摘　要

协同过滤在推荐系统中的应用非常广泛，已经成为推荐系统中的主流方法。通过应用协同过滤推荐算法，能够利用用户的历史数据来洞察他们的兴趣和偏好，从而提供个性化的推荐服务。这种基于协同过滤的电影推荐系统的设计，对于提高电影推荐的准确性、改善用户体验、为电影行业带来经济效益以及增强电影推荐系统的可扩展性具有重要的意义。

本文论述了基于协同过滤的电影推荐系统的设计与实现。首先，介绍了电影推荐系统的研究背景和目的，并简要回顾了电影推荐系统的研究历史。接着，文章详细阐述了推荐系统的设计过程，包括收集训练数据、选择推荐算法、模型训练和部署等。针对冷启动问题、推荐准确性差和实时性差等问题针对性地提出了加入用户自定义初始标签、在推荐优先级中加入偏移项等方法来解决相应问题。对算法性能进行了分析并比较了不同算法的时间开销。最后，文章总结了电影推荐系统的意义，并列出了本项目的设计内容和设计指标。

总体来说，基于协同过滤的电影推荐系统是一个非常热门的研究课题，并且有很多不同的研究方向和应用领域。通过基于协同过滤的电影推荐系统，可以提高电影推荐的准确性、提升用户体验、为电影行业带来经济效益以及提高电影推荐系统的可扩展性。

关键词：推荐算法；数据收集；模型训练；电影推荐

Abstract

Machine learning is widely used in recommendation systems and has become the mainstream method in recommendation systems. Machine learning recommendation algorithm can learn users' interests and preferences based on users' historical data, and then provide personalized recommendations for users based on these information. Therefore, designing a movie recommendation algorithm based on machine learning is of great significance for improving the accuracy of movie recommendation, improving user experience, bringing economic benefits to the film industry and improving the scalability of the movie recommendation system.

This paper discusses the design and implementation of movie recommendation algorithm based on machine learning. First of all, the article introduces the research background and purpose of movie recommendation system, and briefly reviews the research history of movie recommendation system. Then, the paper elaborates the design process of the movie recommendation algorithm based on machine learning, including selecting machine learning algorithm, collecting and preparing data, feature engineering, model training and deployment. Finally, the paper summarizes the significance of the movie recommendation algorithm based on machine learning, and lists the design content and design indicators of this project.

In general, the movie recommendation algorithm based on machine learning is a very hot research topic, and there are many different research directions and application fields. The film recommendation algorithm based on machine learning can improve the accuracy of film recommendation, improve user experience, bring economic benefits to the film industry and improve the scalability of the film recommendation system.

Key words: Recommended algorithm; Data collection; Model training; Movie Recommendation

目　录

第1章 绪 论

1.1 课题背景

随着互联网技术的发展，电影推荐系统已经成为了电影行业的重要组成部分。它可以帮助用户找到自己感兴趣的电影，并为电影行业带来巨大经济效益。然而，现有的电影推荐系统存在着一些问题，如准确性低、推荐结果单一等[1]。因此，设计一种基于协同过滤的电影推荐系统，可以提高推荐准确性、提升用户体验、提高可扩展性以及为电影行业带来经济效益。本研究的目的就是设计并实现一种基于协同过滤的电影推荐系统来解决这些问题。在本研究中，将采用协同过滤技术，对用户的历史浏览记录、喜好和评分进行分析，并使用这些信息来向用户推荐相似用户喜欢的电影[2]。

协同过滤在推荐系统中的应用非常广泛[3]，已经成为推荐算法中的主流方法。协同过滤推荐系统能够根据用户的历史数据来学习用户的兴趣和偏好，然后根据这些信息来为用户提供个性化的推荐。涉及到的方法包括协同过滤、内容基于推荐、基于图的推荐、深度学习等[6]。在协同过滤方面，有很多研究是关于如何更好地利用用户之间的相似度来进行推荐的[7]。在内容基于推荐方面，有很多研究是关于如何利用物品的内容信息来进行推荐的。基于图的推荐方法则是利用图的结构来进行推荐[8]。深度学习在推荐系统中的应用也越来越多，可以通过学习用户的多维行为序列来进行推荐[9]。

推荐算法在电影推荐方面运用的却不是很广泛。目前，主要流行的是基于热度的推荐和基于内容的推荐[10]。类似于协同过滤和实时推荐等算法运用得并不多。为了提高电影推荐的准确性、提升用户体验、为电影行业带来经济效益以及提高电影推荐系统的可扩展性，实现基于协同过滤的推荐系统具有相当的现实意义。

另外，我们还将在本研究中使用特征工程技术来提高模型的准确性，并使用大量的数据来训练和评估模型。最后，我们将在本研究中提供详细的实验结果，并对本研究的结果进行分析。

总之，本研究的课题背景是基于协同过滤技术设计和实现一种电影推荐系统来解决现有系统存在的问题，并为电影行业带来经济效益。

1.2 课题意义

协同过滤的电影推荐系统的目的是使用数据和统计模型等方式来为用户提供个性化的电影推荐。设计基于协同过滤的电影推荐系统有以下几点意义：

(1) 提高电影推荐的准确性。协同过滤算法可以从大量的数据中学习用户的喜好，从而提高电影推荐的准确性。

(2) 提升用户体验。通过向用户推荐他们可能喜欢的电影，可以提升用户的观看体验。

(3) 为电影行业带来经济效益。通过向用户推荐更多的电影，可以帮助电影行业获得更多的收益。

(4) 提高电影推荐系统的可扩展性。协同过滤算法的优势在于它们可以自动学习，因此即使用户数量增加，也可以保证电影推荐系统的可扩展性。

因此，设计基于协同过滤的电影推荐系统对于提高电影推荐的准确性、提升用户体验、为电影行业带来经济效益以及提高电影推荐系统的可扩展性具有重要意义。

1.3 国内外的发展现状

协同过滤在推荐系统中的应用非常广泛，已经成为推荐系统中的主流方法。协同过滤推荐系统能够根据用户的历史数据来学习用户的兴趣和偏好，然后根据这些信息来为用户提供个性化的推荐[11]。

国内外研究都有很多关于推荐算法的研究，涉及到的方法包括协同过滤、内容基于推荐、基于图的推荐、深度学习等。在协同过滤方面，有很多研究是关于如何更好地利用用户之间的相似度来进行推荐的[12]。在基于内容的推荐方面，有很多研究是关于如何利用物品的内容信息来进行推荐的。基于图的推荐方法则是利用图的结构来进行推荐。深度学习在推荐系统中的应用也越来越多，可以通过学习用户的多维行为序列来进行推荐[13]。

研究现状来看，协同过滤在推荐系统中的应用还有很大的潜力，各种方法也在不断发展和改进。未来，期望能够看到更多创新的协同过滤推荐算法的出现，并且能见到它们在实际应用中取得更好的效果。

在国内外，也有很多研究者在研究如何提升协同过滤推荐算法的效率和准确性。例如，一些研究者致力于开发新的协同过滤算法[14-17]，这些算法能够在保证推荐质量的同时降低计算复杂度。另一方面，一些研究者正在研究如何利用深度学习提升推荐的准确性，例如通过学习用户的历史数据来更好地预测用户的喜好。

另外，随着推荐系统在实际应用中的普及，如何保证推荐系统的公平性也成为了一个重要的研究课题[18]。公平性指的是推荐系统不会因为用户的特征（例如性别、种族等）而对用户进行歧视性的推荐。研究者们正在努力开发能够保证公平性的协同过滤推荐算法。随着人工智能技术的发展，许多新的应用领域也开始使用推荐系统。例如，在自动驾驶领域，可以使用推荐系统来为车辆提供道路选择建议；在医疗领域，可以使用推荐

系统来帮助医生更好地为患者提供诊疗建议。因此，协同过滤推荐算法在新的应用领域的应用也成为了研究的热点。

随着计算资源和数据规模的不断增大，协同过滤推荐系统在分布式计算框架上的应用也成为了一个研究热点。分布式计算框架能够帮助更有效地利用计算资源，提高算法的计算效率[19]。

总之，协同过滤在推荐算法研究领域取得了很多成就，但是仍有很大的潜力可以挖掘。未来，协同过滤推荐系统仍将是一个非常热门的研究课题，并且有望在实际应用中取得更好的效果。还有一些其他的研究方向值得一提。例如，有些研究者致力于开发能够应对数据稀疏性和冷启动问题的推荐系统，数据稀疏性是指在推荐系统中，很多用户可能只浏览或购买了很少的物品，因此为这些用户进行推荐就变得困难[20-21]。冷启动问题则是指对于新用户或新物品，由于缺乏历史数据，推荐系统难以进行有效的推荐[22]。因此，如何有效应对这些问题是一个重要的研究课题。总的来说，协同过滤推荐系统是一个非常热门的研究课题，并且有很多不同的研究方向和应用领域。

1.4 课题研究的主要内容

本课题要求实现对电影推荐系统主要功能，并提供算法运行所需的平台。课题的设计内容主要包括：

(1) 实现基于模型的电影推荐算法。

(2) 实现基于协同过滤的推荐算法。

(3) 实现基于内容的推荐算法。

(4) 收集并准备相应的数据，可以从网上获取公开的电影数据集或从电影网站或软件中收集用户的历史浏览记录、喜好、评分等信息。

(5) 进行特征工程，提取出有意义的特征，如电影类型、演员、导演等。

推荐算法常用的的设计指标主要包括[23]：

(1) 准确率：衡量推荐系统向用户推荐的电影中，用户实际喜欢的电影的比例。

(2) 召回率：衡量推荐系统能够向用户推荐到用户喜欢的电影的比例。

(3) 覆盖率：衡量推荐系统能够向用户推荐到的电影种类的数量。

(4) 多样性：衡量推荐系统向用户推荐的电影的种类的多样性。

(5) 新颖度：衡量推荐系统向用户推荐的电影是否为新颖的电影。

(6) 平衡性：衡量推荐系统向用户推荐的电影是否平衡，不会偏向某一类型的电影。

在本课题中使用 RMSE 作为评估准确度的主要指标。

1.5 本文主要工作和结构安排

在深入研究和分析了国内外电影推荐系统的现状之后，以及全面了解了电影推荐所涉及的基本要求，本文结合了电影网站的实际需求，将这些需求有机融入到推荐系统的开发中。接下来，将按照以下组织结构来展开论述：

第 1 章聚焦于分析推荐系统的开发背景、国内外的推荐算法需求、开发推荐系统的意义以及推荐系统开发要实现的目标。

第 2 章旨在介绍推荐系统所采用的关键技术以及项目构建所使用的工具。

第 3 章重点分析项目开发的可行性，通过业务描述、功能性需求和非功能性需求的描述来确定开发目标。

第 4 章着重介绍推荐系统实现的目标，并通过说明各个用户端和重要模块来明确系统实现的目标。最后，设计了相关数据库表格和内容，展现各个实体之间的关系。

第 5 章主要介绍推荐系统的推荐算法并进行解释说明。对各个算法的实现进行详细解释，并提供性能分析和比较，以帮助用户理解推荐系统开发的理念和具体操作。

第 6 章的主要目的是对推荐系统进行测试，对系统的功能进行排查，并对存在问题的功能进行修改和优化，以提升用户的使用体验。

1.6 本章小结

本章主要介绍了推荐系统的课题背景、国内外发展现状、意义等，让多种类别的用户了解到本系统的用途和优势。本章介绍了推荐系统要实现的基本目标，以及每个流程所要实现的内容，对用户能够更好的了解平台的功能和作用。

本章重点涵盖了推荐系统的背景、国内外发展现状和意义，旨在向各类用户介绍本系统的用途和优势。详细阐述了推荐系统所追求的基本目标以及每个流程所要实现的内容，以帮助用户更好地理解平台的功能和作用。通过本章的阐述，可以深入了解推荐系统的重要性，并了解如何通过该系统提供个性化的推荐结果，从而提升用户体验。

第 2 章　系统相关技术

2.1　系统开发及运行环境

本课题以 MovieLens 的视频库为基础，结合某影视网站的实际商业数据，构建了一套影片推荐体系。该模型将提供在线与离线线两种不同的推荐方式，结合协作筛选与以内容为基础的信息处理技术，实现了两种信息的融合。其中包括了系统的前端应用、系统的设计与实施以及系统的开发。本章节主要介绍了本课题所涉及到的主要功能模块。

用户可视化：它的作用是为用户提供互动以及数据呈现，它是利用 AngularJS2 来完成的，并且在 Apache 上进行了部署。

综合业务服务：主要实现业务逻辑，用 Spring 构建并部署在 Tomcat 上。

【数据存储部分】

服务数据库：以 MongoDB 为核心的服务数据的存储平台。

搜索服务器：使用 ElasticSearch 作为模糊检索服务器，实现基于内容的推荐。

高速缓存数据库：采用 Redis 技术进行高速的高速数据采集，实现了实时查询。

【离线推荐部分】

脱机统计服务：利用 SparkCore+SparkSQL 实现批量的统计，对指数级的数据进行统计。

脱机推荐：利用 Spark 核心+Spark MLlib 框架，利用 ALS 算法，完成脱机推荐。

工作调度服务：使用 Azkaban 调度离线推荐部分的任务。

【实时推荐部分】

记录收集服务：使用 Flume-ng 收集商业平台上的一部影片的记录，并将其实时传送至 Kafka 簇。

信息缓冲区：本工程使用 Kafka 对流数据进行缓冲区，接收 Flume 发送的信息。并把这些信息推送给该计划的即时建议系统。

实时推荐服务：本课题以 Spark Streaming 为平台，利用 Kafka 中的缓存数据，开发相应的推荐方法，完成对数据的分析，并在 MongoDB 数据库中进行整合和更新。

2.2　技术简介

2.2.1　机器学习简介

机器学习（Machine Learning，简称 ML）是一种通过数据训练模型来实现任务的技

术。在传统的程序设计中，程序员需要手动编写程序来完成特定的任务。而在机器学习中，开发人员并不直接编写程序，而是提供一些数据作为输入，然后让计算机根据这些数据自己找到规律和模式，生成一个可用的模型。这个模型可以用于完成各种任务，如分类、聚类、预测等。

机器学习技术的应用非常广泛，包括图像识别、自然语言处理、推荐系统、金融风控等领域。

2.2.2 MongoDB 简介

MongoDB 是一个开放源代码，高性能，无模式的文件型数据库，其最初的目的是为了便于开发，便于扩充，属于 noSQL 数据库的一类。它是与关系数据库（MySQL）最相似的一种。

其支持的资料架构很宽松，跟 JSON 差不多，叫做 BSON，因此，其不但能储存更多的资料，而且还很有弹性。

在 MongoDB 中，一条记录是一种文件，是一种包含了一个字段和值对（field: value）构成的一种数据结构，与 JSON 对象相似，也就是把一个文件看作一个对象。栏位的资料型别为字元，其数值可以包含其他文件、一般阵列和文件阵列，以及其他文件阵列。

2.2.3 Redis 简介

Redis 是一个完全开源免费的软件，遵循 BSD 协议。它是一个灵活高性能的 key-value 数据结构存储，可用于数据库、缓存和消息队列。Redis 具有以下三个特点：

(1) 支持数据持久化，可以将内存中的数据保存在磁盘中，重启时可重新加载。

(2) 提供多种数据结构存储，包括 list、set、zset 和 hash 等。

(3) 支持主从复制，实现数据备份和故障恢复。

2.2.4 Elasticsearch 简介

Elasticsearch 是一款用于各种资料的自由和公开的分散式搜寻与分析工具，包括文字，数字，地理空间，结构化与非结构化资料等等。Elasticsearch 基于 Apache Lucene，于 2010 年首先由 Elasticsearch N. V.（现为 Elastic）公布。Elasticsearch 以其简洁的 REST 风格 API，分布式特征，快速和可扩充性而著称。Elastic Stack 是一个自由开放的软件包，用于数据收集，扩展，存储，分析和可视化。Elastic Stack 被广泛地称作 ELK Stack（分别代表 Elasticsearch, Logstash, Kibana），其中包含了大量的轻型数据收集 Agent，它们被称作 Beats，用于 Elasticsearch 中的数据收集。

2.2.5 Azkaban 简介

Azkaban 是一种批工作流程的任务分派工具，它被 Linkedin 所开放。用来在工作流中按照一定的次序执行一系列的工作和过程。Azkaban 通过 KV 文档的形式，在不同的

工作中创建了不同的相关性，并且为网络用户接口的维持和追踪工作流程提供了方便。

具有以下功能性特征：

(1) 网络用户接口。

(2) 便于工作流程的上载。

(3) 便于对各作业间的关联进行设定。

(4) 对工作流程进行安排。

(5) 验证/批准。

(6) 使工作流程可以被关闭和再开始。

(7) 一种模块化、可插入式的插件式插入装置。

(8) 专案办公空间。

(9) 对工作流程、工作流程、工作流程进行记录、审核。

2.2.6 Apache Spark 简介

Apache Spark 是一个面向海量数据的分布式开放源代码处理器。采用了在存储中进行缓冲，并采用了一种最优的方法来实现对任意大小的资料的快速分析。该软件提供了一个 API，使用 Java, Scala, Python, R 等多种工作方式来实现对程序的复用，包括批量处理，互动查询，实时分析，机器学习，图处理等等。

各行业的众多组织都使用它，其中包括 FINRA、Yelp、Zillow、DataXu、Urban Institute 和 CrowdStrike。Apache Spark 已经成为最受欢迎的大数据分布式处理框架之一，在 2017 年拥有 365000 名会定期参加聚会的会员。

2.2.7 ZooKeeper 简介

ZooKeeper 是一种集中的、保存组态资讯、为组态资讯、并为组态资讯、组态资讯及组态资讯而设的伺服器。

这是一款由谷歌 Chubby 开发的开放源码软件，也是 Hadoop 和 Hbase 的关键组成部分。ZooKeeper 的目的就是要将那些容易出现错误的、复杂的、重要的、易于使用的、有效的、稳定的界面展示出来，并同时支持 java 和 C 两种接口。

2.2.8 Flume-ng（next generation）简介

Flume 是一种分布式的，可靠的，并且可以有效地收集，聚合和移动海量的日志。该系统具有一种简单且可伸缩的体系结构，该体系结构以流式数据为基础。该系统鲁棒性强、容错性好、可靠度高，并具备多种失效切换与恢复功能。该软件采用了易于扩充的数据模式，可以进行联机分析。

2.2.9 Kafka 简介

Kafka 是一个开放源代码的流处理器，它是 ApacheSoftware Foundation，用 Scala 和

JAVA 语言编写。旨在构建一个统一的、具有高吞吐能力的、低延迟的、具有高吞吐能力的、具有较强计算能力的系统。其持久层基本上就是一个“基于分布式交易记录体系结构的大量发行/订购信息队列”，这使得其在管理流数据方面具有很高的价值。

2.3 本章小结

在这一章中，着重讨论了以协同过滤为基础的影片推荐系统的开发工具，运行环境。

第 3 章　系统可行性和需求分析

3.1　可行性分析

（1）技术可行性

本研究所使用的技术栈为业界所广泛认可并使用的，因此需要使用的工具和技术在公共互联网上都可以找到。所使用的数据集为 MovieLens，开源所以不存在获取难度。

难点在于将所有组件配置到本地并相互串联，让数据在其中流通，以及服务器（实验项目用虚拟机代替）的配置，以及推荐算法选型，找出最优推荐算法。

（2）时间可行性

计划整个项目完成时间为两到三个月，关键步骤为项目构建和论文撰写。主要使用模块开发法进行项目开发。项目风险主要在于有些技术较新颖以前没有接触过，但是经查阅都有比较详实的教程与社区可以进行学习，所以不会对项目进度有影响。

（3）经济可行性

当项目正式上线运行后，成本主要来源于服务器的租用费用，收益主要来源是页面广告。

（4）法律可行性

项目所用数据与组件都是开源免费的，用户数据仅收集系统运行所必要的。因此在法律上项目具有可行性。

（5）用户可行性

项目主要面向于电影爱好者，帮助他们从大量电影中找出喜欢看的电影。设计参考了目前成熟的电影网站，易于用户使用。相较于对于其他小型电影网站，项目的优势是提供了丰富的推荐方式，方便用户发现潜在的喜欢的电影。因此项目具有用户可行性。

3.2　业务描述

本项目的核心业务是基于电影资料库的电影推荐。目标是向观众推送最符合口味的电影。推荐的方式有统计推荐服务，基于内容的推荐，离线推荐服务和实时推荐服务。

除了推荐服务外，系统还提供了搜索服务还有电影打分功能。另外设计了前台来展示上述服务的结果。

3.2.1　离线统计服务

大致流程是首先从数据库中把电影数据载入然后通过计算和一些 SQL 操作得到电

影的平均评分、评分个数、最近电影评分还有每一类电影的 TOP10。之后将结果再存入数据库等待调用。

3.2.2 基于 LFM 的离线推荐服务

LFM 隐语义模型（Latent Factor Model）作为协同过滤算法的核心方法。LFM 模型通过分析用户和物品之间的关联关系，生成用户对目标物品的评分预测。为了实现推荐功能，系统使用了 ALS（Alternating Least Squares）算法作为协同过滤的优化算法。相对于传统的梯度下降算法，ALS 算法能够并行计算，提高了计算效率。

在这个部分中，系统根据存储在 MongoDB 中的用户评分表和电影数据集，分别计算用户电影推荐矩阵和电影相似度矩阵。为了计算用户电影推荐矩阵，系统首先对用户 ID 和电影 ID 进行笛卡尔积，生成（用户 ID，电影 ID）的元组。然后，利用训练好的 ALS 模型对这些元组进行评分预测。预测结果根据预测分值进行排序，并选取分值最高的 K 个电影作为当前用户的推荐结果。最后，系统将推荐结果保存到 MongoDB 的 UserRecs 表中。

另外，系统还利用 ALS 算法计算了电影间的相似度矩阵，该矩阵用于查询当前电影的相似电影，为实时推荐系统提供支持。通过计算每对电影之间的相似度余弦值，我们可以得到电影间的相似度。这些相似度在一段时间内基本保持不变。最终，系统将计算得到的相似度数据保存到 MongoDB 的 MovieRecs 表中。

服务通过 LFM 隐语义模型和 ALS 算法实现了用户电影推荐和电影相似度计算。通过存储和查询 MongoDB 中的数据，系统能够为用户提供个性化的电影推荐和实时的相似电影查询服务。

3.2.3 实时推荐服务

首先，为了实现实时推荐设计了几个指标。

(1) 计算速度要快。

(2) 推荐的结果不用特别精确。

(3) 有预先设计的推荐模型。

然后，具体的业务流程如下：首先，系统接收到用户对某一电影的评分，通过缓存的电影相似度矩阵将与这部电影相似的电影先选出来，然后通过缓存的用户最近 K 次评分与推荐电影列表做加权计算后得出推荐优先级，这样就得到了推荐列表，最后将待推荐列表存入数据库。

3.2.4 基于内容的推荐

基于内容的推荐就是根据用户对电影的偏好来推荐，使用到了基于物品(item-based)的协同过滤（Item-CF 算法)。首先使用 TF-IDF（term frequency－inverse document

frequency，词频-逆向文件频率）算法对用户选择的电影简介进行特征抽取得到电影的特征向量，然后根据这个电影的特征向量与总电影相似度矩阵做比对按照相似度高低排序得到电影推荐列表，最后将待推荐列表存入数据库。

3.3 功能性需求

通过上述业务描述，了解了各个模块的运行过程。接下来进行需求分析。需求分析的主要任务就是明确推荐系统具体要达成的目标，对推荐系统各个模块提出更为具体以及完整的要求。推荐系统主要功能有：

(1) 离线统计，离线统计可以在用户提供自己的兴趣之前将存储的电影按照统计学的方式推荐出来。策略有按照平均分推荐，按照热度（被打分次数）推荐，按照最近热度（最近被打分次数）推荐。

(2) 内容推荐，内容推荐可以在用户点选了自己感兴趣的电影标签时将存储的电影按照协同过滤的方式推荐出来。

(3) 离线推荐，离线推荐可以在用户在网站上留下了自己的特征之后通过 ALS 算法得到用户的推荐列表。

(4) 实时推荐，实时推荐可以在用户在网站上留下了自己的特征之后通过 ALS 算法以及之前缓存的数据快速地更新用户的推荐列表。

除了推荐算法模块外，项目还应该包含一个用于展示所有推荐算法的平台。其中应该给用户分配账号。用户的用例图如图 3-1 所示。

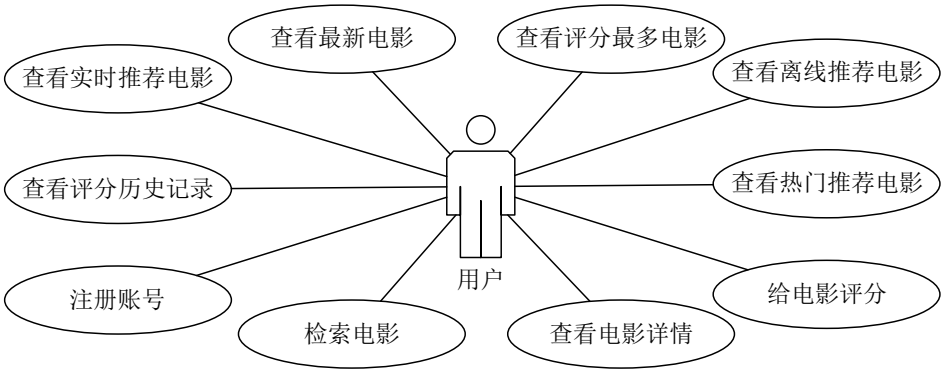


图 3-1　用户用例图

3.4 非功能性需求

系统的开发过程中，除了功能需求之外，非功能性需求也起着至关重要的作用。在软质量模型中，需要考虑软件系统的性能、可靠性、安全性、可维护性、易用性等方面的外部质量和内部质量需求。这些非功能性需求对于确保系统的高效运行和用户满意度

至关重要。

3.4.1 需求分析阶段的性能需求

电影推荐系统中的实时推荐算法对响应速度有要求。系统要求在用户点击请求后不超过 2 秒钟内完成响应，验证方法为运行系统模拟并评估响应时间。

3.4.2 需求分析阶段的可靠性需求

可靠性需求通常指软件系统需要具备的故障恢复能力、数据备份能力等方面的要求。

为了防止同一时间请求数据过多导致数据库宕机，采用了缓存数据库 redis。一是提供了缓冲，防止数据库被过多请求导致宕机。二是将常访问的数据存储下来，加快了数据的存取速度。同时，系统需要每隔 3 天进行一次完整备份，并保留至少 3 个版本的备份数据；验证方法为检查备份记录，并进行数据恢复测试。

3.4.3 需求分析阶段的安全性需求

安全性要求通常指软件系统需要具备的用户数据保护、系统访问控制等方面的要求。

首先，系统要求用户密码不能以明文形式存储，且用户数据需要进行加密保护；验证方法为检查数据存储格式，评估加密方式是否满足要求。其次，只有经过授权的用户才能访问系统功能，且对不同用户角色进行访问权限控制；验证方法为分别使用不同的用户角色进行系统访问测试，评估权限控制效果。

3.5 本章小结

在本章对系统进行了综合分析，考虑了需求和可行性两个方面。通过对系统需求的分析，明确了系统应该具备的功能，并为系统模块的详细设计提供了基础。此外，也通过对系统需求的描述，证明了构建系统是可行的，从而为后续的设计和实施奠定了坚实的基础。

第 4 章　系统设计

4.1　系统概要设计

本系统的主要目标是实现推荐算法和提供一个载体将算法结果表现出来。主要解决的问题有：实现推荐算法，并使算法按预期效果运行；设计交互模块将算法结果展示。因此系统采用 B/S 架构，模块之间通过日志文件进行间接通信，通过请求进行直接通信。

4.2　系统结构设计

根据第 3 章对系统需求的分析，本系统大致分为五个部分。

数据采集和处理模块：负责采集和处理用户行为数据、电影元数据等信息，并对数据进行清洗和转换，为后续的模型训练和推荐服务提供数据基础。

协同过滤模型训练模块：基于采集和处理得到的数据，利用协同过滤算法进行模型训练，建立起能够对用户和电影进行有效推荐的模型。

推荐服务模块：根据用户的行为信息和电影元数据，使用机器学习模型进行实时推荐，向用户推荐可能感兴趣的电影。

用户交互界面模块：提供友好的用户交互界面，支持用户查看电影列表、搜索电影、查看电影详情、对电影进行评分等功能，同时支持个性化推荐服务，为用户提供更好的体验。

系统管理模块：负责系统的配置管理、日志管理、性能监控等工作，保证系统的可靠性和稳定性。

电影推荐系统的总体功能结构图如图 4-1 所示。

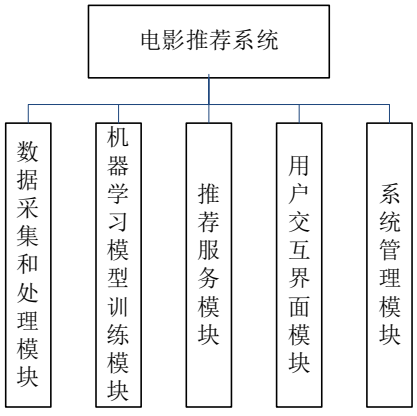


图 4-1　电影推荐系统的总体功能结构图

4.2.1 数据采集和处理模块

数据采集和处理模块的作用是收集和处理原始数据，以生成有用的信息和知识。然后，这些数据会通过算法和技术进行处理和分析，以识别和提取有价值的信息和知识，例如预测趋势、分类数据和优化决策等。这些过程可以帮助用户更好地了解他们的喜好，从而做出更明智的决策和行动。在本项目中，数据的采集是采用的公开的数据集，通过运行程序子模块加载和处理并存入数据库中。数据采集和处理模块功能结构图如图 4-2 所示。

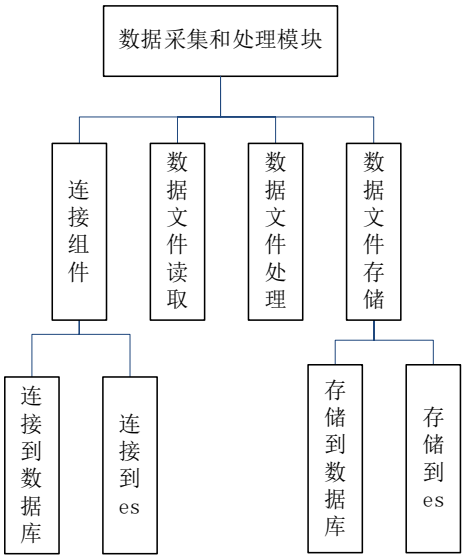


图 4-2　数据采集和处理模块功能结构图

4.2.2 机器学习模型训练模块

机器学习模型训练模块是本项目的核心模块之一，主要用于对推荐模型进行训练。模型训练是机器学习的关键环节，目的是从数据中学习模型的参数或者结构，使得模型能够对未知的数据做出准确的预测。

在机器学习模型训练时，我们需要为模型提供大量的标注数据，根据训练数据不断调整模型的参数或者结构，最终使得模型的预测结果尽可能地接近真实结果。通过反复迭代，调整，最终得出一个性能比较优秀的机器学习模型。

机器学习模型训练模块包含多个子模块，例如数据预处理、特征提取、模型选择、超参数优化等，每个子模块都需要细致调整和优化才能保证整个模型具有良好的性能表现。机器学习模型训练模块功能结构图如图 4-3 所示。

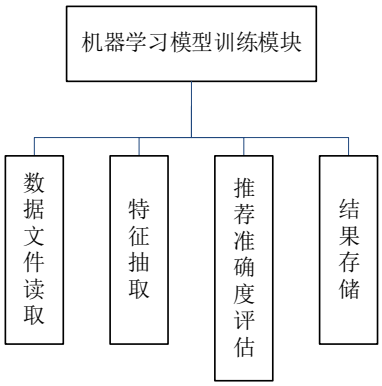


图 4-3　机器学习模型训练模块功能结构图

4.2.3　推荐服务模块

推荐服务模块可以根据用户的多种特征（如产品浏览历史、个人信息、地理位置、评论行为等）来识别出用户喜好，并向其推送内容。本系统主要采用基于协同过滤的算法来进行推荐。

4.2.4　用户交互界面模块

基于 Spring 的网页端模块主要负责用户界面的展示、用户交互和推荐结果的呈现。该模块主要有下面几个功能：

(1) 用户认证和管理：该模块提供用户注册、登录和注销等功能，确保用户身份的认证和管理。用户可以通过注册和登录来访问系统，系统可以根据用户的身份进行推荐结果的个性化展示。

(2) 电影展示和搜索：该模块可以展示电影列表和电影详情页面，用户可以浏览不同类型的电影、查看电影的详细信息，如标题、剧情简介、演员表等。用户还可以使用搜索功能来查找特定的电影。

(3) 用户交互：该模块支持用户与系统之间的交互操作。用户可以对电影进行评分、评论和收藏，系统可以根据用户的行为和反馈来调整和优化推荐结果。

(4) 推荐结果展示：该模块将个性化的推荐结果展示给用户。推荐结果可以根据用户的兴趣、历史行为和其他因素进行计算和排序，以提供用户最相关和吸引力的电影推荐。

4.2.5　系统管理模块

系统管理模块用于对系统的算法模块进行调度，定时地进行离线推荐数据的处理。

4.3 数据库设计

数据库设计与系统功能、效率、一致性和安全性有密切的关系，它使得在进行大量数据存储和访问时减少存储量，提高信息准确度成为可能，以及可以有效的实现多项任务。通过正确的数据库设计，可以显著地减少冗余信息，形成一个合理、一致及高效的数据存储模型。

4.3.1 数据库实体图

数据库实体图是用来描述数据库相关信息的可视化工具。它可以展示数据库内容的逻辑框架，帮助开发人员和用户明白数据库的组成、用途和相互之间的联系，从而有助于业务、分析方面的理解和优化处理。

在电影推荐系统中，主要的实体图就是用户实体图和电影实体图。用户实体应该包括的属性有：用户 ID，用户名，用户密码，是否是第一次登陆，用户偏爱的电影类型，用户创建时间。用户 ID 用于唯一的标识一个用户，是否是第一次登陆用于解决冷启动问题。用户实体图如图 4-4 所示。

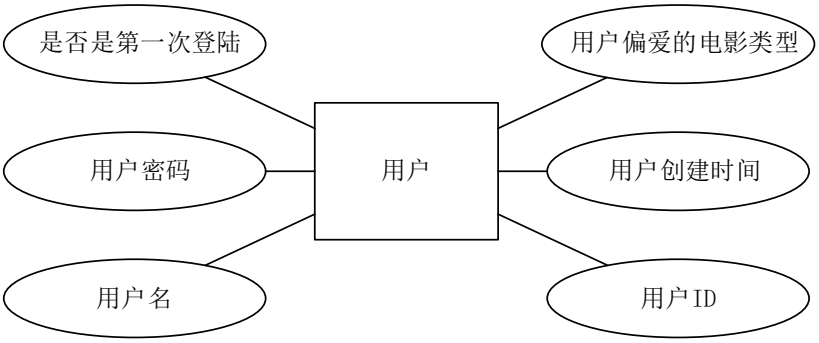


图 4-4　用户实体图

电影实体图应该包括的属性有：电影的 ID，电影的描述，电影的时长，电影拍摄时间，电影发布时间，电影语言，电影所属类别，电影的导演，电影的演员。电影实体图如图 4-5 所示。

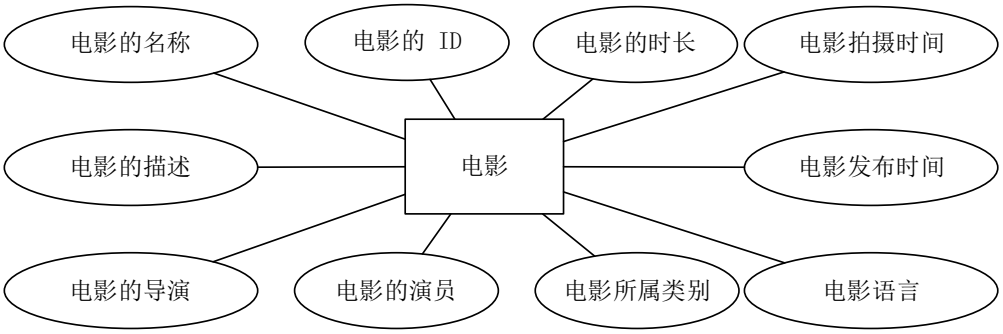


图 4-5　电影实体图

4.3.2 数据库表设计

数据库表设计指的是在数据库中创建表格以存储数据的过程。表设计需要确定每个表格的列和数据类型，以及规划不同表格之间的关系和索引，以便在存储和检索数据时能够高效地完成操作。良好的表设计方案可以提高数据的准确性、可靠性和安全性，并使数据查询和分析更加便于进行。下面介绍各个表的详细信息。

（1）影片资料表用来储存影片资料。主要存储的数据包括了：电影的 ID、电影的名称、电影的描述、电影的时长、电影的拍摄时间、电影的发行时间、电影的语言、电影的归属类别、电影的导演、电影的演员。表的结构如表 4-1 所示。

表 4-1 电影数据表（Movie）

列名	数据类型	长度	可否为空	说明
mid	Int	32	否	电影的 ID
name	String	16	否	电影的名称
descri	String	16	否	电影的描述
timelong	String	16	否	电影的时长
shoot	String	16	否	电影拍摄时间
issue	String	16	否	电影发布时间
language	String	16	否	电影语言
genres	String	16	否	电影所属类别
director	String	16	否	电影的导演
actors	String	16	否	电影的演员

（2）使用者打分表格是用来储存使用者打给影片的打分。其中保存的信息包括：使用者 ID、影片 ID、影片分数、影片打分时间等。表的结构如表 4-2 所示。

表 4-2 用户评分表（Rating）

列名	数据类型	长度	可否为空	说明
uid	Int	32	否	用户的 ID
mid	Int	32	否	电影的 ID
score	Double	64	否	电影的分值
timestamp	Long	64	否	评分的时间

（3）影片标记表格是用来储存由使用者贴上的影片的标记。储存的资料包括使用者 ID、影片 ID、影片标签、影片时长等。表的结构如表 4-3 所示。

表 4-3 电影标签表（Tag）

列名	数据类型	长度	可否为空	说明
uid	Int	32	否	用户的 ID
mid	Int	32	否	电影的 ID
tag	String	64	否	电影的标签
timestamp	Long	64	否	评分的时间

（4）User 表格是用来保存 User 数据的。储存的资料包括使用者的 ID、使用者名称、

使用者的口令、使用者是否为首次登入、使用者喜欢的影片种类、账户建立时间。表的结构如表 4-4 所示。

表 4-4　用户表（User）

列名	数据类型	长度	可否为空	说明
uid	Int	32	否	用户的 ID
username	String	64	否	用户名
password	String	64	否	用户密码
first	boolean	1	否	是否第一次登录
genres	List<String>		否	用户偏爱的类型
timestamp	Long	64	否	用户创建的时间

（5）最新影片分数统计表格是用来为影片保存最新的评估。储存的资料包括影片的 ID、影片的分数、评分时间。表的结构如表 4-5 所示。

表 4-5　最近电影评分个数统计表（RateMoreMoviesRecently）

列名	数据类型	长度	可否为空	说明
mid	Int	32	否	电影的 ID
count	Int	32	否	电影的评分数
yearmonth	String	64	否	评分的时段

（6）影片分数统计表格是用来储存影片的最新分数数目的。储存的资料包括影片的 ID、影片的评价数目。表的结构如表 4-6 所示。

表 4-6　电影评分个数统计表（RateMoreMovies）

列名	数据类型	长度	可否为空	说明
mid	Int	32	否	电影的 ID
count	Int	32	否	电影的评分数

（7）影片平均分数表是用来储存影片的平均分数的。储存的资料包括影片的 ID、影片的平均评分。表的结构如表 4-7 所示。

表 4-7　电影平均评分表（AverageMoviesScore）

列名	数据类型	长度	可否为空	说明
mid	Int	32	否	电影的 ID
avg	Double	64	否	电影的平均评分

（8）影片相似度矩阵是用来储存影片的影片相似度矩阵。主要储存的资料是与影片最类似的影片和一组影片的 ID。表的结构如表 4-8 所示。

表 4-8　电影相似性矩阵表（MovieRecs）

列名	数据类型	长度	可否为空	说明
mid	Int	32	否	电影的 ID
recs	Array[(mid:Int,score:Double)]		可	该电影最相似的电影集合

（9）使用者影片建议矩阵是用来储存向使用者推荐的影片。主要的储存资料是向其

建议的一组影片和使用者的 ID。表的结构如表 4-9 所示。

表 4-9　用户电影推荐矩阵表（UserRecs）

列名	数据类型	长度	可否为空	说明
mid	Int	32	否	用户的 ID
recs	Array[(mid:Int,score:Double)]		可	推荐电影集合

（10）利用一个动态的影片推荐矩阵来储存影片相似度矩阵。主要的储存资料是向其建议的一组影片和使用者的 ID。表的结构如表 4-10 所示。

表 4-10　用户实时电影推荐矩阵（StreamRecs）

列名	数据类型	长度	可否为空	说明
mid	Int	32	否	用户的 ID
recs	Array[(mid:Int,score:Double)]		可	实时推荐集合

（11）前十名的影片目录是用来储存每种影片的前十名。储存的资料有影片种类，前十名影片。表的结构如表 4-11 所示。

表 4-11　电影类别 TOP10 表（GenresTopMovies）

列名	数据类型	长度	可否为空	说明
genres	String	64	否	电影类型
recs	Array[(mid:Int,score:Double)]		可	TOP10 电影

4.4　本章小结

在本章中详细介绍了电影推荐系统所需实现的功能。对系统中各个实体之间的关系进行了分析，并设计了系统所需的数据库。同时，确定了数据库中需要包含的数据表以及每个表中具体的数据。

第 5 章　系统详细设计与实现

本章着重描述各个模块和子系统的实现，包括程序的算法设计、代码实现等。

5.1　实验设置

5.1.1　实验环境

系统实验环境如表 5-1 所示。

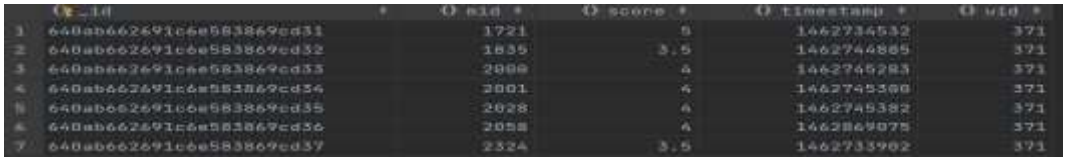
表 5-1　系统实验环境

	应用服务器	数据库服务器	客户端
硬件配置	CPU：Intel 酷睿 i5 9300H Memory：2GB HD：20G SSD	CPU：Intel 酷睿 i5 9300H Memory：2GB HD：20G SSD	CPU：Intel 酷睿 i5 9300H
软件配置	OS：CentOS 7 JDK 1.8.0 Tomcat 5.5 Mongodb 3.4 Elasticsearch 5.6 Redis 2.9 Kafka 3.4 Spark 2.1	OS：CentOS 7 JDK 1.8.0 Tomcat 5.5 Mongodb 3.4 Elasticsearch 5.6 Redis 2.9 Kafka 3.4 Spark 2.1	OS: Windows 11

5.1.2　实验数据

为研究系统的推荐效果，本实验采用经过修改的 MovieLens 数据实例，数据含义参照章节 4.3.2 数据库表设计。其中 movies 表如图 5-1 所示。其中 ratings 表如图 5-2 所示。其中 tags 表如图 5-3 所示。

图 5-1　movies.csv



	_id	mid	score	timestamp	uid
1	640ab6622691c6e583869cd31	1721	5	1462734532	371
2	640ab6622691c6e583869cd32	1835	3.5	1462744885	371
3	640ab6622691c6e583869cd33	2000	4	1462745283	371
4	640ab6622691c6e583869cd34	2001	4	1462745300	371
5	640ab6622691c6e583869cd35	2028	4	1462745382	371
6	640ab6622691c6e583869cd36	2058	4	1462869075	371
7	640ab6622691c6e583869cd37	2324	3.5	1462733902	371

图 5-2　ratings.csv

图 5-3　tags.csv

5.2　基于统计的离线推荐服务

离线统计推荐的目的是在不调用用户特征的情况下，通过统计分析给用户推荐可能喜欢的电影。

离线统计服务的算法描述是首先从数据库中把电影数据载入然后通过 SQL 操作得到电影的平均评分、评分个数、最近电影评分还有每一类电影的 TOP10，之后将结果再存入数据库等待调用。算法的具体流程见算法 5-1。

算法 5-1　离线统计算法原理
Data：电影信息表（Movie），用户评分表（Rating），电影标签表（Tag） Result：最近电影评分个数表（RateMoreMoviesRecently），电影平均评分表（AverageMoviesScore），电影评分个数（RateMoreMovies），电影类别 TOP10 表（GenresTopMovies） (1) 初始化变量：最近电影评分个数，电影平均评分，电影评分个数，电影类别 TOP10 (2) 连接数据库读取电影信息，用户评分，电影标签 (3) 电影评分个数:select mid count(mid) as count from ratings group by mid (4) 最近电影评分个数:select mid, count(mid) as count, yearmonth from ratingOfMonth group by yearmonth, mid order by yearmonth desc, count desc (5) 电影平均评分:select mid, avg(score) as avg from ratings group by mid (6) 电影类别 TOP10:对电影平均评分按电影类型分类后降序排列 (7) 存入数据库

5.3　基于内容的离线推荐服务

基于内容的推荐就是根据用户对电影的偏好来推荐类似的电影，使用到了基于物品(item-based)的协同过滤（Item-CF)。

5.3.1　TF-IDF 算法

全称为词频—逆文档频率（Term Frequency-Inverse Document Frequency），是一种

用于资讯检索和文本挖掘的常用加权技术。

当某个词在文章中的 TF-IDF 越大，那么一般而言这个词在这篇文章的重要性会越高，所以通过计算文章中各个词的 TF-IDF，由大到小排序，排在最前面的几个词，就是该文章的关键词。迁移到电影推荐来说，就是通过对电影评论使用该算法获得电影的特征向量。

TF-IDF 的计算步骤为首先计算出词频 TF。计算公式如下：

$$TF_{i,j} = \frac{n_{i,j}}{n_{*,j}} \tag{5-1}$$

其中，$TF_{i,j}$表示词语 i 在文档 j 中出现的频率，$n_{i,j}$表示 i 在 j 中出现次数，$n_{*,j}$表示 j 的总词数。

然后，计算逆向文件频率 IDF。计算公式如下：

$$IDF_i = \log\frac{N+1}{N_i+1} \tag{5-2}$$

其中，IDF_i代表词语 i 在文档集合中的逆文档频率，N 代表文档总数，N_i表示文档集合中包含 i 的文档数。

最后，通过两者相乘得到词语 i 的 TF-IDF 值。

5.3.2 算法设计

该方法的基本思想是：利用 TF-IDF 方法提取影片的基本信息，从而获得影片的特征值。在此基础上，求各影片的特征值，获得影片的相似性矩阵；在此基础上，利用相似性矩阵中的相似性，将影片从高到低进行分类，从而获得影片的推荐名单。

在进行相似性分析时，可用欧氏相似性和余弦相似性两种相似性分析方法进行分析。在这个单元里，我们采用了余弦相似性方法来进行影片间的相似性的计算。而余弦相似性则是利用矢量间的角度来衡量矢量间的相似性。

在此基础上，针对不同类型的影片，利用其特征矢量间的相似性度量其相似性。当两部影片之间的距离较近时，则表明两部影片之间具有较多的共同特性，从而更有可能被推荐。根据不同的相似性，对不同类型的影片进行分类，从而获得最终影片的推荐名单。

通过使用 TF-IDF 进行特征抽取和余弦相似度进行相似度计算，本模块能够为用户提供个性化的电影推荐。其中，欧氏距离的计算公式是：

$$d(x,y) = \sqrt{\sum_i (x_i - y_i)^2} \tag{5-3}$$

余弦相似度的计算公式是：

$$\cos\theta = \frac{a \cdot b}{||a|| * ||b||} = \frac{\sum_i x_i y_i}{\sqrt{\sum_i {x_i}^2} * \sqrt{\sum_i {y_i}^2}} \tag{5-4}$$

最后将待推荐列表存入数据库。算法的具体流程见算法 5-2。

算法 5-2　内容推荐算法原理
Data：电影信息表（Movie），基准推荐电影 Result：电影推荐列表 (1) 初始化变量：电影推荐列表，电影特征向量，电影相似度矩阵 (2) 连接数据库读取电影描述(Movie. descry) (3) While Movie. descry != Null do (4) |对 Movie. descry 进行分词 (5) |根据公式（5-1）和公式（5-2）计算得到每个词的 TF-IDF 值，并把每个词作为电影的特征向量，更新电影特征向量。 (6) |返回电影特征向量。 (7) end (8) 通过公式（5-4）计算基准推荐电影和其他电影的相似度，去除自乘，再去除小于 0.6 的。 (9) 按照相似度将所有电影降序排列存入数据库。这样就得到了基准电影的相似电影推荐列表。

5.4　基于隐语义模型的离线推荐服务

基于模型的推荐就是根据模型来推荐电影，使用到了基于模型((model based)的协同过滤（model-CF)。在基于模型的协同过滤中矩阵分解模型的单一精度较高，因此采用矩阵分解模型。ALS 是实现矩阵分解的具体算法。

5.4.1　推荐服务详细介绍

基于隐语义模型的离线推荐服务一般分为两个主要步骤：离线训练和在线推荐。

在离线训练阶段，首先需要对原始的用户-电影评分矩阵进行预处理和特征提取，得到一个低维的隐语义空间。具体的做法是使用矩阵分解模型，如交替最小二乘法等，对评分矩阵进行分解，得到用户和电影的隐向量表示。这些隐向量可以看作是对用户和电影进行了特征提取，表示了用户和电影的一些潜在特征，如喜好、属性等。然后，基于这些隐向量训练一个协同过滤模型来预测用户对电影的评分。在训练过程中，一般使用均方根误差（RMSE）或者平均绝对误差（MAE）等指标来评估模型的预测精度。

在在线推荐阶段，根据用户的历史评分行为，利用训练好的模型来预测用户对每个

电影的评分，然后将得到的预测结果进行排序，推荐给用户。使用的排序算法是基于评分的排序。

总的来说，基于隐语义模型的离线推荐服务主要涉及到矩阵分解、特征提取、机器学习模型训练和推荐排序等算法，其中矩阵分解和交替最小二乘法是其中的核心算法。

5.4.2 损失函数

损失函数的作用是用来评估模型在训练过程中的预测误差，它反映了协同过滤模型从数据映射到最终预测值的能力。通过不断优化损失函数可以使我们有效地将预测误差最小化，并将模型越训练越好，正确率也越高。在 spark 的 ALS 类中显式反馈的损失函数为：

$$J(U,V)=\sum_{i}^{m}\sum_{j}^{n}\left[\left(r_{ij}-u_i v_j^T\right)^2+\lambda\left(\left||u_i|\right|^2+\left||v_j|\right|^2\right)\right] \tag{5-5}$$

其中，U 表示用户对隐含特征的偏好矩阵，V 表示电影对隐含特征的归属矩阵，m 表示用户个数，n 表示电影个数，λ 表示正则项系数。

5.4.3 隐语义模型

隐语义模型的训练采用了矩阵分解的思想。这种方法的主要思想是将用户-电影评分矩阵分解为两个较小的矩阵，即用户-隐含特征矩阵 U 和电影-隐含特征矩阵 V。通过这种分解，可以捕捉到用户和电影之间的潜在关联和隐含特征。

实现隐语义模型的训练过程中，使用了交替最小二乘法（ALS）算法。这是一种优化算法，属于协同过滤算法中的一种。ALS 算法的核心思想是交替更新 U 和 V 两个矩阵，直到误差达到足够小的程度。在每次迭代中，ALS 算法固定一个矩阵，然后通过最小化均方误差的方式来更新另一个矩阵。这个过程会反复进行，直到模型收敛，即达到一个稳定的状态。

算法 5-3　隐语义模型训练

Data：用户评分表（Rating）

Result：用户-隐含特征矩阵 U，电影-隐含特征矩阵 V，预测矩阵

(1) 初始化变量：用户-隐含特征矩阵 U，电影-隐含特征矩阵 V

(2) 通过 Rating 矩阵使用 ALS 算法求解 U，V

(3) 返回用户-隐含特征矩阵 U，电影-隐含特征矩阵 V

(4) 返回矩阵 $\begin{bmatrix} \boldsymbol{U_1}*\boldsymbol{V_1} & \cdots & \boldsymbol{U_1}*\boldsymbol{V_j} \\ \vdots & \ddots & \vdots \\ \boldsymbol{U_i}*\boldsymbol{V_1} & \cdots & \boldsymbol{U_i}*\boldsymbol{V_j} \end{bmatrix}$ 作为预测矩阵，其中 $\boldsymbol{U_i}*\boldsymbol{V_j}$ 表示用户 i 对电影 j 的预测评分。

通过应用 ALS 算法进行隐语义模型的训练，可以从用户-电影评分矩阵中提取出潜

在的用户和电影特征，从而实现对用户对新电影的评分预测。这种方法可以有效地捕捉到用户和电影之间的关联，并为推荐系统提供个性化的推荐结果。具体来说，ALS 算法将原问题转化为两个子问题：固定 U 求解 V，固定 V 求解 U。在每个子问题中，将一个矩阵固定，然后通过最小化均方误差来更新另一个矩阵。算法流程如算法 5-3 所示。

5.4.4 测评指标

在训练中，以 RMSE（Root Mean Square Error）均方根误差作为模型优劣的指标。均方根误差是预测值与真实值偏差的平方与观测次数 n 比值的平方根。选用 RMSE 作为评价指标有如下优点：

(1) 直观易懂：RMSE 的计算方法简单，结果也容易理解，可以直接反映出预测值与真实值之间的误差。

(2) 稳定性高：RMSE 对异常值的敏感性较低，能够较好地应对数据中的噪声和离群点。

(3) 数学性质好：RMSE 是对误差平方的平均值开根号，具有良好的数学性质，可以方便地进行统计分析和优化。

计算公式为：

$$RMSE = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(boserved_t - predicted_t)^2} \tag{5-6}$$

其中 observed 为真实值，predicted 为预测值。N 为观测次数。RMSE 的值越小代表模型预测越精准。

5.4.5 模型训练

在理论上，实验通过 ALS（交替最小二乘法）训练隐语义模型，返回用户-隐含特征矩阵 U，电影-隐含特征矩阵 V，预测矩阵。

实际过程中，通过调用 sparkMllib 的 ALS 算法来实现。代码调用为：

$$Val\ model = ALS.train(trainData, rank, iterations, lambda)$$

其中，trainData 代表训练集，rank 表示特征向量的维度，iterations 是循环次数，lambda 代表正则项系数。

5.4.6 参数调整

通过将数据集按照四比一的比例随机切分为训练集和测试集，正式开始模型的训练。通过训练集将模型训练出来后，用测试集评估预测效果。测评指标如公式（5-6）在此，为了考虑性能与效率，将迭代次数规定为 5 次。训练数据如表 5-2 所示。

表 5-2　模型训练数据

Rank	Lambda	Rmse
50	0.01	1.378646
100	0.01	1.462907
200	0.01	1.492453
300	0.01	1.512931
50	0.1	0.926513
100	0.1	0.92457
200	0.1	0.924181
300	0.1	0.924322
50	1	1.345155
100	1	1.345156
200	1	1.345401
300	1	1.345395

5.4.7　实验结果及分析

训练结果如图 5-4 所示。其中，三条线分别代表了 λ 的不同取值，rank 表示特征向量的维度，lambda 代表正则项系数即 λ 。从图中可以看出，对推荐准确度影响最大的因素是 λ，即损失函数对推荐准确度影响最大，要想进一步提高准确度，应该优化损失函数。

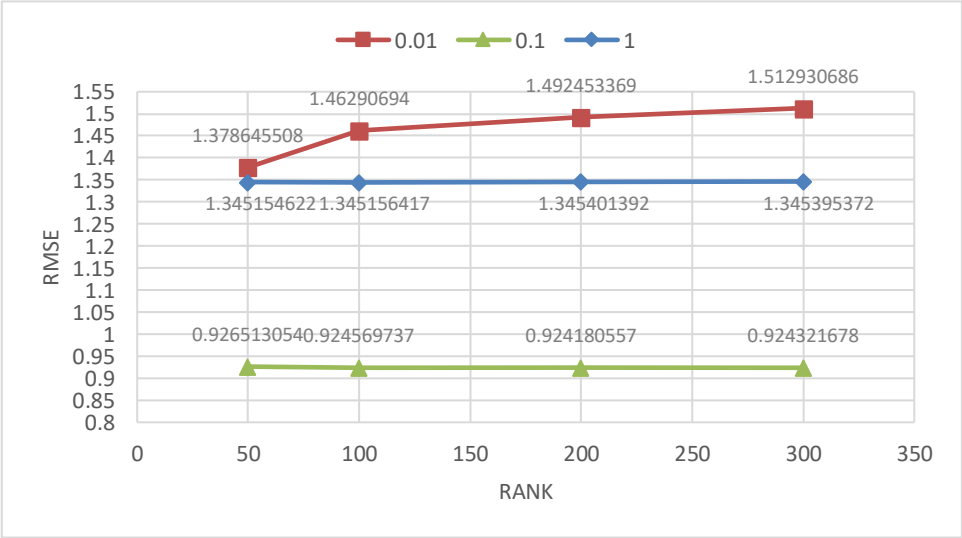


图 5-4　模型训练结果

根据训练结果可知，模型参数（rank,iterations,lambda）的较优取值为（200，5，0.1）。此时的 RMSE 有区间最小值为 0.92418。

5.4.8　隐语义模型更新

在模型训练完成之后，如果有新用户加入，就应该对隐语义模型进行更新，这样才能对新用户进行较为准确的推荐。

在新用户加入时，推荐系统需要将该用户加入到原有的用户-电影评分矩阵中，然后重新训练隐语义模型，得到更新后的用户-特征矩阵和电影特征矩阵。通常的做法是，将新用户的特征向量初始化为随机值，然后利用 ALS 算法对用户-电影评分矩阵进行分解，并不断迭代优化新的用户-特征矩阵和电影特征矩阵，直到模型收敛。这样就可以得到更新后的模型，从而进行推荐。

5.4.9 推荐算法设计

通过 ALS 训练出来的 Model 来预测当前用户的电影的推荐列表。算法具体流程见算法 5-4。

算法 5-4　基于模型的推荐算法原理
Data：电影信息表（Movie），用户表（User），用户评分表（Rating），ALS 模型 Result：用户电影推荐矩阵表（UserRecs） 1. 初始化变量：电影推荐列表，电影特征向量，电影相似度矩阵 2. UserId 和 MovieID 做笛卡尔积，产生（uid，mid）的元组 3. While（uid，mid） do 4. |ALS（uid，mid）通过 ALS 模型预测（uid，mid）的元组。 5. |返回预测结果。 6. end 7. 根据预测分值排序，返回分值最大的 K 个电影，更新 UserRecs 表。

在此基础上，利用 ALS 算法求解影片之间的相似性，实现影片之间的相似性搜索，从而实现影片之间的相似性搜索，为影片的实时推荐提供依据。

算法 5-5　电影相似矩阵求解
Data：用户评分表（Rating） Result：电影相似度矩阵（MovieRecs） 1. 初始化变量：用户-隐含特征矩阵 U，电影-隐含特征矩阵 V 2. 通过 Rating 矩阵使用 ALS 算法求解 U，V 3. 对$\boldsymbol{U_i}$，$\boldsymbol{V_j}$使用公式（5-4）求得$\boldsymbol{\cos（U_i，V_j）}$表示$\boldsymbol{U_i}$，$\boldsymbol{V_j}$间的相似程度 4. 返回矩阵$\begin{bmatrix} \cos（U_1，V_1） & \cdots & \cos（U_1，V_j） \\ \vdots & \ddots & \vdots \\ \cos（U_i，V_1） & \cdots & \cos（U_i，V_j） \end{bmatrix}$作为相似度矩阵，更新 MovieRecs 表

将 ALS 应用于 Rating 矩阵后，可获得一个用户特性矩阵 U(m*k)，其中，每一个用户都有 k 个特性来表示；一个 V（n*k）矩阵，它代表了一个影片的特征量，而每一个影

片又可以用 k 个特征量来描述。尽管每个维的特性含义未知，但 k 维的数理矢量代表了这条线所对应的影片的特性。

然后，可以对任意两个电影的特征向量求余弦相似度值，用来表达这两个电影间的相似程度，其求解公式见公式（5-4）。影片间的相似性在一定时期里是一个固定的数值，因此不必经常更新。详细的计算过程参见方法 5-5。

5.5 实时推荐服务

实时推荐与离线推荐最大的不同在于实时推荐强调算法的实时性，因此，耗时巨大的 ALS 算法并不适用于此。本模块的算法基于 item-CF 进行修改。

5.5.1 实时推荐模型

本模块基于 item-CF 并使用了加权的方法来优化推荐结果。推荐模型为：

$$E_{uq} = \frac{\sum_{r \in RK} sim(q,r) * R_r}{sim_{sum}} + \log \max\{incount, 1\} - \log \max\{recount, 1\} \qquad (5\text{-}7)$$

其中，E_{uq} 代表计算出的电影推荐分值，E_{uq} 的表达式由三部分组成，第一部分为基准项，评分依据主要来源于该项，后两项为偏移项用于调节推荐力度。

设置代表以时间次序最近的 K 个得分的参数 RK。针对使用者 u 与影片 r，将其描述为 Rr，其中 sim（q, r）代表影片 q 与影片 r 之间的类似程度。当影片 q 与影片 r 之间的相似性小于 0.6 时，就会被视作无关而被忽略。

对于影片 Q，我们会根据使用者 u 的最新 K 次评价来分析。我们将每个影片的相似性定义为与影片 q 的相似性超过最小门限的影片数目。incount 指这 K 个分数中类似影片 q 的分数更高（3 或更大）的影片数目。recount 指的是这 K 个分数中与影片 q 类似但分数更低的影片数目（低于 3）。

通过对这些指标的计算和分析，我们可以进一步了解电影 q 在用户 u 的评分历史中的相关性和受欢迎程度。

5.5.2 算法设计

这个算法的逻辑是这样的：如果一个人给一部影片 p 打了分数，就会引发一个关于他的推荐结果的更新。当使用者对影片 p 进行打分时，使用者与影片 p 最相近的影片间的推荐强度也会随之改变。所以，我们将选取最接近影片 p 的 K 部影片。

通过“建议优先权”的加权，每一部被选中的影片都被计算出了它在使用者 u 中的“建议优先权”。通过 u 的几个近期的评价，这些“候选影片”被计算出对 u 的推荐优先级。在此基础上，通过对已有影片的排序，将其与最近一期影片的排序进行融合，从而获得最新的影片推荐。详细的计算过程参见方法 5-6。

算法 5-6　实时推荐评分求解
Data：用户评分缓存，用户评分表（Rating），电影相似度矩阵（MovieRecs） Result：用户 u 的推荐列表 1. While（用户对某一电影评分）do 2. |根据公式（5-7）更新用户 u 的推荐列表 3. |返回用户 u 的推荐列表 4. End 5. 更新用户实时电影推荐矩阵（StreamRecs）

5.6　各推荐方式比较

通过记录各推荐算法运算时间，对四种不同的推荐方式进行时间开销方面的对比。对比如图 5-5 所示。通过对比，发现实时推荐的时间性能最好。

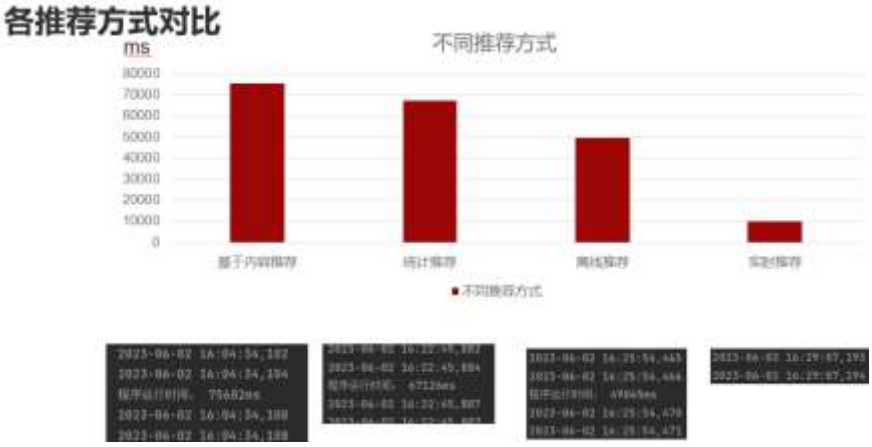


图 5-5　时间开销对比

5.7　本章小结

本章主要介绍了推荐算法的详细设计，各个模块和子系统的实现，包括程序的算法设计、代码实现等，以及不同算法之间的比较。

第 6 章　系统测试

6.1　系统测试概述

本章主要介绍了该系统的测试环境，测试方法和步骤，包括实验室测试和用户测试。实验室测试是在专业的软件/硬件测试实验室中进行的；用户测试是把新发布的软件交付给用户之前的最后一步。

6.2　测试环境

系统测试环境如表 6-1 所示。

表 6-1　系统测试环境

硬件环境	应用服务器	数据库服务器	客户端
硬件配置	CPU：Intel 酷睿 i5 9300H Memory：2GB HD：20G SSD	CPU：Intel 酷睿 i5 9300H Memory：2GB HD：20G SSD	CPU：Intel 酷睿 i5 9300H
软件配置	OS：CentOS 7 JDK 1.8.0 Tomcat 5.5 Mongodb 3.4 Elasticsearch 5.6 Redis 2.9 Kafka 3.4	OS：CentOS 7 JDK 1.8.0 Tomcat 5.5 Mongodb 3.4 Elasticsearch 5.6 Redis 2.9 Kafka 3.4	OS: Windows 11

6.3　登录功能测试

本测试面向前台登录界面，需要检测是否页面跳转正常和登录正常，这个功能模块可以完成从登录端口跳转到推荐页面。登录测试如表 6-2 所示。

表 6-2　后台登录测试用例表

测试编号	测试用例	测试数据	期望的结果	实际的结果
01	账号正确，密码不正确	账号：2019021194 密码：2019021194	出现账号或者密码错误提示	出现账号或者密码错误提示
02	账号没有输入	账号： 密码：2019021194	请填写账号	请填写账号
03	密码没有输入	账号：2019021194 密码：	请填写密码	请填写密码

测试通过，达到了预想的目的，登录跳转模块、验证模块是否正常。

6.4 运行效果

登陆界面如图 6-1 所示。

图 6-1 登陆界面

项目首页，为解决冷启动问题要求用户输入感兴趣电影类型，此时界面如图 6-2 所示。

图 6-2 用户首次登陆

此时用户选完自己喜欢的电影进入首页，实时推荐和离线推荐没有的原因是用户还没有对电影进行评分，因此无法推荐。如图 6-3 所示。

图 6-3 用户未评价时首页

因此选择一部电影进入详情页面进行评分，如图 6-4 所示。

图 6-4　电影详情页

当用户评价完电影后，实时和离线推荐更新。二者相同的原因是样本量太少，所以推荐结果一致。如图 6-5 所示。

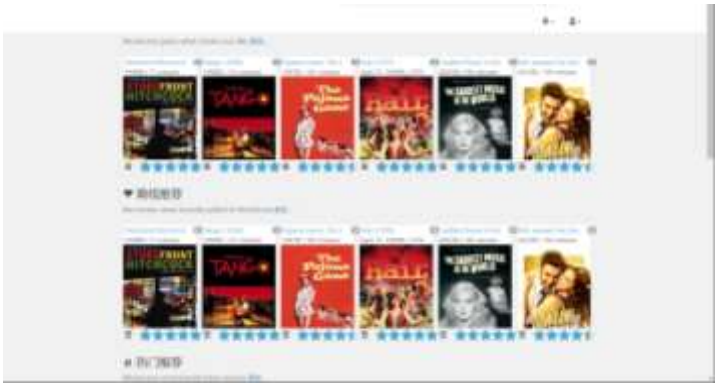


图 6-5　实时和离线推荐更新

当评论数量增多后，就可以看出二者不同，如图 6-6 所示。

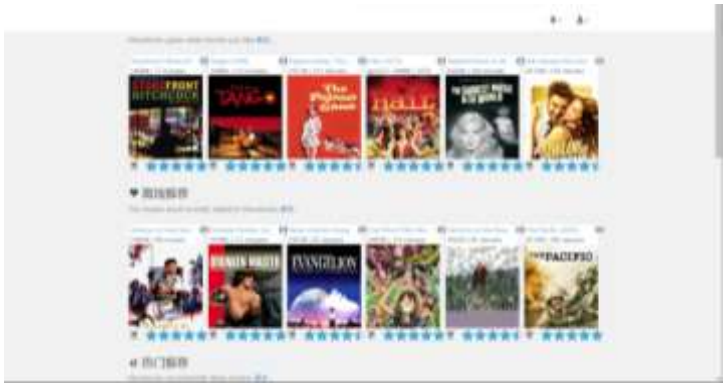


图 6-6　推荐出现差异

实时推荐相较于离线推荐时刻变动，体现出实时推荐的实时性。如图 6-7 所示。

图 6-7　体现实时性

以下是剩余的几种推荐方式，如图 6-8 所示。

图 6-8　其余推荐方式

6.5　本章小结

本章主要介绍了系统测试的目的，测试方法和原则，并选择了部分功能测试。

结　论

本文主要研究了一种基于协同过滤的电影推荐系统，通过对用户行为数据的分析和处理，实现了个性化的电影推荐功能。在实验中，我们使用了常用的评估指标来测试本系统的性能，结果表明，本系统具有较高的准确率，能够有效地提高用户对推荐电影的满意度和观影率。

本系统具有以下技术优点：首先，采用了多种特征工程技术，能够更全面地提取用户行为数据的信息，从而更好地进行电影推荐。其次，采用了多种协同过滤算法，能够更准确地预测用户的电影偏好。最后，本系统采用了分布式存储和计算技术，能够高效地处理大规模数据，实现了快速的推荐功能。同时，本项目还利用了现有的开源技术和工具，如 Spark、Kafka、MongoDB 等，实现了一套完整的电影推荐系统，具有良好的可扩展性和灵活性。本项目的设计和实现为电影推荐系统的开发提供了可参考的经验和方法。

虽然本系统在实验中表现出了较高的性能，但是还有一些方面需要改进。比如需要进一步优化推荐算法，提高推荐效果和速度；需要加强对用户隐私数据的保护和管理；需要加强系统监控和日志分析等方面。相信在不断迭代和优化的过程中，本项目可以不断发挥其价值，为用户和平台带来更多的方便。未来可以考虑如下改进方向：首先，采用更多的用户行为数据，以提高推荐的准确性和个性化程度。其次，进一步优化协同过滤算法的训练和参数调整，以提高预测准确率和推荐效果。最后，结合深度学习等新的技术手段，进一步提高系统的推荐性能和用户体验。

综上所述，本项目的设计和实现为电影推荐系统的开发提供了一种可行的思路和方法，同时为该领域的相关研究提供了一定的参考价值。虽然本项目仍存在一些待优化和改进的地方，但是其具有良好的可扩展性和商业前景，值得进一步研究和发展。

参考文献

[1] 胡琪,朱定局,吴惠斐,巫丽红.智能推荐系统研究综述[J].计算机系统应用,2022,31(04):47-58.

[2] 黄剑波,陈方灵,丁友东,吴利杰.基于情感分析的个性化电影推荐[J].计算机技术与发展,2020,30(09):132-136.

[3] 王康伟.基于机器学习的金融产品推荐算法研究[D].哈尔滨:黑龙江大学,2019.

[4] 周鹏.基于词嵌入的文本分类及新闻推荐算法研究[D].南京:南京邮电大学,2022.

[5] 李浩.基于评论文本的个性化推荐算法研究[D]. 南宁:广西大学,2022.

[6] 孙海峰,甘明鑫,刘鑫,吴越.国外电影推荐系统网站研究与评述[J].计算机应用,2013,33(S2):119-124.

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

[8] He X, Deng K, Wang X, et al. Lightgcn: Simplifying and powering graph convolution network for recommendation[C]//Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval. 2020: 639-648.

[9] 吴银豪. 面向婚恋系统的推荐算法研究[D].杭州:杭州师范大学,2021.

[10]李光明,房靖力.Spark 平台下电影推荐系统的设计[J].计算机应用与软件,2020,37(11):28-34.

[11]吕学强,王腾,李雪伟,董志安.基于内容和兴趣漂移模型的电影推荐算法研究[J].计算机应用研究,2018,35(03):717-720+802.

[12]周蔚生. 基于机器学习的混合电影推荐系统的设计与实现[D].北京:首都经济贸易大学,2021.

[13]冯萍,钱阳,李国梁,刘笑涵.融合图神经网络的深度学习电影推荐系统设计与实现[J].白城师范学院学报,2021,35(05):49-56.

[14]姜雨霖,冯相忠.基于协同过滤和用户画像的流媒体推荐系统研究[J].电脑与信息技术,2021,29(05):37-41.

[15]魏子钦,单豫洲,梁艳美.基于混合推荐算法的电影推荐研究[J].信息与电脑(理论版),2022,34(09):166-168.

[16]李雪婷,杨抒,赛亚热·迪力夏提,赵昀杰.融合内容与协同过滤的混合推荐算法应用研究[J].计算机技术与发展,2021,31(10):24-29.

[17]李宏志,李苋兰.融合改进的内容与协同过滤的博客推荐方法[J].湖南科技大学学报(自然科学版),2021,36(03):104-112.

[18]李文俊.推荐系统中公平性问题研究现状与展望[J].价值工程,2023,42(05):166-168.

[19]周望. 基于机器学习的推荐系统关键技术及其应用研究[D].成都:电子科技大学,2020.

[20]Liang Guanzhong,Wen Junhao,Zhou Wei. Individual Diversity Preference Aware Neural Collaborative Filtering[J]. Knowledge-Based Systems,2022,258.

[21]Zhang Yihao,Zhao Chu,Yuan Meng,Chen Mian,Liu Xiaoyang. Unifying attentive sparse autoencoder with neural collaborative filtering for recommendation[J]. Intelligent Data Analysis,2022,26(4).

[22]曹珂崄,张天舒,孙娟,彭二帅.基于社区结构的推荐系统中冷启动问题研究[J].现代信息科技,2023,7(01):30-32+35.

[23]杨正成,刘浩.基于 LightGBM 的广告商品平台推荐系统设计与应用[J].科技创新与应用,2022,12(30):1-6.

谢！