摘要

随着信息化社会的发展，推荐系统成为了解决信息过载的有力工具。本课题将使用用户画像来改善目前推荐系统存在的用户-评分矩阵数据稀疏的问题。设计且实现了基于用户画像的自适应推荐系统。系统的功能包括数据初始化、结合用户操作推荐相关产品等。

关键词：推荐系统，自适应，用户画像

**Abstract**

**Keywords:**

目录

1绪论

1.1 研究背景与意义

从互联网的诞生到现在的全面普及，数以万计的信息通过不同途径进入人们的视野，但是用户在面对大量的信息时，从中获得自己真正所需要的信息是十分麻烦的。为此，近些年来，推荐系统应运而生，具有巨大市场需求和研究价值。近年来，搜索引擎技术和信息检索技术发展迅速，在一定程度上缓解了信息过载的问题，如Google，Baidu等。然而这些系统并没有考虑单个用户所具有的不同兴趣喜好，检索出来的信息仍存在着大量的冗余信息，无法满足不同用户的不同需求。

随着Internet和信息技术的飞速发展，“信息过载”和“信息迷向”的问题愈发凸显，助推了个性化推荐系统的蓬勃发展[2]。推荐系统起源于互联网的沃土，并在这片土壤中蓬勃发展，尤其在电商、新闻、音乐，电影等领域体现出推荐系统的重要性。自适应推荐系统为用户从大量的数据中过滤掉无用信息，只给用户展示真正需要的信息，大大提高了用户体验以及用户对商家的依赖度。同时可以将用户需求转为商家的利润。目前众多知名网站都使用了不同形式的推荐系统来为自己的用户提供推荐服务，如淘宝，当一个用户检索一类商品时，网站根据用户的特点推荐出适合用户购买的商品，用户往往会在网站的推荐商品中购买到自己心仪的商品。这样，推荐系统给用户和商家提供了极大的便利。再如网易云音乐的推荐系统，它在每天会根据每个用户的音乐喜好形成一份用户特有的歌单[12]，极大的提高了用户对于此应用的粘性。等等，众多的应用程序表明了推荐系统在过滤无用信息、针对每个用户推荐适合自己的互联网资源上发挥的重要的作用，具有极高的使用价值和研究热点。

推荐系统一般基于用户的浏览记录、操作行为以及评分等行为，来推测出用户可能所需要和感兴趣的信息。推荐系统建立在海量的数据挖掘上，这些海量的数据我们可以通过系统的后台日志获取或者可以通过网络爬虫获取。

现阶段的推荐系统主要分为基于内容的推荐、协同过滤推荐、基于关联规则推荐、基于效用推荐以及基于知识的推荐和组合推荐。目前，协同过滤推荐算法是推荐系统中使用最多的算法，本文也将使用协同过滤算法，但是，传统的协同过滤算法会随着数据规模的增大出现数据稀疏性的问题，严重影响了该算法的应用，如淘宝等大型电商平台，用户过亿，项目数量也是大量的，但是也只有少部分用户给个别的产品进行过评级，这就造成了用户项目评分矩阵极端稀疏。所以，在本文中，将用户画像引入到协同过滤算法中，以提高推荐系统准确率。

用户画像是大数据的新兴概念，可将用户信息标签化来获得高度提炼的特征标识，本文将在协同过滤算法中结合用户画像，可在协同过滤给出的评分数据基础上根据用户偏好的附加信息给出推荐项目。

本文将使用MovieLens数据集作为推荐系统的数据来源。MovieLens-1m包含了6040名用户对3952部电影的1000000条评分。同时，还包含了用户的其他信息（性别、年龄、职业、邮编）和电影的信息（电影名、类型）。本文将在这些数据集的基础上构建出一个影片推荐系统。

综上所述，本文基于用户画像的自适应推荐系统可以对海量的用户网络行为进行数据分析，挖掘用户的特点，提取用户相关的特征信息，进行动态分析，让系统更好的了解用户喜好，从而给出自适应的推荐结果。

1.2 国内外研究现状

1.2.1 推荐系统

推荐系统自20世纪90年代发展成为一门独立的学科后，推荐系统飞速发展。推荐算法作为推荐系统的核心部分，也在不断优化发展。在目前的推荐算法中，以基于内容的推荐算法和基于协同过滤的推荐算法为主。基于内容的推荐算法（Content-Based,CBF）是最早提出并在推荐系统中实现的算法。其实现思路很简单，首先分析用户过去感兴趣的项目，向用户推荐与过去感兴趣项目的相似项目。CBF不存在项目冷启动的问题，但是随着数据量的激增，对用户发掘潜在兴趣的能力不足以给用户推荐合适的项目。

随着Goldberg提出的协同过滤算法的概念的兴起。协同过滤的研究和应用受到广泛关注。协同过滤算法不需要关心用户或者项目的信息，只需要了解用户对于项目操作。1994年，GroupLens团队提出了基于协同过滤算法的开源框架GroupLens【18】。GroupLens框架是很多推荐系统的基础，为了验证GroupLens框架的的准确性，GroupLens开发了MovieLens的推荐系统，由此出现了MovieLens数据集。

尽管协同过滤算法可以解决基于内容的推荐算法中存在的不足，但是随着用户数和项目数的增加，数据的稀疏性问题成为眼下亟需解决的问题。例如，国内最大的电子商务平台淘宝，截止2010年，淘宝网的用户达到3.7亿，商品数量8亿，最多每天有六千万人访问淘宝。若利用传统的协同过滤算法，构建用户-项目矩阵，那么这个矩阵大小将有6000万\*8亿。如此大数量的矩阵，计算效率非常低。而且每个用户评价访问的商品数量不超过800个，那么在用户-项目矩阵中，稀疏度将小于百万分之一。通过这个矩阵算出的结果，结果将近似于0。而本次实验将使用的MovieLens的稀疏度也将达到4.5%。所以，目前的国内外对推荐系统的研究主要集中在如何解决数据稀疏性上。目前，主要有两种技术手段来解决数据稀疏性的问题。其一是将高维的评分矩阵映射到低维的评分矩阵中，但是这样的做法使得用户-评分矩阵的信息无法全部利用到，降低了推荐的精度，如Sarwar等人通过奇异值的分解降低用户-评分矩阵的维度【19】。再如Koren等人，将隐式评分信息加入到传统的因子分解机模型中，创造性的提出了SVD++算法【20】。另一种解决稀疏性的办法则是结合其他的信息，让推荐系统计算近似用户集时更加准确。吴一帆等人提出了结合用户背景信息的协同过滤算法，其思想是在进行传统的协同过滤算法前利用用户背景信息构建一个不再稀疏的用户-评分矩阵，提高了推荐的精确到且不会对性能造成影响【21】。

1.2.2 用户画像

用户画像最早诞生于20世纪90年代后期。用户画像在推荐系统中的应用主要是根据用户的社会属性、生活习惯、操作行为等不同维度信息的分析，抽象出用户画像，并加入用户标签模型，对不用的用户打上不同的标签，最后根据标签将项目推荐给适合的用户群。胡兆山利用用户画像生产用户信息度量模型并将用户画像与协同过滤融合，提高了协同过滤算法的准确性，缓解了数据稀疏性的问题【22】。张颖提出了基于用户画像特征数据集的个性化推荐算法，利用现有数据作为细粒度的用户画像，并提出用户标签作为粗粒度的用户画像，在得到整体的用户画像特征数据集，该数据集优于其它单一特征的数据集【23】。

1.3 论文组织结构

第一章，绪论。在绪论里说明了本文的研究背景与意义，并对国内外推荐算法、用户画像做出了简要归纳。

第二章，相关技术与理论综述。第二章主要介绍了用户画像的相关原理以及简要介绍了推荐系统的分类和基本原理。

第三章，用户画像与协同过滤。在第三章中，提出了将用户画像和基于用户的协同过滤融合的方法。通过计算用户画像的相似度和用户相似度的权重值来得到最终的相似度。

第四章，系统设计与实现。第四章主要介绍了系统的实现方式以及主要功能的截图说明等。

第五章，总结与展望。

2相关技术与理论综述

2.1 用户画像

用户画像（Users’Profile）这个概念是最早由交互设计之父Alan Cooper所提出的，对同一类型的用户进行不同维度的刻画，从不同维度展现用户的信息全貌。他提出“用户画像是目标用户的一个具体代表”，这句话表明了用户原型是真实用户的虚拟代表，是建立在一系列真实数据之上的目标用户模型[5]，通过研究用户画像，分析用户的行为模式，兴趣爱好，挖掘用户的个性化和潜在需求，有助于开展针对用户的个性化知识资源服务，增强用户的黏度和信誉度。

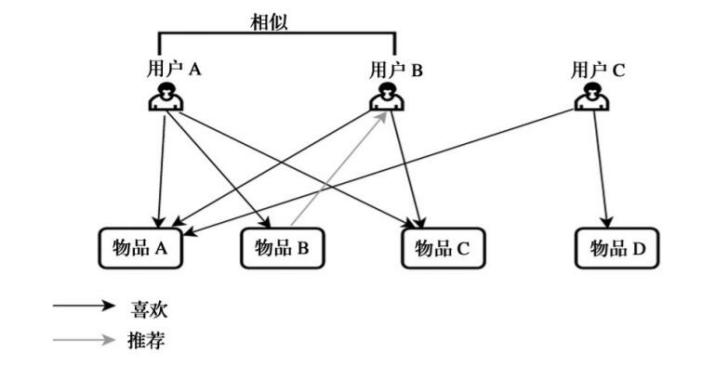
用户画像是任何一个推荐系统中必不可少的一部分[6]，推荐系统可以根据用户画像得出较为准确的用户行为偏好，从而更好的为用户提供推送服务。不同的业务系统同一个用户具有的用户画像是不同的，比如微信中的用户画像偏向于社交，而同一个人在支付宝中则更加偏向于支付，用户画像是由数据依据模型构建而成的。所以我们需要结合具体的业务来制定相关的标签，通过对标签数据的分析来达到预测用户偏好的目的[7]。

李丹、高建忠在基于用户画像的图书馆推荐服务初探一文中[15]展示了如何构建用户画像，通过读者入馆行为数据为读者提供推荐服务。在这篇文章中，讨论了基于用户画像的图书推荐过程：首先通过显式或隐式方式相结合读取读者信息，构建出读者的用户画像；再次将具有相同特征的用户划归为一类，分析用户借阅等行为信息，给出个性化推荐，采用协同过滤算法进行Top\_N推荐，向读者推荐兴趣度排名前N的内容项。

2.2 推荐系统

推荐系统能够有效的解决信息超载的问题，能够在海量的数据上，结合用户之前的行为模式等信息用户信息，为用户推荐其感兴趣的内容。现阶段的推荐系统算法主要包括协同过滤推荐、基于内容的推荐、基于二部图的推荐、基于关联规则的推荐和基于社交网络的推荐[8]，以下将逐一介绍各个推荐算法。

（1）协同过滤算法是由Goldberg等[9]所提出，最初的协同过滤算法用于过滤邮件，经过多年发展，协同过滤算法成为了推荐系统中最重要的算法之一。协同过滤算法利用大量用户与产品之间的关联历史数据来计算用户与产品之间的相似度，查找与用户相似度较高的近邻集，并通过近邻集用户对其他产品的评分来预测目标用户对该产品的预测评分，从而产生推荐的产品集合，再根据Top\_N来产生推荐项。如图2-1所示。可通过计算得知，用户A与用户B相似，用户A的兴趣偏好为物品A、B、C，用户B的兴趣偏好为物品A、C，所以可将物品B推荐给用户B。



**图2-1 协同过滤算法简化过程**

（2）基于内容的推荐算法考虑了协同过滤算法中忽略的用户信息（如年龄、性别等）和产品信息。基于内容的推荐算法通过提取用户-产品特征，学习用户兴趣模型，考量用户信息与待选产品间的匹配度，再将匹配度最高的产品推荐给用户[10]。基于内容的推荐算法第一步是建立起推荐对象模型，然后建立用户兴趣模型，再计算推荐对象模型和用户兴趣模型的相似度，最后按照匹配程度的高低进行相关产品的推荐。

（3）基于二部图的推荐算法有Aggarwal由1992年提出[11]，基于二部图的推荐算法只关注用户是否选择了某项，而不关心用户与产品是何种形式。在二部图算法中，用户与产品都被看作为二部图的节点，若用户选择了某产品，用户与某产品之间则存在一条边。所以通过研究节点之间的相关性，可以得出用户感兴趣的产品。

（4）基于关联规则的推荐算法的核心是关注用户的行为数据，从大量的数据中获取潜在有用的关联规则，从而产生推荐产品。关联规则的推荐算法广泛应用于电商领域，它通过用户的操作日志分析用户的每次购买之间的联系，形成用户的购买模式，当用户在进行购买活动时，可进行相关的推荐，达到交叉销售的目的。现阶段，基于关联的推荐算法主要使用Apriori算法以及FP-Grouwth算法，这类算法具有不可避免的缺点：需要对数据集进行频繁的扫描，计算量大，耗时长。同样也具有冷启动和数据稀疏的问题。

（5）基于社交网络的推荐算法利用社交网络数据得到用户兴趣偏好及好友信息，再根据获取到的数据为用户进行产品推荐、好友推荐及会话推荐。

对推荐系统的评价指标包括：平均绝对误差、均分根误差、标准平均误差、召回率、准确率。平均绝对误差用来衡量用户预测评级与实际评级的平均绝对误差；召回率表示推荐列表中用户实际感兴趣的产品占系统中用户感兴趣的所有产品的占比；准确率表示推荐列表中用户感兴趣的产品在所有被推荐的产品中的所占比例。

3用户画像与自适应协同过滤算法

3.1 问题的提出

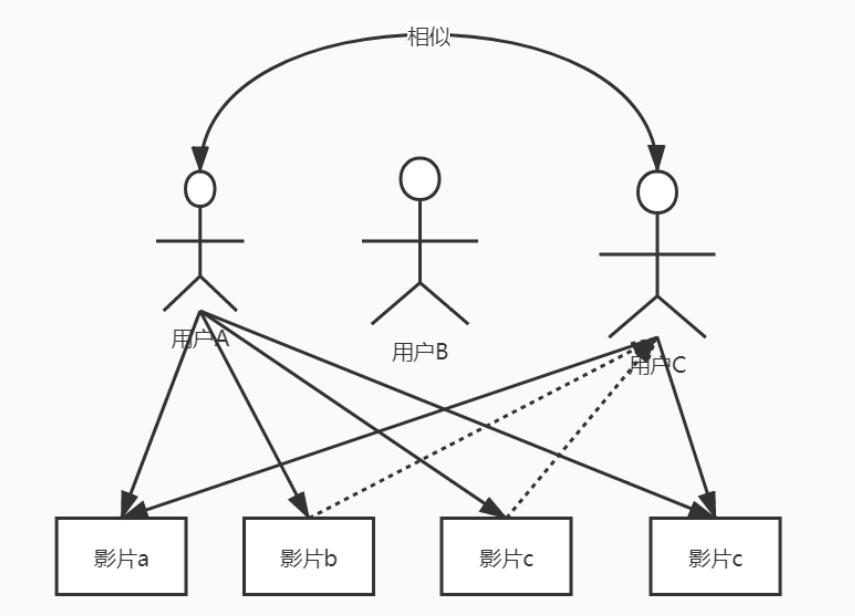
在1.2.1节中讨论过，推荐系统中的数据稀疏性问题制约了推荐系统的发展。而随着用户画像的出现，可以很好的弥补这一问题。据调查显示，约80%的用户愿意在平台中填写自己的个人姓名、性别、职业、教育情况等个人信息，另有约28%的用户不介意向电商平台或者其他平台共享自己的个人信息。【24】

所以，本文将使用最易获取到的用户信息来构建用户画像，并将其与基于用户的协同过滤算法融合，达到解决用户-评分矩阵稀疏的问题的目的，由此提高推荐系统的推荐准确性。

3.2 协同过滤算法

3.5.1 UserCF原理

UserCF的核心思想很简单：当用户A与用户C同时对影片a和影片b感兴趣并给出评分，则经过近邻集计算后，将用户A与用户C视为相似用户，则当需要为用户C推荐影片时，则可以把用户A感兴趣的且用户C并未评分或者了解的影片b、c推荐给用户C。



**图3-1 基于用户的协同过滤算法简图**

3.5.2 数据来源与数据处理

本文采用movielens-1m数据集并结合运行时产生的新数据作为项目数据来源。Movielens数据集是一个关于电影评分的数据集，包含了IMDB、The Movie DataBase上用户对电影的反馈信息。Movielens数据集经常被用来作为推荐系统，机器学习算法系统中的数据集，很多著名的论文也都基于Movielens数据集之上。所以本项目也将使用Movielens数据以验证基于用户画像的自适应推荐系统的可靠性。

Movielens数据集有众多版本，主要是数据集大小、数据集的新旧程度来区别。本项目将使用Movielens-1m，它包含了一百万条用户对于电影的评分记录。Movielens-1m可在grouplens的官网上进行下载使用，网站为：[https://grouplens.org/datasets/movielens/。Movielens-1m数据集中包含了](https://grouplens.org/datasets/movielens/。Movielens-1m数据集中包含了3)三个主要文件：movies.data、ratings.dat、users.dat，以下将逐一介绍这三个文件的内容以及作用：

movie.dat文件包含了3246条电影记录的信息。数据格式为：MovieID::Title::Genres。MovieID为每部电影的Id，Title为电影标题，Genres为电影类别，表3.1展示的是movie.dat文件内前三条的记录。

**表3.1 movie.dat文件内前三条数据**

|  |  |
| --- | --- |
| 序号 | 数据 |
| 1 | 1::Toy Story (1995)::Animation|Children's|Comedy |
| 2 | 2::Jumanji (1995)::Adventure|Children's|Fantasy |
| 3 | 3::Grumpier Old Men (1995)::Comedy|Romance |

rating.dat文件包含了1000209条用户对电影的评分记录。数据格式为：UserID::MovieID::Rating::Timestamp。UserID为评价用户Id，MovieID为评价电影Id，Rating为用户评分，按一颗星的规模递增（1stars-5stars），Timestamp为自1970年1月1日零点后到用户提交的时间的秒数，表3.2展示的是rating.dat文件内前三条的记录。

**表3.2 rating.dat文件内前三条数据**

|  |  |
| --- | --- |
| 序号 | 数据 |
| 1 | 1::1193::5::978300760 |
| 2 | 1::661::3::978302109 |
| 3 | 1::914::3::978301968 |

user.dat文件包含了6040条用户记录。数据格式为：UserID::Gender::Age::Occupation::Zip-code。UserID为用户Id，Gender为用户性别（M为男性，F为女性），Age为用户年龄，其中低于18岁，Age为1，Occupation为用户的职业，以数字0-20表示21个职业，详细见表3.4。Zip-code为用户所占邮政编码，表3.2展示的是user.dat文件内前三条的记录。

**表3.3 user.dat文件内前三条数据**

|  |  |
| --- | --- |
| 序号 | 数据 |
| 1 | 1::F::1::10::48067 |
| 2 | 2::M::56::16::70072 |
| 3 | 3::M::25::15::55117 |

**表3.4 occupation与职业对照表**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Occupation | 职业名称 | Occupation | 职业名称 | Occupation | 职业名称 |
| 0 | 其他/不确定 | 7 | 行政/管理 | 14 | 销售/营销 |
| 1 | 学者/教育家 | 8 | 农民 | 15 | 科学家 |
| 2 | 艺术家 | 9 | 家庭主妇 | 16 | 个体 |
| 3 | 文书/管理 | 10 | 低龄学生 | 17 | 技术员 |
| 4 | 大学/研究生 | 11 | 律师 | 18 | 商人/工匠 |
| 5 | 客户服务 | 12 | 程序员 | 19 | 失业 |
| 6 | 医生/保健 | 13 | 已退休 | 20 | 作家 |

获得数据后，首先将三个数据文件全部统一存入Mysql数据库的三张表中，数据字典如下表所示：

**表3.5 movie表数据字典**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 默认值 | 允许非空 | 备注 |
| id | int(11) |  | NO | 自增主键 |
| movie\_id | int(11) |  |  | 电影id |
| title | varchar(255) |  |  | 电影名 |
| genres | varchar(255) |  |  | 电影分类 |
| created\_time | timestamp | CURRENT\_TIMESTAMP | NO | 创建时间 |
| status | int(2) | 0 | NO | 0（正常）、1（删除） |

**表3.6 rating表数据字典**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 默认值 | 允许非空 | 备注 |
| Id | int(11) |  | NO | 自增主键 |
| user\_id | int(11) |  |  | 用户id |
| movie\_id | int(11) |  |  | 电影id |
| rating | int(2) |  |  | 评分 |
| created\_time | timestamp | CURRENT\_TIMESTAMP | NO | 创建时间 |
| status | int(2) | 0 | NO | 0（正常）、1（删除） |

**表3.7 users表数据字典**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 默认值 | 允许非空 | 备注 |
| id | int(11) |  | NO | 自增主键 |
| user\_id | int(11) |  |  | 用户id |
| gender | varchar(255) |  |  | 性别 |
| age | int(3) |  |  | 年龄 |
| Occupation | varchar(255) |  |  | 职业 |
| zip\_code | varchar(255) |  |  | 邮政编码 |
| created\_time | timestamp | CURRENT\_TIMESTAMP | NO | 创建时间 |
| status | int(2) | 0 | NO | 0（正常）、1（删除） |

**表3.8 rated\_users表数据字典**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 默认值 | 允许非空 | 备注 |
| id | int(11) |  | NO | 自增主键 |
| user\_name | varchar(255) |  |  | 用户名 |
| password | varchar(255) |  |  | 用户密码 |
| age | int(3) |  |  | 用户年龄 |
| genres | varchar(255) |  |  | 性别 |
| occupation | varchar(255) |  |  | 用户职业 |
| user\_type | int(2) | 0 | NO | 用户类型0（普通）、1（超级） |
| created\_time | timestamp | CURRENT\_TIMESTAMP | NO | 创建时间 |
| status | int(2) | 0 | NO | 0（正常）、1（删除） |

3.5.3 算法实现

第一步，在协同过滤算法中一般采用Jaccard相似系数公式或者余弦相似度公式来计算两个用户的相似度

其中，给定用户和用户,设表示用户感兴趣的集合，表示用户感兴趣的集合，表示用户和用户之间的相似度。

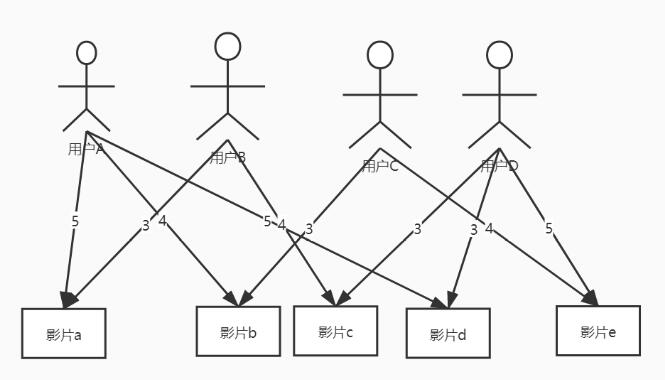
采用Jaccard相似系数公式或者余弦相似度公式的时间复杂度为，当用户数量很大时，运算效率很低，原因在于并不是所有的用户之间都有相同的感兴趣集合。也就是公式中分子为0，这样可以减少我们的运算。所以，我们建立倒查表。例，用户A感兴趣商品a、b、d，用户B感兴趣商品a、c，用户C感兴趣商品b、e，用户D感兴趣商品c、d、d。此时的倒查表为表3.9：

**表3.9 倒查表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 用户A | 用户B | 用户C | 用户D |
| 用户A | 0 | 1 | 1 | 1 |
| 用户B | 1 | 0 | 0 | 1 |
| 用户C | 1 | 0 | 0 | 1 |
| 用户D | 1 | 1 | 1 | 0 |

第二步，计算出用户之间的兴趣相似度后，通过公式来计算用户对物品的感兴趣程度：

其中，表示了和用户相似度最高的个用户，表示对物品产生过评级行为的用户集合，指用户和用户的兴趣相似度，表示用户对物品的兴趣程度，值为用户给物品的评分，表示用户对物品的感兴趣程度。

下面，将举例实际情况来体现基于用户的协同过滤算法的工作流程。

**图3-2 实际情况下的协同过滤算法**

图3-2所示中用户A对影片a、b、d的评分分别为5、4、5；用户B对影片a、c的评分分别为3、4；用户C对影片b、c的评分分别为3、4；用户D对影片c、d、e的评分分别为3、3、5。

通过公式计算出的用户之间相似度为表3.10：

**表3.10 用户A、B、C、D间的相似度**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 用户A | 用户B | 用户C | 用户D |
| 用户A | 1 |  |  |  |
| 用户B |  | 1 | 0 |  |
| 用户C |  | 0 | 1 |  |
| 用户D |  |  |  | 1 |

若此时，我们将对用户A进行物品推荐，选定K值为3所以，用户A此时没有对影片c、e有过行为记录，所以计算，,因此如果只对用户A推荐一件物品，算法将推荐物品

3.5.4 推荐算法结果度量

本文将全部数据集划分为训练集与测试集，其中训练集占全部数据的70%，测试集占全部数据集的30%。在处理过程中，按照随机的方式划分数据集。

（1） 均方根误差和平均绝对误差

均方根误差（RMSE）和平均绝对误差（MAE）是评估预测评级准确性的两个指标。MAE指标和RMSE指标分别使用公式以及公式进行计算：

其中，和分别表示了用户对产品i的实际评分和预测评分。表示推荐算法评级总数。

（2）召回率、准确率、覆盖率、新颖度

召回率（Recall）指推荐系统产生的推荐数量与总的需要推荐的数量的比。精准率（Precision）指推荐系统产生的推荐数量与所有产品的比。计算召回率和精确率的公式分别为公式和公式：

其中，指推荐系统对用户产生的推荐列表，指测试集上用户感兴趣的集合，指所有用户集合。

覆盖率（Coverage）也是衡量推荐系统优劣的重要指标，覆盖率反应了推荐算法发掘长尾的能力，覆盖率越高，则推荐算法越能将长尾中的物品推荐给用户[17]。覆盖率表示推荐列表占所有产品的比例。覆盖率计算公式：

其中，表示所有产品集合，指推荐系统对用户产生的推荐列表。

新颖度（Popularity）指推荐产品的热门程度，若推荐产品热门，则说明推荐产品的新颖的较低，反之，推荐产品的新颖度较高。在计算平均流行度时，将会对每个产品的流行度取对数，因为产品的流行度是满足长尾分布曲线的，将流行度取对数后，流行度的平均值更加稳定。新颖度的计算公式如下：

其中，代表推荐系统给用户的推荐列表，代表产品的流行度，指产品在所有用户推荐列表中出现的频次。

3.3 构建用户画像

在构建用户画像时，由于性别只有两个维度，所以对于性别的这个属性的构建很简单，但是，用户属性除了性别外，还有职业，年龄等，这些属性的维度太多，无法合理的得出用户画像相似度。因此，我们将年龄和职业按照社会属性进行划分为固定几个维度上，方便进行计算用户画像相似度。所以本文将Movielens数据集中的用户属性进行分类，具体信息见表。

**表3.1 MovieLens数据集划分对照表**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Gender | | Age | | Occupation | |
| 标签 | 属性 | 标签 | 属性 | 标签 | 属性 |
| M | 男性 | Teen | 青年（24岁以下） | Tec | 6、11、12、17 |
| F | 女性 | Ad | 成年（25-44岁） | Rec | 1、2、4、15、20 |
|  |  | Mat | 壮年（45-55岁） | Ope | 3、7、16 |
|  |  | Old | 老年（56岁以上） | Obj | 5、8、14、18 |
|  |  |  |  | Else | 0、9、10、13、19 |

在将数据集分类后，就可以利用公式 计算不同用户之间的用户画像相似度：

其中。表示用户的第个属性特征。通过计算可以得到用户和用户之间的用户画像的相似度。

3.4 基于用户画像的协同过滤

在上文中，我们利用了公式 计算得到了用户与用户之间的余弦相似度，又利用公式 计算得到了不同用户之间的用户画像相似度，在本节中，将介绍一种方法将两者相似度进行组合，得到组合相似度，在利用公式进行计算用户对物品的感兴趣程度。

由于用户相似度与用户画像相似度都能表示用户之间的相似程度，但是二者的侧重点却不一样。用户相似度主要从用户的历史操作记录来分析计算用户之间的相似程度，而用户画像相似度是从用户的社会属性特征来分析计算用户之间的相似程度。所以，如何正确的在二者之间选择合适的值作为相似度成为主要问题。一旦，用户画像相似度在相似度的比重过高，则推荐系统将大大忽略用户对项目的历史评分结果，导致推荐的准确率降低，反正，比重过低，则用户画像在推荐系统中起到的作用将大打折扣。因此，本文将利用公式来确定用户画像相似度在相似度中的权重。

其中，表示用户产生过评分的所有项目，为用户画像在组合相似度中的权重，表示为用户和用户共有项目集合与用户和用户的所有项目集合之比。

其中，表示用户的余弦相似度在组合相似度的比重。

通过计算出的两种相似度在组合相似度中的权重，我们可以通过公式计算出最终得到的组合相似度，再利用公式计算用户对产品的感兴趣程度。

其中，表示用户与用户的组合相似度，表示用户与用户之间的用户画像相似度，表示用户与用户之间的余弦相似度。

3.5 本章小结

在本章中，首先说明了将用户画像与推荐系统相结合是为了解决用户-项目矩阵稀疏的问题，接着介绍了基于用户的协同过滤的基本原理与实现方式以及在项目中数据来源的说明。然后又介绍了如何构建一个用户画像。最后交代了将用户画像与基于用户的协同过滤融合的方式。

本文将在下一章节中，介绍系统的实现方式与系统展示。

4系统设计与实现

3.1 技术概要

该课题主要采用前后端分离的技术，前端负责用户操作日志采集，推荐可视化等功能，后端主要提供算法支持，Api提供等。

所用的编程语言主要有Golang、Python、JavaScript、Sql、Html等；主要采用Mysql和Redis作为系统数据库；采用的前端框架为VueJs。

其他的后端技术有Kafka、gRPC、Docker等。这些技术应用细节将在下文中详细介绍。

3.2 编程语言

本次项目的编程语言选择了Golang和Python。Golang的轻便以及高并发使得它在过去十年里飞速发展，Golang由Google公司开发完成，开发理念为兼顾开发效率、编译效率以及运行效率，被誉为云计算时代的C语言，结构简单，易学易用。Golang的高性能HTTP Server提供了对Http协议良好的支持，可以使用几行代码开启一个Http服务，十分适合于本项目的后端服务器实现[1]。而使用Python的原因在于Python在科学计算方面有着其他编程语言无法达到的效率和优势，且Python是人工智能领域的首选语言，有着很多性能优越的第三方库供我们选择使用，在本项目中，核心算法部分将由Python完成，再使用RPC的方法与Golang服务器进行通信。所以，在本次项目中，我将使用Golang和Python完成这次的推荐系统项目。

3.3 数据库

本次项目中，将使用Mysql数据库作为主要的数据存储数据库，在伍志聪发表的《Mysql数据库在中小型业务系统的应用》[3]中，Mysql的数据类型、与系统的兼容性以及安全性都相较于其他数据库有较大的优势。Mysql由一个mysqld服务器守护进程和客户程序及库组成，提供了适合于众多编程语言的Api接口，且多平台支持，支持多线程、多用户。

将Redis作为用户Token数据库以及数据缓存。Redis是目前应用十分广泛的键值对数据库，逐渐取代了Memcache成为现在系统中首选的缓存服务器，它支持事务、I/O持久化和订阅发布功能，都对系统数据实时存储提供强有力帮助[6]。在本次课题中，主要是依靠Redis具有的告诉读写，纯内存操作等性能，使用Redis存储用户登录时产生的Token，保证用户的持久化登录，提高用户操作友好度。

3.3.1 数据库设计

Mysql

3.4 前端框架

本次项目将使用Vue.js作为前端框架。Vue.js是一个轻巧、高性能、可组件化的前端框架，同时拥有者非常容易上手的API，Vue.js不同于其他重量级的框架，它只关注视图层，采用自底向上增量开发的设计，并且学习成本低，非常容易学习，可以很轻松的将其它库与已有项目整合。Vue.js的灵活与渐进式的特点都适合本次项目，因为不管我们最后的前端项目做到何种程度，在代码方面都易于管理。Vue.js提供NPM来安装依赖。Vue.js有着活跃的社区，促进框架的发展，因此学习Vue.js的途径也变得多种多样。

Vue.js采用了MVVC的设计模式。MVC模式是传统软件工程中的一种设计模式，它将项目进行了逻辑层面的划分，其中MVC分别是Model、View和Controller首字母的缩写。View指软件的视图界面，Model指视图中所需要的数据，Controller控制View和Model之间的数据传输。随着互联网的发展，研究人员在MVC的基础上衍生除了MVVM的开发模式。MVVM指Model View ViewModel。重点在于用事件驱动的UI平台开发。MVVM的核心是通过数据绑定链接View和Model，使得数据的变化自动映射为视图的更新。Vue.js对页面中的每个Dom操作都进行了封装工作，让数据发生变化的Dom自动动态做出改变，简化了系统开发的投入，降低了开发难度和日后维护成本[16]。

3.5 核心算法

本项目的核心算法部分将由Python完成。本项目将大量使用Python的一个扩展程序库Numpy。Numpy支持大量的维度数组与矩阵运算，Numpy底层使用C语言编写，数组中直接存储对象而不是对象指针，所有其运行的效率远远高于Python代码。不仅如此，Numpy内置了并行计算功能，当系统有多个核心是，Numpy会自动的进行并行计算，大大提高了运行效率。Numpy的特点，运算效率高以及编码难度低都适合在推荐系统核心算法中使用。

在本项目中，采用了基于用户的协同过滤算法(User Based Collaborative Filtering, UserCF)作为推荐算法

3.6 后端服务器

后端服务器将由Golang实现。Golang原生支持并发编程，可以十分简单的开启并发执行单元，开发人员无需了解执行细节，调度器会自动将这些任务分配给合适的线程执行。Golang在创建时就具有了Python的编码高效、C语言的编译迅速以及C/C++的运行高效。它是编译型的静态语言，相比于解释型语言具有天生的性能优势。

在本项目中还将使用消息队列Kafka来将用户（Producer）与主题（Topic）进行绑定（Subscribe），使得用户数据可以自适应的被推荐系统所接受。同时，Kafka还具有消息的持久化功能，保障了数据传输的稳定性。本课题使用Kafka代替传统的消息代理，Kafka有着更高的数据吞吐量、高性能以及容错机制，都让Kafka成为了大规模消息处理应用软件系统的理想解决方案。之所以Kafka具有高性能的数据传输表现，主要是因为Kafka在发送少量数据时，会将数据暂时缓存在内存中，当超过一定的大小或一定时间后再进行批量发送，同时Kafka还会将数据进行压缩以减小网络负载。在本课题中，使用Kafka传输用户的操作日志给算法程序形成用户画像，分析用户兴趣，再通过Kafka返回推荐项目，取代了传统的http方式，使得数据不易丢失，提高性能[13]。

本课题中，后端系统将会放入Docker容器中进行发布。Docker区别于传统的虚拟机，传统的虚拟机运行在寄主操作系统之上，然后通过对CPU、内存、IO存储设备等进行虚拟化来达到资源隔离和调度的目的。而Docker使用Namespace来进行系统环境的隔离，加上CGroups实现资源的限制，使用了和寄主机相同的内核。Docker轻量化、启动速度快、运行效率高的特点都让Docker成为取代传统虚拟机的条件。使用Docker可以快速的搭建自己想要的环境，快速构建出适合本课题的环境，并且可以打包成镜像文件，方便在寄主机间进行转移 [14]。

3.6.1 单点登录设计

单点登录（Single Sign-On，SSO）现如今被广泛应用于web应用系统中。主要为了解决http连接的无状态带来的麻烦，减少登录次数。本次项目中将使用Token作为持久化登录的令牌。基于Token的用户验证一般常用语前后端分离的项目中，解决了跨域的问题。用户在进行登录后，服务端依据HS256等加密方式生产Token发给客户端，客户端可将Token放入Cookie或者localStorage中，在每次请求时在http的header中都带上Token，服务端收到后解析即可验证身份。

JWT（JSON Web Token）是为了在网络应用环境间传递声明而执行的一种基于JSON的开放标准。JWT生成的Token适合于分布式站点的SSO场景中。JWT由头部、载荷、签证三部分构成，中间以“.”连接成一个完整的字符串。头部由两部分组成：Token的类型（JWT）和加密算法名称（例如HS256）再用Base64编码得到JWT的头部。载荷部分用来存放实际需要传递的数据，JWT规定了7个官方字段使用：签发人、过期时间、主题、受众、生效时间、签发时间、编号。载荷也要经过Base64编码。最后一个部分是签名部分。是对前两个部分的签名，以防止数据被篡改。需要服务端指定一个密钥，使用HMACSHA256对头部和载荷进行签名。最后再把三个部分用“.”连接。

Golang中使用JWT生产Token的代码：

tokenkey := fmt.Sprintf("%s%d", username, time.Now().Unix())

token := jwt.New(jwt.SigningMethodHS256)

claims := make(jwt.MapClaims)

claims["exp"]=time.Now().Add(time.Minute\*time.Duration(120)).Unix() //过期时间为2小时

claims["iat"] = time.Now().Unix() //签发时间

claims["jti"] = tokenkey

token.Claims = claims

tokenString, err := token.SignedString([]byte("hello")) //指定密钥为hello

3.6.2 MVC设计

MVC作为架构模式的一种，非常适用于web系统的开发。MVC的全称是Model View Controller，是模型、视图、控制器的缩写。它将业务逻辑、数据、前端界面分离。MVC的设计模式使得更改视图层代码不用重新编译模型和控制层代码。因为模型与控制器和视图的分离，让更改应用程序的数据部分和业务逻辑变得更加容易。使整体架构变得高内聚、低耦合。不仅如此，MVC模式允许不同的视图层代码访问同一个服务器的数据层和控制层代码，所以，只需要写一个服务器代码，就可以让不同的客户端使用。

在本项目中，将MVC的架构设计思路拓展开来，分离出Handle层（提供Api）、Dao层（与数据库交互）、Model层（逻辑处理层）、Rpc层（远程过程调用）等不同作用的模块层。保障了推荐系统的自适应响应，利于扩展功能。

3.6.3 restful设计

表述性转态转移（Representational State Transfer，REST）最早由Roy Fielding在2000年提出，REST指一组架构约束条件和原则。开发人员遵循REST架构设计开发出一套易于扩展、简洁、表达明确的HTTP服务。

在本项目中，所有接口分为user和movie两种，表示两种资源，在对资源的动作的描述，都从URL上很明显可以看出，又可以从接口的请求方式上看出是向服务器传输数据还是请求数据。这样的API设计，可以让前后端开发人员很清楚的知道接口的作用，易于管理维护。

**表4.1 系统接口说明表**

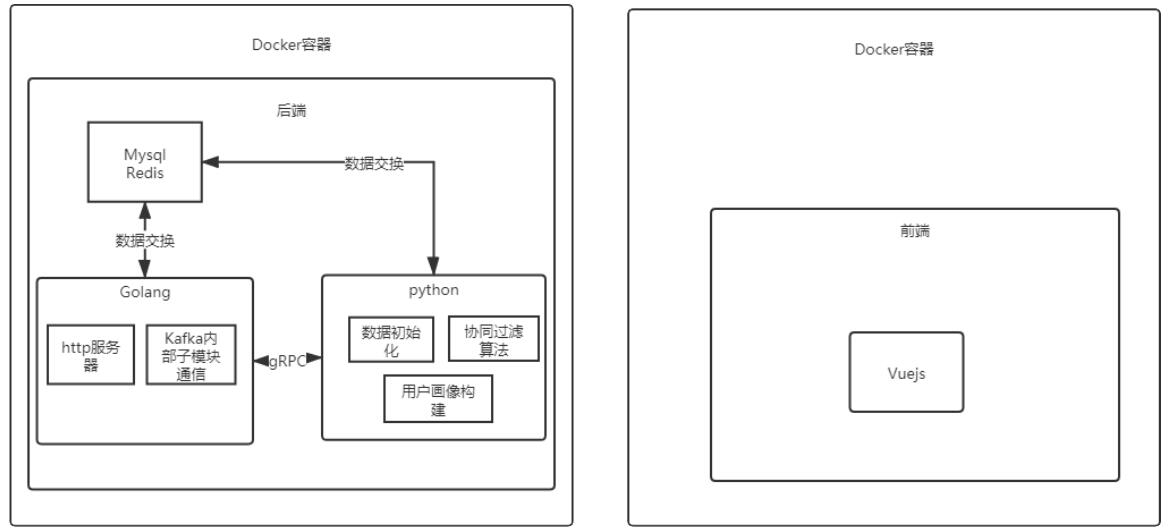
|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 接口URL | 请求方式 | 接口说明 |
| 1 | user/login | POST | 登录 |
| 2 | User/logou | GET | 登出 |
| 3 | User/addUser | POST | 添加用户 |
| 4 | User/get\_info | GET | 利用Token获取用户信息 |
| 5 | Movie/pageMovie | GET | 获得分页电影信息 |
| 6 | Movie/movieRate | GET | 电影评分 |
| 7 | Movie/Recommend | GET | 获得推荐电影 |

3.6.4 RPC层

远程过程调用（Remote Procedure Call，RPC），这是一种通过网络从远程计算机程序上请求服务，而不需要了解底层网络技术的思想。一个完整的RPC框架包含了服务发现、负载、容错、网络传输、序列化等组件。RPC的核心功能就是将数据请求序列化后，通过协议编码，再通过底层的套接字或者其他协议传输给请求端，请求端收到请求后，再反向操作后得到数据，RPC的核心功能主要有五个部分组成：客户端、客户端Stub、网络传输模块、服务端、服务端Stub。不仅如此RPC既具有跨语言通信的能力，其性能也比普通的HTTP服务快，所以在跨语言通信或微服务系统中，多选择使用RPC进行服务间的通信。

因此，在本项目中，作为算法核心的Python代码部分想要与Golang提供的HTTP服务间进行通信，需要RPC的使用。开启RPC服务后，在HTTP服务器上只需要想调用本地方法一样，就可以调用Python提供的算法代码。

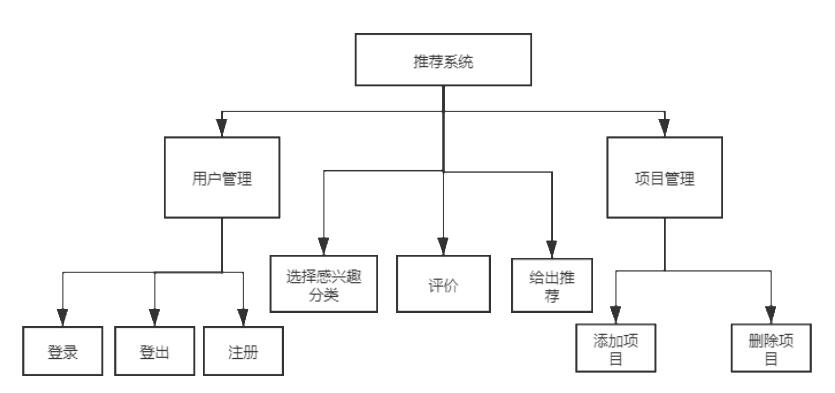
3.7 架构设计



**图3-1 架构图**

本项目的系统架构图由图3-1所示。前后端分别放入Docker容器中进行管理编排。Golang代码负责提供http服务并实现接口部分，内部以微服务进行划分子模块，其子模块间采用Kakfa进行通信。Python代码部分负责初始化数据集，实现协同过滤算法与用户画像的结合。Python代码与Golang代码均可直接与数据库进行数据交换。前端主要以Vuejs为主，负责请求发送数据以及接受响应并动态渲染数据。

3.7 系统设计



**图3-2 系统设计图**

本项目的系统设计图由图3-2所示。系统主要分为用户管理、项目管理两部分。用户可登陆、登出以及注册新用户。此外用户还可以添加新的项目，并给予评价。用户在首次进入系统时要求选择感兴趣的分类，以便解决推荐系统中存在的冷启动问题。由此随着用户给出的评价越多，推荐的准确率也将提高。只有管理员用户才可以删除项目。

3.7 系统展示

3.7.1 登录界面



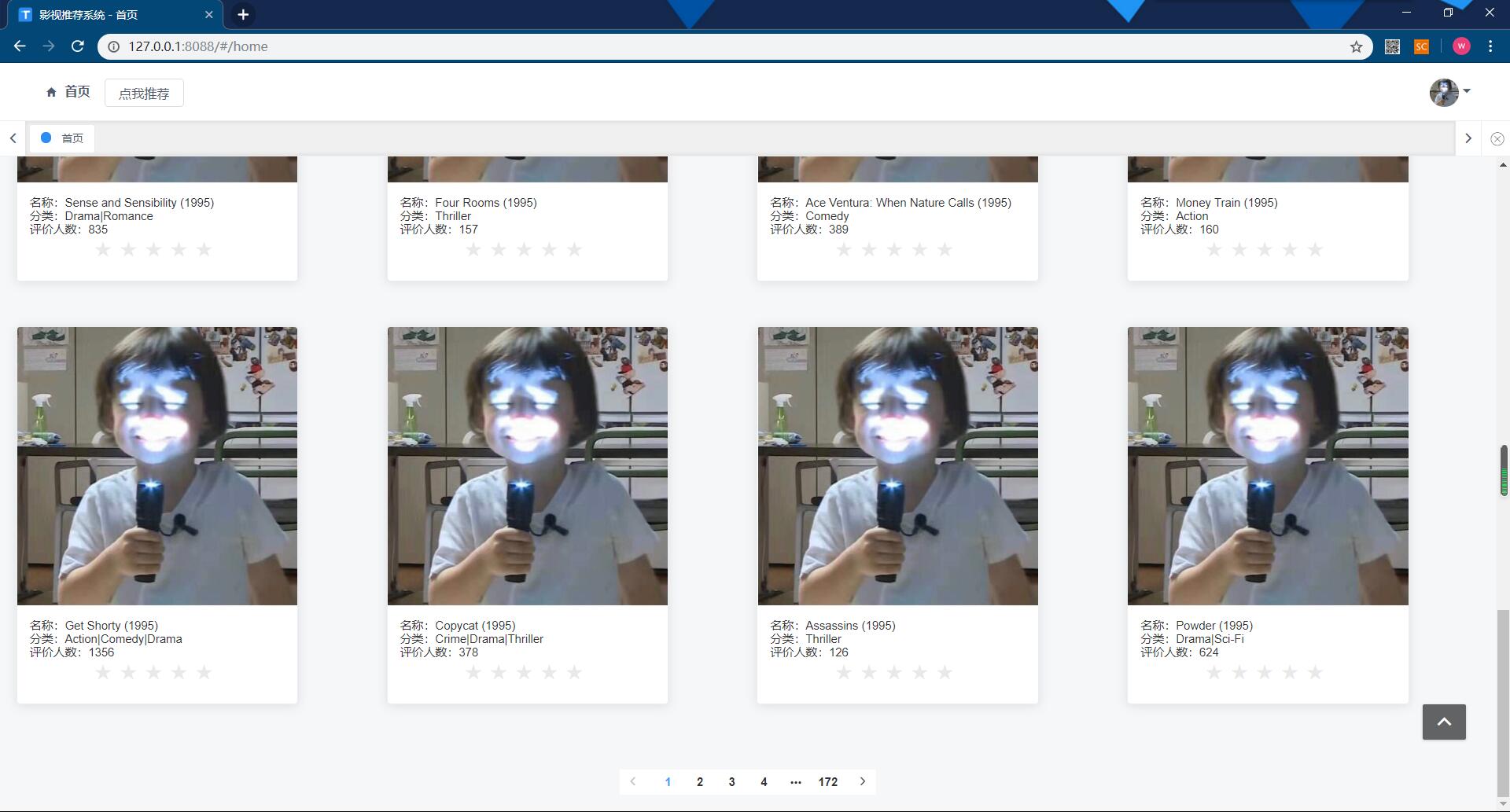
系统运行后，首页为登录界面，将会对用户的账号密码，用户类型进行判断。若用户登录成功过且Token时间未到期，则将会直接进入用户主页面。并且通过此界面可进入到注册界面。

3.7.2 注册界面



系统可对未注册用户进行注册。输入注册的用户名、密码、年龄，并选择性别和职业类别即可完成用户注册。

3.7.3 用户主界面



在系统用户主界面中，主要展示的是所有数据集的内容以及自定义的数据内容，以分页的形式给出，在跳转页码时，系统将会重新请求请求页的少量数据，避免了一次请求大量数据带来的网络负载以及数据库压力。每项数据都以卡片的形式给出，卡片中给出了每部电影的名称、分类、评价人数以及平均分。并且用户可以对项目进行打分评价，

3.7.4 推荐页面

在推荐页面中，展示的是后台为当前用户计算出来的推荐项，并将当前推荐的召回率、准确率、覆盖率以及新颖度结果展示给用户。

5总结与展望

5.1 总结

5.2 展望

5.2.1 单一的评级方式

在系统中，用户对项目的评级方式就是单一的评价打分，没有对用户的其他操作进行评价，如点击、访问、查看详情等其他操作。

5.2.2 系统实时处理与离线处理相结合

在。

5.2.3 单一的评级方式

在

、

参考文献

[1] 曹廷,付宇.Golang编程语言特性分析[J].电脑迷,2018,11:55-56.

[2] 石雨晴. 基于中图—专业分类映射的高校图书馆图书推荐方法的研究与实现[D].辽宁大学,2017.

[3] 伍志聪.MySQL数据库在中小型业务系统的应用[J].数字技术与应用,2011(11):122.

[4] 刘俊龙,刘光明,张黛,喻杰.基于Redis的海量互联网小文件实时存储与索引策略研究[J].计算机研究与发展,2015,52(S2):148-154.

[5] COOPER A. The Inmates are Running the Asylum [M][. S.l.]:Macmillam Pulihing co.lnc,1999,53:17.

[6] Leung K ,Lee D.Deriving Concept- Based User Profiles Fro Search Engine Logs[J]. IEEE Trans. Knowledge And Data Engg., 2010, 22(7).

[7] 臧勇琪.基于用户画像的精准营销研究[J].全国流通经济,2019(25):14-16.

[8] 崔春生,吴祁宗,王莹.用于推荐系统聚类分析的用户兴趣度研究[J].计算机工程与应用,2011,47(7):226-228.

[9] GOLDBERG D,NICOLS D. Using collaborative filtering to weave an information tapestry[J].Communications of the ACM,1992,35(12):61-70

[10] 常璐.高校图书馆 E-learning支持服务研究[D].上海:东华大学,2013.

[11] 蔡红蕾.二部图网络结构算法在推荐系统中的应用[D].秦皇岛:燕山大学,2014.

[12] 李兰馨.大数据智能算法范式下的用户黏性研究——以网易云音乐为例[J].新媒体研究,2019,5(04):4-6.

[13]刘邦,余华平.Kafka分布式消息队列的高性能研究[J].电脑知识与技术,2019,15(32):4-6.

[14] 赵冠臣,王冬妮,刘至洋,孟振江.浅谈Docker容器技术[J].有线电视技术,2019(09):85-88.

[15] 李丹,高建忠.基于用户画像的图书馆推荐服务初探[J].图书馆,2019(07):66-71.

[16] 徐頔,朱广华,贾瑶.基于VueJs的WEB前端开发研究[J].科技风,2017(14):69.

【17】刘向举,袁煦聪,刘鹏程.基于长尾理论的物品协同过滤Top-N推荐算法[J].齐齐哈尔大学学报(自然科学版),2019,35(02):1-4+9.

【18】FM Harper, JA Konstan. The Movielens Datasets: History and Context[C]. In: ACM Transactions on Interactive Intelligent Systems, 2015.

【19】X Luo, MC Zhou, S Li, Z You, Y Xia. A non-negative latent factor model for large-scale spares matrices in recommender systems via alternating direction method[C]. IEEE Trans. Neural Netw, vol 27, No.3, 2016:579-592.

【20】Koren Y. Factorization meets the neighborhood：a multifaceted collaborative filtering model[C]. Proceedings of International Conference on Knowledge Discovery and Data Mining，2008：426-434.

【21】吴一帆，王浩然．结合用户背景信息的协同过滤推荐算法[J]．计算机应用，2008，28 (11) : 2973－2974．

【22】胡兆山. 基于用户画像与协同过滤的混合推荐系统研究[D].云南财经大学,2019.

【23】张颖. 基于用户画像特征数据集的个性化推荐算法研究[D].西安电子科技大学,2018.

【24】曾春，邢春晓，周立柱. 个性化服务技术综述[J]. 软件学报，2002, 13( 10) : 1952－1961．

致谢

唐稳稳

2020年XX月XX日