

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对用户做出推荐，最大程度地提升推荐效果。

**关键词：**

ABSTRACT

**KEYWORDS：**

目录

1. 引言

本章首先对课题的研究背景，学籍信息管理信息系统的研究现状、课题的研究目的和意义、本文的研究内容等进行了阐述和分析。并在本章最后给出了论文的行文结构。

* 1. 课题背景与意义

北京无限讯奇信息技术有限公司（以下简称“无限讯奇”）是中国移动12580业务独家合作伙伴，负责开发和建设中国移动12580业务核心系统、内容采编与发布、广告销售、用户拓展与服务等运营工作[1]。

阿迪沙发北京无限讯奇信息技术有限公司（以下简称“无限讯奇”）是中国移动12580业务独家合作伙伴，负责开发和建设中国移动12580业务核心系统、内容采编与发布、广告销售、用户拓展与服务等运营工作[1]。

* 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。

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。

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

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. 论文组织结构

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. 本章小结

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。

* + 1. 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。

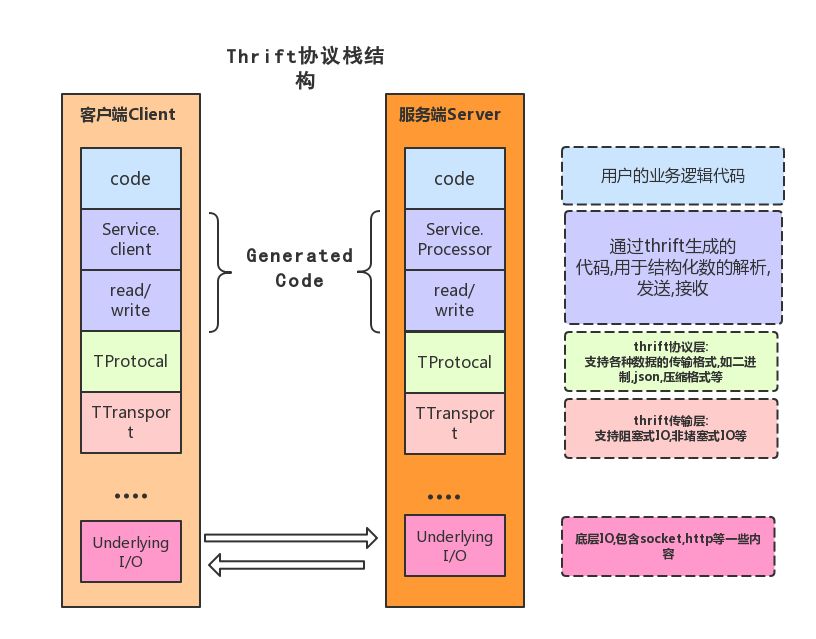
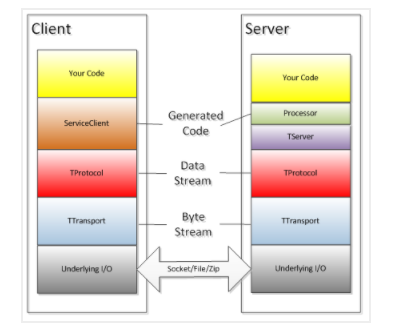


图4-2 系统流程图

Figure 4-2 System Flow Diagram



* + 1. 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。

* + 1. Django Web架构概述
    2. 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. 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
   1. 系统需求概述
   2. 系统功能性需求分析

文章数据同步与用户标签提取模块

基于多种推荐算法的混合推荐引擎模块

* + - 1. 相关推荐
      2. 个性化推荐
      3. 推荐解释的生成
      4. 推荐热点候补
      5. 已推荐过滤及新颖性排名

推荐点击预估模块

* + - 1. 用户兴趣建模
      2. 文章特征计算
      3. 特征工程
      4. 模型训练

推荐效果可视化模块

标签系统

* 1. 系统非功能性需求分析
  2. 本章小结

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。推荐点击预估模块主要通过机器学习的方法训练点击预估模型，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。

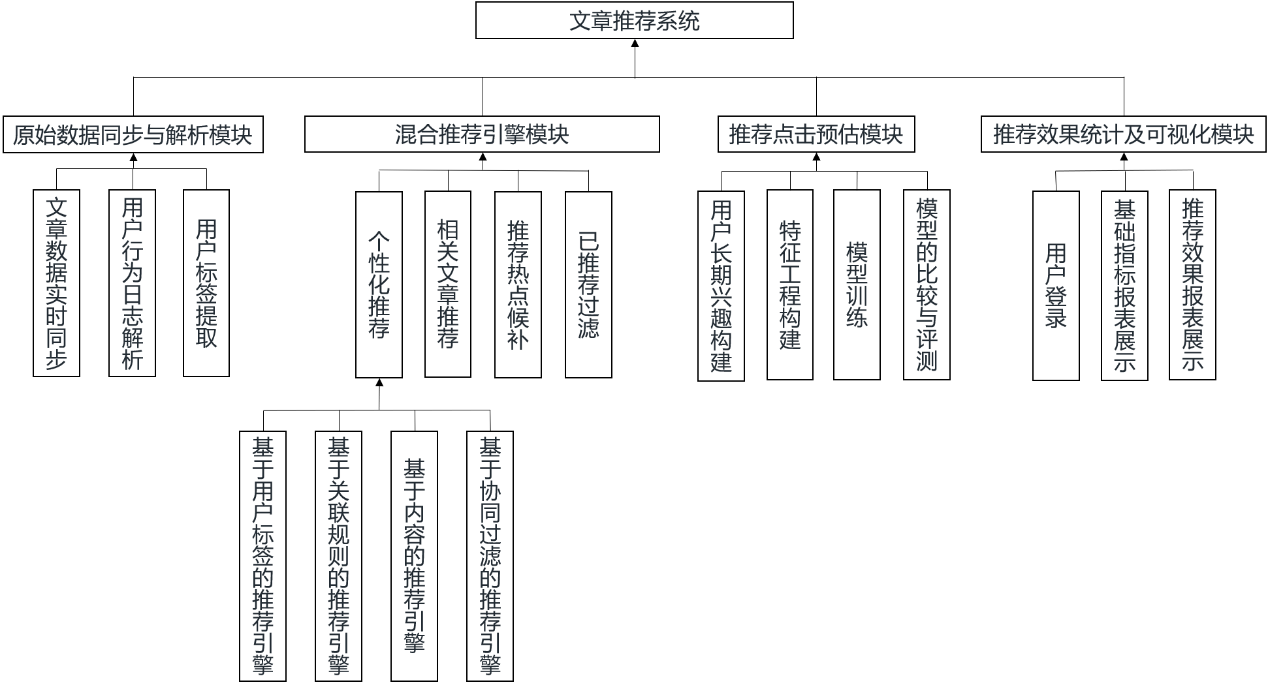
图4-1 系统模块设计图

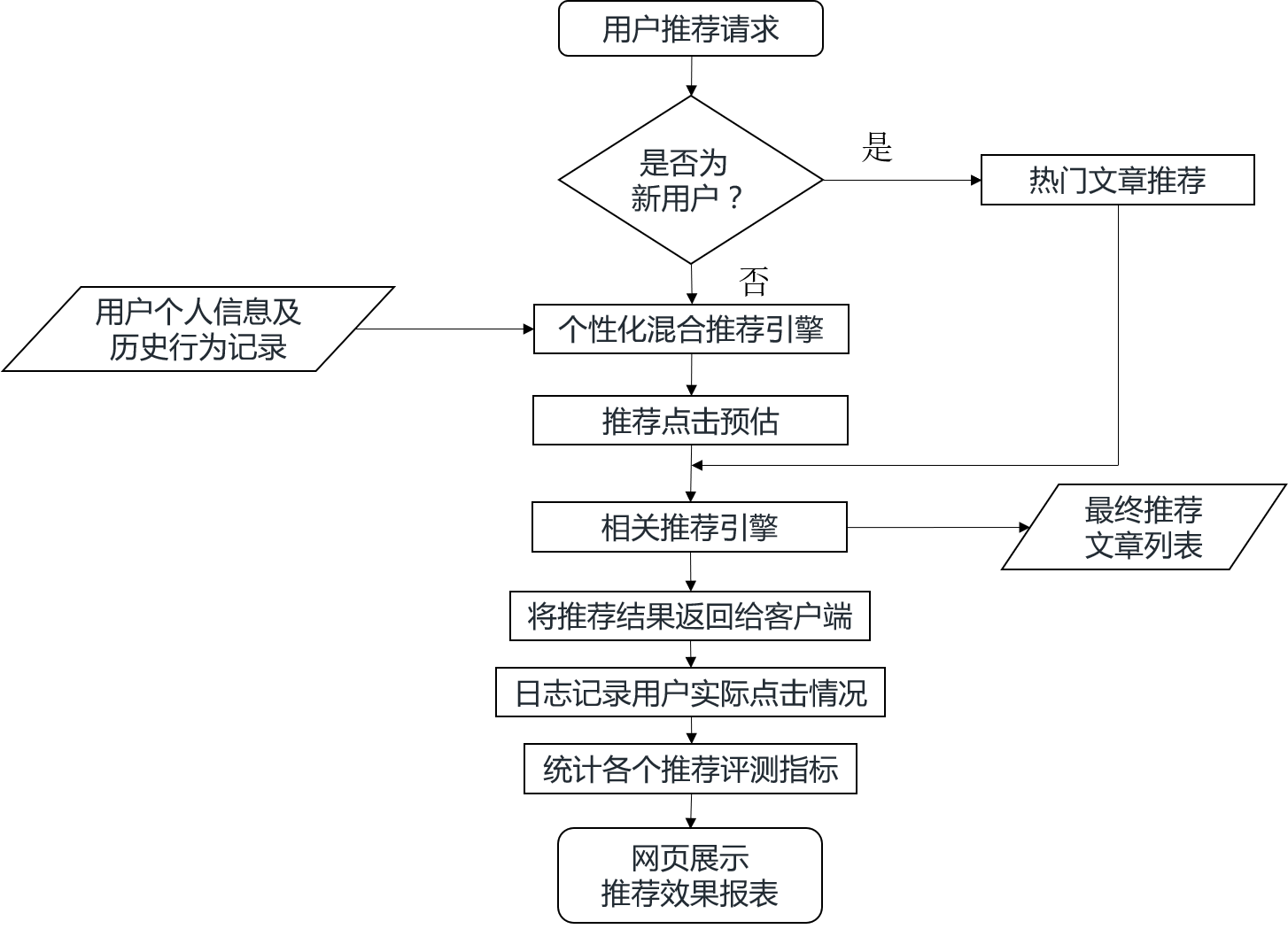
Figure 4-1 System Module Diagram

* 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。

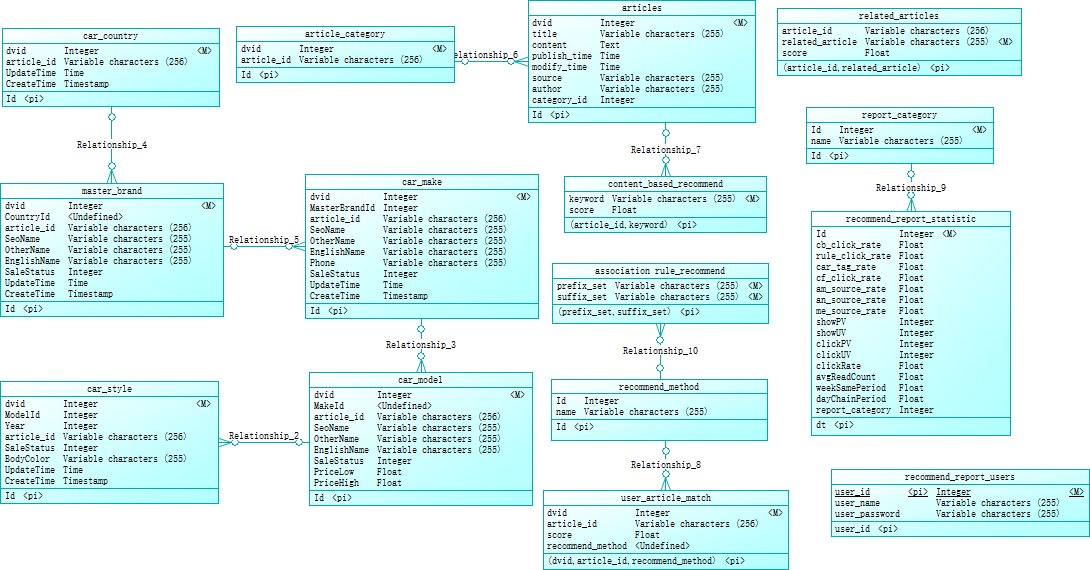


图4-3 E-R图

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。

* + 1. 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**

* + 1. 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。

* + 1. 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。

* + 1. 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。

* + 1. 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。**

* + 1. 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。

* + 1. 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。

* + 1. 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。

* + 1. 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MapReduce分布式离线计算，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？？？？？？？？？？？？？？）

* + 1. 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。

* + 1. 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。

* + 1. 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。

* + 1. 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。

* + 1. 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. 点击预估模型的设计
     1. XGBoost算法设（模型使用的什么参数）
     2. GBDT算法设计
     3. XGBoost算法与GBDT算法比较
     4. 时间复杂度
     5. 空间复杂度
  2. 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，**