需要使得整个系统大改动，实现一个高内聚、低耦合的系统。

## 4.3 系统设计

### 4.3.1 项目架构设计

整个系统使用 Spring Boot 框架开发，基于 MVC 的思想[41]，共分为三层：表现层（View）、存储层（repository）和逻辑控制层（Controller），总体架构如图 4-5 所示。

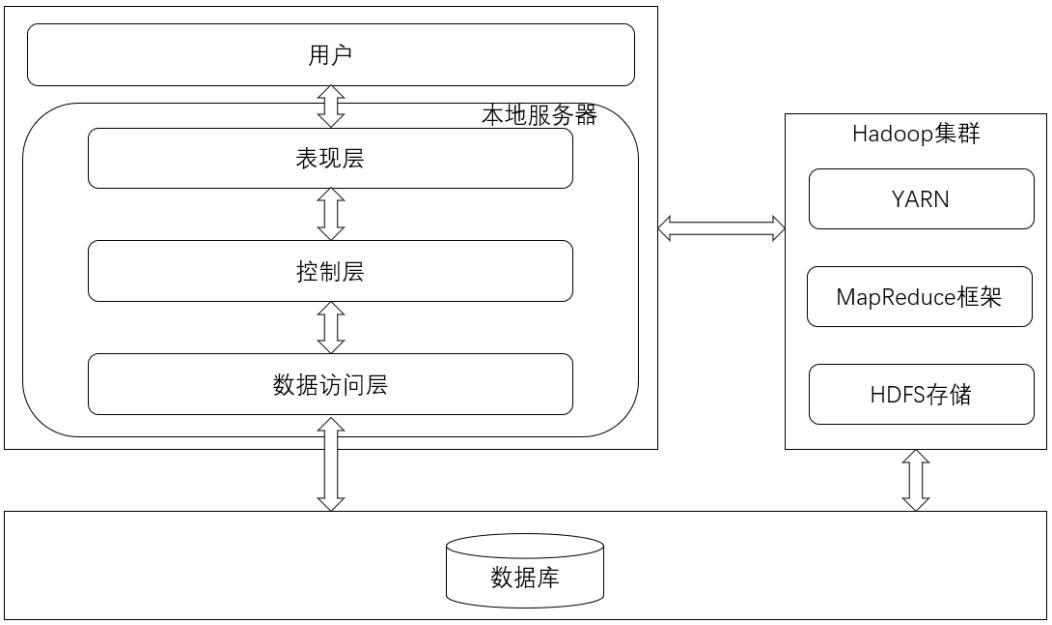


**图 4- 5 系统架构图**

表现层：主要用于展示结果、接收用户输入，比如信息输入或者搜索，提供用户和系统交互的界面。

逻辑控制层：负责请求跳转、业务逻辑处理、调用接口等等。

数据访问层：与数据持久化相关，对数据库的一些基本操作进行封装，对外提供接口，被逻辑层调用[42]。

Hadoop 集群：负责提供分布式计算服务，返回数据。

项目的模块设计如下图 4-6 所示：

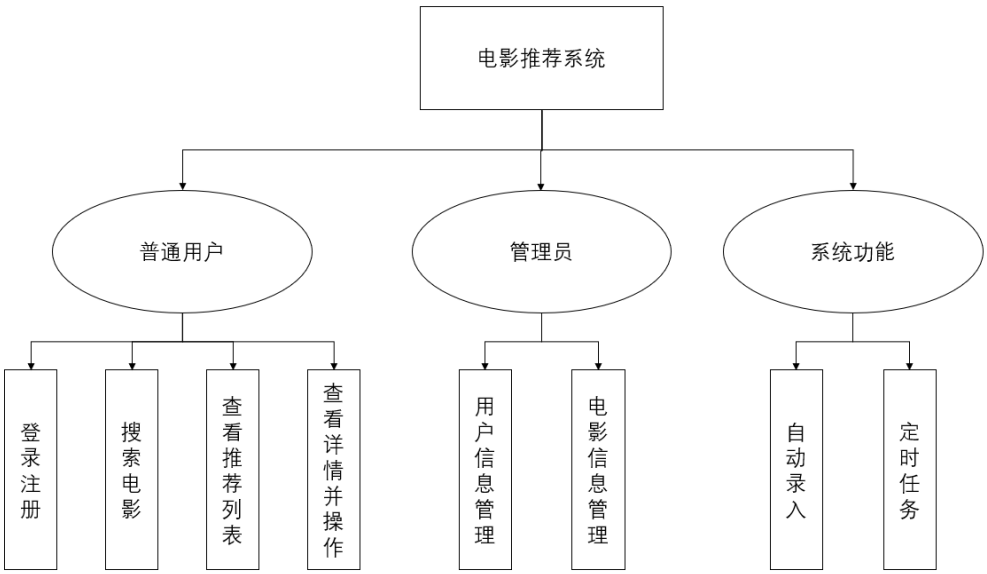


图 4- 6 系统功能模块图

### 4.3.2 数据库设计

本推荐系统所使用到的对象主要包括用户表、管理员表、电影信息表和用户观影表。

用户表是指和用户相关的各类信息，包括唯一 id 标识，登录使用的用户名密码、用户特征、对电影类型的偏好、该用户已存在的推荐列表，推荐列表是为该用户推荐的电影列表，考虑到推荐具有短时间的稳定性，所以存储下来，不用每次都去进行计算，节约了计算资源，每个用户最多保存 3 个推荐列表，并随着时间更新。管理表是指和管理员相关的信息，因为管理和用户的功能、行为差别较大，所以不适合共用一张表，而是单独设计一张表，避免了存储资源的浪费。电影信息表是指电影的唯一 id 标识，电影名、类型。考虑到电影的数量庞大，所以用户的观影表单独存储，避免用户表庞大和大量的空数据。用户观影表存储的是用户观看过的电影一句用户对该电影的评分，目的是和用户表、电影信息表一起做连接查询。

数据库逻辑设计，模型中的实体包含以下 4 个：

用户表：ID，username，password，role，age，character，gender，drama，comedy，love，biography，action，history，science_fiction，suspense，

foreign，HK&Taiwan，china，real_actor，animation，others，recolist

管理员表：ID，admin_name，admin_pwd，role

电影信息表：ID，name,drama，comedy，love，biography，action，history，science_fiction,suspense,foreign,HK&Taiwan,china,real_actor,animation,others

用户观影表：user_ID，movie_ID，rating，comment

下面以用户表 user 为例做具体的结构设计，用户表的设计如表 4-1 所示：

**表 4- 1 用户表**

| 字段 | 类型 | 含义 |
|---|---|---|
| id | long | 用户的唯一身份标识 |
| name | varchar | 姓名 |
| password | varchar | 密码 |
| role | int | 角色，0 是管理员，1 是用户 |
| age | varchar | 年龄 |
| gender | varchar | 性别 |
| character | varchar | 性格 |
| drama | flout | 剧情偏好得分 |
| comedy | flout | 喜剧偏好得分 |
| love | flout | 爱情偏好得分 |
| biography | flout | 传记偏好得分 |
| action | flout | 动作偏好得分 |
| history | flout | 历史偏好得分 |
| science_fiction | flout | 科幻偏好得分 |
| suspense | flout | 悬疑偏好得分 |
| foreign | flout | 国外偏好得分 |
| HK&Taiwan | flout | 港澳台偏好得分 |
| china | flout | 国产偏好得分 |
| real_actor | flout | 真人偏好得分 |
| animation | flout | 动画偏好得分 |

| others | flout | 其他偏好得分 |
|---|---|---|
| recolist | varchar | 已存储的推荐列表 |

### 4.3.3 Hadoop 集群设计与搭建

由于分布式计算的需要，需要搭建 Hadoop 集群，初步把集群数量设定为 5 台，也可以根据后续需求进行添加或删除，集群各个主机节点之间的从属关系如表 4-2 所示：

表 4- 2 集群各个主机节点关系表

| | hadoop001 | hadoop101 | hadoop102 | hadoop103 | hadoop104 |
|---|---|---|---|---|---|
| HDFS | NameNode<br>DataNode | SecondaryNameNode<br>DataNode | DataNode | DataNode | DataNode |
| YARN | NodeManager | NodeManager | NodeManager<br>ResourceManager | NodeManager | NodeManager |

需要注意的是 NameNode 和 SecondaryNameNode 需要配置在不同的主机上，这样才能起到备份的作用。YARN 的 ResourceManager 也会占用很大的资源，也需要单独在一台主机上，这样的设计可以让集群的稳定性、安全性达到最高。根据表中的结构，可得集群的整体架构如图 4-7 所示：

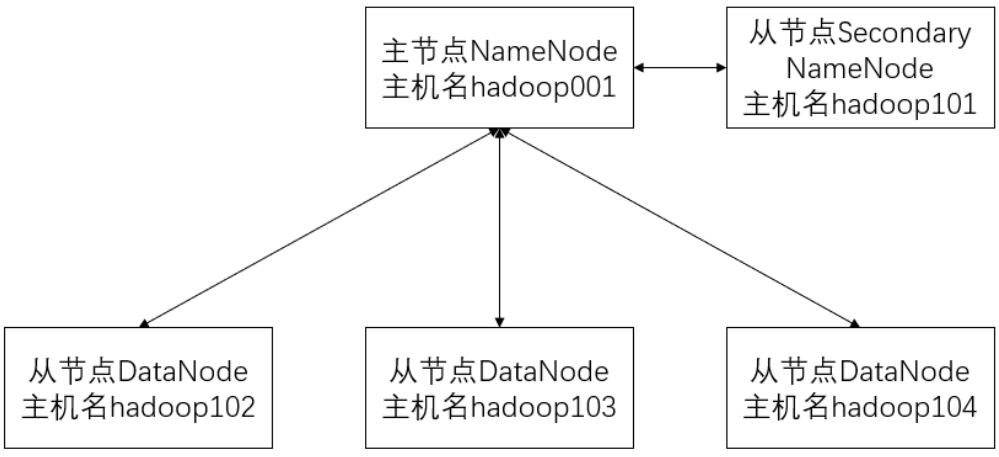


图 4- 7 集群架构示意图

本文在实现 hadoop 集群搭建的时候，选择了 Linux 操作系统，因为在 Linux 系统下更加安全和稳定，选用的版本是 CentOS6，搭建流程如图 4-8 所示：

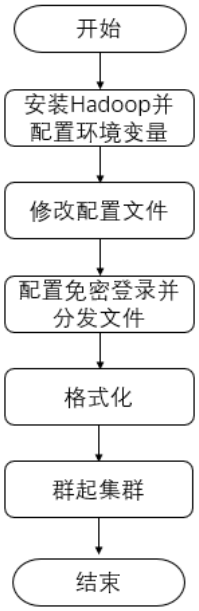


**图 4-8 集群搭建流程图**

（1）文件配置：下载解压和安装的过程不再详细说明，在安装好以后，就需要对 Hadoop 做一些基本的配置，配置文件都在 hadoop-2.7.2/etc/hadoop 文件夹下面，主要的几个配置文件和作用如表 4-3 所示：

**表 4-3 主要配置文件表**

| 文件名 | 作用 |
| --- | --- |
| hadoop-env.sh | 配置 java 环境变量 |
| core-site.xml | 指定 NameNode 的主机节点位置：将 fs.defaultFS 的属性值设置为 hdfs://hadoop001:9000<br>指定 Hadoop 运行过程中产生文件的存储目录，而且 hadoop.tmp.dir 也是 hadoop 文件系统依赖的基础配置，很多路径都依赖它：将 hadoop.tmp.dir 的属性值设置为 /opt/hadoop-2.7.2/data/tmp |
| hdfs-site.xml | 配置文件的副本数量，默认值为 3：将 dfs.replication 属性值设置为 3<br>配置辅助节点的路径，目的是在某一台主机节点宕机后还能从其他节点获取同样的数据，这也是数据可靠性的一个体现：将 dfs.namenode.secondary.http-address 属性值设置为 hadoop101:50090 |
| yarn-site.xml | 配置 Reducer 获取数据的方式：将 yarn.nodemanager.aux-services 属性值设置为 mapreduce_shuffle<br>指定 YARN 的 ResourceManager 的主机节点位置：将 yarn.resourcemanager.hostname 设置为 hadoop102 |

| | |
|---|---|
| mapredsite.xml | 把 mapred-site.xml.template 重命名为 mapred-site.xml，将 mapreduce.framework.name 属性值设置为 yarn，这个属性决定 MapReduce 作业是提交到 YARN 集群执行的，如果不配置则默认使用本地执行器执行 |

以上只是完成了一台主机的配置，由于集群主机数量很大，所以还需要有一个文件分发的功能，这样就可以将相同的文件分发到每一台主机。

（2）免密登录：在文件分发之前，需要解决一个问题，那就是在 Linux 主机上使用一个账号通过 ssh 远程连接另一台主机时需要输入密码，为了使得操作方便，需要配置免密登录的功能。免密登录的原理如图 4-9 所示：

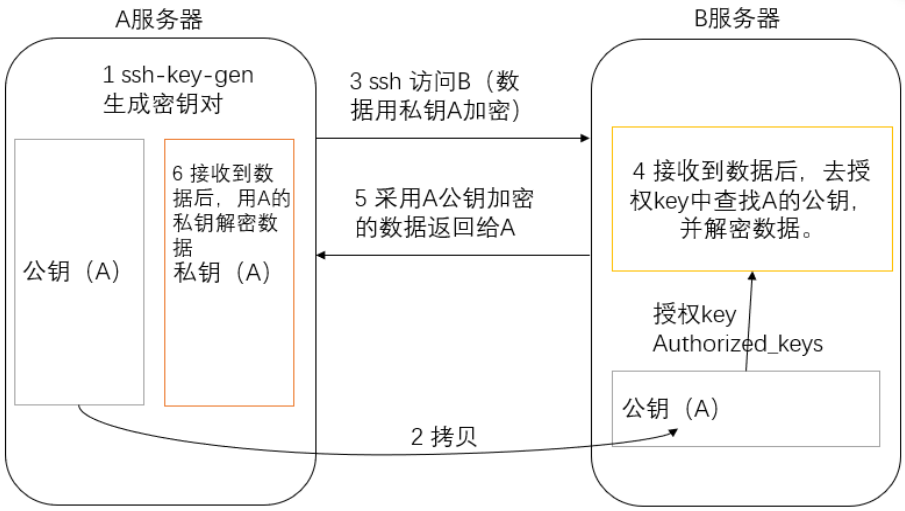


**图 4-9 免密登录示意图**

具体做法是，在 A 主机中使用 ssh-keygen -t rsa 命令获得私钥，然后再使用 ssh-copy-id 命令将 A 生成的公钥复制到 B 主机上，这样 A 就可以免密登录到 B。考虑到集群的主机数量很大，所以本文采用的方式是，让集群中所有的主机都将公钥复制到 hadoop001，然后在 hadoop001 中将 authorized_keys（存放的是授权过得无密登录服务器公钥）分发到所有主机，这样每两台主机都可以实现免密登录了。

（3）文件分发：Linux 中文件备份和镜像的命令常用 rsync，它的优点是速度快、不会复制相同内容，而且支持符号链接的。Scp 命令同样也是用于复制文件，但是 rsync 的速度更快，因为其只对差异文件做更新，而 scp 是把所有文件都复制

过去。本文选用的是 rsync 命令，并实现了一个分发文件 shell 脚本，每次按照命令格式调用脚本，就能实现文件的统一分发，十分方便。

（4）格式化和群起集群：在 hadoop/etc 的 slaves 文件中把集群所有的主机名逐行写入，并且不能有空格。如果是第一次启动，就需要先格式化 NameNode 和 DataNode。格式化完成后就可以在 NameNode 的节点启动 hdfs，命令是 sbin/start-dfs.sh。在 ResouceManager 的节点启动 YARN，命令是 sbin/start-yarn.sh。至此，分布式环境搭建完成。

## 4.4 本章小结

本章对系统的功能进行了分析，搞清楚系统的目标和主要功能，划分好了功能模块，并且对系统进行了设计，主要是系统架构设计和数据库设计。由于需要使用分布式集群，本章还重点对 Hadoop 集群进行了架构设计和搭建。在下一章中，将依据本章的设计进行具体的实现和测试。

# 第五章 系统实现与测试

本章根据第三章提出的改进算法和第四章的系统分析进行系统的开发实现。

## 5.1 系统开发环境

开发环境：Idea2018.3.5，jdk-12.0.2，hadoop2.7，springboot2.1.7，mysql5.7。

操作系统：Windows10，CPU 为 intel Core i5，主频 1.8GHz，内存 8G，硬盘 512G，CentO S6。

开发语言：Java，SQL，JavaScript，Shell 脚本

## 5.2 系统功能实现

根据系统设计的架构，首先进行数据层的实现。MySQL 在 Web 应用开发方面非常流行，也是最好的关系型数据库管理系统应用软件之一[43]。由数据库设计可得，数据实体类共有 4 个，分别是用户类，管理员类，电影类和观看列表类。在 Mysql 中建立相对应的表，这里以用户类为例，其他实体类的表类似，如图 5-1 所示：

```
CREATE TABLE IF NOT EXISTS User(
  user_id INT(11) NOT NULL AUTO_INCREMENT,
  name VARCHAR(45) DEFAULT NULL,
  password VARCHAR(45) DEFAULT NULL,
  role INT DEFAULT NULL,
  age VARCHAR(45) DEFAULT NULL,
  gender VARCHAR(45) DEFAULT NULL,
  character VARCHAR(45) DEFAULT NULL,
  drama FLOUT DEFAULT NULL,
  comedy FLOUT DEFAULT NULL,
  love FLOUT DEFAULT NULL,
  biography FLOUT DEFAULT NULL,
  action FLOUT DEFAULT NULL,
  history FLOUT DEFAULT NULL,
  science_fiction FLOUT DEFAULT NULL,
  syspense FLOUT DEFAULT NULL,
  foreign FLOUT DEFAULT NULL,
  HK&Taiwan FLOUT DEFAULT NULL,
  china FLOUT DEFAULT NULL,
  reai_actor FLOUT DEFAULT NULL,
  animation FLOUT DEFAULT NULL,
  others FLOUT DEFAULT NULL,
  recolist VARCHAR(100)DEFAULT NULL,
  PRIMARY KEY (task_id)
) ENGINE=InnoDB;
```

图 5- 1 用户表实现图

数据库表建立完成后，接下里是软件系统的开发。首先在 IDEA 中建立一个 Spring Boot 项目，并引入相对于的 starter，如图 5-2 所示：

```
<groupId>com.xu</groupId>
<artifactId>movie_recommedation</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>movie_recommedation</name>
<description>Demo project for Spring Boot</description>

<properties>
    <java.version>1.8</java.version>
</properties>

<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jdbc</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
```

图 5- 2 starter 引入图

数据层包含两部分，DAO 和 Service 的实现。DAO 的作用是直接从数据库中读取表的内容然后返回，提供一些基本的方法，在 Spring Boot 中，每个实体对应的 Mapper 就相当于是 DAO 层，系统的各个 DAO 及功能如图 5-3 所示：

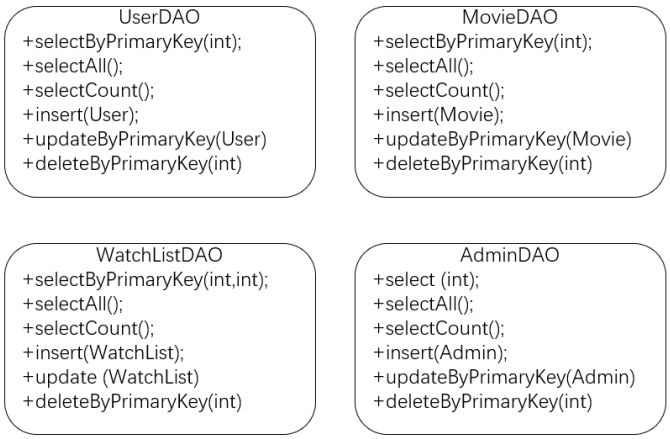


图 5- 3 DAO 及其方法图

DAO 完成后，需要对 DAO 里面的方法进行 service 封装，并且有些功能会涉及到不止一张表，所以一个 service 可能会对应不止一个 DAO，系统的主要 service 类图如图 5-4 所示：

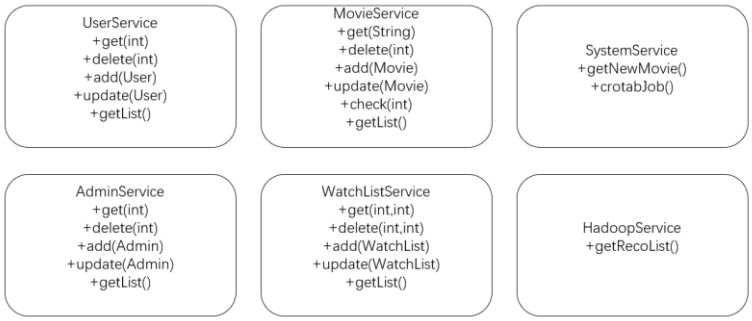


图 5- 4 service 类图

完成了数据层的实现，第二层是 Controller 层，负责相应用户的请求和返回数据，并选择恰当的视图以用于显示，系统主要 Controller 的类图如 5-5 所示：

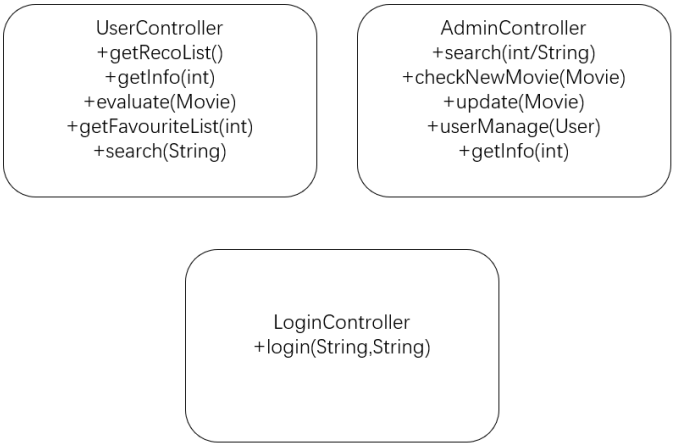


图 5- 5 Controllor 类图

表现层使用 html 和 themyleaf 视图解析器。整体的架构就完成了，由于有些功能的实现过程很相似，接下来以“查看电影并操作”以及“查看推荐列表”两个功能加以描述。

### 5.2.1 查看电影详情并操作

用户可以对电影进行搜索，查看和评价，管理员可以对电影进行审核，修改和删除。根据以上的功能需求可以得到该功能实现类之间的依赖如图 5-6 所示：

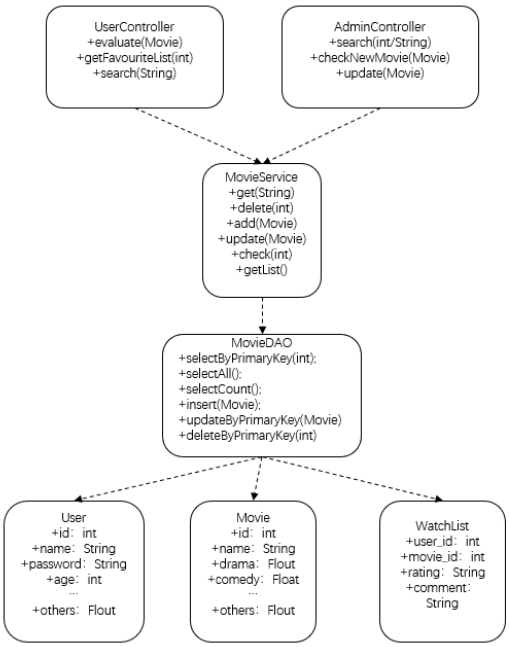


**图 5- 6 电影操作功能实现类依赖图**

用户或者管理员的操作请求会进入对应的 Controller 的对应方法，有关于电影的操作会使用 MovieService 类完成具体功能，MovieService 依赖于 Movie DAO，后者负责对数据库进行交互和数据持久化。由于用户的操作会使得电影偏好属性发生改变，而且会留下观看记录，所以涉及到的 Entity 有三个 User、Movie 和 WatchList。

### 5.2.2 查看推荐列表

此功能是本系统的重点功能，突出的特点就是在 Spring Boot 项目中使用分布式计算服务。当用户点击页面按钮时，请求转到 UserController，其中 getRecoList()方法设置@RequestMapping("/hadoop/getRecoList")，接下来就会去从数据库中读取信息，上传到 HDFS 文件系统，然后进行 MapReduce 的聚类操作，该功能的类图如 5-7 所示：

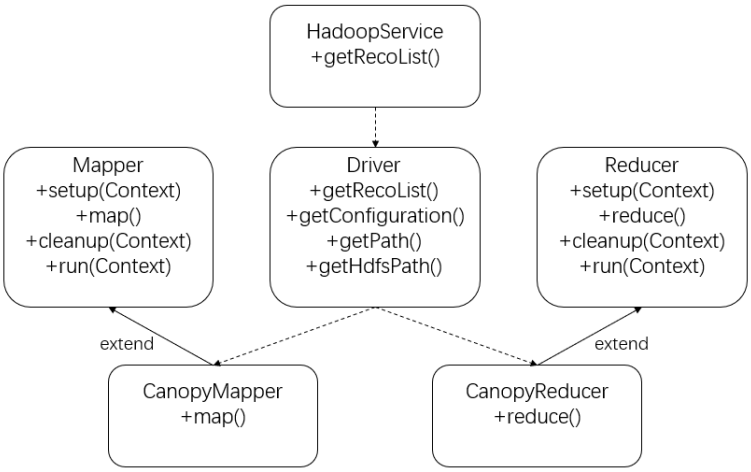


**图 5- 7 查看推荐列表功能类图**

## 5.3 系统测试

系统开发完成之后，就要对系统进行测试。系统测试的定义是：在规定的条件下对程序进行操作，以发现程序错误，衡量软件质量，并对其是否能满足设计进行评估的过程[44]。常用的测试策略有单元测试和集成测试。本节使用单元测试的策略对系统进行测试，模拟使用整个系统，测试系统是否达到预期的要求，包括功能和性能两方面。功能体现在每个模块是否正确实现，性能主要体现系统的稳定性和安全性。

### 5.3.1 功能测试

这里以对电影管理为例，介绍在开发过程中的测试，测试用例如表 5-1 所示：

**表 5- 1 测试用例表**

| 编号 | 功能点 | 测试用例 |
| --- | --- | --- |
| 1 | 电影信息审核 | 1. 点击审核通过按钮，期望返回审核通过并且已将电影加入库中<br>2. 点击修改，期望跳转到电影修改页面并且把现有的电影信息一起传过去。 |
| 2 | 电影信息修改 | 1. 传入正确电影 id，其他名称字段为空，期望返回成功，并且此时该电影的信息无变化 |

| | | |
|---|---|---|
| | | 2. 传入正确的电影 id 以及修改后的字段值，期望返回成功，并且此时该电影的信息变为修改后的新值<br>3. 传入错误的电影 id，期望返回修改失败，找不到该电影的信息 |
| 3 | 电影信息删除 | 1. 输入已存在的电影 id，期望返回删除成功<br>2. 输入不存在的电影 id,期望返回找不到该电影<br>3. 输入已删除的电影 id,期望返回该电影已删除不能重复操作 |
| 4 | 获取电影信息 | 1. 输入正确的电影 id，期望返回该 id 的电影的信息<br>2. 输入错误的电影 id,期望返回找不到该电影信息 |
| 5 | 获取电影列表 | 1. 输入正确的电影 id 若干个，期望返回对应个数的电影信息列表<br>2. 输入 n 个正确电影 id 和 m 个错误电影 id，期望返回 n 个电影信息和找不到以下 m 个电影 |

### 5.3.2 用户界面测试

用户进入网站后即跳转到登录界面，如图 5-8 所示：

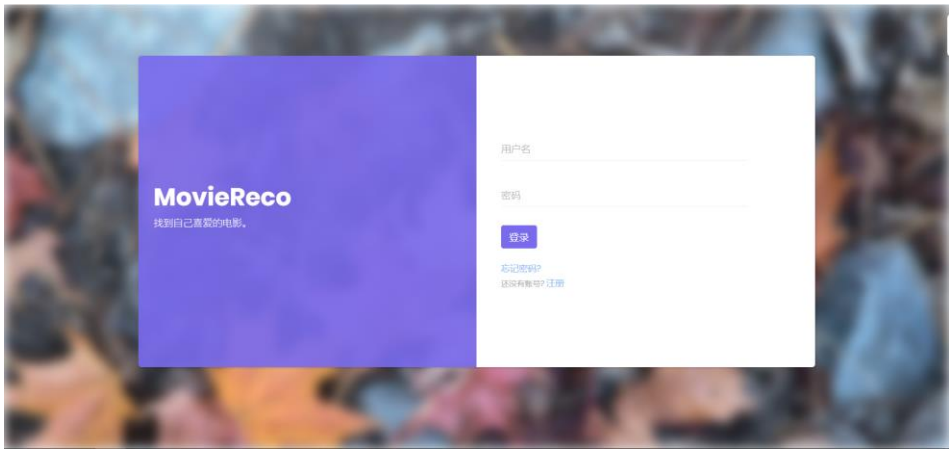


图 5- 8 登陆界面图

登录成功以后，是主界面，如图 5-9 所示，默认展示的是近期热门的电影，最上面是搜索框，用户可以搜索自己喜欢的电影。

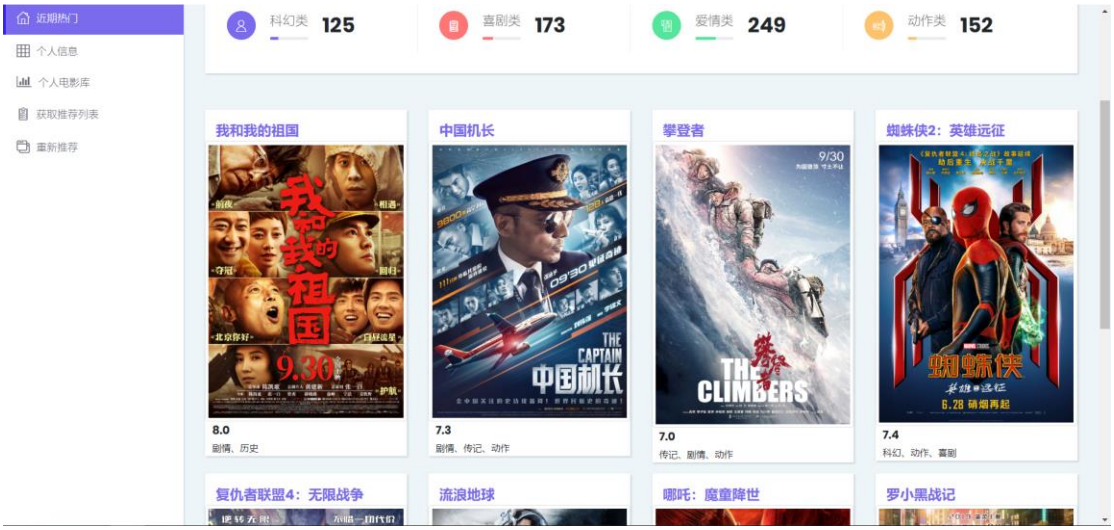


图 5- 9 主界面图

单击某部电影图片，即可查看电影详情页面，如图 5-10 所示：

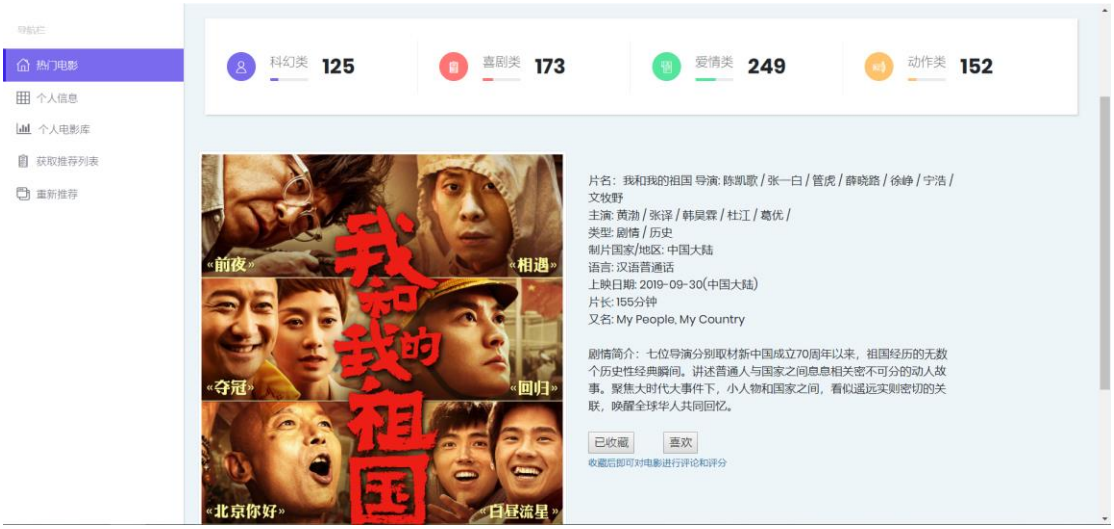


图 5- 10 电影详情界面图

左边是用户可以选择的操作，分别是“近期热门”、“个人信息”、“个人电影库”、“获取推荐列表”和“重新推荐”。当用户点击个人电影库的时候，就可以看到自己观看过的电影，并可以挑选其中的某一部电影进行评分和评论等操作，如图 5-11 所示：


图 5- 11 电影评论、评分界面图

接下来是对推荐功能的测试，用户再个人信息里可以看到自己对于电影偏好的饼图，如图 5-12 所示，系统会将个人偏好最高的 4 个类型展示出来：


图 5- 12 电影类型偏好饼图

当他在查看推荐列表时，就会得到以下的结果，如图 5-13 所示：

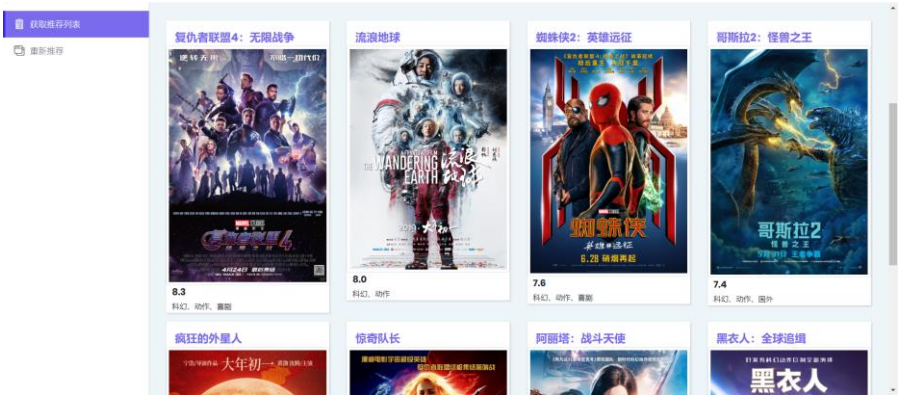


图 5- 13 推荐列表图

### 5.3.3 安全性测试

作为一个公开可访问的电影推荐系统，安全性的测试非常重要[45]，这决定了该系统是否可以正常投入到生产环境中去，如果系统安全性很低，那么很可能会被非法用户入侵，盗取到用户的信息，造成巨大损失，所以对于安全性的测试必须给予极大的重视[46]。本节将会就系统在收到安全攻击时的抗风险能力进行测试，详细内容如表 5-2 所示：

**表 5- 2 安全性测试表**

| 安全性测试分类 | 检查内容 |
| --- | --- |
| 登录注册<br>安全性测试 | 1. 用户名、密码校验是否正确<br>2. 用户角色区分是否正确<br>3. 是否存在 SQL 注入的可能性<br>4. 是否可以通过其他方法登录系统<br>5. 没有登录的前提下是否可以访问主页面 |
| 数据库<br>安全性测试 | 1. 数据库身份验证<br>2. 系统数据是否加密处理<br>3. 系统数据是否备份<br>4. 数据库的日志记录是否正常 |

## 5.4 本章小结

本章根据第四章设计的系统架构，选取了合适的开发环境，然后逐一实现了数据层、控制层和展示层，通过类图来说明各个 java 类之间的关系，并针对查看电影并操作和查看推荐列表两个功能进行具体的说明。最后一节对系统的功能和性能进行了全面的测试，功能测试证明系统按照设计实现了各个模块的功能，性能上表现良好，用户界面测试证明了系统的各个界面能够很友好地显示数据，安全性测试证明了系统的安全性达到了一定的要求[47]。

# 第六章 总结与展望

## 6.1 总结

本文在阅读了大量文献资料后，完成了一个简单的电影推荐系统，主要的工作如下：

1.本文引入了用户特征属性来应对冷启动的问题，重新定义了用户-评分矩阵，将用户电影偏好作为用户的属性，并且随着时间的推移，使用时间因子来改变用户属性和用户电影偏好的比例。但是存在的问题是，用户特征属性不能太多，一是用户在输入的时候影响用户体验，二是用户特征属性只能粗略地估计出用户的喜好，所以存在一定的误差。

2.在使用聚类算法减少近邻选择的过程中，本文对 K-means 算法进行了改进，引入了 Canopy 进行预处理，主要的目的是自动获得优质的 K 值，以此来减少后续 K-means 聚类的计算量和迭代次数，并且通过使用分布式计算框架，将串行算法并行化实现。经过实验验证，该方法有效地提高了 K-means 的聚类效率。

3.基于混合推荐算法的思想，在系统中实现了基于用户协同加基于内容协同的混合推荐算法，既保证了多样性，又保证了准确性。

## 6.2 展望

本文实现的原型系统的数据规模依旧较少，只能简单是展示出所研究和改进的算法在推荐系统中的应用，无法明确地验证该算法在大数据下的效果。后续还需要在大规模的数据下进行深层次的验证。

近年来以深度学习为指导思想的推荐算法开始出现，本文并没有对这方面的算法进行深入探究。应该结合时代的发展，将最新的技术应用于推荐系统，这也是本文以后应该要探索的方向。

# 参考文献


[1] 韩家慧.加速推动媒体融合发展 构建全媒体传播格局.新华社，2019-03-15，www.xinhuanet.com/politics/leaders/2019-03/15/c_1124240350.htm

[2] 朱湘. 面向社交网络的信息传播关键技术研究[D]. 国防科学技术大学, 2017.

[3] Najafabadi M K, Mahrin M N, Chuprat S, et al. Improving the accuracy of collaborative filtering recommendations using clustering and association rules mining on implicit data[J]. Computers in Human Behavior, 2017, 67: 113-128.

[4] 李峰. 移动社交网络中推荐系统安全检测技术的研究和实现[D]. 南京邮电大学, 2017.

[5] Bendechache M, Tari A K, Kechadi M T. Parallel and distributed clustering framework for big spatial data mining[J]. International Journal of Parallel, Emergent and Distributed Systems, 2019, 34(6): 671-689.

[6] 纪科. 融合上下文信息的混合协同过滤推荐算法研究[D]. 北京交通大学, 2016.

[7] Wang Z, Yu X, Feng N, et al. An improved collaborative movie recommendation system using computational intelligence[J]. Journal of Visual Languages & Computing, 2014, 25(6): 667-675.

[8] Resnick P, Iacovou N, Suchak M, et al. GroupLens: an open architecture for collaborative filtering of netnews[C]//Proceedings of the 1994 ACM conference on Computer supported cooperative work. ACM, 1994: 175-186.

[9] 刘辉，郭梦梦，潘伟强. 个性化推荐系统综述[J]. 常州大学学报：自然科学版, 2017, 29(3): 51-59.

[10] 张远荪. 基于信任关系的协同过滤推荐算法研究[D].湖南大学,2018.

[11] Said A, Bellogín A. Replicable evaluation of recommender systems[C]//Proceedings of the 9th ACM Conference on Recommender Systems. ACM, 2015: 363-364.

[12] Yin R, Li K, Zhang G, et al. A deeper graph neural network for recommender systems[J]. Knowledge-Based Systems, 2019, 185: 105020.

[13] Karatzoglou A, Hidasi B. Deep learning for recommender systems[C]//Proceedings of the eleventh ACM conference on recommender systems. ACM, 2017: 396-397.

[14] Tianxing M, Baimuratov I R, Zhukova N A. A Knowledge-Oriented Recommendation

System for Machine Learning Algorithm Finding and Data Processing[J]. International Journal of Embedded and Real-Time Communication Systems (IJERTCS), 2019, 10(4): 20-38.

[15] 陆嘉恒. Hadoop 实战 2.0，机械工业出版社，2017

[16] 翟周伟. Hadoop 核心技术，机械工业出版社. 2015

[17] 王晓林. 基于内容的视频信息检索技术研究与实现[D]. 西安电子科技大学, 2013.

[18] Koohi H, Kiani K. User based collaborative filtering using fuzzy C-means[J]. Measurement, 2016, 91: 134-139.

[19] Jorro-Aragoneses J L, Recio-García J A, Díaz-Agudo B, et al. RecoLibry-core: A component-based framework for building recommender systems[J]. Knowledge-Based Systems, 2019, 182: 104854.

[20] 孙冬婷，何涛，张福海. 推荐系统中的冷启动问题研究综述[J]. 计算机与现代化, 2012 (5): 59-63.

[21] 林婉莹. 图书推荐系统中提升 Top-N 列表多样性算法研究[D]. 北京邮电大学, 2019.

[22] 谢振东，张绪升，苏浩伟，等. 基于大数据的广州一站式出行服务平台构建研究[J]. 现代信息科技, 2019 (7): 60.

[23] 朱永强，周珂，李丹，等. HDFS 小文件读写优化策略[J]. 计算机时代, 2016 (9): 9-12.

[24] 饶磊，杨凡德，刘东. 基于分布式 NameNode 节点的 HDFS 优化研究[C]//第八届全国信号和智能信息处理与应用学术会议会刊. 2014.

[25] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters[J]. Communications of the ACM, 2008, 51(1): 107-113.

[26] 黄开科. Hadoop Map Reduce Shuffle 过程优化方案研究[J]. 华中科技大学, 2016: 1-z.

[27] 司雅楠.Hadoop2.0 平台概述[J].科技与创新,2019(05):65-66.

[28] Johnson R, Hoeller J. Expert One-on-one J2EE Development without EJB[J]. 2003.

[29] Walls C. Spring Boot in action[M]. Manning Publications, 2016.

[30] 张峰. 应用 SpringBoot 改变 web 应用开发模式[J]. 科技创新与应用, 2017 (23): 193-194.

[31] 张忠林，曹志宇，李元韬. 基于加权欧式距离的 k_means 算法研究[J]. 郑州大学学报：工学版, 2010 (1): 89-92.

[32] 彭凯，汪伟，杨煜普. 基于余弦距离度量学习的伪 K 近邻文本分类算法[J]. 计算机工程与

设计, 2013, 34(6): 2200-2203.

[33] 郝德华, 关维国, 邹林杰, 等. 基于 Pearson 相关系数的快速虚拟网格匹配定位算法[J]. 计算机应用, 2018, 38(3): 763-768.

[34] 柏宇轩. Kmeans 应用与特征选择[J]. 电子技术与软件工程, 2018 (1): 186-187.

[35] 周志华. 机器学习, 清华大学出版社, 2017.

[36] 钟熙, 孙祥娥. 基于 Kmeans++ 聚类的朴素贝叶斯集成方法研究[J]. 计算机科学, 2019 (2019 年 z1): 439-441,451.

[37] Tong J F. User clustering based on Canopy+ K-means algorithm in cloud computing[J]. Journal of Interdisciplinary Mathematics, 2017, 20(6-7): 1489-1492.

[38] 余长俊, 张燃. 云环境下基于 Canopy 聚类的 FCM 算法研究[J]. 计算机科学, 2014 (S2): 316-319.

[39] 牛怡晗, 海沫. Hadoop 平台下 Mahout 聚类算法的比较研究[J]. 计算机科学, 2015 (S1): 465-469.

[40] 邹倩颖. 基于 Hadoop 平台的 Canopy 算法研究及应用[J]. 福建电脑, 2016, 32(1): 116-117.

[41] 吴波. 基于 MVC 架构的十二年一贯制学生信息管理系统的设计与实现[D]. 东南大学, 2016.

[42] 刘旺. MVC 架构下的教培信息发布系统的设计与实现[J]. 电脑知识与技术, 2019 (23): 49.

[43] Bucea-Manea-Tonis R, Bucea-Manea-Tonis R. How to Design a Web Survey Using Spring Boot with Mysql: A Romanian Network Case Study[J]. Annals of Spiru Haret University, 2017 (2).

[44] 梅磊, 徐伟. 基于程序结构的软件测试数据自动生成系统[J]. 中国科技信息, 2015 (5): 101-103.

[45] 甄海涛, 王金玉, 杨卓林. 基于 hadoop 大数据平台的数据安全研究[J]. 自动化技术与应用, 2019 (8): 38.

[46] 张振华. 大数据背景下软件测试的挑战及其展望探析[J]. 电子技术与软件工程, 2016 (6): 61-61.

[47] 刘珊, 王伟. 大数据时代下计算机网络信息安全[J]. 电子技术与软件工程, 2019 (6): 150.

# 致谢

时光飞逝，转眼间我已经在华东师范大学度过了两年半的时光，在这两年多的学习生活中，我遇到了很多人，包括老师、同学以及室友，在他们的帮助下，我在学习、工作和生活各个方面都取得了很大的进步。在即将毕业之际，我要在这里向所有帮助过我的人表达最衷心的感谢。

首先我要感谢我的导师——王江涛老师，感谢他在我撰写论文的过程中对我的帮助和指导。在我本科毕业，刚进入实验室后，王老师渊博的专业知识，严谨的治学态度，精益求精的工作作风，诲人不倦的高尚师德，朴实无华、平易近人的人格魅力对我影响深远。导师不仅授我以文，而且教我做人，虽然只有短短的两年多，却让我终生受益无穷。本论文从选题到完成，修改了许多次，每一次都是在王老师的指导下完成的，倾注了导师大量的心血，在此我要向王老师表示深切的谢意与祝福！

接下来我要感谢实验的各位师兄师姐、同级伙伴和学弟学妹们，和你们在一起学习的时光总是充满了欢声笑语，丝毫不会感觉到工作的繁重和无聊。正是由于你们，我才会觉得科研之旅非常得开心。

然后我要感谢我的父母，在我读研期间，他们全力支持我，让我可以全身心地投入到学习中去。每当遇到难过的事情或者困惑的时候，我都会和他们倾诉，父母也会竭尽全力开导我，这时我就会感到只要有家庭的支持，任何困难都没什么大不了的，感谢父母的无私奉献！

我还要感谢我亲爱的室友们，徐文起、徐向阳、谢永康和余阳，从实验室回到寝室后，我们总是会聊聊今天的收获，还有所见所闻，一天的疲劳都烟消云散了。在两年内，我们互相切磋技术、共同进步，这些都是我非常珍贵的回忆，愿我们每一个人都前程似锦。

最后我要感谢参与本论文评审的各位专家老师，很荣幸能够得到你们的指导，正是因为这样，我才能更好地完善我的论文，再次表示感谢。

# 攻读硕士学位期间发表论文和科研情况

- **已发表软件著作权**

[1] 在线脑神经元信号实时显示软件　　授权号：2019SR0614979

- **参与的科研课题**

[1] 上海市科委重大专项课题:《基于微芯片技术的脑活动多道记录系统》