

硕士专业学位论文

基于混合推荐模式的汽车类文章推荐系统的设计与实现

Design and Implementation of Automotive Article Recommendation System based on Mixed Recommendation Mode

作者：申玉聪

导师：孔令波

北京交通大学

2018年3月

学位论文版权使用授权书

本学位论文作者完全了解北京交通大学有关保留、使用学位论文的规定。特授权北京交通大学可以将学位论文的全部或部分内容编入有关数据库进行检索，提供阅览服务，并采用影印、缩印或扫描等复制手段保存、汇编以供查阅和借阅。同意学校向国家有关部门或机构送交论文的复印件和磁盘。学校可以为存在馆际合作关系的兄弟高校用户提供文献传递服务和交换服务。

（保密的学位论文在解密后适用本授权说明）

学位论文作者签名： 导师签名：

签字日期： 年 月 日 签字日期： 年 月

学校代码：10004 密级：

北京交通大学

**专业硕士学位论文**

基于混合推荐模式的汽车类文章推荐系统的设计与实现

Design and Implementation of Automotive Article Recommendation System based on Mixed Recommendation Mode

作者姓名：申玉聪 学 号：16126192

导师姓名：孔令波 职 称：讲师

工程领域：软件工程 学位级别：硕士

北京交通大学

2018年3月

摘要

随着时代的不断进步和科学技术的不断发展，人们的生活水平逐年提高，生活环境也得到了极大的改善，汽车已经成为了人们出行最主要的代步工具，人们的购车需求与日俱增，那么对于即将买车的人来说，找到一个能为用户提供专业而丰富的汽车资讯服务的平台显得尤为重要，而移动互联网和智能手机的普及给手机应用的蓬勃发展带来了生机，越来越多的用户倾向于通过手机App来获取更多的汽车资讯信息。基于以上一个背景，我们可以发现，当一个还在犹豫买什么车、或者不知道这款车怎么样的用户，面对日益增长的海量汽车资讯信息，传统的搜索引擎服务已经难以满足用户的需求了，如何智能地向用户推荐一些个性化的汽车资讯信息，达到较高的用户满意度，并提升APP的点击率以及转化率已经成为开发汽车资讯平台面临的主要问题。

推荐系统是一种汇集集体的信息，过滤出个性化信息的一种软件工具，旨在挖掘用户兴趣，在信息严重过载的时代下为用户提供个性化服务。推荐系统和搜索引擎对于用户来说是互补的两个工具，在汽车资讯应用中，推荐系统很大程度上弥补了检索功能上的不足，主动为用户提供一套用户感兴趣的汽车资讯文章。推荐系统主要是通过分析用户在App上的历史行为，然后为用户建立专属兴趣模型，进而为每个用户提供专属的个性化文章。

随着推荐系统的不断发展，涌现出了各种各样的推荐算法，但是每种推荐算法都有各自的优点与局限性，因此，为了弥补各种推荐算法的不足之处，本次项目使用混合推荐模式将多种推荐算法及策略结合起来共同为用户生成推荐候选集列表，然后对候选集列表的各个文章使用机器学习的方法进行点击率预估进而生成最符合用户兴趣的文章列表，最大程度地提升推荐效果，其中主要使用了基于内容的推荐算法、基于关联规则的推荐算法、基于协同过滤的推荐算法以及根据汽车类资讯文章的特点设计的基于车标签的的推荐策略。

本文将使用App中产生的用户行为日志和汽车资讯文章数据为数据基础，使用Thrift作为推荐系统服务端的框架，利用Hadoop相关技术对海量日志进行分布式存储并对推荐算法及策略进行分布式计算，实现了数据同步、个性化推荐、相关文章推荐、热点文章推荐等功能，并利用Django和Echarts实现了推荐效果数据的可视化，使得推荐的具体效果能够以报表的形式展示出来，方便开发者对推荐的流程及算法进行改进。

本文对整个项目的需求分析、设计、系统实现以及系统测试都进行了详细的阐述，其中重点说明了推荐算法及策略的设计、点击率预估模型的设计以及数据可视化在推荐效果报表中的具体应用。

在本次的项目中，本人对项目进行了需求分析、系统的概要设计、推荐算法及相关策略的设计、系统的详细设计、数据同步模块、混合推荐引擎模块、推荐点击率预估模块、基于数据可视化的推荐效果报表模块的具体实现与测试。

**关键词：**混合推荐模式；推荐系统；数据可视化；点击率预估

ABSTRACT

**KEYWORDS：**

目录

1. 引言

本章将首先介绍本次项目的背景与意义，然后分析本次研究课题在国内外的研究现状，最后阐述本人在本次项目中的主要工作以及论文的组织结构。

* 1. 课题背景与意义

随着互联网以及信息技术的迅猛发展以及人们的生活水平的提高，越来越多的人加入到购车的行列中，人们对与汽车资讯文章的需求也是与日剧增，对于一个致力于为用户提供汽车类资讯文章的App来说，能为用户从海量的汽车资讯信息中提取出用户感兴趣的信息，并提供定制的个性化推荐列表显得尤为重要。近年来各大研究机构和公司都为推荐系统的发展做出了巨大的贡献，各种推荐算法不断涌现出来，并不断完善，为了弥补不同的推荐算法的不足之处，将各个不同算法进行融合，然后使用机器学习的方法对推荐候选集列表进行重排序的策略也逐渐成为了各个公司的开发推荐系统的首选方案。

开发汽车类文章推荐系统的意义如下所述：（1）如果能为用户主动提供用户感兴趣的专属个性化汽车资讯内容，不仅能提升App自身的点击率与转化率，还能使这款App在同行业的竞争中脱颖而出。（2）对于大部分购车意向还不明确的用户来说，在推荐系统的推荐下能够引导用户逐渐明确自己的购车意向，能在一定程度上提升用户的满意度，增强用户对App的信任，从而增加用户留存率。（3）推荐系统能将一些长尾中的文章推荐给用户，提升非热门文章的点击率。

* 1. 国内外研究现状

就目前国内外学籍信息管理系统的发展状况来说，国外院校的重视程度较高，且由于其计算机水平起步早发展快，因此，相对起步较早，发展较快且利用率较高；国内从上世纪90年代传入国内，起步较晚，但是由于我国政府政策支持和倡导，相应的学籍信息管理信息系统发展较快，且在政府部门和企业都有了较快的发展和广泛的应用，但是针对具体不同类型的高校而言，缺乏学籍管理的针对性。

* + 1. 国内研究现状

就目前国内外学籍信息管理系统的发展状况来说，国外院校的重视程度较高，且由于其计算机水平起步早发展快，因此，相对起步较早，发展较快且利用率较高；国内从上世纪90年代传入国内，起步较晚，但是由于我国政府政策支持和倡导，相应的学籍信息管理信息系统发展较快，且在政府部门和企业都有了较快的发展和广泛的应用，但是针对具体不同类型的高校而言，缺乏学籍管理的针对性。

就目前国内外学籍信息管理系统的发展状况来说，国外院校的重视程度较高，且由于其计算机水平起步早发展快，因此，相对起步较早，发展较快且利用率较高；国内从上世纪90年代传入国内，起步较晚，但是由于我国政府政策支持和倡导，相应的学籍信息管理信息系统发展较快，且在政府部门和企业都有了较快的发展和广泛的应用，但是针对具体不同类型的高校而言，缺乏学籍管理的针对性。

* + 1. 国外研究现状

就目前国内外学籍信息管理系统的发展状况来说，国外院校的重视程度较高，且由于其计算机水平起步早发展快，因此，相对起步较早，发展较快且利用率较高；国内从上世纪90年代传入国内，起步较晚，但是由于我国政府政策支持和倡导，相应的学籍信息管理信息系统发展较快，且在政府部门和企业都有了较快的发展和广泛的应用，但是针对具体不同类型的高校而言，缺乏学籍管理的针对性。

* 1. 本人主要工作

本人的主要工作是基于混合推荐模式的汽车类文章推荐系统的需求分析、系统概要设计、推荐算法及策略设计、系统详细设计以及数据同步模块、混合推荐引擎模块、点击率预估模块、基于数据可视化的推荐效果报表模块的实现与测试。

* 1. 论文组织结构

本次项目是严格按照软件开发的基本理论和流程进行基于混合推荐模式的汽车类文章的推荐系统的开发工作的，为了方便读者阅读，特将论文的组织结构安排如下：

第一章，引言。向读者介绍了本次项目的研究背景与意义、国内外研究现状、以及本人在项目中负责的工作。

第二章，理论基础与关键基础介绍。本章向读者介绍了在开发过程中用到的主要的理论知识以及关键技术，主要包括推荐系统定义、基于内容的推荐算法、基于协同过滤的推荐算法、基于关联规则的推荐算法、混合推荐模式介绍、点击率预估介绍、Thrift服务端框架、Django Web 框架、Echarts数据可视化。

第三章，需求分析。主要包括系统的可行性分析、系统角色用例分析、系统的整体流程分析、系统功能性需求分析、系统非功能性需求分析。

第四章，系统概要设计。主要包括对系统的功能模块的划分，以及数据库的概要设计。

第五章，推荐算法设计。XxXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

第六章，系统详细设计与实现。本章主要介绍了**数据流同步模块、**混合推荐引擎模块、点击率预估模块、基于数据可视化的推荐效果报表模块。

第七章，系统测试。本章介绍了系统的功能性测试和性能测试的过程。

第八章，总结与展望。将本文的内容进行总结，并对推荐系统今后的改进方向做出展望。

1. 理论基础与关键技术介绍

**就目前国内外学籍信息管理系统的发展状况来说，国外院校的重视程度较高，且由于其计算机水平起步早发展快，因此，相对起步较早，发展较快且利用率较高；国内从上世纪90年代传入国内，起步较晚，但是由于我国政府政策支持和倡导，相应的学籍信息管理信息系统发展较快，且在政府部门和企业都有了较快的发展和广泛的应用，但是针对具体不同类型的高校而言，缺乏学籍管理的针对性。**

* 1. ***~~Thrift框架~~***

***~~Thrift是一个开源的服务框架，他最初来源于Facebook的其中一个项目，但是后来Facebook在2007年将Thrift贡献给Apache，在2008年Thrift正式成为是Apache的开源项目之一。~~***

***~~Thrift的出现是为了解决不同语言环境下的通信问题，使得多种语言之间可以通过RPC（Remote Procedure Call）进行通信，并且将代码生成引擎和软件栈联合起来创建跨平台的服务。我们可以通过一个简单的定义文件定义传输的数据结构和服务接口，然后将代码生成引擎生成的代码导入自己的工程中就可以编写跨语言环境通信的代码，比如可以使用python编写客户端，用java编写服务端。~~***

***~~Thrift架构概述~~***

***~~Thrift在整体架构上使用的C/S模式，具体的架构图如图2-1所示，从图中可以看到Thrift是拥有一个完整的堆栈结构用于构建他的服务端和客户端。~~***

***~~从堆栈的底层开始看，最底层的I/O模块主要负责客户端和服务端之间数据的传输，数据包括文件、压缩的数据流、Socket等。~~***

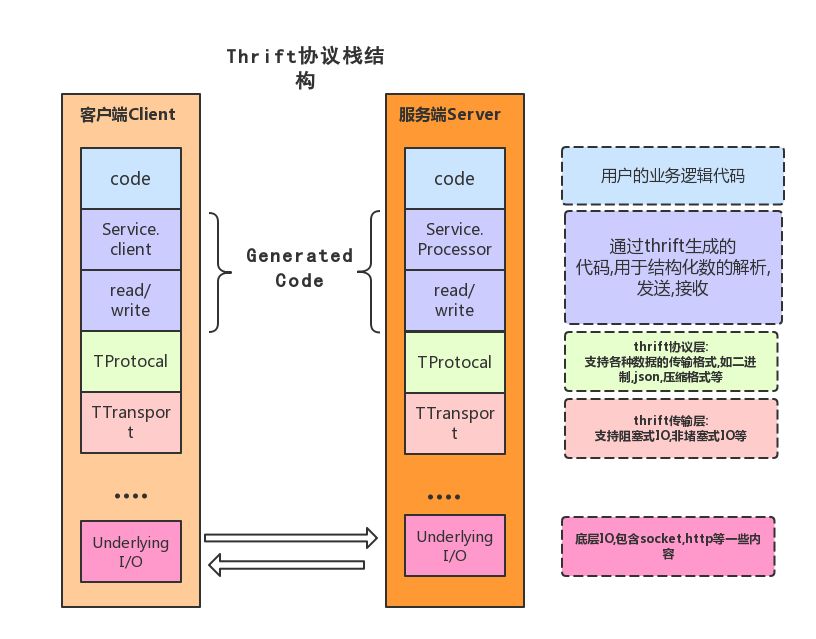
***~~TTransport层代表传输层，是数据读取及传输的渠道，和I/O层联系紧密，每一个I/O模块都会根据传输介质的不同选择相对应的TTransport来处理，如传输介质是Socket的时候，对应的传输层使用TSocket进行传输，传输介质是文件的时候，对应的传输层使用TFileTransport进行传输。~~***

***~~TProtocol代表协议层，协议层的作用有两个，一个是充当数据传输的消息队列，另一个是消息的编码和解码。协议层将结构化的数据转化为二进制流信息交给传输层进行传输。~~***

***~~TServer主要负责接收客户端发出的请求，然后将请求信息转发到Processor，Processor执行具体的处理操作，而TServer的作用是快速处理客户端的请求，尤其是要满足高并发环境下对于客户端请求的快速转发。~~***

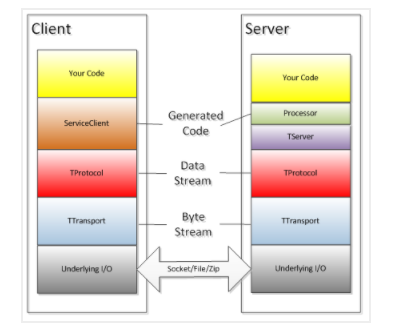
***~~TProcessor负责处理对客户端的请求，并作出响应，是服务端在Thrift和用户逻辑之间的重要转折点，同时TProcessor也负责向消息队列中写入数据。~~***

***~~为了实现以上的堆栈模块，Thrift定义了一套IDL（Interface Definition Language），用户在使用Thrift时，需要将数据结构和接口的定义卸载IDL文件中，Thrift的代码生成引擎会帮我们实现这个协议栈，用户只需要在接口的实现类中实现用户的逻辑即可。~~***

***~~~~***

***~~图4-2 系统流程图~~***

***~~Figure 4-2 System Flow Diagram~~***

***~~~~***

***~~Thrift 的优势和劣势~~***

***~~Thrift的优点如下：（1）作为一个软件框架，Thrift强大的代码生成引擎，为用户在代码编写上提供了极大的方便。（2）Thrift的应用范围很广，支持的语言非常多。（3）Thrift的跨平台特性使得不同的系统之间得以通信。（4）Thrift支持的数据格式很多，如二进制、json的序列化以及反序列化、压缩格式等。~~***

***~~Thrift也有许多不足的地方，如Thrift存在文档缺失的问题，如果要彻底理解Thrift的原理，只能阅读Thrift的源码。这给开发着也带来了极大的不便。~~***

* 1. Django Web框架

Django是python的一个开源的WEB框架，它采用了MTV的模式，及M（model）模型，模板（Template），和视图（view）使用Django可以非常简单快捷的开发WEB后台，构建自己的网站系统，适合敏捷开发。

Django Web架构概述

Django Web优势和劣势

使用Django是因为有很多优势：（1）框架免费并且开源源代码，可以进行定制。（2）可以进行快速高效的开发。（3）比较成熟的MVC框架。（4）有非常强大的生态和扩展性。（5）可以快速构建后台管理系统，非常方便。

除此之外，Django也有很多不足的地方：（1）在效率方面没有JAVA等开源框架高。（2）框架生态完成度，框架生态成员没有JAVA等框架多。（3）系统耦合性较高，没有java那么高的层次之间的分层。

* 1. ECharts 数据可视化工具

Echarts是用于数据可视化的开源库，Echarts是基于html5 Canvas实现的一个纯JavaScript的图表库，他不仅几乎兼容所有的浏览器，而且在PC和移动设备上运行也非常流畅，因此Echarts从一开始就受到广大开发者的喜爱，Echarts的特点如下所述：

1. 丰富又酷炫的图表界面：Echarts不仅为我们提供了常规的饼图、折线图、柱状图、散点图等，还为我们提供了用于统计、地理数据可视化、关系数据可视化、多维数据可视化、以及用户BI的图表，更人性化的是还能支持图与图之间的混搭。如果对于以上内置的图表不满意，我们还可以使用自定义图表，只需要传入一个函数，就能将你的数据全部映射到你需要的图表上。
2. 使用起来非常简单，Echarts被封装的很好，我们只需要掌握一些简单的前端知识就能将图表完美地展示出来。
3. 关于Echarts的文章资料十分丰富，而且描述十分详细，为开发者提供了极大的便利条件。
4. Echarts支持动态数据的展示，数据驱动图表的变化，我们在实现时只需要获取到数据，然后填入相应的数据，Echarts就能自动寻找两组数据之间差异，从而选择合适的动画效果将这些变化显现出来
   1. Hadoop生态系统

Hadoop是一个开源的分布式软件框架，在大数据时代应运而生，满足了传统技术架构和存储方式不能满足的需求。Hadoop平台能够对海量数据进行分布式的存储和计算，具有高可靠性、高扩展性以及良好伸缩性。Hadoop生态系统包含的内容很多，如图2-2所示，本节将重点介绍Hadoop生态系统的三个核心部分：HDFS、MapReduce和Yarn。

* + 1. HDFS概述

HDFS是分布式文件系统，它源自于一篇名为GFS的论文。HDFS在Hadoop生态系统中负责数据的分布式存储，为海量数据的存储提供了保障，下面我们将介绍HDFS的架构设计及优缺点。

* + - 1. HDFS架构

易于扩展的分布式文件系统，可以运行在大量普通廉价机器上，提供容错机制 ，为大量用户提供性能不错的文件存取服务

高容错性，数据自动保存多个副本，副本丢失后，自动恢复 ，适合批处理，移动计算而非数据

，数据位置暴露给计算框架，适合大数据处理GB、TB、甚至PB级数据，百万规模以上的文件数量，10K+节点规模

* + - 1. HDFS优势和劣势

提供了很多分布式的能力，流式文件访问，一次性写入，多次读取，保证数据一致性，可构建在廉价机器上，通过多副本提高可靠性 ，提供了容错和恢复机制。

同时HDFS有很多缺点，比如不支持毫秒级别的数据访问，小文件存取存取效率很低，因为在存取效率占用NameNode大量内存，寻道时间超过读取时间，并发写入、文件随机修改能力很弱，而且HDFS一个文件只能有一个写者，仅支持append

* + 1. MapReduce概述
       1. MapReduce架构

MapReduce将计算过程分为两个阶段：Map和Reduce阶段，Map阶段并行处理输入数据，Map阶段由一定数量的Map Task组成

Reduce阶段对Map结果进行汇总，Reduce阶段由一定数量的Reduce Task组成。shuffle连接Map和Reduce两个阶段

* + - 1. MapReduce优势和劣势

MR具有很多优点

首先易于编程，同时MapReduce具有良好的扩展性，高容错性，适合PB级以上海量数据的离线处理

缺点：

MapReduce同时也有很多缺点

1.实时计算能力比较弱,不能像MySQL一样，在毫秒级或者秒级内返回结果

2.对于流式计算无法处理，MapReduce的输入数据集是静态的，不能动态变化 MapReduce自身的设计特点决定了数据源必须是静态的

3.不擅长图计算，多个应用程序存在依赖关系，后一个应用程序的输入为前一个的输出

4.利用磁盘，读写效率比较低

* + 1. Yarn概述
       1. Yarn架构

HADOOP集群负责负责集群资源的统一管理和调度，处理客户端需求，同时启动监控applicationMaster

，监控监控NodeManager，负责资源的分配调度

* + - 1. Yarn优势和劣势

1.对整个集群的分配调度进行解耦，使hadoop可以在一套集群内使用多个计算框架

2.有自动纠错机制，防止在hadoop内部的分布式计算内容出现错误。

3.可以同时处理多个计算框架的计算任务。

* 1. 推荐系统

随着互联网技术的发展，推荐系统应运而生，早在推荐系统这个概念被正式提出之前，协同过滤的思想就已经被提出，基于协同过滤的设计思想，1997年推荐系统这个概念被Resnick正式提出，但是此时的推荐系统还是仅仅以协同过滤为核心。但是随着推荐系统的发展，越来越多的公司和学者也投入到了推荐系统的研究行列，关于推荐系统的研究成果层出不穷，尤其是各种推荐算法被提出，极大地促进了推荐系统的发展，使得推荐系统得到了广泛的应用。

* + 1. 推荐系统定义

推荐系统可以被认为是一种根据用户的偏好，对信息进行过滤的软件工具。随着互联网时代的到来和信息技术的迅猛发展，网络已经是用户获取信息的主要渠道，但是互联网上的信息早就呈现一种过载的状态，用户想要从大量的网络资源中提取自己感兴趣的东西越来越难，因此，搜索引擎在信息过载的时代应运而生，人们可以通过关键词对自己想要的信息进行检索，这种方式在一定程度上解决了信息过载的问题，但是现实的问题是，大部分用户并不能为搜索引擎提供一个准确有效的关键词，因此检索的结果也不尽人意。推荐系统的出现在一定程度上完成了搜索引擎完成不了的任务，将用户感兴趣的内容主动地提供给用户，这样不仅给信息的索取者提供了便利，也为信息的生产者提供了推销自己信息的机会。

搜索引擎和推荐系统不同的地方就是，推荐系统能够主动地分析用户的历史行为，并给用户建立专属的兴趣模型，推荐系统就是根据得到的用户兴趣的模型为用户产生推荐，因此使用推荐系统时，用户并不需要主动提供自己的兴趣关键词，推荐系统就会帮助用户发掘他们的兴趣爱好，从而产生推荐。这种推荐的方式使得用户得到了自己感兴趣的内容，信息生产者也提升了网站或者App的点击率，对于信息的生产者和信息的消费者来说是双赢的。推荐系统的应用领域也十分广泛，如电子商务、社交网络，音乐、电影、阅读、邮件、广告、和基于位置的服务等。

推荐系统的发展过程中，各式各样的推荐算法也层出不穷，但是他们都基于一个目的，就是想要将用户和内容联系起来，下面我们将分别介绍几种常用的推荐算法。

* + 1. 基于关联规则的推荐算法

所谓的关联规则就是不同事务之间一种潜在的联系，基于关联规则的推荐算法主要就是通过一系列用户行为发掘出这种潜在的联系。关联规则的出现最初源自于一个名为“啤酒和尿布”的故事，人们发现啤酒和尿布经常会被一起购买，这恰恰反应了一种用户行为模式，爸爸在给孩子买尿布的同时会同时购买自己喜欢喝的啤酒，如果针对这种现象，我们将啤酒和尿布放在一起，那么将会大大提升物品的销量。基于关联规则的推荐主要就是想要挖掘出用户群体在行为上的共性，然后将这种共性运用在推荐上。

关联规则分析常用的算法有Apriori、FP-growth，其中Apriori算法是最经典的关联规则分析的算法之一，这种算法简单有效但是效率较低，FP-growth算法在Apriori算法的基础上做出了改进，极大地提高了算法的效率，而Eclat算法和以上两种算法不同的地方是将数据集使用垂直表示方式，减少了数据集的遍历次数，在一定程度上提升了算法的效率。

基于关联规则的推荐算法的主要应用场景是电商网站中的捆绑销售，但是研究发现在文章推荐领域中，不同用户浏览的不同内容之间也会有一定的相关性，完全不相关的用户也可能拥有相似的阅读兴趣，因此关联规则给文章领域的推荐也带来了机会和挑战。

* + 1. 基于协同过滤的推荐算法

协同过滤算法是最早被提出的推荐算法，该算法是在1992年被Goldberg等人提出，其中基于邻域的推荐算法是协同过滤中最常用的算法，在业界中也得到了广泛的认可，由于基于邻域的算法中最重要的部分是找寻互相邻近的用户或者文章，因此才称为基于邻域的算法，其中互相邻近的用户和文章是通过相似度计算得来的。基于邻域的协同过滤算法主要分为两大类：一类是基于用户的协同过滤算法，另一类是基于项目的协同过滤算法，下面我们将主要介绍这两类方法。

基于用户的协同过滤算法致力于寻找相似的用户群体，为用户推荐其他相似用户看过的文章。在协同过滤中，我们是利用用户的历史行为记录来断定两个用户的相似性的，如果两个用户总是喜欢一些相同的文章，那么我们就可以认为这两个用户的相似度比较高，那么我们在做出的推荐的时候，就给用户推荐和这个用户相似度比较高的用户喜欢的，并且该用户以前没看过的文章。

基于项目的协同过滤主要是通过项目的相似度来产生推荐的，核心是给某个用户推荐和该用户历史上感兴趣的文章相似的文章。其中文章相似度的计算和用户的历史行为紧密相关，如果两篇文章被很多用户同时喜欢的话，我们就认为这两篇文章的相似度很高，产生推荐的时候，首先获取用户的历史浏览记录，然后为该用户推荐和该用户历史浏览过的文章相似度比较高的文章。

随着一个网站或者App的发展，其用户规模会逐渐增加，对于基于用户的协同过滤来说，他产生了很多不足的地方：（1）计算用户的相似的会变得越来越复杂。（2）产生的用户的相似度矩阵会非常稀疏，容易浪费大量的存储空间，而且计算的复杂度也会增加。（3）对于一些历史行为记录较少的用户来说，产生的推荐效果并不太好。因此以上几种缺点基于项目的协同过滤才被提出，基于项目的协同过滤弥补了基于用户的协同过滤的一些缺点，使得相似度的计算变得比较容易，而且只要用户对一篇文章产生了行为，就能立刻为用户产生合理的推荐。由于本项目所使用的用户行为记录数据规模庞大，考虑到基于用户的协同过滤的以上不足之处，本次项目使用的协同过滤算法是基于项目的协同过滤算法。

* + 1. 基于内容的推荐算法

基于内容的推荐算法的思想非常简单，就是给用户推荐和该用户以前喜欢的文章相似的文章，这种思想和基于项目的协同过滤的推荐的思想类似，但是文章相似度的计算方式有所不同，基于项目的协同过滤的文章相似度的计算方式是基于用户行为的文章相似度计算，而基于内容推荐的文章相似度是通过计算文章内容本身的相似度得来的。

基于内容的推荐算法一般包括三个步骤：（1）构造文章特征向量，一般我们是通过一些文本处理的方式将文章的关键词信息抽取出来，用来表示整篇文章。（2）构造用户特征向量，一般我们将用户的历史行为进行分析，使用用户历史上感兴趣的关键词信息作为用户的特征向量。（3）产生推荐，我们计算用户特征向量和文章特征向量的相似度，将那些和用户特征向量相似度高的文章推荐给用户。

基于内容的推荐算法被广泛地应用到各个领域中，这种推荐方式要求项目能够被抽取成有意义的特征，而且计算的过程不涉及到其他用户的信息，是一种简便有效的推荐算法。

* + 1. 混合推荐模式

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

* 1. 点击率预估

Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

* 1. Spark Mllib
  2. 特征工程
  3. 本章小结

本章主要介绍了本次项目涉及到的一些理论知识以及关键技术，主要包括Thrift框架，Hadoop生态系统，Django Web框架、Echarts数据可视化框架、推荐系统涉及的概念以及常用算法。具体的推荐算法的设计在后面的章节将会做出详细的阐述。

1. 需求分析

本章主要介绍了汽车类文章推荐系统的系统需求分析、功能性需求分析和非功能性需求分析。

* 1. 系统需求分析

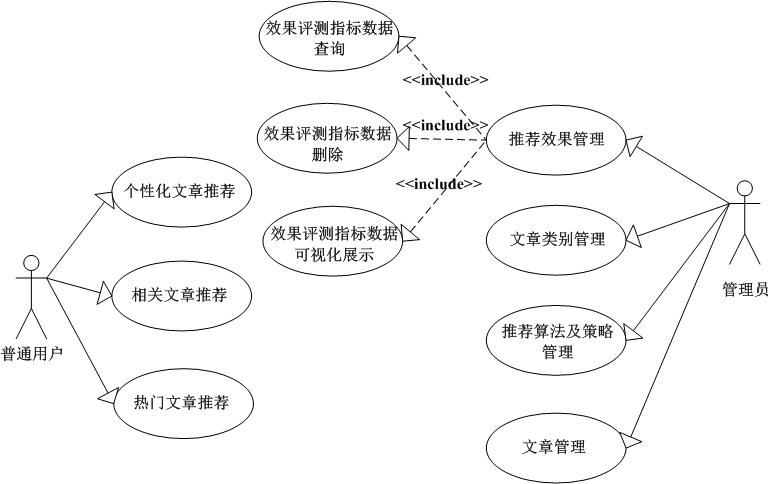
本文所介绍的推荐系统涉及到的用户角色主要有两类，分别是文章的普通读者和管理员，普通读者主要涉及到的功能主要有个性化推荐、相关推荐、热点推荐。管理员用户涉及到的功能主要有文章管理、文章类别管理、推荐算法及策略管理、推荐效果数据管理及可视化，具体的系统用例图如图3-1所示。

图3-1 系统用例图

Figure 3-1 Use Case Diagram of System

在整个系统中，普通用户可以通过个性化推荐功能收到来自于推荐系统返回的的个性化文章列表，在这基础上我们为每篇文章生成跟这篇文章内容最相关的几篇文章共同推荐给用户，这就是相关推荐的推荐过程。当新用户第一次来到汽车资讯App时，这时没有用户的历史行为记录，无法对新用户做出推荐，我们采取的策略就是给新用户推荐热门的文章，除此之外，当用户历史行为记录较少，个性化推荐的结果较少时，我们可以将热门文章作为候补文章，补充到个性化推荐列表中。

推荐系统是一个典型依靠数据作为驱动的产品，推荐系统的开发离不开大量的数据支持，推荐系统的改进也同样需要统计大量的推荐效果数据，以此来评测推荐系统的好坏，因此推荐系统的后台管理模块的一个最重要的部分就是对于推荐效果数据的统计、管理以及可视化展示。作为系统管理员用户，最基本的操作就是对文章以及文章类别的增加、修改、删除、查询。除此之外，由于本文介绍的推荐系统是基于多种算法的混合推荐系统，因此我们根据实际情况为每种算法设置权重值，在多种算法融合的阶段，需要对各个算法推荐的结果进行加权操作，因此推荐算法管理模块的主要功能就是根据实际的推荐效果对推荐算法及其权重进行不断调整，以此来达到最优的推荐效果。

* + 1. 系统数据流

用户在使用汽车资讯App时会产生很多行为，如浏览文章，查看车型信息等行为，这些用户行为会以日志的形式保存下来作为推荐系统做用户行为分析时最主要的数据来源。

原始用户行为日志有两个去向，以下我们将分别介绍用户的两个去向：第一，由于考虑到日志数据非常庞大，传统的关系型数据库已经满足不了海量日志的存储，因此采用Hadoop生态系统中的HDFS（分布式文件存储系统）来存储海量的原始用户行为日志；第二，为了构造用户的短期兴趣标签，原始用户行为日志会被实时地处理和解析成用户的短期兴趣标签，由于我们只保留近期少量的兴趣标签，因此使用非关系型数据库Redis不仅能满足短期兴趣标签的存储的要求，Redis数据库作为内存数据库，其快速存取的特性为线上调用用户兴趣标签的响应速度提供了保障。

由于多种推荐算法及策略涉及到对HDFS上大量数据的处理与计算，因此我们使用MapReduce对存储在HDFS上的大量数据进行分布式的离线计算，同时为了保障线上的响应速度，我们将离线计算的结果周期性地更新到关系型数据库中进行存储。

线上运行的混合推荐引擎将获取到关系型数据库中存储的数据和非关系型数据库中存储的用户兴趣标签数据，然后进行处理，进而生成推荐候选文章列表。

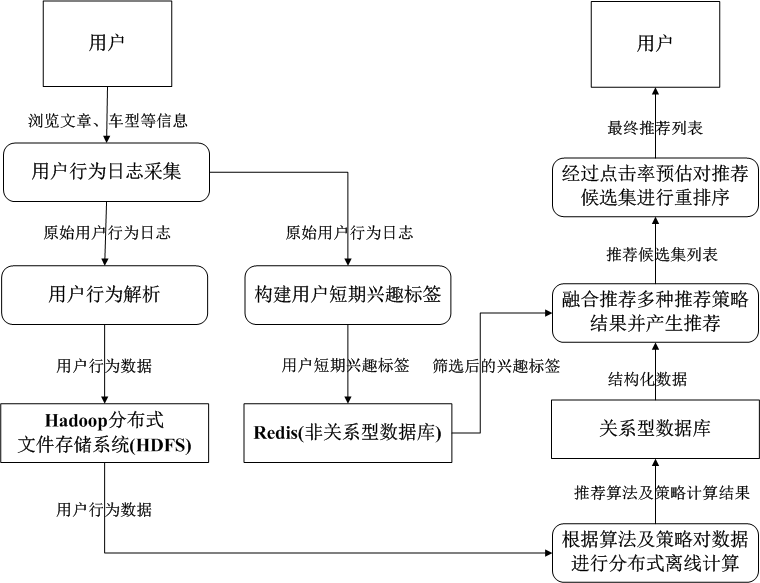
我们将训练好的点击率预估模型对推荐候选文章列表中的各篇文章进行点击率预估，得出每篇文章被点击的概率，我们按照这个概率对文章进行重排序，生成最终的推荐列表返回给用户。具体的数据流图如图3-2所示。

图3-2 系统数据流图

Figure 3-2 Data Flow Diagram of System

* + 1. 开发环境

开发语言：主要使用Java语言和Python语言进行开发，部分模块使用了Scala语言。其中推荐系统服务端使用Java语言，推荐系统后台管理模块使用Python语言进行开发，其中涉及到点击率预估模型的训练是在Spark上进行的，因此也涉及到了Scala语言的使用。

集成开发环境：IntelliJ IDEA

数据库：使用了非关系型数据库Redis和关系型数据库MySQL，其中Redis的版本是Redis3.1，MySQL的版本是MySQL5.7。

**要不要写Hadoop的版本信息**

* 1. 系统功能性需求分析

本文介绍的基于混合推荐模式的推荐系统按照功能可以分为源数据同步与处理模块，混合推荐引擎模块，点击率预估模块，后台管理模块。

* + 1. 数据同步与处理模块

在本项目中，原始数据被分成三类，分别是原始用户行为日志数据、文章数据、注册用户数据，下面我将分别介绍这三类数据的同步与处理过程以及数据存储的方式。

原始用户行为日志数据的产生源于用户在App上的各种操作行为，用户一旦发生操作，日志数据会被服务端放入消息队列中，原始的日志数据未经过处理，因此含有很多无效的信息。本模块要做的就是消费消息队列中的原始用户行为日志数据，然后从日志数据中抽取出有效的信息，然后按照规范的结构存储到HDFS（Hadoop Distributed File System）中。

文章数据是我们要推荐的主要数据，当前的推荐系统很难作为一个独立的产品而出现，往往需要结合不同的业务形态，很多情况下需要跨部门的数据传输，本项目使用的文章数据也不例外，本项目需要的文章数据是通过接口从App的后台数据库同步至推荐系统的数据库进行存储，这里使用的数据库是关系型数据库。

注册用户数据和文章数据类似，也是从App的后台数据库同步至推荐系统数据库进行存储，这里使用的数据库是关系型数据库。

* + 1. 基于多种推荐算法的混合推荐引擎模块

基于多种推荐算法的混合引擎模块最主要的功能是个性化推荐，除此之外还有相关推荐和热门推荐两个功能。

个性化推荐的结果是多种不同的算法融合的结果，其目的是为了产生更精确的推荐，主要是根据用户的历史行为记录为用户生成个性化推荐列表。

相关推荐主要是根据文章内容的相似度产生推荐，在文章详情页的底部为用户生成和该篇文章内容相似的文章作为延伸阅读推荐给用户。

热门推荐主要是根据文章的热度进行推荐，优先推荐热度高的文章。当某用户的历史行为记录太少，导致个性化推荐结果过少时，使用热门推荐进行补充。

* + 1. 点击率预估模块

点击率预估模块的主要功能是对推荐候选文章列表进行点击率的预估，得到每篇文章被用户点击的概率，并使用该概率值对文章推荐列表进行重排序。由此可以看出来，对点击率预估模型的开发是本模块的重点，开发模型涉及到对用户特征的建立、文章特征的建立、特征工程的建立、以及模型的训练及评估四个部分，以下我们将详细介绍每个部分各自的功能。

用户特征的建立部分除了获取用户的基本信息，还需要根据用户的历史行为记录挖掘出用户的偏好，以此来刻画一个用户的画像，用户的特征主要包括xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx。

文章特征的建立是建立和文章相关的有效特征，以此来刻画文章。文章特征分为两部分，一类是文章的基本特征，另外一类是文章的统计特征。文章的基本特征包括文章的标题、作者、来源、分类、内容、发布时间、修改时间、关键词等信息。文章的统计信息包括文章、文章类别以及作者的近一周的点击量、点击率数据，是基于海量的用户历史数据分布式离线统计的结果。

建立特征工程的目的是构建训练点击率预估模型所需要的特征数据集，换句话说就是从原始的数据中抽取出有效的特征用来训练模型，使得模型达到比较好的效果，特征工程对于机器学习中模型的训练非常重要，如果特征选的好，就算是最简单的算法也可以达到很好的效果。特征工程建立可以分为特征组合、特征选择、特征映射三个步骤，最终将构造好的特征数据集用于模型的训练。

模型的训练及评估部分主要是根据选择好的模型，使用我们特征工程构建的特征数据集进行模型的训练，并根据每次的训练结果调整参数，直到模型得到一个较好的效果。

* + 1. 后台管理模块

推荐系统后台管理模块主要包括以下几个功能：推荐算法管理、推荐效果管理、文章管理、文章类别管理。

推荐算法及策略管理主要包括对算法及其权重的增加、修改、删除、查询。由于本项目涉及多种推荐算法，每种推荐算法都有不同的权重，在进行推荐时，对不同的算法进行加权，从而产生最终的推荐结果，我们会经常根据推荐效果调整各个算法的权重值使得推荐效果达到最优，因此推荐算法管理功能在推荐后台管理系统中是很重要的功能。

推荐效果管理是推荐系统后台管理模块中最重要的部分，我们通过对各项评测指标的统计结果来反映推荐效果的好坏，以此来对推荐系统进行优化。推荐效果管理包括两部分，第一部分是评测指标对应的统计数据的删除、查询功能，其中评测指标包括推荐量、点击量、点击率、人均阅读率等指标。第二部分是评测指标数据的可视化，其中，文章类别和文章作者的推荐量、点击量、点击率、人均阅读率使用饼图表示，各个算法的推荐量、点击量、点击率、人均阅读率用饼图和折线图表示，App整体的推荐量、点击量、点击率、人均阅读率使用折线图表示。

文章管理部分是对文章的基本数据进行管理，具有对文章增加、修改、删除、查询的功能。

文章类别管理部分具有对文章的内容类型增加、修改、删除、查询的功能。

* 1. 系统非功能性需求分析

本节主要介绍系统的非功能性需求，包括响应时间和易用性。

* + 1. 响应时间

为了保证大量的计算过程不影响线上的响应时间，我们对复杂的计算过程采用离线计算的方式，并将离线计算的结果周期性地更新到关系型数据库中供线上调用，同时为了提升数据库的查询速度，还为数据库表建立了索引。

推荐系统服务端对90%的请求的响应时间不超过49毫秒，平均响应时间保持在34毫秒左右。

* + 1. 可靠性

数据库使用了复制架构，即主库的数据会实时同步到从库上，如果主库发生宕机等事故时，数据库访问请求会转移到从库上，从而使系统能够正常运行，保证了系统的可靠性。

* + 1. 易用性

推荐系统为用户提供了简洁易用的界面，尤其是推荐系统后台管理模块中推荐效果报表的展示的部分，采用了折线图、饼图等图形对各项评测指标进行了展示，用户可以清晰地查看推荐效果及其变化趋势等，方便对推荐系统做出改进。

* 1. 本章小结

本章先介绍了系统需求分析，分析了系统用例以及系统数据流，然后对系统的功能性需求做了详细的说明，最后简要地介绍了系统的非功能性需求。

1. 推荐系统概要设计

本章主要阐述了汽车类文章推荐系统的概要设计，包括系统总体架构设计、系统模块划分、系统业务流程、数据库设计。

* 1. 系统总体架构设计

推荐系统整体对外提供Thrift服务，App客户端通过发送Thrift请求获取推荐列表。

* 1. 系统模块划分

汽车类文章推荐系统划分为原始数据同步与解析模块、基于多种推荐算法的混合推荐引擎模块、推荐点击率预估模块、推荐效果统计及可视化模块。推荐系统的原始数据来源有两个，其中一种数据来源方式是通过WebService接口同步外部文章数据，另一种数据来源方式通过Kafka消息队列获取原始的用户行为日志数据。其中原始数据同步与解析模块主要负责实时同步文章数据，并将原始的日志数据按照一定的规则进行解析，并根据用户的浏览、收藏、搜索、分享等行为为用户打上相应的兴趣标签存储到数据库中为推荐引擎提供数据来源。基于多种推先算法的混合推荐引擎模块主要负责利用多种推荐算法计算用户对文章的兴趣度、文章关键词、文章内容相似度、文章的关联规则，然后根据计算结果筛选出推荐候选集列表，最终对推荐候选集进行已推荐过滤及新颖性排名操作。推荐点击率预估模块主要通过机器学习的方法训练点击率预估模型，利用训练好的模型预测推荐的文章被用户点击的概率，相当于使用模型预测的概率值对上一个模块产生的推荐候选集进行重排 序，再选择概率最高的文章N篇文章作为最终的推荐结果列表，此外训练模型使用的特征数据的提取涉及大量的分布式离线计算，也是本模块的重点开发任务。推荐效果统计及可视化模块主要负责统计每天的推荐效果，如每种算法的PV、UV、点击率、人均阅读次数、用户留存率等并以图表的形式在网页中展示，方便我们查看推荐效果的趋势，进而帮助我们不断改进推荐算法。系统模块设计图如4-1图所示。

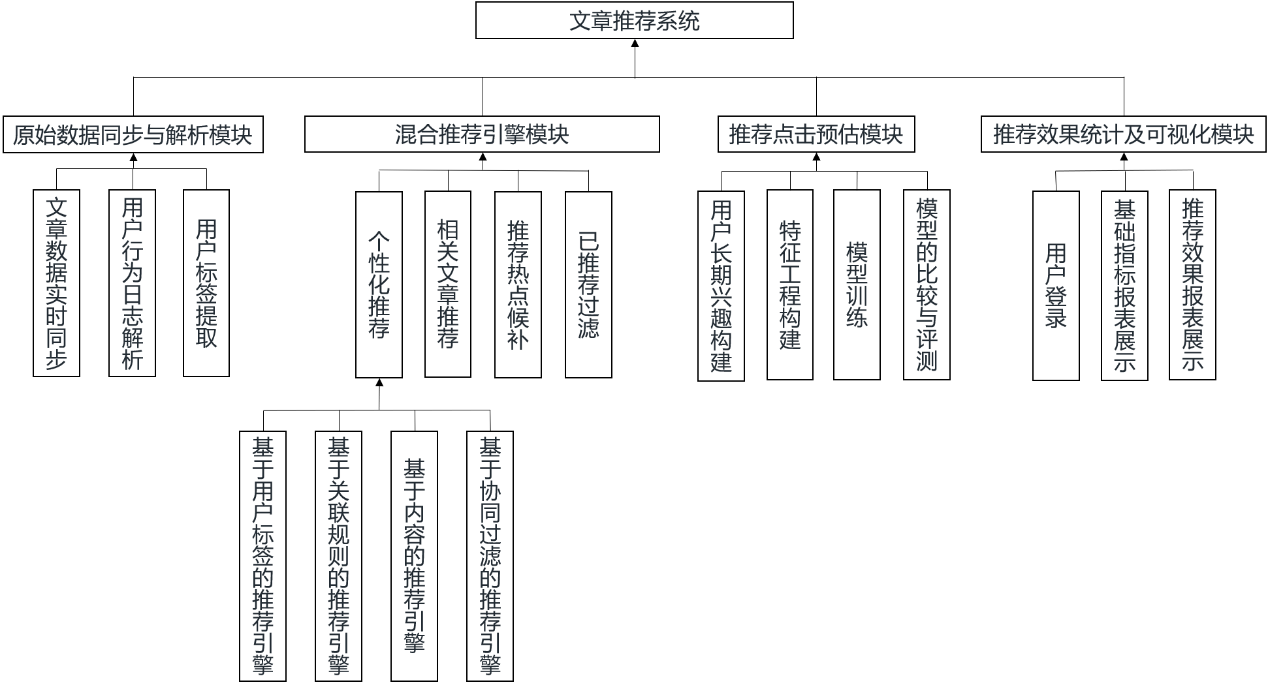
图4-1 系统模块设计图

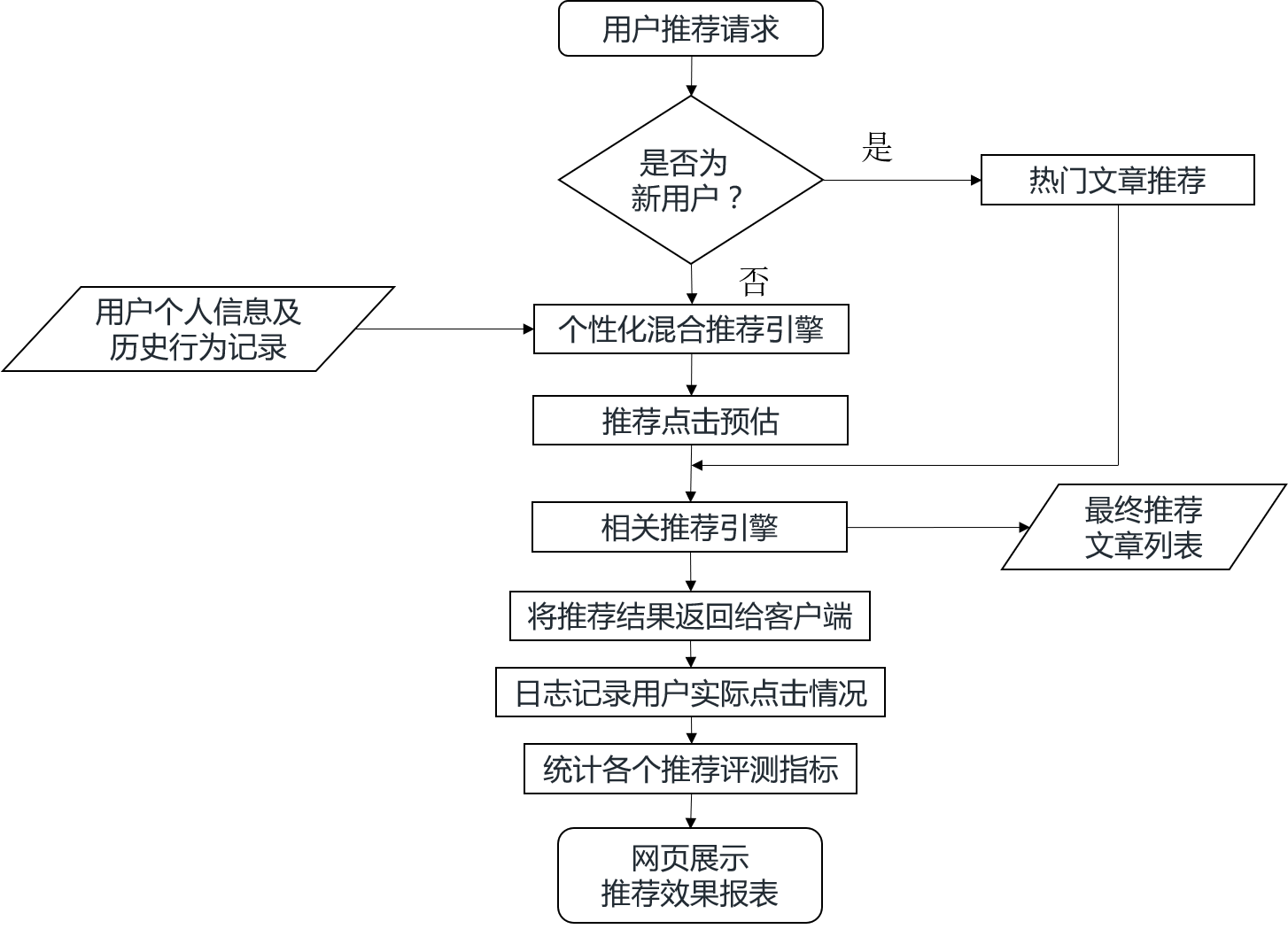
Figure 4-1 System Module Diagram

* 1. 系统业务流程

本节主要介绍推荐系统整体的业务流程，首先用户在APP客户端发出推荐请求，推荐系统服务端首先根据用户的设备ID判断用户是新用户还是老用户，如果该用户新用户，我们直接给用户推荐热门文章列表，此时推荐完成。如果该用户是老用户，那么我们只需要将老用户的个人信息输入到基于多种算法的混合推荐引擎中，即可得到该用户的推荐候选列表，为了使我们的推荐更加准确，推荐候选列表会输入到我们已经训练好的推荐点击率预估模型中，对用户的点击行为进行预测，得出的结果是候选列表中每篇文章被该用户点击的概率值，选择点击率预估概率最高的N篇文章作为最终的推荐结果，然后将最终的N篇文章输入到相关文章推荐引擎中得到每篇文章的相关文章，最终将个性化推荐结果和相关文章推荐结果作为最终的推荐结果返回到APP客户端，除此之外我们会对每天线上的点击率、人均阅读率等在线指标进行统计，并以报表的形式可视化地展现在网页上，方便对推荐算法进行改进。整体的系统流程如4-2图所示。

图4-2 系统流程图

Figure 4-2 System Flow Diagram



* 1. 数据库设计

本节主要说明推荐系统的数据库设计，包括E-R图和物理表结构设计。

E-R图设计

由于本次项目主要针对于与汽车相关的文章的推荐，因此汽车相关和文章相关的数据都是必不可少的，除此之外还有与推荐策略相关的表，以及数据可视化涉及到的推荐效果报表数据。表之间的联系如图4-3所示。

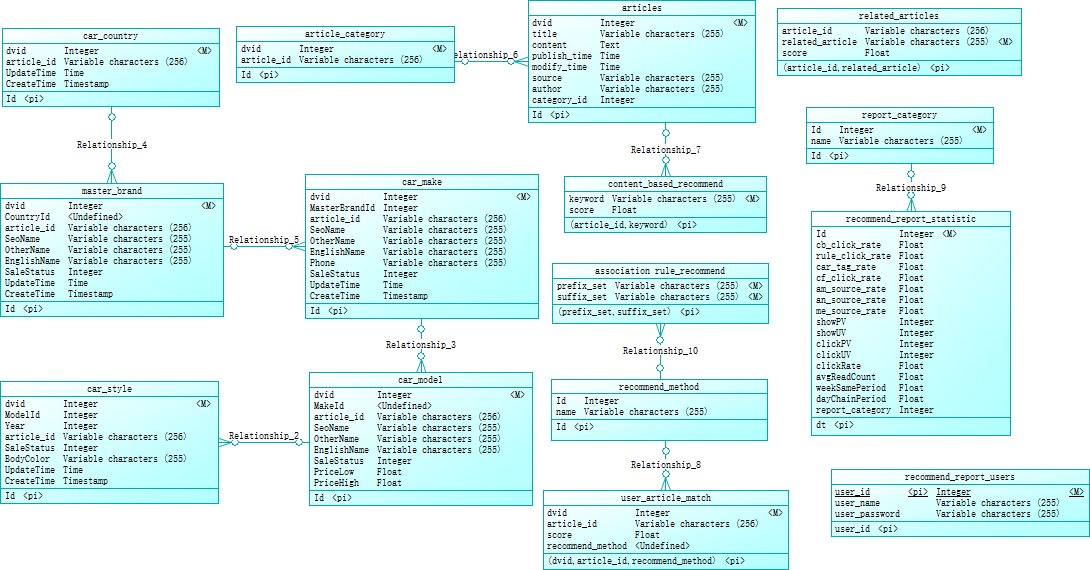


图4-3 E-R图

Figure 4-3 E-R Diagram

物理表结构设计

涉及到本次项目的表共有15张，本节将利用表格的形式说明每张表的表结构及其含义与作用，具体信息如表4-1至表4-11所示。

1. 文章基本信息表

如表4-1所示，文章基本信息表主要用于记录文章的ID、标题、内容、发表时间、修改时间、文章来源、文章作者、文章类型等内容。

表4-1 文章信息表

Table 4-1 article information sheet

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 字段说明 | 备注 |
| id | varchar (12) | 文章ID | 主键 |
| title | varchar (500) | 文章标题 | 非空 |
| content | mediumtext | 文章内容 | 非空 |
| publish\_time | datetime | 发表时间 | 非空 |
| modify\_time | datetime | 修改时间 |  |
| source | varchar (100) | 文章来源 | 非空 |
| author | varchar (500) | 文章作者 | 非空 |
| category\_id | int | 文章类型 | 非空 |

表4-2 文章类别信息表

Table 4-2 article category information sheet

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 字段说明 | 备注 |
| id | int | 文章类别ID | 主键 |
| name | varchar (500) | 文章类别名称 | 非空 |

表4-3 车辆国别表

Table 4-4 article category information sheet

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 字段说明 | 备注 |
| id | int | 国别ID | 主键 |
| name | varchar (500) | 国别名称 | 非空 |
| update\_time | timestamp | 更新时间 | 非空 |
| create\_time | timestamp | 创建时间 | 非空 |

表4-4 车辆品牌表

Table 4-3 article category information sheet

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 字段说明 | 备注 |
| id | int | 品牌ID | 主键 |
| name | varchar (100) | 品牌名称 | 非空 |
| country\_id | int | 国别ID | 外键 |
| seo\_name | varchar (100) | App显示名称 |  |
| other\_name | varchar (500) | 其他别名 |  |
| English\_name | varchar (100) | 英文名称 |  |
| Sale\_status | int | 销售状态 | 非空 |
| Update\_time | timestamp | 更新时间 | 非空 |
| Create\_time | timestamp | 创建时间 | 非空 |

表4-5 车辆制造厂商表

Table 4-5 article category information sheet

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 字段说明 | 备注 |
| id | int | 厂商ID | 主键 |
| name | varchar (100) | 厂商名称 | 非空 |
| Brand\_id | int | 品牌ID | 外键 |
| seo\_name | varchar (100) | App显示名称 |  |
| other\_name | varchar (500) | 其他别名 |  |
| English\_name | varchar (100) | 英文名称 |  |
| Sale\_status | int | 销售状态 | 非空 |
| Update\_time | timestamp | 更新时间 | 非空 |
| Create\_time | timestamp | 创建时间 | 非空 |

表4-6 车型表

Table 4-6 article category information sheet

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 字段说明 | 备注 |
| id | int | 车型ID | 主键 |
| name | varchar (100) | 车型名称 | 非空 |
| make\_id | int | 厂商ID | 外键 |
| seo\_name | varchar (100) | App显示名称 |  |
| other\_name | varchar (500) | 其他别名 |  |
| English\_name | varchar (100) | 英文名称 |  |
| Sale\_status | int | 销售状态 | 非空 |
| Price\_low | Double | 最低价格 | 非空 |
| Price\_high | double | 最高价格 | 非空 |
| Update\_time | timestamp | 更新时间 | 非空 |
| Create\_time | timestamp | 创建时间 | 非空 |

表4-7 车款表

Table 4-6 article category information sheet

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 字段说明 | 备注 |
| id | int | 车款ID | 主键 |
| name | varchar (100) | 车款名称 | 非空 |
| model\_id | int | 车型ID | 外键 |
| Year | Double | 车款年份 | 非空 |
| Sale\_status | Int | 销售状态 |  |
| Body\_color | Varchar(100) | 车身颜色 |  |
| Update\_time | timestamp | 更新时间 | 非空 |
| Create\_time | timestamp | 创建时间 | 非空 |

1. 用户-文章相关表

如表4-7所示，用户-文章相关表主要用于记录用户和文章的匹配度，而表格中匹配度是通过不同的推荐策略计算而得出的结果，具体使用的推荐策略存储在推荐算法类别表中。

表4-8 用户-文章相关表

Table 4-8 article category information sheet

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | | 字段类型 | 字段说明 | 备注 | |
| dvid | int | | 车款ID | | 联合主键 |
| Article\_id | varchar (100) | | 文章ID | | 联合主键 |
| Score | Double | | 匹配度 | | 外键 |
| Rcmd\_method | Int | | 推荐策略类别 | | 联合主键 |

1. 推荐策略类别表

如表4-9所示，推荐策略类别表中存储的是计算用户-文章匹配度所使用的多种推荐策略，推荐策略类别表中主要存储的是推荐策略ID、推荐策略名称。

表4-9 推荐策略类别表

Table 4-9 article category information sheet

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 字段名称 | | 字段类型 | | 字段说明 | 备注 | |
| Id | int | | 推荐策略ID | | | 主键 |
| name | varchar (100) | | 推荐策略名称 | | | 非空 |

1. 文章关键词表

如表4-10所示，文章关键词表中包括文章ID、关键词、关键词得分三个字段。文章使用TF-IDF算法进行关键词的提取，并将关键词及其得分存储在文章关键词表中。

表4-10 文章关键词表

Table 4-10 article category information sheet

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 字段名称 | | 字段类型 | | 字段说明 | 备注 | |
| Article\_id | varchar (100) | | 文章ID | | | 联合主键 |
| keyword | varchar (100) | | 文章关键词 | | | 联合主键 |
| score | double | | 关键词得分 | | | 非空 |

1. 相关文章表

如表4-11所示，相关文章表中有三个字段分别是文章ID、相关文章ID、两篇文章的相似度，其中文章ID和相关文章ID联合起来共同作为相关文章表的主键。

表4-11 相关文章表

Table 4-11 article category information sheet

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 字段名称 | | 字段类型 | | 字段说明 | 备注 | |
| Article\_id | varchar (100) | | 文章ID | | | 联合主键 |
| Related\_article | varchar (100) | | 相关文章ID | | | 联合主键 |
| score | double | | 文章相似度 | | | 非空 |

1. 关联规则表

如表4-12所示，假设A、B、C为文章，（A,B）->C代表的含义为看过A和B文章的人也会看C文章，此时，我们认为（A,B）这个集合和C具有关联性，我们称（A,B）为前件，C为后件，关联规则表中存储的就是我们通过计算得出的文章之间的关联性，即关联规则中的前件和后件两个字段。

表4-12 关联规则表

Table 4-12 article category information sheet

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 字段名称 | | 字段类型 | | 字段说明 | 备注 | |
| prefix\_set | varchar (100) | | 前件 | | | 联合主键 |
| suffix\_set | varchar (100) | | 后件 | | | 联合主键 |

1. 评测指标类型表

如表4-13所示，评测指标类型表中主要存储用于评测推荐效果的指标信息，如各个算法的点击率、人均阅读率等，包括两个字段，分别是指标的ID和指标的名称

表4-13 评测指标类型表

Table 4-13 article category information sheet

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 字段名称 | | 字段类型 | | 字段说明 | 备注 | |
| Id | varchar (100) | | 报表类型ID | | | 主键 |
| Name | varchar (100) | | 报表类型名称 | | | 非空 |

1. 指标统计结果表

如表4-14所示，我们将每天统计各个指标的值存储到指标统计结果表中，这些指标用于评测推荐系统在线上的推荐效果，帮助我们对推荐算法做出改进。

表4-14 指标统计结果表

Table 4-14 article category information sheet

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 字段名称 | 字段类型 | | | 字段说明 | 备注 | |
| dt | Datetime | | | 日期 | 联合主键 | |
| Id | | integer | 指标ID | | | 联合主键 | |
| Value | | double | 指标值 | | | 非空 | |

1. 推荐报表系统用户表

如表4-15所示，推荐报表系统用户表主要用于存储报表系统的用户信息，包括用户ID、用户名、用户密码等信息。

表4-15 推荐报表系统用户表

Table 4-14 article category information sheet

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 字段名称 | | 字段类型 | | 字段说明 | 备注 | |
| User\_id | | Integer | | 用户ID | 主键 | |
| User\_name | Varchar(100) | | 用户名 | | | 非空 | |
| password | Varchar(100) | | 密码 | | | 非空 | |

1. 推荐算法及策略设计
   1. 基于内容的推荐

基于内容的推荐算法主要根据用户的历史兴趣特征与文章内容的相似度来产生对该用户的文章推荐列表。基于内容的推荐算法主要包含4个步骤：用户兴趣特征构造，文章内容关键词提取，计算相似度并产生推荐。

基于内容的推荐流程

用户的兴趣特征主要通过分析用户在历史上看过的一些文章内容来构造出用户感兴趣的关键词信息。通过实时消费用户行为日志，记录用户看过的文章信息，然后对文章进行关键词提取操作，将文章关键词信息作为用户的特征进行记录，通过用户行为的不断更新，用户特征的权重也会不断更新，用户感兴趣的关键词权重会越来越高，为了能够更好地表达用户最近一段时间的兴趣，我们采用权重的时间衰减策略，对于长时间没有更新过权重的特征进行降权。

文章关键词的提取操作是在文章数据同步与更新时进行的，当系统收到文章更新请求时，首先将文章的基本信息进行存储，然后调用文章分词接口将文章进行分词，最终利用文章的分词结果计算TF-IDF值作为文章词语的权重值，输出权重值最高的词语作为文章最终的关键词，文章的关键词数据会被结构化地存储在数据库中供线上实时调用。

当基于内容的推荐引擎收到某一个用户推荐请求时，首先将该用户的特征数据和文章关键词数据向量化，计算该用户特征向量与候选集文章向量的余弦相似度，选择相似度最高的文章作为基于内容的推荐引擎的推荐结果列表。

**计算app一整天的用户量，决定是否将一天之内的内容标签存储在redis中或者mysql中。。。。。。或者es存储也可以考虑，只存储一天之内的用户，作为唯一实时的的推荐引擎，车标签的推荐引擎没办法实时，因为涉及到复杂的离线计算了**

基于内容推荐的优势与劣势

基于内容的推荐算法有如下优点：（1）用户之间是相互独立的，计算某一个用户的兴趣偏好的时候不需要依赖其他用户的行为数据，很好地解决了数据稀疏性问题。（2）当有新的文章加入到推荐候选集中，可以立即得到被推荐的机会，而且新文章被推荐的机会和老文章是一致的。（3）具有良好的推荐解释性，可以很容易地向用户提供合理的推荐理由，比如推荐的文章和用户关注的哪些关键词匹配。

基于内容的推荐算法也会有一些局限性：（1）文章特征的抽取过程中，只能抽取一些容易提取的文本类内容，而对于文章中的图片等多媒体数据的特征抽取具有一定的难度。（2）不能挖掘用户的潜在兴趣，只能为用户推荐和已有兴趣相似的文章内容。

* 1. 基于车标签的推荐策略设计

由于本项目主要讨论的是关于汽车领域文章的推荐，绝大多的用户也主要是浏览和车相关的内容，根据本次项目的实际特点，本人采取如下推荐策略：第一步首先发掘文章和车的相关性，为文章打上和车相关的一些标签；第二步是发掘用户和车的相关性，为用户打上和车相关的标签，最后，通过车标签将用户和文章关联起来，下面几节中将分别介绍这三步的详细设计思路。

车标签简述

在汽车领域，和车相关的属性有很多，我们设计车标签的目的主要是挖掘用户感兴趣的车型，并将含有用户感兴趣的车型的相关文章推荐给用户。我们本次采取用的车标签主要包括汽车的车型、品牌、车身、级别、厂商、国别、价格。汽车还有很多相关的参数名词，如轴距、驱动方式、进气形式、百公里加速时间等，因为这些参数几乎每个车型都会有，因此不能表示出一个明确的车型，因此我们在车标签类别中不考虑这些参数数据。

文章的车标签提取（车标签的示意图可以画）

文章的车标签提取和基于内容的推荐过程中对文章关键词的提取有一些类似，不同的地方是，文章关键词代表的是文章的整个内容，而车标签只是要提取文章中关于车型的词语，而且分词完成后需要通过车型标签推导出品牌、车身、级别等其他相关的标签，具体的提取策略如下所述。

首先将文章内容进行分词，分词时使用自定义词典将每篇文章中的车型名称提取出来，这里所说的自定义词典指的是含有所有车型名称及其各种别名的一个数据集，在分词时，自定义词典中含有的词会被分词程序认为是一整个词从而不会被切分开，而且有了自定义词典方便了只对车型名称的提取操作。至此，文章分词的结果是文章中出现的所有车型名称。

使用**TF-IDF算法**计算文章中每个车型关键词的得分，以此来评估每个车型关键词在文章中的重要程度，并将得分值作为该关键词的权重，然后使用车型表将车型名称关键词映射成车型ID，至此得到每篇文章对应的多个车型ID及其权重。

通过车型ID及其权重生成品牌、车身、级别、厂商、国别、价格等标签，具体的计算策略为：如果一篇文章对应的所有车型ID都属于同一个品牌，则产生品牌ID标签，权重为车型权重的一半，如果不属于同一个品牌，则不生成品牌标签。车身、级别、厂商、国别的计算策略同品牌一样，至此为每篇文章产生了一系列和车型相关联的车标签及其权重。

用户车标签的提取

用户车标签提取的数据来源是用户行为日志，提取用户车标签需要的用户行为有浏览文章、浏览车型综述页、收藏车型、查看车款详情、车款询价、条件选车、违章查询，用车服务。这些行为中都包含着具体的车型信息，我们对不同的行为赋予不同的权重，并将权重对应到具体的车型ID上，如我们规定用户浏览车型页的权重是2，用户收藏车型的权重是4，我们认为用户浏览的车型ID对应的权重是2，用户收藏的车型ID对应的权重是4，除此之外，我们对权重按照时间进行衰减，将长时间不关注的车型的权重降低，举个例子，如果最近7天的权重为1，那么将过去的8至14天的权重设为0.9，最后得到的每个车型最终的权重是以上两种权重相乘的结果。

通过用户对应的车型ID及其权重生成品牌、车身、级别、厂商、国别、价格等标签，权重为车型权重的一半，若不同车型对应相同的品牌，则对品牌的权重累加，同理，累加车身、级别、厂商、国别、价格的权重。**以后可以将这个计算方式改进成公式的形式。**

用户和文章的关联度计算

通过以上两个小节，我们能得到文章对应的车标签及其权重，以及用户对应的车标签及其权重。本节我们将介绍如何通过车标签将用户和文章连接起来。

具体关联度的计算公式为：

1. 其中TW的计算公式是xxxxxxxxxxxx，代表的含义将文章和用户对应的相同的车标签ID的权重相乘再相加TP = log(5,1+Pu/(1+|Pu-Pa|))，a = 40/∑厂商权重之和 ；如果a>=1则让a=1，用户和文章匹配的最终值=a\*TW\*TP
   1. 基于关联规则的推荐

基于关联规则的推荐算法主要是要发掘文章之间一些潜在关联性，并将这些相关性运用在我们的推荐上。举个例子，假如我们发掘出了如下一条关联规则：（A，B）->C，这条规则的含义是，同时看过文章A和文章B的文章有很大的可能性都会看文章C，根据这条规则，我们可以很容易地对用户做出推荐，为同时看过A和B文章的用户推荐C文章。由以上的例子我们可以发现，对于关联规则的挖掘是本节的重点。

支持度、置信度、提升度计算方式

支持度、置信度、提升度的计算和关联规则的挖掘紧密相关，是我们度量关联规则的三个重要指标，下面将用实例介绍支持度、置信度、提升度的具体计算方式。

支持度用来表示文章I1和文章I2同时被同一个用户点击的概率，如果支持度大代表两篇文章的关联程度就大，支持度具体计算方式如下所示，其中N(I1,I2)代表同时阅读过文章I1和文章I2的用户数量，N(u)代表用户总数。

置信度则表明了如果一个用户阅读过I1文章，则这个用户阅读文章I2的概率有多大，如果置信度为100%，则代表如果一个用户阅读了文章I1，则这个用户一定会阅读文章I2，置信度具体的计算方式如下所示，其中N(I1,I2)代表同时阅读过文章I1和文章I2的用户数量，N(I1)代表阅读过文章I1的用户数。

提升度表达的意思是文章I1是否被阅读能否影响文章I2被阅读，提升度具体的计算方式如下所示，其中代表置信度，即阅读文章I1后阅读文章I2的概率，P(I2)代表没有阅读文章I1但是确阅读文章I2的用户出现的概率。

频繁项集生成算法的比较

所谓项集，在本项目中指的是文章的组合，K-项集指的是具有K篇文章的项集，而频繁项集就是指在所有用户浏览记录中出现最频繁的文章的组合，我们要找文章之间的关联性，就要首先发掘频繁项集，目前主流的频繁项集生成算法主要有Apriori、FP-growth、Eclat算法，在本节中我们将对这三种算法进行优缺点的分析，选择最适合本项目的算法来生成频繁项集。

Apriori算法是一种经典的频繁项集算法，采用一种迭代搜索的方式，首先找到1-项集，对1-项集进行剪枝操作，将不满足最小支持度的项集删除，然后将剩余1-项集中前缀相同的项集进行连接组合生成2-项集。根据上述方法一直迭代下去，直至不能再生成满足最小支持度的项集。整个过程看成是剪枝，连接两个步骤不断地重复迭代，虽然算法简洁明了，但是这种算法的缺点也是显而易见的：（1）每次进行剪枝操作时都需要遍历一次整个数据集，大量的I/O操作造成了该算法的性能瓶颈。（2）连接操作过程中产生大量的无效的候选集，使得计算的空间复杂度非常高。

FP-growth算法中最大的亮点是对FP-tree（Frequent Pattern Tree）的构建，该算法首先对数据集进行遍历，将数据集中的信息全部保存到树结构中，而且保存了项集之间的关联关系，很好地避免了大量候选集的产生，构建完树结构后，然后遍历FP-tree的头表产生一系列条件模式基，每个条件模式基就是一个长度为1的频繁项集的前缀，然后继续分别对条件模式基构建条件FP-tree，最终产生频繁项集。综合来看该算法引入了树结构，并采用分而治之的策略，相对于Apriori算法优点显而易见：（1）计算的过程不需要产生大量候选集（2）只需要遍历两次数据集就可以完成整个算法。虽然FP-growth比较高效，但是在FP-tree的构建过程对内存的消耗很大，容易造成内存溢出，因此减少内存的消耗也是今后要改进的方向。

Eclat算法采用的是倒排策略，将数据集中每条记录按照文章项来进行汇总，该算法只需要遍历一次数据集，得到所有1-项集，然后对1-项集求交集得到2-项集，依次迭代下去，直至K-项集包含所有的文章项，此时选择包含元素个数大于阙值的交集对应的项集做为最终的频繁项集。Eclat和以上两种算法相比，明显的优点是：（1）运算过程非常简便，只需要不断求集合的交集运算即可。（2）只需要遍历一次数据集，比以上两种算法的遍历次数都少。但是当数据集规模较大时，求交集的运算也会随之变得非常复杂，大量的交集运算极有可能变成算法性能的瓶颈。

本项目中使用的数据集是从用户行为日志中解析出来的，随着APP用户数量的增加，单节点的数据存储已经不能满足日益增加的日志数量，因此，我们将用户行为日志存储在分布式文件系统HDFS上。我们下面将根据本项目的实际情况来选择适合本项目的算法作为生成频繁项集的算法。

Apriori算法是三个算法中效率最低的，频繁的遍历数据集，进行剪枝和链接操作是该算法的核心，造成严重的I/O瓶颈是不可避免的，改进的空间非常有限，因此我们不采用这种算法。FP-growth算法虽然通过引入树结构来改善了Apriori算法的两个严重缺点，而且针对于小规模数据集来说，在构造树的过程中对内存的消耗问题完全可以忽略，但是本项目采用的数据集规模庞大，在构造FP-tree的时候势必会占用大量的内存空间，因此这种算法对于本项目来说也是不可行的。Eclat算法运算简单，而且只需要遍历一次数据集就可以完成整个算法，比以上两种算法的遍历次数都少，而且计算的复杂度也优于以上两种算法，除此之外Eclat算法在庞大的数据规模下面临的频繁求交集的问题可以使用分布式计算MapReduce来解决，通过MapReduce的分而治之的思想，能过很好地地减少Eclat算法在大规模数据下计算量。

这部分描述可以画fp-tree、还有eclat的那个示意图。

推荐过程

具体的推荐步骤主要分为三步，分别是：根据原始数据集生成频繁项集，根据频繁项集生成强关联规则，根据强关联规则产生推荐。

关联规则算法使用的数据集是用户浏览文章的历史记录，每一条记录代表一个用户历史浏览过的文章的集合。

第一步，我们利用上一节讨论过的FP-growth算法对原始数据集进行处理，这个过程不断采用支持度进行剪枝操作，将不满足最小支持度的文章项删除，直到所有文章项集满足最小支持度，即生成频繁项集。

第二步，根据第一步产生的所有频繁项集生成强关联规则，是否为强关联规则通过计算置信度来判断，删除不满足最小置信度的关联规则，直至所有关联规则都满足最小置信度，即生成强关联规则。

第三步，计算上一步生成的所有强关联规则的提升度，当提升度小于等1时，为无效的关联规则。举个例子，若等于1，则证明I1和I2是相互独立的，也就是不管I1是否出现，对I2的出现都毫无影响，这种情况下，即使得到了很高的支持度和置信度，但是它也不是一条有效的关联规则，至此，删除所有无效关联规则，即生成最终的有效强关联规则。

第四步，生成用户推荐列表时，根据用户的历史行为记录，按照关联规则进行筛选，得到和用户浏览过的文章相关联的文章生成推荐。

由于用户行为记录数据规模很大，因此将数据集分布式地存储在HDFS上，关联规则的计算方式采用离线计算，并将离线计算的结果周期性更新到数据库中，保证线上调用时的响应速度。（离线计算的结果是用户的推荐结果还是关联规则？？？？？？？？？？？？？？）

关联规则推荐优缺点

基于关联规则的推荐有如下优点：（1）能产生清晰有效的规则。（2）这种推荐一般能够提高一些文章的转化率，比如当用户阅读了一篇频繁项集中的文章，那么该用户阅读频繁项集中的其他文章的概率将会大大提升。

基于关联规则的推荐也有不足的地方：（1）由于这种算法采用的也是根据用户的历史行为数据进行推荐，也不可避免地产生了冷启动问题。（2）由于热门物品出现的次数非常多，这种算法可能造成热门物品被过度地推荐，而造成了推荐结果的新颖性不高。

* 1. 基于项目的协同过滤的推荐算法

基于项目的协同过滤推荐算法通过用户评分矩阵和文章相似度矩阵计算用户与文章之间的相似度。该算法是推荐系统中最常用的基本算法，而且早在业界中得到了广泛的应用。

该方法的基本思想就是给用户推荐的文章是与该用户之前喜欢的文章相似的文章，其中文章相似度的计算方式和基于内容的推荐算法的相似度计算方式不同，基于项目的协同过滤算法的相似度计算是通过分析用户的历史行为记录得到的，比如文章A和文章B的相似度很高，是因为喜欢文章A的用户很大一部分也喜欢文章B。

基于邻域的算法主要包括两类，一类是基于用户的协同过滤算法，另外一类是基于项目的协同过滤算法，由于随着用户数据的不断增加，使用基于用户的协同过滤算法的在计算上时间复杂度和空间复杂度都非常大，因此我们在本项目中采用基于项目的协同过滤算法。

文章相似度的计算及改进方式

首先根据用户的历史行为计算文章之间的相似性，计算相似性的方式为统计喜欢A文章的用户中有多少比例的用户同时也喜欢B文章，具体的计算公式为，这个公式中分母代表喜欢文章i的用户数量，分子代表同时喜欢文章i和文章j的用户数量，这样看来，这个公式是有一个缺点的，因为当喜欢文章j的用户非常多，文章j是非常热门的文章的时候，这个公式的值会趋近于1，造成的结果就是任何一篇文章和热门文章的相似度都很高，很显然我们并不希望每次推荐的结果都是热门商品，因此我们根据这个缺点对公式做出改进。

将公式改进为，改进后的这个公式对文章j的权重进行惩罚，从而减小了热门文章和其他文章的相似度。

通过以上的公式，我们可以看到，在基于项目的协同过滤算法中，两篇文章具有相似度是因为这两篇文章同时被很多用户感兴趣，也就是说，很多用户对相似度的产生都有贡献，但是由于用户的活跃度不同，因此用户对相似度的贡献也不尽相同。~~经过分析，我们发现有些用户虽然很活跃，但是很多行为并不是因为自身的兴趣而产生的，因此我们引入IUF （Inverse User Frequence）来对文章相似度的计算进行改进。IUF是用户活跃度的对数的倒数，改进之后的公式对活跃用户进行了惩罚，使得相似度的计算更加合理，具体的计算公式为XXXXXXXXXXXXXXXXXXXXXXXXXXX。~~

用户评分矩阵计算

用户在app上对文章的浏览、收藏等行为属于隐形反馈数据集，这些隐形反馈数据集在一定程度上能够反映用户对文章的评分，我们制订了如下规定，当用户u对文章i有收藏行为的时候，用户u对文章i的评分为3，当用户对文章浏览时间大于域值但是没有收藏文章时，用户u对文章i的评分为2，当用户u对文章i的浏览时间小于域值的时候，用户u对文章i的评分为1，当用户u对文章i没有任何行为的时候，用户u对文章的评分为0。根据以上规定对所有用户行为进行统计可以得到m行n列的用户评分矩阵Rm×n，其中评分矩阵中第i行，第j列的元素rij表示用户i对文章j的隐性评分。

由于每个用户对文章的行为及其有限，因此评分矩阵中评分非零的元素远远小于矩阵元素的总数，是一个稀疏矩阵，因此我们在对矩阵进行存储时，为了节省空间，我们只存储非零元素，将（用户，文章，评分）三元组结构化地存储在数据库中。

基于项目的协同过滤推荐的基本流程

基于项目的协同过滤算法的实现主要包括三个部分，第一部分计算文章相似度矩阵，第二个部分计算用户评分矩阵，最后综合前面两部分计算用户对文章的兴趣度，产生用户的推荐列表集合。

评分矩阵和文章相似度矩阵的计算涉及对大量行为日志的处理和分布式离线计算，因此我们对评分矩阵和文章相似度矩阵进行周期性离线更新，在线上避免复杂的计算过程，从而保证了线上能够实时处理推荐请求。

项目之间的相似度计算可以使用5.3.1节介绍的经过IUF参数修正的相似度计算公式，计算完文章相似度矩阵和用户评分矩阵后，我们使用如下公式计算用户u对一个物品j的兴趣：

puj=∑i∈N(u)∩S(j,K)ruiwij

上式中N(u)表示用户感兴趣的文章的集合，S(j,K)表示和文章j相似度最高的K篇文章，wij表示文章i和文章j的相似度，rui表示用户u对于文章i的评分，该公式表明和用户以往喜欢的文章相似度越高的文章，最后在该用户的推荐列表中排名越高。

根据以上兴趣度公式的计算选择用户兴趣度最高的前N篇文章作为该用户的推荐集合，将离线计算好的推荐结果结构化地更新到数据库中，供推荐系统在线上调用。

基于项目的协同过滤算法的优势和不足

基于项目的协同过滤算法有如下优点：（1）用户的推荐结果和用户的行为紧密相关，如果用户有了新的行为，一定会体现在最终的推荐结果中。（2）对于汽车类的文章来说，用户的兴趣相对比较稳定，可以将复杂的计算工作离线进行，并周期性更新，这样可以保证线上的响应速度。（3）相似度的计算来源于用户的历史行为，能够合理地为用户提供推荐理由，增加用户的信任度。

基于项目的协同过滤算法也有一些不足的地方：（1）需要维护一个文章相似度矩阵，随着文章数量的增加，所需要的空间代价也随之增加。（2）不能解决冷启动问题，新用户没有历史行为记录，因此不能为新用户做出合理的推荐。（3）新文章没有被任何用户浏览过，因此也不会被推荐给用户。

* 1. 点击率预估模型的设计

XGBoost算法设（模型使用的什么参数）

GBDT算法设计

XGBoost算法与GBDT算法比较

时间复杂度

空间复杂度

* 1. 基于多种算法的混合推荐策略

以上几个小节中介绍了几种不同的推荐方式，包括基于内容的推荐、基于车标签的推荐、基于协同过滤的推荐、基于关联规则的推荐，综合看来这些推荐策略都是各有各的优缺点，因此我们将这几种算法混合使用，这样能够保留各个算法各个的优势，从而得到更全面的推荐列表。

下面将介绍几种常用的混合推荐技术：（1）加权型混合推荐技术，具体的做法是对不同的推荐算法赋予不同的权值，将产生的推荐结果项的得分按照推荐算法的权值重新进行加权操作，得到新的推荐结果集。（2）整体混合型推荐技术，具体的做法是将不同的推荐算法产生的结果全部输出，用户根据实际情况对推荐结果进行调整。（3）分级混合推荐技术，具体的做法是将多种推荐算法按照算法的效果划分层次，优先采用效果好的推荐算法。

开始

获取用户历史记录

利用推荐点击率预估模型对候选集重排序

本项目采用的是加权型混合推荐技术，**文章的推荐流程如图5-x所示**，当用户向推荐系统发出推荐请求时，系统会首先获取用户的历史记录，之后的具体步骤如下。

若用户的历史记录不为空，则图中所示的四种推荐算法将利用用户的历史行为记录为该用户产生个性化推荐，由于采用了加权法，所以对不同推荐算法产生的结果进行加权操作，从而产生最终的推荐结果。如果用户的历史记录为空，则需要根据文章的热度对文章进行推荐。

由于多种算法混合推荐，因此产生的推荐结果包含的文章数量可能会非常大，但是我们每次为用户返回的推荐列表集合中文章的数量是有限的，因此我们将混合推荐引擎产生的推荐候选集合输入到点击率预估模型中对用户的点击行为进行预测，生成每篇文章被用户点击的概率，我们利用输出的概率值进行推荐候选集合的精准排序，最终选择概率最高的前N篇文章作为推荐结果返回给客户端。

* 1. 本章小结

**本章主要对本项目使用的推荐算法进行了详细的阐述，**

1. 系统详细设计与实现

本章将介绍系统的详细设计与实现的过程，主要包括数据同步与处理模块、基于多种推荐策略的混合推荐引擎模块、点击率预估模块、推荐后台管理模块的详细设计与实现。

* 1. 系统类详细设计

本节将详细介绍推荐系统中涉及到的各个主要的类的设计与实现，包括各个类的功能介绍，以及类包含的方法的详细说明，具体的类图如图6-1所示。

* 1. 数据同步与处理

数据同步与处理模块包括用户行为日志、文章数据、注册用户数据的同步与处理，从而为推荐算法及策略的计算提供了数据来源。

* + 1. 用户行为日志同步与处理

原始用户行为日志是从外部服务器产生并通过Kafka消息队列传输到推荐系统服务端的，然后推荐系统服务端通过消费Kafka消息队列中的数据来获取原始用户行为日志，最后从原始用户行为日志中提取出有效的数据并解析成清晰的结构化数据存储至HDFS（Hadoop分布式文件系统）中。

解析完成后的用户行为日志包括以下几个字段：用户的设备（手机）ID、设备品牌、设备操作系统、设备型号、注册ID、发生行为的时间、行为类型、行为对象、发生行为时所在的省份、城市，具体的字段如表6-1所示。

表6-1 用户行为表

Table 6-1 User Behavior sheet

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段说明 | 字段内容 | 备注 |
| deviceId | 用户设备ID | A00000643597FB | 手机标识码 |
| deviceBrand | 用户设备品牌 | vivo | 手机品牌 |
| deviceOS | 用户设备操作系统 | android | 手机操作系统 |
| deviceModel | 用户设备型号 | X20 | 手机型号 |
| uid | 用户注册ID | 860162035413761 | 用户注册时分配的ID |
| actionTime | 发生行为的时间 | 2017-12-12 16:35:06 |  |
| actionType | 行为类型 | 浏览文章 |  |
| actionItem | 行为对象 | am100012 | 文章ID |
| provice | 发生行为时所在省份 | 河北 |  |
| city | 发生行为时所在城市 | 保定 |  |

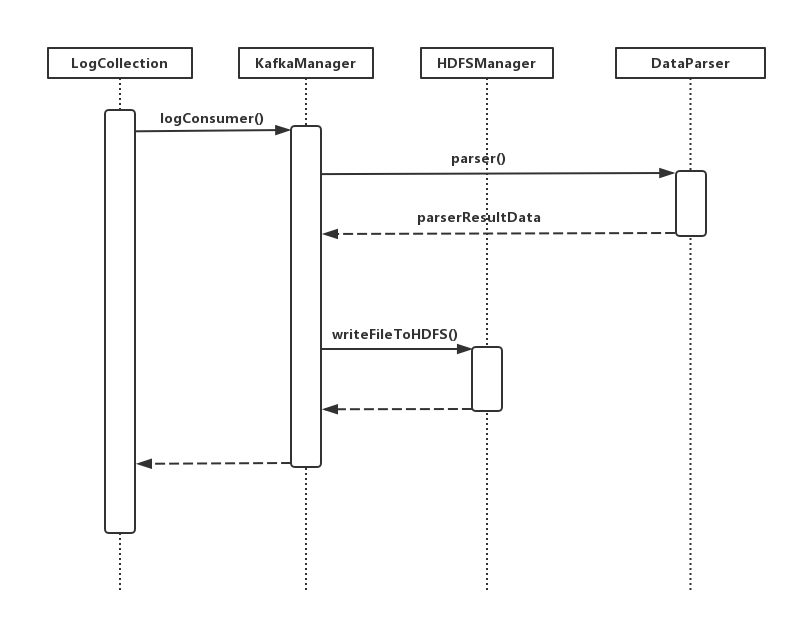
用户行为数据同步及处理的时序图如图6-1所示。其中LogCollection是负责以不同的方式收集日志数据的类，我们这里使用的用户行为日志数据是通过Kafka消息队列进行传输的，KafkaManager是负责管理Kafka消息队列中的数据的类，该类实现了消费原始用户行为日志数据的logConsumer方法，该方法会实时监控Kafka消息队列中的数据，一旦有用户行为数据产生就会发生消费行为将数据取出。DataParser类主要负责数据的转化，该类实现了parser方法，该方法对从Kafka消息队列中消费到的原始用户行为日志数据进行解析，然后返回结构化的有效用户行为数据。HDFSManager类主要负责管理HDFS上存储的文件，该类实现了writeFileToHDFS方法，该方法将解析完成的用户行为数据结构化地存储到了HDFS，其中存储到HDFS时，按照日志的更新时间进行分区。

图6-1 用户行为数据同步时序图

Figure 6-1 Sequence Diagram of User Behavior Data Synchronization

* + 1. 注册用户数据同步与处理

注册用户数据包括已经在汽车资讯App上注册过的用户信息，包括用户的性别、年龄等基本信息以及用户收藏夹数据。推荐系统依赖于外部系统的数据支持，已注册用户的数据就是从外部系统获取。注册用户的基本信息用户于构造用户的基本特征，为推荐点击率预估模型的开发提供了一部分数据，由于模型是在离线的状态下进行开发，因此，用户数据的同步采用周期性的离线更新方式。

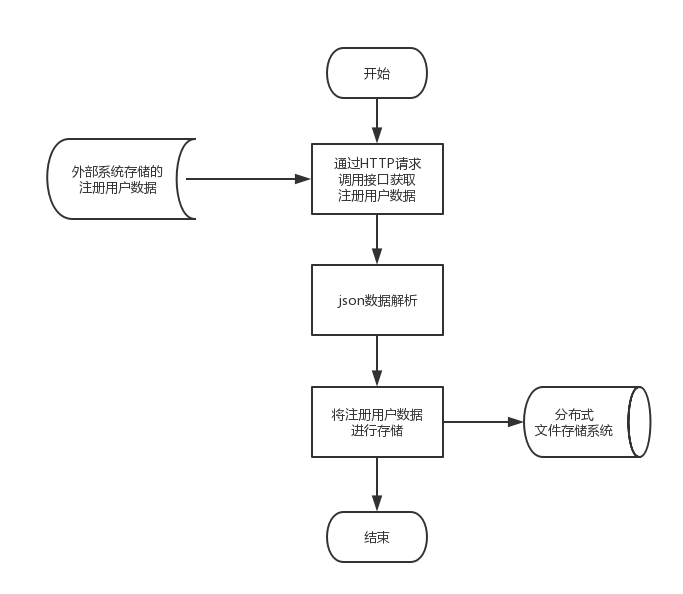
用户数据同步的流程图如图6-2所示，首先通过HTTP请求调用外部接口来获取存储在外部系统的用户数据，然后对获取到的用户数据进行解析，最终存储在HDFS（Hadoop分布式文件系统）中，为点击率预估模型的分布式离线训练提供了数据来源。

图6-2 用户数据同步流程图

Figure 6-1 Flowchart of Registered User

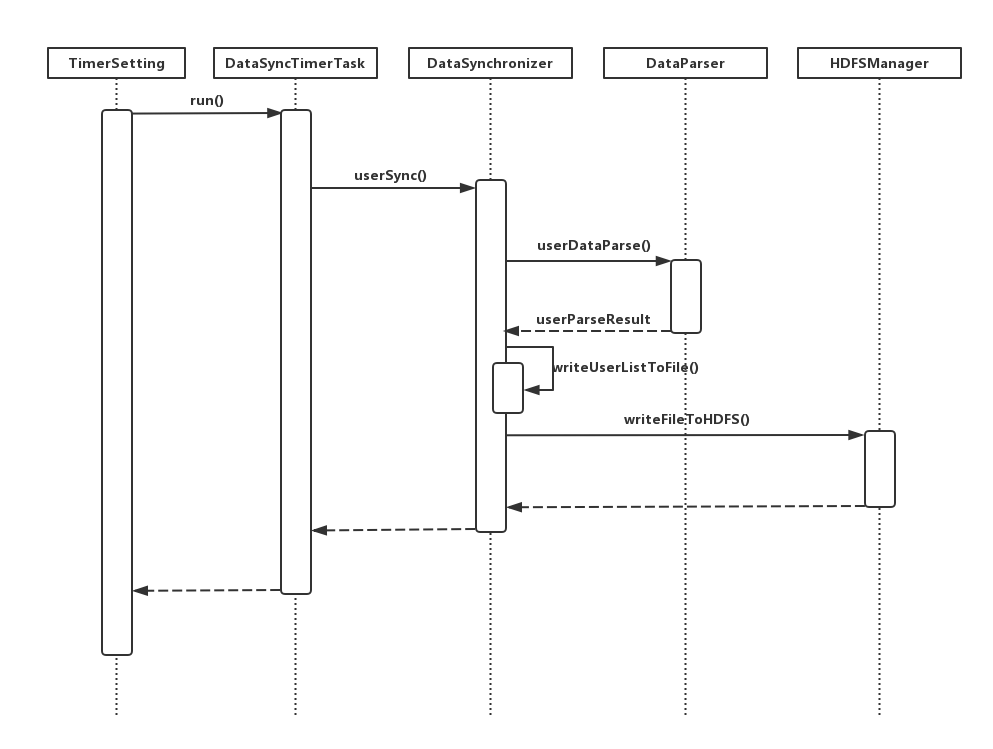
用户数据同步的时序图如图6-3所示，TimerSetting是定时器设置类，可以设置定时器第一次执行时间和执行的周期，定时器设置完成后，会执行DataSyncTimerTask类中的run方法，该类继承了TimerTask类，并重写了run方法，run方法中调用DataSynchronizer类中的userSync方法来实现用户数据同步的逻辑，用户数据同步的过程中调用userDataParse方法对用户数据进行解析成结构化的格式，然后调用writeUserListToFile方法将结构化的用户数据存储至文件中，最后将文件存储至其中存储在HDFS上的数据按照更新时间进行分区。

图6-3 用户数据同步时序图

Figure 6-1 Flowchart of Registered User

* + 1. 文章数据同步和处理

文章基本信息和用户基本信息类似，由于推荐系统本身需要依赖外部系统的数据支持，因此文章信息也是通过接口的方式从外部系统获取，并周期性地更新用户数据。在进行文章数据存储时，为了方便线上的调用和线下的离线计算，不仅将文章数据存储到MySQL数据库中，在HDFS上也会进行文章的存储。

文章信息包括文章ID、作者、标题、内容、关键词等字段，具体的字段如表6-2所示。

表6-2 文章信息表

Table 6-2 Article Information sheet

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段说明 | 字段内容 | 备注 |
| id | 用户设备ID | A00000643597FB | 手机标识码 |
| author | 用户设备品牌 | vivo | 手机品牌 |
| title | 用户设备操作系统 | android | 手机操作系统 |
| content | 用户设备型号 | X20 | 手机型号 |
| publishTime | 用户注册ID | 860162035413761 | 用户注册时分配的ID |
| modifyTime | 发生行为的时间 | 2017-12-12 16:35:06 |  |
| category | **文章的车标签** | 浏览文章 |  |
| actionItem | 行为对象 | am100012 | 文章ID |
| provice | 发生行为时所在省份 | 河北 |  |
| city | 发生行为时所在城市 | 保定 |  |

如图6-4所示是文章数据同步的时序图，同用户数据同步类似，使用TimerSetting类对定时器的第一次执行时间和更新周期进行设置，定时器设置完成后会调用DataSyncTimerTask类中的run方法执行定时任务，其中DataSyncTimerTask类继承了TimerTask类，并重写了该类中run方法，run方法负责填充数据同步的业务逻辑。

在执行文章数据同步时，定时器调用的是articleSync方法，该方法在DataSynchronizer类中负责同步增量的文章数据，其中增量的文章数据包括新增的文章和修改过的文章。

获取到所有的增量文章数据后需要调用KeyWordsExtract类中的extract方法提取增量的文章数据的关键词信息，并将关键词信息返回到DataSynchronizer类中。

为了方便线上的推荐系统获取到文章数据，我们将文章数据存储到MySQL数据库中，ArticleDao类负责和MySQL数据库进行交互，包括对文章数据的增加、修改、查询、删除等功能。将获取到的文章增量数据进行存储时，首先对增量文章数据进行遍历，并判断数据库中是否含有该文章ID，如果含有该文章ID，那么将新的文章数据更新到数据库的文章表中，如果不含有该文章ID，那么将该文章数据插入到数据库的文章表中。

为了方便线下的推荐策略的离线计算，我们还将文章数据存储在HDFS上，HDFSManager类主要负责和HDFS(Hadoop分布式文件系统)进行交互，在完成MySQL数据库的文章数据存储之后，将调用HDFSManager类中的writeDataToHDFS方法将增量的文章的数据全部存储至HDFS中，并按照文章的更新时间进行分区。

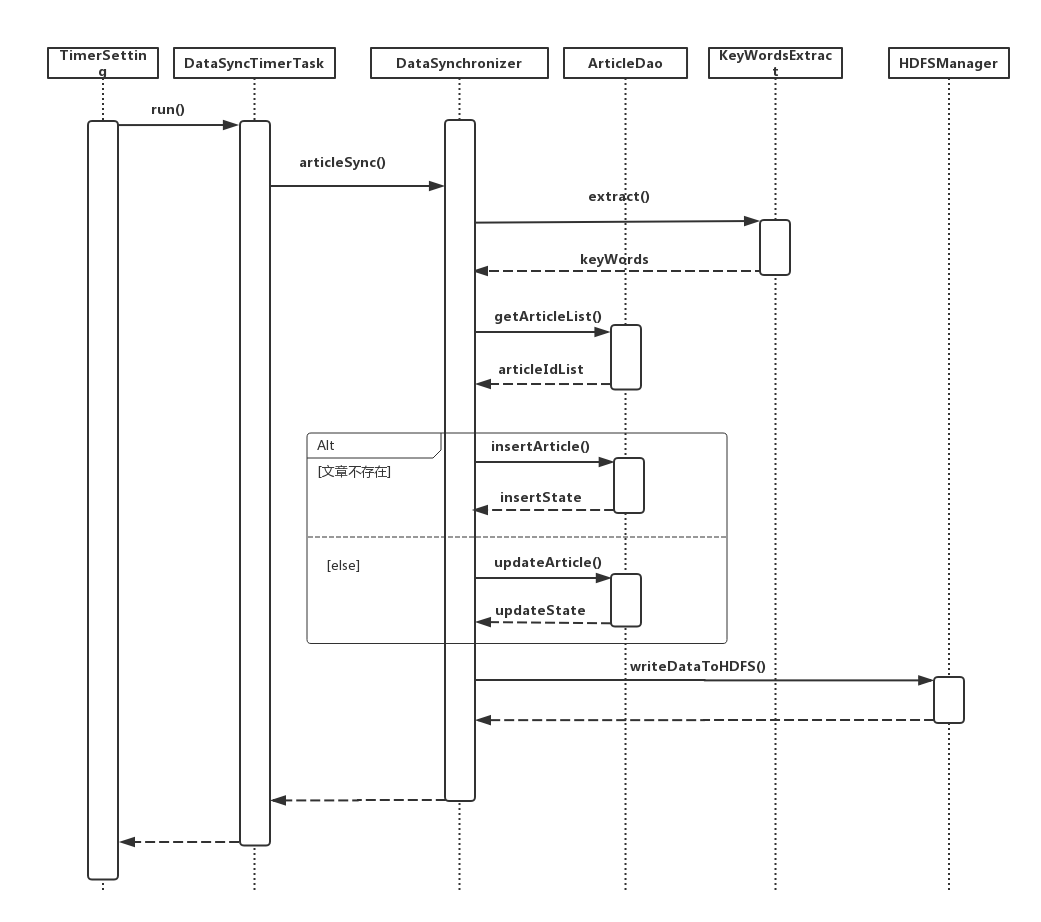


图6-4 文章数据同步时序图

Figure 6-1 Flowchart of Registered User

* 1. 基于多种推荐策略的混合推荐引擎

基于多种推荐策略的混合推荐引擎是整个推荐系统的核心模块，原始的用户行为日志、文章数据、用户数据经过混合推荐引擎模块的分析、计算最终转化为推荐列表。推荐列表的生成过程中涉及大量的离线计算与分析，如果将这个过程在线上进行会延长推荐服务器的响应时间，会严重影响用户满意度。因此需要对推荐算法及策略进行周期性的离线的分析和计算，并将生成的推荐信息更新到关系型数据库中，当推荐系统服务器收到推荐请求时，会实时地从数据库中查询推荐信息并做简单的处理即可推荐候选集列表。

* + 1. 个性化文章推荐

文章的个性化推荐涉及多种推荐策略，最终产生推荐候选集时将多种的推荐策略的结果进行加权融合。具体涉及到的推荐算法及策略包括基于内容的推荐、基于协同过滤的推荐、基于关联规则的推荐、基于车标签的推荐，其中基于车标签的推荐是根据本次项目和车辆信息有着紧密联系的特点，进而设计的一种推荐策略，具体四种推荐策略的实现详情我们将在本节进行具体的介绍。

* + - 1. 基于协同过滤的推荐

基于协同过滤的推荐的计算过程中需要存储在HDFS上的用户行为日志数据，具体的计算过程使用MapReduce分布式离线计算，计算步骤主要分为三步：第一步是计算用户评分矩阵，第二步是根据用户的历史行为记录计算文章的同现相似度矩阵，第三步是结合以上两步的计算结果来计算用户对文章的兴趣度，第四步是按照兴趣度进行排序，保留用户最感兴趣的10篇文章并存储到MySQL关系型数据库中，供推荐系统在线上调用。

协同过滤算法使用的初始数据集是经过解析的用户行为日志数据，数据集有三个字段，分别是用户ID、用户行为类型、文章ID，其中用户行为类型分为点击和收藏。如图6-5所示是基于协同过滤的推荐的主要计算流程，下面将根据流程图详细介绍算法的计算过程。

第一步，根据用户行为数据集得到用户评分矩阵。其中对于评分来说，我们有如下规则：用户行为类型为点击时，用户对该文章的评分为1，用户行为类型为收藏时，用户对该文章的评分为2。如图所示，UserScoreMatrixMapper类和UserScoreMatrixReducer类构成了一个完整的用于计算用户评分的MapReduce程序，其中UserScoreMatrixMapper是整个MapReduce程序的Map阶段，负责对输入数据进行处理，将原始的数据集中的用户对文章的行为类型转化成了用户对文章的评分，并将用户ID作为key，文章ID和评分作为value进行输出。UserScoreMatrixReducer类是整个MapReduce程序的Reduce阶段，这个阶段将根据Map阶段的输出结果将相同用户ID的数据项进行汇总，并将用户ID和用户对所有文章的评分作为用户评分矩阵输出。

第二步，根据用户评分矩阵得到文章同现相似度矩阵。由图中可以看到，本步骤的输入数据集是上一步中的MapReduce输出的用户评分矩阵。ItemOccurenceMapper类和ItemOccurenceReducer类构成了一个完整的用于计算文章同现相似度矩阵的MapReduce程序，其中ItemOccurenceMapper是整个MapReduce程序的Map阶段，负责对输入数据进行处理，由于计算文章同现值不需要用到评分数据，因此我们只将用户对应的多篇文章进行两两组合，以文章组合作为key，以1作为value进行输出，ItemOccurenceReducer类代表整个MapReduce程序的reduce阶段，负责将Map阶段的输出数据按照key值进行汇总，这里是对文章组合进行汇总，得到各个文章组合出现的次数，即为文章的同现相似度矩阵。

第三步，根据第一步得到的用户评分矩阵和第二步得到的文章同现相似度矩阵得到各个用户分别对各个文章的兴趣度。从数学的角度出发，要得到用户对文章的兴趣度只需要将以上两个矩阵相乘即可，而矩阵相乘的操作可以被分解成相乘、相加两部分，于是考虑到计算程序的并行化，本步骤将矩阵相乘的操作切分成两个MapReduce程序共同计算两个矩阵相乘的结果。首先MatmulMapper类作为Map阶段负责接收用户评分矩阵和文章同现相似度矩阵的数据，为了方便区分输入数据，我们将使用flag进行标识，A代表用户评分数据，Map端使用文章ID作为key，用户及评分数据作为value进行输出，B代表文章同现相似度数据，Map端使用一篇文章ID作为key，另一篇文章及两篇文章的同现相似值作为value进行输出。下面举例说明MatmulReducer类作为reduce端的计算过程，由于以上两份数据都是以文章ID作为key输出的，因此文章ID相同的数据会进入到同一个reduce中进行汇总，假设汇总后的数据为mapA：<item1,(item1:1,item2:3,item3:4)>，mapB：<item1,(u1:1,u2:3)>，以上数据有如下含义：对于item1来说，它的同现文章及其同现值存放在mapA中，所有用户对它的评分数据放在mapB中，reduce阶段，只需要遍历mapA，将和item1同现的所有的文章的同现值和mapB中每个用户对于item1的评分相乘，得到以用户ID为key，文章ID和相乘结果为value的输出结果。AddMapper和AddReducer是最后一个MapReduce处理过程，这个MapReduce将上一个MapReduce的输出结果作为输入，在Map阶段的输出key是用户ID，输出value是文章ID和相乘结果，在Reduce阶段将按照相同的用户ID进行汇总，将相同用户ID且相同文章ID的分数相加，即得到最终的用户评分矩阵和文章同现矩阵的相乘结果，也就是用户对文章的兴趣度矩阵。

第四步，将上一步得到的文章兴趣度矩阵按照兴趣度的大小进行排序，对于每一个用户只该保留该用户最感兴趣的10篇文章，并将排序和筛选的结果存储到关系型数据库MySQL中。

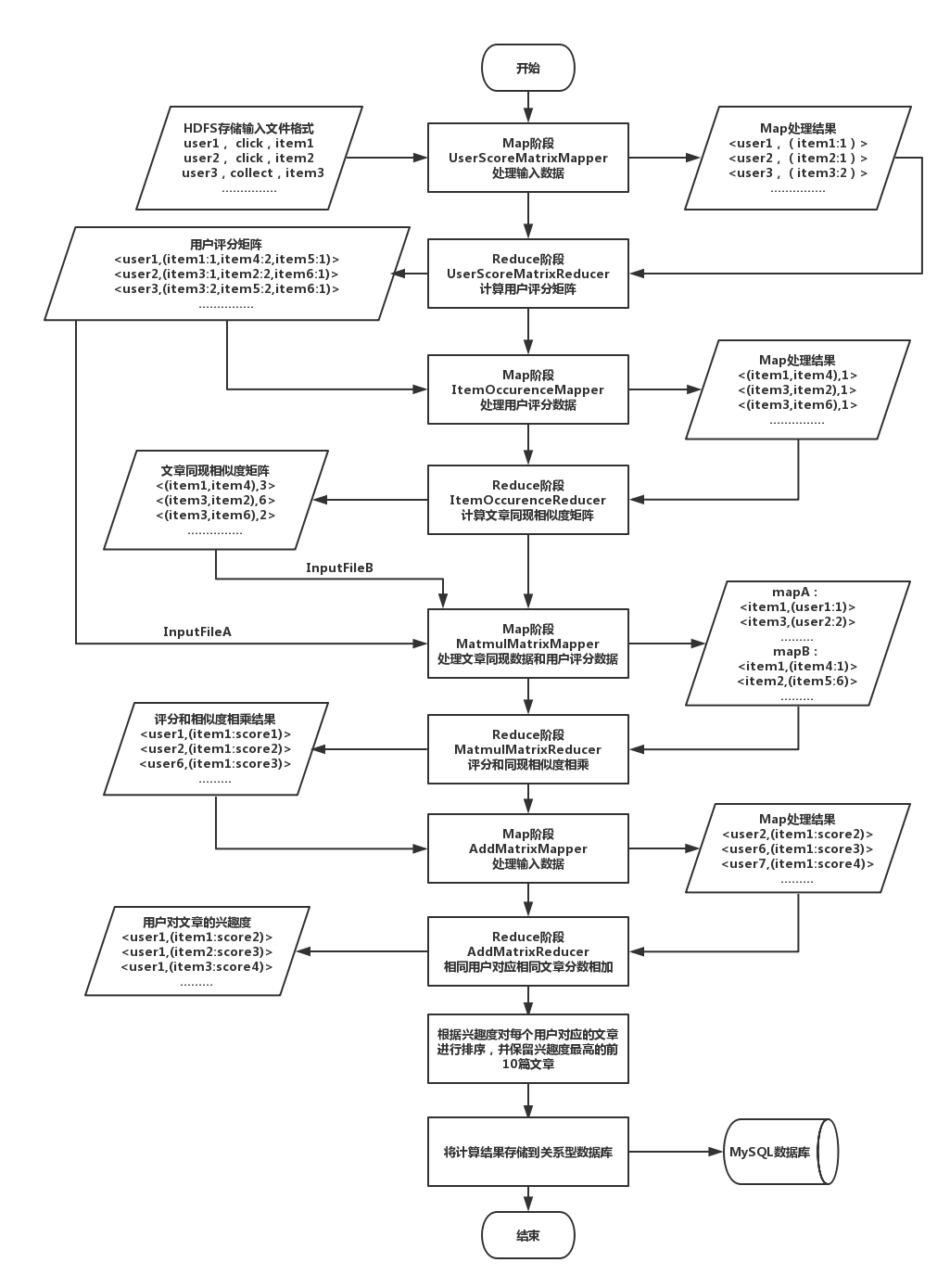


图6-4 文章数据同步时序图

Figure 6-1 Flowchart of Registered User

* + - 1. 基于关联规则的推荐

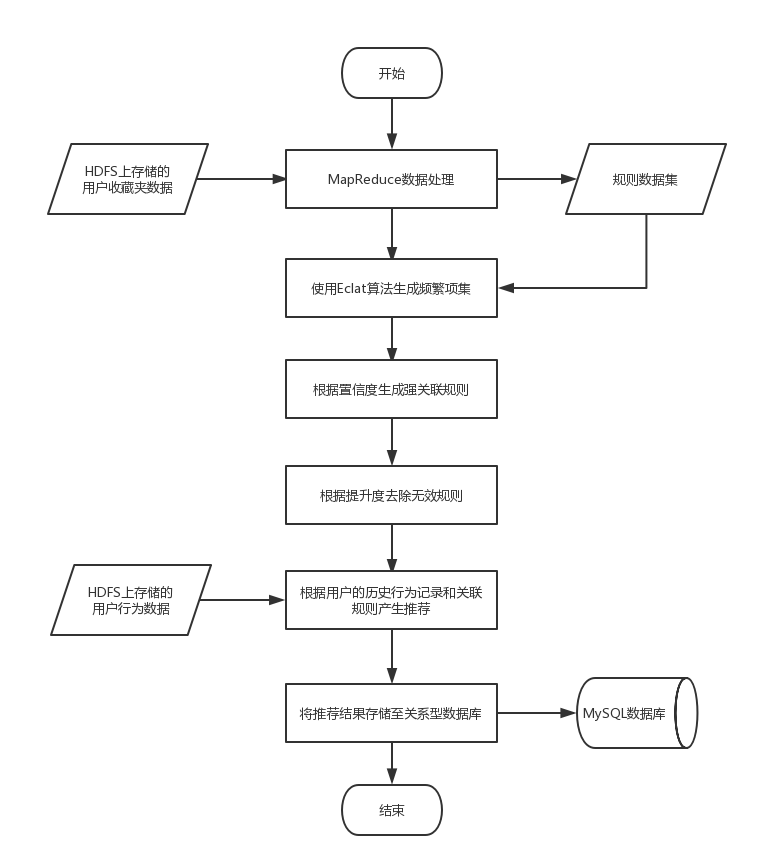
基于关联规则的推荐使用的原始数据是用户的收藏夹数据，我们将每个用户的收藏夹中的所有文章看成是一条记录，从不同用户的收藏行为中挖掘出文章之间的关联性，具体的计算过程如图6-5所示，一共分为五个步骤：第一步，将HDFS上存储的用户收藏夹数据进行预处理，使之变为规则的格式，具体的数据集格式形如<user1，（item1，item2，item3）>。第二步，使用Eclat算法生成频繁项集。第三步，通过计算置信度筛选出强关联规则。第四步，计算强关联规则的提升度进而去除无效的强关联规则。第五步，根据用户的历史行为数据和关联规则产生对用户的推荐列表并存储至关系型数据库MySQL中。

图6-5 关联规则计算流程图

Figure 6-1 Flowchart of Registered User

如图6-6所示是关联规则整个计算过程的时序图，图中清晰的展示了各个类及方法的调用过程。首先OozieTimer是使用Oozie实现的定时器，Oozie是一个工作流调度引擎，由于关联规则的计算过程是离线进行并周期性更新的，因此使用Oozie可以定时触发需要执行的任务。DataProcessMR是负责数据预处理的MapReduce程序，定时器通过调用DataProcessMR类的main方法来触发MapReduce任务，进而生成用于计算关联规则的数据集。RuleCalculate类负责实现关联规则计算的整个过程，在计算的过程中首先调用HDFSManager类中的readDataFromHDFS方法从HDFS中获取处理好的数据集以及用户行为数据，然后调用FPAlgorithm类中的eclat方法计算频繁项集，其中FPAlgorithm类中实现了多种计算频繁项集的方法，这里我们只使用eclat方法进行频繁项集的计算，具体选择eclat算法的原因我们已经在算法的设计章节介绍过了。生成频繁项集后，调用RuleCalculate类中的genStrongRule方法生成关联规则，并计算各个关联规则的置信度，然后筛选出置信度大于域值的关联规则作为强关联规则输出。下面一步是调用RuleCalculate类中的genEffectRule方法生成有效关联规则，具体的实现步骤是，计算上一步得到的强关联规则的提升度，若提升度小于等于1则证明关联规则无效，因此根据提升度的计算结果将无效关联规则去除。生成有效的强关联规则后，结合用户行为数据调用getRecommend方法为用户生成推荐文章列表，并使用关联规则的置信度作为文章的权重，举例说明生成推荐列表的具体实现过程：假设(A,B)->C是有效强关联规则，这条规则代表如果用户同时看过A文章和B文章，那么用户有很大概率会看C文章，假设经过分析用户行为数据得到某用户同时看过A和B文章，那么生成推荐结果时，我们会将C文章推荐给该用户。生成推荐结果后调用StrategyDao类中的saveData方法将推荐结果存储至MySQL关系型数据库中，至此关联规则推荐完成。

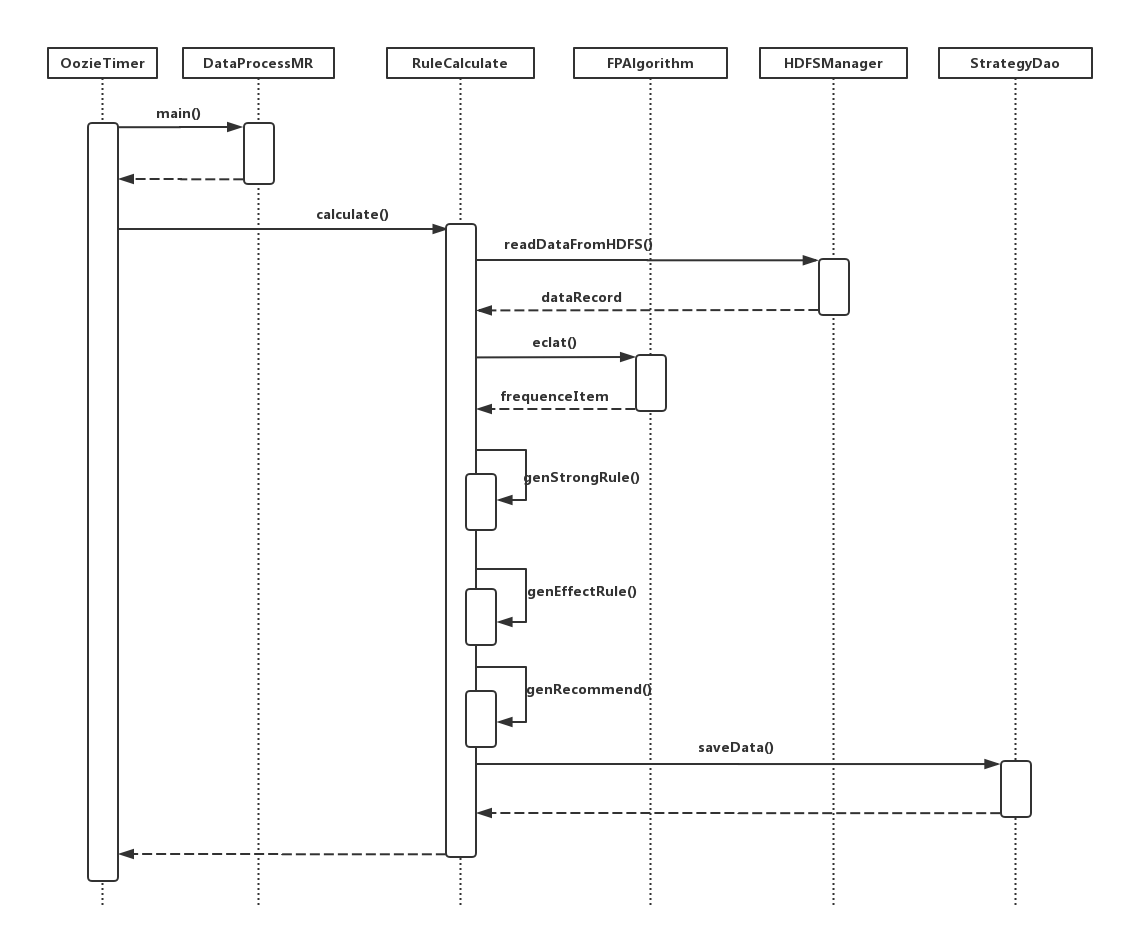


图6-6 关联规则计算流程图

Figure 6-1 Flowchart of Registered User

* + - 1. 基于内容的推荐

基于内容的推荐的计算过程需要的数据是存储在HDFS上的用户行为数据和文章关键词数据，整个计算过程分为三步：第一步，根据用户行为数据和文章关键词数据构造用户兴趣标签，第二步，计算用户兴趣标签和文章关键词的余弦相似度，第三步，根据相似度产生用户的推荐列表并存储至MySQL关系型数据库中。整个计算过程离线进行，并周期性地将离线计算的结果更新到关系型数据库中，如图6-6所示是基于内容推荐的计算流程图。

图片包含 屏幕截图

已生成高可信度的说明图6-7 关联规则计算流程图

Figure 6-1 Flowchart of Registered User

基于内容的推荐过程中需要的用户行为数据和文章关键词存储在HDFS上且数据量很大，因此次计算过程采用分布式的并行离线计算，整个产生推荐的过程由三个MapReduce组成，分别是用于统计用户对文章行为次数的MapReduce、用户构造用户兴趣标签的MapReduce、用于计算余弦相似度的MapReduce，下面将详细介绍这三个MapReduce的处理细节。

DataProcessMapper类和DataProcessReducer类组成了用于统计用户对文章行为次数的MapReduce。其中DataProcessMapper类是Map阶段，主要负责处理输入的用户行为数据，得到在一段时间内用户对文章的行为次数，我们认为收藏行为的重要程度大于点击行为，因此如果用户对文章的行为是点击，就将用户对该篇文章的行为次数设为1，如果用户对文章的行为是收藏，就将用户对该篇文章的行为次数设为2，Map阶段对输入的用户行为数据进行处理，然后以用户ID为key，以文章ID和初始兴趣度为value作为结果进行输出。DataProcessReducer类是Reduce阶段，这个阶段将使用Map阶段的输出结果作为输入，对key相同的数据项进行汇总，由于同一个用户可能会多次浏览同一篇文章，因此，通过Reduce阶段，我们可以将同一个用户对同一篇文章的多个行为次数进行累加，最终得到的是一段时间内，各个用户对各篇文章的总行为次数。

UserItemTagMapper类和UserItemTagReducer类组成了用于构建用户兴趣标签的MapReduce。其中UserItemTagMapper类是Map阶段，该阶段的输入数据由两部分组成，一部分是是上一个MapReduce得到的用户对文章的行为次数，另一部分是文章的关键词数据。由于我们要得到用户对应文章的关键词数据，因此需要将以上两部分数据在Map阶段做Join操作，具体Map join的方式是：由于文章关键词数据数据量较小，因此选择将将文章关键词数据放入分布式缓存中，在UserItemTagMapper类的setup方法中将文章的关键词数据从缓存中读取出来，并以map类型进行存储，其中setup方法是在Map阶段之前进行的，在Map阶段将具有相同文章ID的用户行为数据和存放在map类型的关键词数据进行对应连接操作，并将文章关键词的权重和对应的文章行为次数相乘得到用户对关键词的综合权重，最终Map阶段的输出以用户ID作为为key，以用户对应的兴趣标签及权重的组合作为value，将结果输出到Reduce阶段进行汇总。UserItemTagReducer类是Reduce阶段，这个阶段将用户ID相同的数据进行汇总，累计同一用户对应的各个关键词及综合权重作为用户的兴趣标签进行输出，至此用户兴趣标签构建完成。

ContentBasedMapper类和ContentBasedReducer类是用户产生推荐结果的MapReduce处理过程，这个阶段的输入由两部分组成：一部分是上一步产生的用户兴趣标签数据，另一部分是文章的关键词标签。ContentBasedMapper类是Map阶段，由于我们需要得到各个用户对各个文章的兴趣度，因此在Map阶段需要对这两部分数据做全连接操作，使用和上一步类似的Map join将用户数据和文章数据连接起来并计算对应的用户标签和文章关键词的余弦相似度，然后以用户ID和文章ID的组合作为key，以计算得出的相似度作为value进行输出。Reduce阶段将接收Map阶段的输出作为自身的输入，然后得出最终的推荐结果输出到关系型数据库MySQL中。

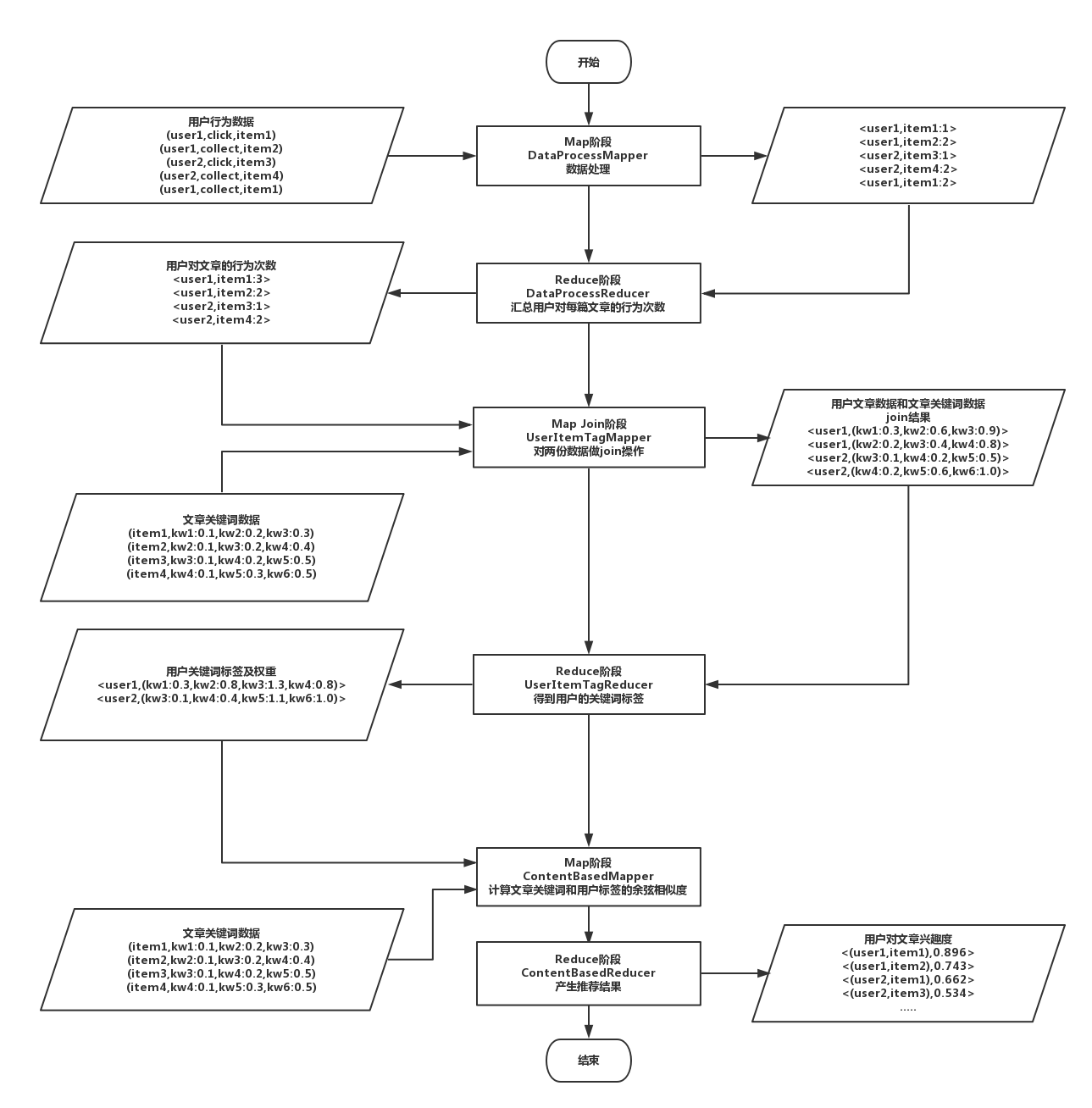
图6-8 关联规则计算流程图

Figure 6-1 Flowchart of Registered User

* + - 1. 基于车标签推荐详细设计与实现

通过对以上几种推荐策略的计算流程的详细阐述，可以发现，以上的几种推荐策略的计算过程需要的用户行为都是用户对于文章的行为数据，这样做的原因是本项目开发的推荐系统是对文章的推荐，因此大部分策略都致力于根据用户对文章的行为来挖掘用户兴趣，进而产生推荐。和以上几种策略不同的是，本节介绍的基于车标签的推荐策略不仅使用了用户对文章的行为数据，还使用了用户在汽车资讯App上产生的和车型相关的行为数据，如浏览车型详情页、收藏车型、搜索车型、车型询价等行为，使用这些行为数据的原因是本项目开发的推荐系统是服务于汽车资讯平台的，和汽车领域有紧密的联系，大部分用户使用汽车资讯App的原因也是想要获取和汽车有关的信息，因此本节介绍的基于车标签的推荐策略的实现过程中，主要的部分是挖掘用户对于车的兴趣，找到用户感兴趣的车型、品牌等信息，进而做出合理的推荐。

基于车标签的推荐主要的**实现流程**如图6-7所示。主要分为四步，分别是构建用户车型标签、用户和文章车标签扩展、根据车标签计算用户和文章的余弦相似度、根据相似度产生推荐结果并存储到关系型数据库MySQL中，下面将详细介绍这四部分的实现细节。

第一步，构建用户车型标签。这部分的实现过程中需要的数据是用户行为数据和文章相关车型数据。首先根据每一条用户行为数据刻画用户在本次行为中对某种车型的兴趣度，然后根据一段时间内的用户行为数据得出用户在这段时间内对多种车型的兴趣度。其中，用户的行为分为操作车型的行为以及操作文章的行为，对于用户操作车型的行为数据来说，直接将行为数据中对应的车型ID赋予用户，而对于用户操作文章的行为数据来说，我们将和这篇文章相关的车型ID赋予用户，根据不同的行为类型我们会对车型标签产生不同的权重值，比如从收藏车型的行为中提取的车型标签的权重会大于从点击车型的行为中提取的车型标签权重。按照以上策略将一段时间内的用户行为数据进行统计和计算，最终可以得到每个用户对每中车型的累计权重值，用户车标签也构建完成。

第二步，用户和文章车标签扩展。第一步用户车标签的构建过程中只产生了车型标签及其对应权重，但是对于汽车信息来说，包含的维度不仅有车型，还有品牌、厂商、车身等。为了挖掘用户对于其他维度的兴趣，这一步将根据车型标签对其他维度的车标签进行扩展。扩展策略如下：如果用户对应的多个车型标签都属于同一个品牌，那么将该品牌标签赋予该用户，且品牌的权重是多个车型标签的平均值，如果用户对应的多个车型标签不属于同一个品牌，则不产生车型标签。车身、厂商、级别、国别等相关车标签的扩展方法同品牌一样。文章的车标签扩展策略和用户车标签扩展策略相同。这一步完成后产生用户对应的车型、车身、品牌、厂商、级别、国别等标签及权重。

第三步，根据车标签计算用户和文章的余弦相似度。这一步的输入数据是用户对应车标签及权重和文章对应车标签及权重，相似度的计算方法采用余弦相似度，至此用户和文章就通过车标签联系起来。

第四步，根据相似度对用户对应的文章列表进行排序，选择相似度较高的文章作为推荐结果，并存储到关系型数据库MySQL中。

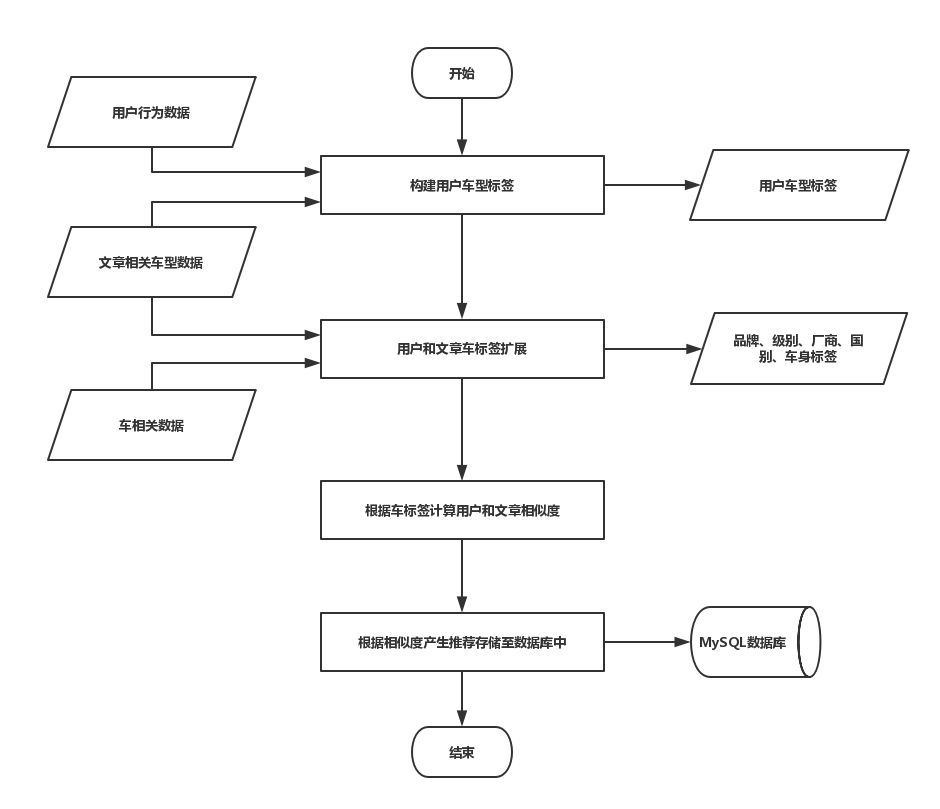
图6-6 关联规则计算流程图

Figure 6-1 Flowchart of Registered User

* + 1. 相关文章推荐的实现
  1. 点击率预估
  2. 推荐后台管理