SJ0202-2016 版

分类号：TP312　　U D C：D10621-408-(2016)2842-0

密　级：公 开　　编 号：2012124076

成 都 信 息 工 程 大 学
学 位 论 文

基于数据分析的物联网便民超市系统设计与实现

论 文 作 者 姓 名：蔡行

申 请 学 位 专 业：物联网工程

申 请 学 位 类 别：工学学士

指导教师姓名(职称)：石磊（讲师）

论 文 提 交 日 期：2016 年 06 月 17 日

基于数据分析的物联网便民超市系统设计与实现

摘　要

随着物联网、互联网的概念逐渐被人们熟知，我们的生活已经逐渐被互联网所渗透。适应于“互联网+”的浪潮下，我们提出基于数据分析的物联网便民超市系统。该系统将传统行业与互联网行业相融合，实现互联网的移动，造就更多的创新，以此给我们的生活带来更多的方便。

本系统实现的主要功能是：对于超市管理员，系统对其提供更科学的供货、销量预测等管理计划；对于消费者，系统提供线上/线下购物两种消费方式。本系统利用机器学习算法中的关联规则算法和 RFID 技术来分析用户购买习惯以及实现商品数量管控。结合 python 数据科学计算实现销量和库存的统计，同时，利用机器学习算法中基于协同过滤的推荐算法对每个线上用户可能喜欢的商品进行推荐。该系统不仅为超市管理员和消费者提供便利，而且较大程度促进了超市的销量。

关键词：数据分析；关联规则；协同过滤；预测；推荐

Design and Implementation of the IoT Convenient Supermarkets System Based on Data Analyzing

Abstract

As the concept of the Internet and IoT Technology is gradually been well known, our life has potentially been penetrated by the Internet. Under the huge influence of the "Internet +", we proposed the IoT convenient supermarkets system based on data analyzing. The system combines the traditional industries with the Internet industry, and utilizes the mobile Internet to be more creative, bringing us more convenience and innovation in our daily life.

The main function of the system is: providing more scientific supply, sales forecasting management plan for the administrator of the supermarket; and providing both online/offline shopping for customers. The system used the association rules algorithm of machine learning and the RFID tech to analyze consumer's potential purchasing habits and effectively control goods' supply and needs. Combining scientific data computing of Python enables achieving sales and stock statistics, and Algorithms based on collaborative filtering of machine learning helps recommending favorable commodities for online users. Not only does the system provide more facilities for supermarket administrator and consumers, but also greatly promoted the supermarket sales.

Key words: data analyzing; association rules algorithm; algorithm based on collaborative filtering; forecasting; recommending

目　录

论文总页数：31 页

1 引言

1.1 课题背景

近几年关于“互联网+”，“物联网”，“智慧生活”等概念逐渐渗透到人们的生活中，尤其是以微信为代表的即时通讯，支付宝为代表的电子支付，淘宝为代表的购物正逐渐影响并改变着我们的生活方式。所谓“智慧”、“互联网+”，不仅仅是互联网的移动与泛在，以及与传统行业的融合及应用，更加入了无所不在的计算、数据、知识，造就了无所不在的创新，也引领了创新驱动发展的“新常态”。而物联网便民超市则是处于这样的创新环境下将互联网与传统零售行业的结合，来给我们的生活带来更多实际的方便。

1.2 国内外研究现状

目前国内外智慧超市的建设也不断加强，但是现在的智慧超市都是关注在对大型商场的布局，如国内众多大型商场或者商圈都在提档升级打造智慧商圈，例如对于商圈停车场打造智慧停车场，实现自动寻空位及导航的功能；对于商场提供免费 WIFI 并开发 APP 进行商圈室内导航、信息搜索等功能，或者是热衷于使用 RFID 技术实现物品定位，自动结算和扣费等智慧功能的研究，但是这种布局目前来讲需要对设备和整个超市的货物设备的摆放都有要求，部署起来规模较大，成本也较高，而且这种智慧超市的“智慧”只能是在实体店购物时才能享受到。

1.3 本课题研究的意义

相比于大型商场或者商圈的智慧建设，对于小规模便民超市与互联网接入的关注度还不够高，当前正处于互联网高速发展的阶段，同时物联网，大数据，云计算等概念也被越来越多的人们所熟知和重视，处于这样的时代背景下，有必要将便民超市这类小规模零售业与互联网结合起来，通过获取有价值的数据，创造更多的盈利机会。而且由于目前的小区或社区超市、便利店等，都很少有能够通过在线购物并享受送货上门的服务。

由于现在的人们越来越热衷于网购，究其原因是网购物品的价格比实体店便宜，再加上现在的移动互联网技术的迅速发展，用户可以在手机上下单付款，以及物流行业的迅速发展为网购创造了良好的基础，也的确为人们的购物生活带来了更多的新鲜和方便。所以“基于数据分析的物联网便民超市系统”则是建立在人们热衷于互联网带来的便利这一条件上，而且现在中国已经快速进入数字、网络化时代，所以未来将小规模零售业接入互联网也将是一种趋势，并实现真正的互联网全覆盖，实现真正的物联互通，智慧生活。

1.4 技术背景

要实现该系统，需要用到以下几个主要技术：

1）RFID 技术。由于本系统需要对商品进行管理，因此使用 RFID(全称为 Radio Frequency Identification)技术来实现。其中文名字即射频识别技术，它的原理是利用射频信号，通过空间耦合（电磁场或者交变磁场）来实现无接触的信息传递，并通过所传递的信息以此达到自动识别的目的。

由于 RFID 标签成本便宜，且相对于传统的二维码、条形码等信息存储方式，RFID 更具有抗污染、存储量大、信息更完整可靠等特点。一套 RFID 主要由阅读器、天线和标签三大件组成，其中标签是由微型天线、耦合元件、芯片组成的，每个标签内部都存有唯一的电子编码。RFID 标签的数据存储方式可分为三种：铁电随机存取存储器（FRAM）、电可擦可编程只读存储器（EEPROM）、静态随机存取存储器（SRAM）。同时标签根据是否内置电源，也可以分为三种类型：主动式标签、半主动标签以及被动式标签[1]。

2）MongoDB 数据库。MongoDB 是以 BSON 文档格式存储数据的非关系型数据库，它没有外键、事务、表、SQL 等概念。在 MongoDB 中一个文档代表了一个存储单元，在 RDBMS 中存储单元被称为行，不过文档中包含的信息要比行多，因为它们可以存储更复杂的信息如列表、词典甚至是词典的列表。在 MongoDB 中要求每个文档必须有唯一标识符（_id）。文档由键和值组成，且键和值总是成对出现的[2]。

要在本系统中使用 MongoDB，还需要安装 PyMongo 和 PHP 的 Mongo 驱动来使我们的程序能够连接上数据库。

3）Python 语言。Python 是一种面向对象、直译式的计算机程序语言。它包含了一组功能完备的标准库，能够轻松完成很多常见的任务，而且它语法简单，它使用缩进来定义语句块，这与其它大多数程序设计语言使用大括号不一样。近年来由于 Python 不断改良的库（最主要是 pandas），使其成为数据处理任务的优先选择。结合其在通用编程方面的强大实力，我们完全可以只使用 Python 这一种语言，去构建以数据为中心的应用程序[3]。

因此我们在系统中对数据分析以及对实现后面提到的机器学习算法都采用 Python 作为编程语言，主要原因是：（1）Python 的语法清晰；（2）易于操作纯文本文件；（3）使用广泛，存在大量的开发文档。

4）PHP 网站开发。php 全名超文本预处理器，是一种开源的脚本语言，大部分的 web 服务器都支持 php。php 网站开发采用的是 html 中内嵌 php 代码的方式，这种编写方式比起 cgi 来说更方便，虽然 php 不能像.net 那样提供大量的控件和库，但是 php 的优点在于轻便、跨平台。更重要的是 php 添加驱动非常方便。本系统使用 MongoDB 作为数据库，只需要在 github 中下载 mongo-php-driver 驱动编译后，通过模块添加的方式添加到 php 中，就可以对

mongodb 进行操作。

5）数据挖掘技术在商务智能的应用已成为各行业的必然趋势，系统在运行过程中会累积很多数据，我们需要的就是从这些数据中找到有用的信息，来对我们的运营作出规划，所以需要用到机器学习的相关技术。所谓机器学习，就是把无序的数据转换成有用的信息。机器学习的主要任务就是分类、回归。分类和回归都归属于监督学习，所谓监督学习，就是指这类算法必须知道预测什么，即目标变量的分类信息。与监督学习相对应的是无监督学习，无监督学习的数据没有类别信息，也不会给定目标值。在无监督学习中，将数据集合分成由类似的对象组成的多个类的过程被称为聚类。

2 需求分析

科技的高速发展，的确给我们的生活带来了很多颠覆式的改变。比如 QQ、微信等社交软件给我们的交流带来了方便，微博、知乎等社区工具带领我们实时了解并分享世界各地不同的信息，淘宝、支付宝、微信红包等也改变了我们以往的消费模式和观念，不得不说，这些都是互联网高速发展的成果，它使得我们每个人能获取的信息越来越对称，并且随着人工智能、机器学习以及深度神经网络学习的发展，我们的生活将更加高效快捷和智能。

当我们使用淘宝、京东或者亚马逊在线购物时，可以选择多家商店，多个种类，并且当我们进入这些网站时，我们可以看见系统给我们推荐的商品，这样方便我么快速定位到自己所想要的商品。但是这样的购物并不能实时满足实时性（比如当我们在家做饭发现没米了，这时候再选择网购就不太现实），因此现在的电商都是基于大平台大用户群，而忽略了分散的小平台小群体。

基于数据分析的物联网便民超市系统便是针对分散的小平台小群体开发的，是为了给人们的日常生活带来更多方便以及更多智能化体验。本系统需要完成的主要功能是基于“互联网+”体现出来的，且本系统面向两类用户群体：

面向超市管理员实现的主要功能为:

（1）商品库存获取

（2）查看热销产品

（3）查看被购买商品的关联性

（4）订单管理

面向消费者实现的主要功能为：

（1）网页端在线购物

（2）查询订单

（3）查看推荐商品

基于数据分析的物联网便民超市系统的主要作用是将互联网融入到传统行业，这也正好体现了“互联网+”的概念，同时“互联网+”中的移动互联网也受到人们的追捧，因此站在消费者的角度，开发一个 020 的线上购物平台可以迎合我们当前所热衷的网购消费模式，并且能够获取推荐以便更快找到所喜欢的商品。

有了这样的系统和平台，就会产生大量的数据，而如今数据挖掘也是一门发展很快的技术，包括人工智能、机器学习的本质也都是数据挖掘。所以本系统最关键的就是对于数据的处理，也是最有价值的。站在超市管理员的角度，通过对消费者的购物数据进行分析，可以得出当前时间段(单位：月)的一些热销商品，以及商品被购买的关联性，以此来对供货量，价格定价或做优惠活动促销作出判断和抉择。

3 系统设计

3.1 系统整体结构

依据以上研究思路和需求概述，提出如下系统设计方案，包括系统软硬件平台的选择，系统采用的架构，系统整体结构图如图 3-1 所示。

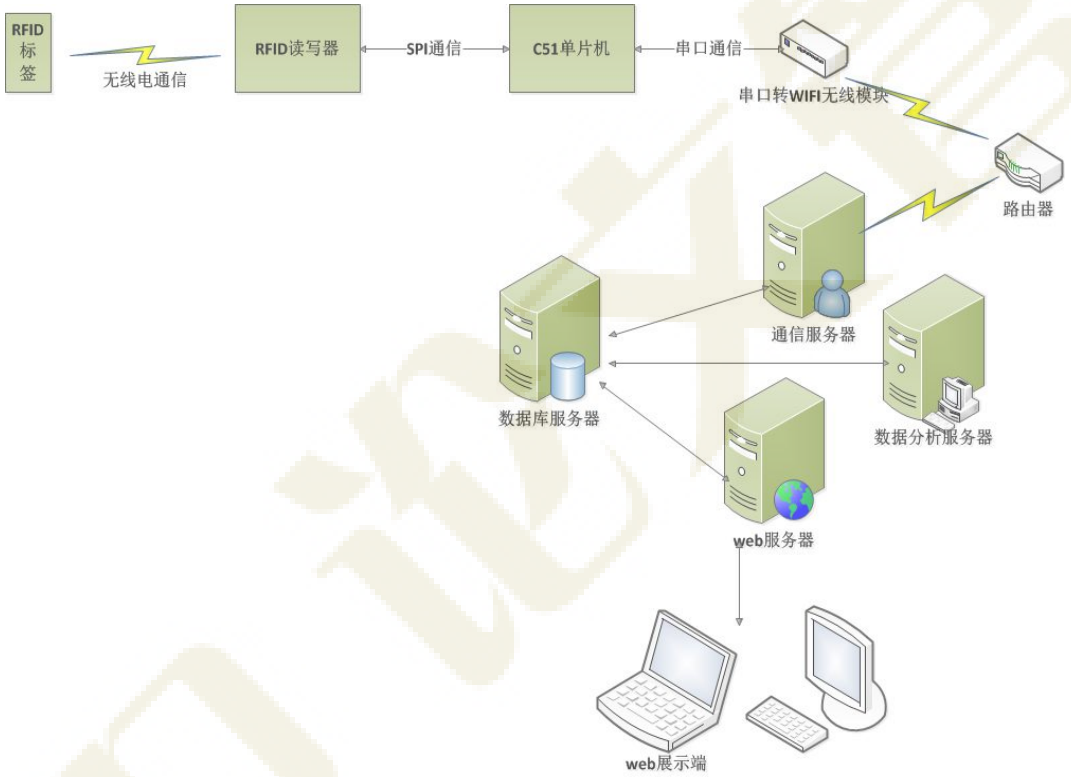

图 3-1 系统整体结构图

本系统主要分为了硬件平台端的数据采集与通信，服务器端的后台处理以及网页端的展示三部分。硬件平台端的数据采集主要是通过对 RFID 标签的读写，RFID 标签用于商品上，相同的商品具有相同的 ID，因此通过统计相同标

签 ID 可以对商品进行库存量统计。硬件平台端的数据通信包括读写器与 51 单片机之间的通信，51 单片机与网络通信模块之间的通信以及网络模块与通信服务器之间的通信。服务器端主要包括通信服务器、数据分析服务器、web 服务器和数据库服务器。通信服务器主要功能是接收并处理从硬件端传送过来的数据，并进行处理入库，分析服务器主要完成数据分析的功能并将结果入库，web 服务器对网页端数据进行处理和交互。网页端则主要面向超市管理员和消费者提供不同功能并能对数据分析的结果进行展示。该系统主要功能集中在服务器端，后面会进行详细介绍。

3.2 硬件平台数据通信过程设计

硬件平台数据通信过程设计如图 3-2 所示。

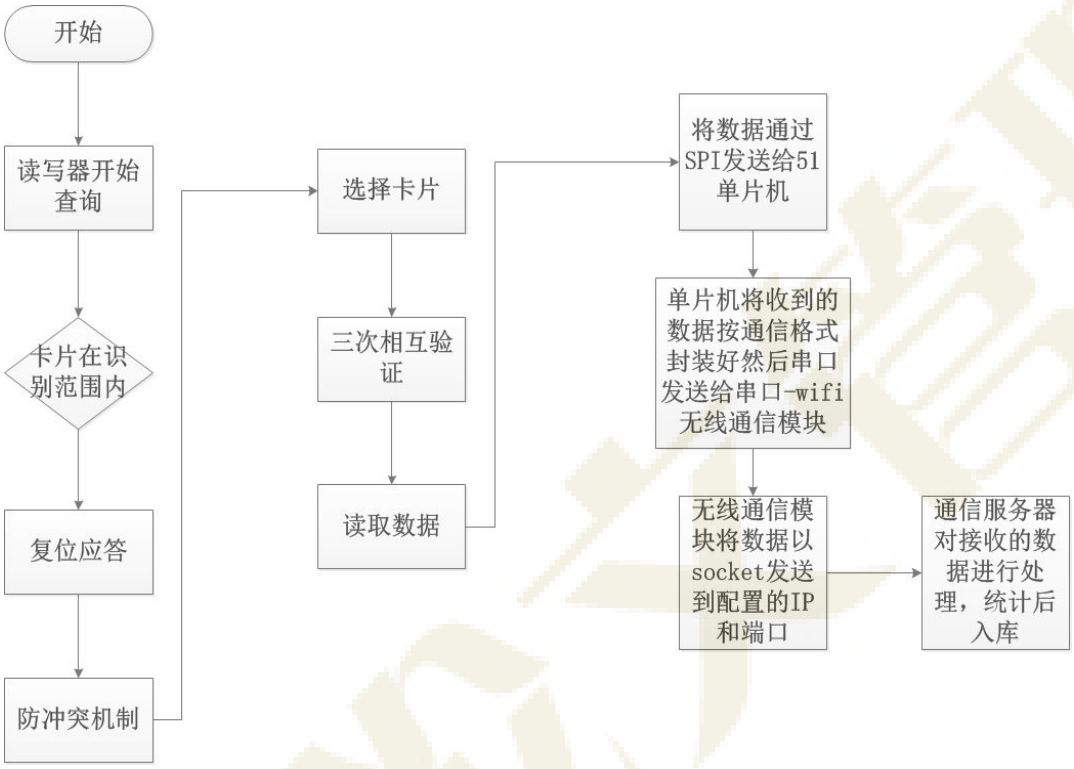

图 3-2 硬件平台数据通信过程设计

RFID 选用 MF_RC522 的读写器以及卡片，读写器通过 RFID 识别和通信等方法，读取在识别范围内的 RFID 标签的数据，然后读卡器通过十分稳定的 SPI 协议将数据发送给 51 单片机，51 单片机收到后按照通信服务器设计好的通信格式（见 3.4）将数据进行封装，然后通过串口将数据发送到透明传输的无线通信模块（该模块实现的是串口转 WiFi，通过其串口的数据都会被原封不动的通过中继路由器无线发送到指定的 IP 的服务器），无线通信模块再通过路由器将数据中转到指定 IP 的服务器[4]。

3.3 数据库表设计

由于采用 MongoDB 非关系型数据库，所以不涉及主/外键等。

表 3-1：TAccount 数据库表

作用：该表主要用于记录消费者的一些信息，包括消费者 ID，该 ID 是注册时由系统分配的，其他信息都是由消费者注册时填写的。

表 3-1 TAccount（消费者注册表）

字段名	英文编码	类型	备注
消费者 ID	USER_ID	char 10	
消费者姓名	USER_NAME	char 20	
密码	PASSWORD	char 10	
用户性别	USER_SEX	char 2	m/f
用户年龄	USER_AGE	int	

表 3-2：TOrder_当前年月数据库表

作用：存储当前年月下的所有消费者的购物订单。已购商品 ID 和数量采用词典形式存储，可以直观的反应该次所购买的所有商品和数量之间的关系。

表 3-2 TOrder_当前年月（当月订单表）

字段名	英文编码	类型	备注
消费者 ID	USER_ID	char 10	表 1 里有对应 ID 时有效
已购商品 ID 及数量	GOODS_AMOUNT	document	{ID0:10,ID2:3,ID7:2...}
购买日期	PURCHASE_DATE	int 20	

表 3-3 TGoods 数据库表

作用：存储商品的 ID 及对应的商品名称、属性、标签、图片路径信息。其中商品 ID、名称、属性、标签都是按一定规则指定的。

表 3-3 TGoods（商品信息表）

字段名	英文编码	类型	备注
商品 ID	GOODS_ID	int	
商品名称	GOODS_NAME	char 20	
商品属性	GOODS_ATTRIBUTES	char 10	
商品标签	GOODS_TAG	char	
商品图片路径	PIC_PATH	char	对应的图片位置

表 3-4：TInventory_当前年月数据库表

作用：该表用于对商品的数量进行统计，当硬件端执行一次标签数据统计并通过通信服务器处理后，就将对该表进行更新。

表 3-4　TInventory_当前年月（库存表）

字段名	英文编码	类型	备注
商品 ID	GOODS_ID	int	表 3 里有对应 ID 时有效
商品数量统计时间	LAST_TIME	int	
商品统计数量	LASTTIME_AMOUNT	int	

表 3-5：TGoods_Price 数据库表

作用：帮助管理员对各个商品的价格进性管理以及将价格与对应的商品关联进行展示。

表 3-5 TGoods_Price（商品价格表）

字段名	英文编码	类型	备注
商品 ID	GOODS_ID	char 10	表 3 里有对应 ID 时有效
商品进价	GOODS_PRICE	float	
商品当前售价	SALE_PRICE	float	

表 3-6:TApriori 数据库表

作用：存储使用 apriori 算法分析的结果并用于页面端展示

表 3-6 TApriori（apriori 算法分析结果）

字段名	英文编码	类型	备注
关联商品 1	GOODS_LISTA	list 列表	例：[1001001,1010012]
关联商品 2	GOODS_LISTB	list 列表	例：[1010023]
置信度值	CONF	float	
消费者 ID	USER_ID	char	

表 3-7：TRecommend 数据库表

作用：存储协同过滤算法分析的结果并用于页面端展示

表 3-7 TRecommend（推荐表）

字段名	英文编码	类型	备注
推荐列表	GOODS_LISTA	dict	例：{‘1010001’:4.5,’1010024’:4.3}
消费者 ID	USER_ID	char	

表 3-8：TSaleAmount 数据库表

作用：存储销量统计结果并用于页面端展示

表 3-8 TSaleAmount（销量统计结果）

字段名	英文编码	类型	备注
商品 ID	GOODS_ID	int	
销售量	SALED_AMOUNT	int	
统计时间	COUNT_TIME	int 20	

由于通信服务器、数据分析服务器都是采用 python 语言开发并且需要与数据库进行交互，对于 Python 的 mongoDB 接口定义如下：

数据库连接的类定义:

class MongoConnector(object):

主要接口定义：

1）def　DAInsert(table, data, type, safe_on = False):

接口用于插入数据

输入：

table：str 类型，表名

data：dict 类型，插入数据

type：DEVIDE_TYPE 类型，设备类型

safe_on：bool 类型，安全插入标志，默认 False

返回值：

tag：opcode 类型，>=-2 成功，-2 失败

2）def　DAQuery(table, con, sort, field, skip, limit, type):

接口用于数据查询

输入：

table：str 类型，表名

con：dict 类型，查询条件

sort：list 类型，list 元素为 dict 类型，排序规则

field：dict 类型，查询列

skip：int 类型，起始记录行

limit：int 类型，返回记录最大数

type：DEVIDE_TYPE 类型，设备类型

返回值：

tag：opcode 类型，>=-2 成功，-2 失败

cursor：dict 类型，结果集

3）def　DAUpdate(table, con, value, type):

接口用于更新数据库

输入：

table：str 类型，表名

con：dict 类型，更新条件

value：dict 类型，更新值

type：DEVIDE_TYPE 类型，设备类型

返回值：

tag：opcode 类型，>=-2 成功，-2 失败

3.4 通信服务器设计

通信服务器的处理逻辑设计如图 3-3 所示。

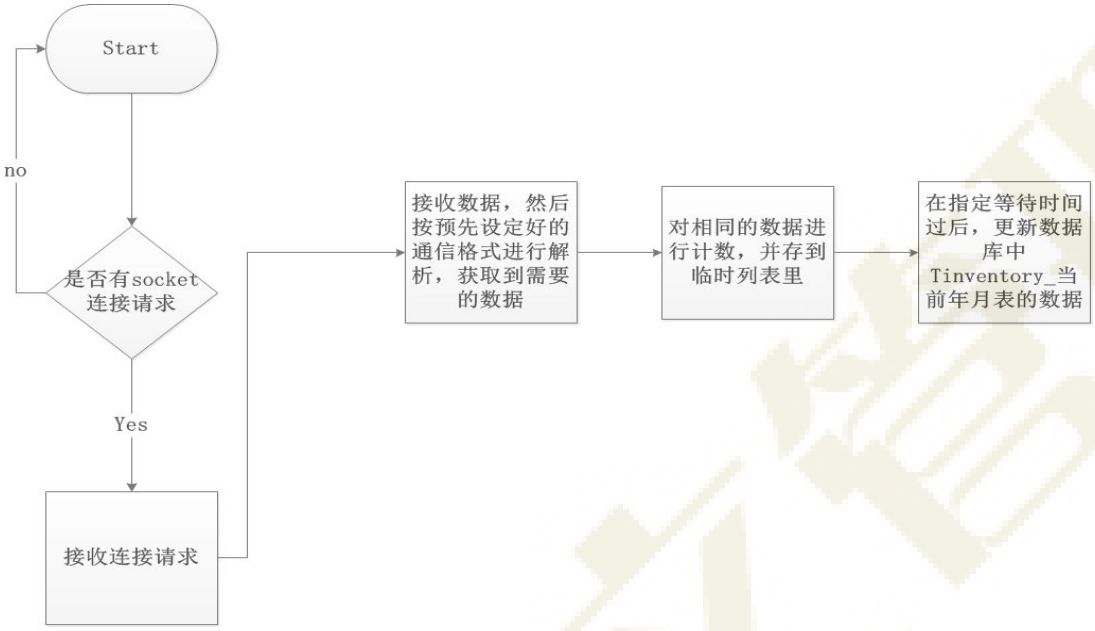

图 3-3 通信服务器逻辑处理

通信服务器的主要功能就是通过建立 socket 通信，接收从无线通信模块发送过来的数据，并对数据按预先定义的通信格式进行解析，获取对应的商品 ID 的数据，然后对商品 ID 进行计数。然后再指定时间内更新数据库的 TInventory_当前年月表。

通信服务器要与无线模块之间进行通信，则需要按照一定的格式来进行，因此将通信服务器与无线模块之间的通信格式定义为：“begin#RFIDDATA#要发送的数据#end”。

3.5 Web 服务器和网页设计

B/S 架构全称为“Browser/Server”，即浏览器/服务器架构，是 WEB 兴起后的一种网络结构模式，统一了客户端然后将系统实现功能的核心放在服务器上，简化了系统的开发、维护、使用。客户机上只需要安装有浏览器，服务器安装数据库（包括关系型或非关系型），浏览器通过 Web Server 同数据库进行数据交互。本系统 web 的功能是提供用户进行网上购物，并将数据分析的结果

以及推荐结果进行展示。

由于 web 系统实现的是购物网站，因此需要用户先进行注册/登录。

主页：主页分为两大部分，top 和 main。top 部分主要是实现登录页面跳转链接，商品搜索框等工具。main 部分显示商品列表、点击商品查看详细信息并将商品加入购物车等。

在 Web 系统中，商品是最重要的部分，如何在主页中展示出所有的商品，是网站设计的一个重点。由于 php 是一种内嵌式的脚本语言，所以我们可以根据数据库中的商品动态生成主页。网站另一个重点就是 php 代码如何响应用户的操作，也就是 html 代码如何调用 php 代码。本网站设计中，我们用到了三种不同的技术实现 html 代码调用 php 代码。第一、通过 form 表单进行调用，当用户点击提交按钮的时候，浏览器会自动生成 post 或 get 参数，并调用 action 中的 php 代码；第二、通过 js 间接调用，与 form 表单的方式相比，这种方式灵活性更高，但是提交参数等需要自己获得；第三、ajax 方式，上面两种方式在提交过后，页面都会重新刷新，每次按键都需要刷新一次页面会严重的降低用户体验，但是使用 ajax 结合 js 则实现了只刷新修改了的地方，从而大大的提高用户体验度。

Web 端主要面向超市管理员和消费者两类用户提供不同功能，具体如图 3-4 所示：

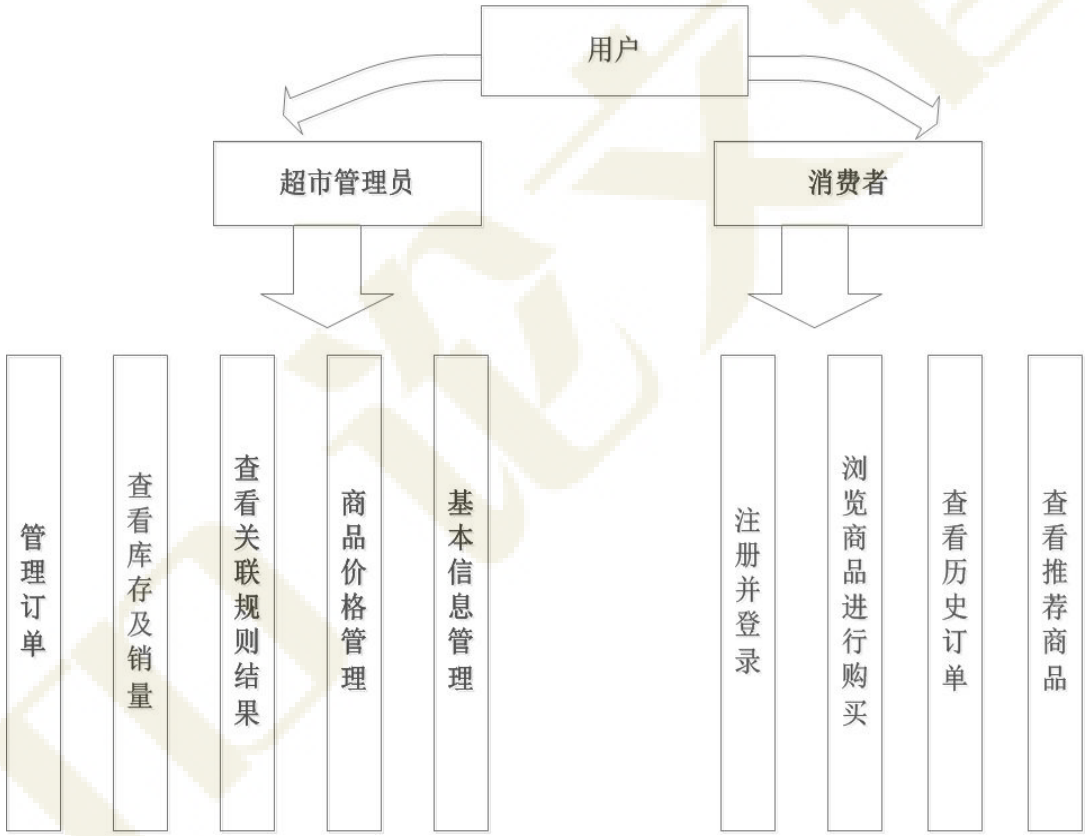

图 3-4 web 网页端具体功能

3.6 数据分析服务器设计

数据分析服务器，顾名思义就是对数据分析并得出我们想要的结果。

从数据库中 TOrder_当前年月表中我们可以获取到消费者 ID、该消费者此次购买的商品 ID 以及每个商品所购买的数量，从 TInventory_当前年月表获取商品库存量及入库时间。通过这些已有的数据，我们希望能够从中挖掘出哪些商品是被经常在一起购买的，这可以帮助超市管理员了解消费者的购买行为，通过这些行为分析的结果可以作为超市管理员在制定商品定价、市场促销、存货管理等计划时的科学依据。这种通过从大规模数据中寻找物品之间的隐含关系被称作关联规则学习或者关联分析。由于在海量的数据里面寻找物品的不同组合是很复杂的，所需的计算代价也是很高的，蛮力搜索方法并不能解决这个问题，因此需要用到更智能的方法在合理的时间内完成。因此我们将采用机器学习中的 Apriori 算法来解决这个问题。

同时我们希望从这些已有的数据中，去帮助消费者发现他们未曾接购买过但是其他有类似购买行为的消费者比较喜爱的商品，然后对他们进行推荐。这种通过将用户和其他用户的数据进行比对来实现推荐的，叫做协同过滤推荐算法。因此我们还将采用机器学习中的协同过滤算法来实现对各个登录系统的消费者进行推荐。

最后我们还希望通过从已有数据中去获取当月各个商品的销售量，然后通过库存量和销售量计算当前的剩余库存量，也可以通过时间差计算平均销售速率。所以此处我们可以通过 python 的科学计算库 NumPy 和 pandas 来进行统计。

3.6.1 Apriori 关联规则算法设计

Apriori 算法是根据指定的数据格式，然后按照一定的最小支持度从数据中找出频繁项集，然后从频繁项集中根据最小置信度挖掘出关联规则。

频繁项集是指一起出现的频率较高的物品组合。对于频繁的定义可以用支持度和置信度来量化[5]。

支持度是指某项集在整个数据集中所占的比例。

置信度是针对一条关联规则来定义的，例如针对 A 的置信度被定义为“支持度({A,B})/支持度({A})”。使用 Apriori 算法的流程设计如图 3-5 所示。

根据算法流程图，定义函数 loadDataSet()，该函数主要功能是

1）利用数据库接口 DAQuery()来获取数据库中的订单数据

2）对查询的数据进行清理，利用 Python 的科学计算库 pandas 整理出每一条只包含商品 ID 的购物记录，生成原始数据集 dataset

3）指定最小支持度和最小置信度

然后根据原始数据集、最小支持度和最小置信度调用 Apriori 算法来找到

关联规则，最后将规则入库并在网页端展示。

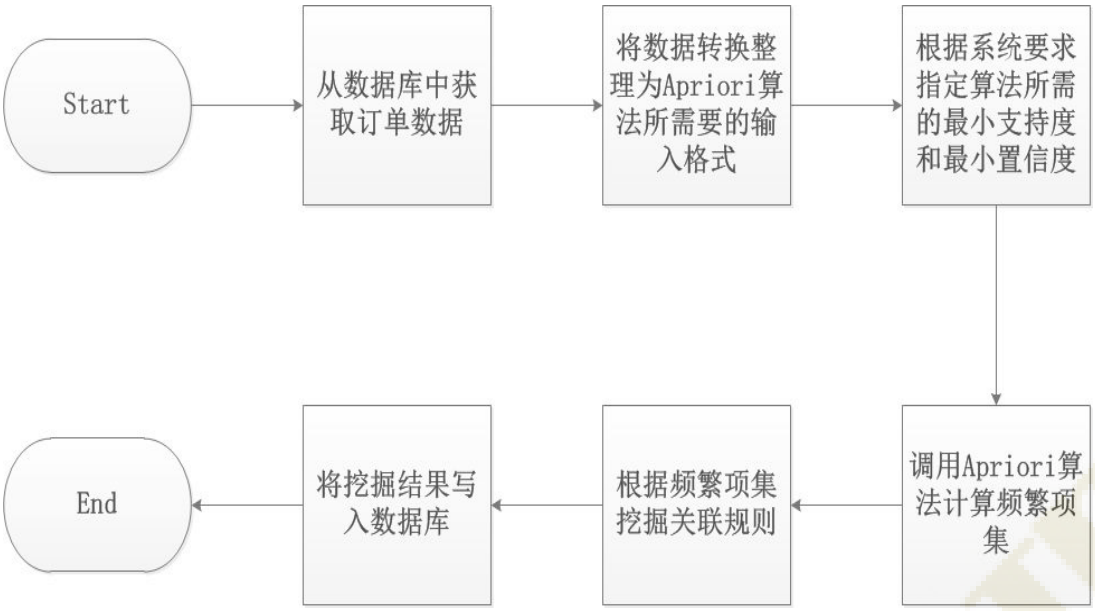

图 3-5 应用 Apriori 算法流程设计

3.6.2 协同过滤算法设计

本系统采用的基于用户的协同过滤算法是根据邻居用户的偏好信息产生对目标用户的推荐，通过相似性度量方法计算出相似用户，并将相似用户购买习惯的结果作为推荐预测结果返回给用户[6]。

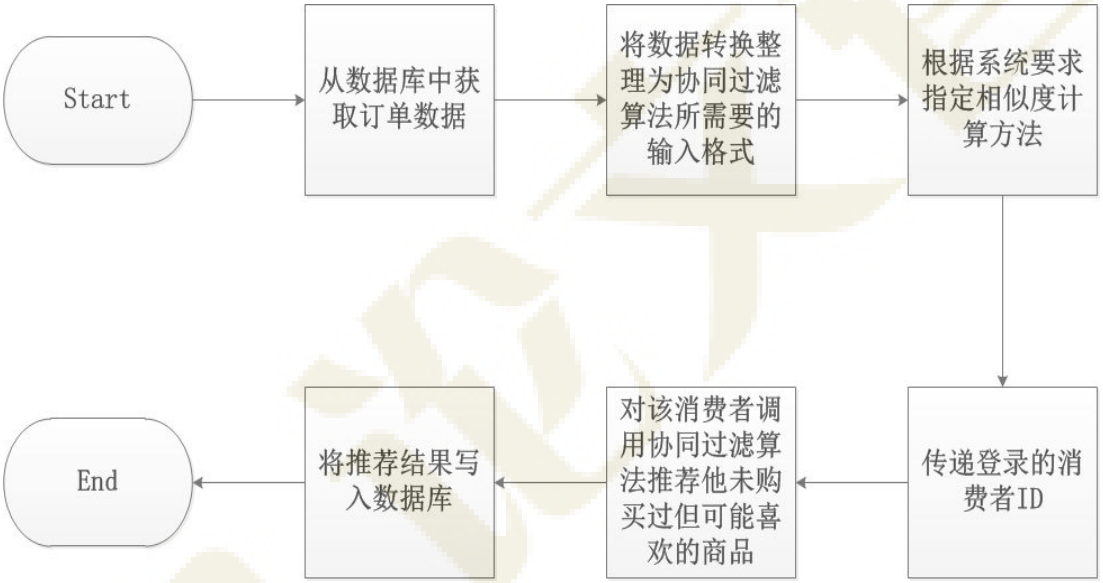

图 3-6 应用协同过滤算法流程设计

因此我们首先需要从数据库获取数据，定义 loadDataSet()函数，该函数主要对查询的数据进行整理，最终可以形成一个以消费者 ID 为行，商品 ID 为列，对应商品的购买总数（商品购买数量从一定程度上反应了消费者对该商品的喜爱度）为值所构成的矩阵，格式如图 3-7 所示。当采用这种方式组织数据时，我们就可以比较消费者或商品之间的相似度了。

	1001001	1001002	1001005	1001006	1001012	1011002	1011003
000000001	976	875	488	584	584	584	584
000000003	0	0	0	26	26	26	26
000000012	144	72	72	0	0	0	0
000000014	0	0	0	70	70	70	70
000000023	166	83	83	59	59	59	59
000000123	0	91	0	0	0	0	0
000000124	0	35	0	0	0	0	0

图 3-7 协同过滤算法输入矩阵

然后根据我们指定的相似度计算方法，以及需要推荐的消费者 ID，调用协同过滤算法来对该消费者进行推荐，并将推荐结果入库最后在网页端展示。

3.6.3 库存及销量统计方法设计

对于库存和销量的统计，可以直接用 Python 的 NumPy 和 Pandas 两个科学计算库。从数据库查询 TOrder_当前年月表，然后按以下步骤处理：

1）使用 pandas 库的 DataFrame()将数据转化为表格形式

2）从表格数据中提取出 GOODS_AMOUNT 字段的数据，遍历其中的数据，并对相同的商品 ID 进行数量累计

3）最后对累计结果进行排序即可得到销量排行。

同样对商品库存通过查询 TInventory_当前年月表与即可获取总量，然后通过前面统计的销量，计算每种商品的当前剩余库存量，并且可以根据当前时间与商品入库时间差，来计算平均销售速率。

4 系统主要功能实现

4.1 51 单片机读取 RFID 的实现过程

单片机程序开始运行后，程序会通过 51 单片机上的按键操作来判断对 RFID 是读或者是写功能，当有按键操作时，就会启动天线寻找范围内的卡片，然后进行防冲突、选卡、计算密码操作，然后根据按键类型，来执行读或者是写操作[7]。当按键为读卡操作时，则先验证密码，然后调用 PcdRead()函数来读取数据，当为写操作时，同样先验证密码，然后调用 PcdWrite()函数来写入数据。

```
KeyVal = get_key();
switch( KeyVal )
{
    case KEY_1:
        OptMode = OPT_READ_MODE;
        break;

    case KEY_2:
        OptMode = OPT_ADD_MODE;
        break;
```

```
            default:
                break;
        }
        if( PcdRequest( PICC_REQIDL, &CardRevBuf[0] ) != MI_OK )
        {
            if( PcdRequest( PICC_REQIDL, &CardRevBuf[0] ) != MI_OK )
                return;
        }
        if( PcdAnticoll( &CardRevBuf[2] ) != MI_OK )
            return;
        if( PcdSelect( &CardRevBuf[2] ) != MI_OK )//选卡
            return;

        cal_keyA( CardKeyABuf );        //计算密码
        switch( OptMode )
        {
            case OPT_ADD_MODE:      //写入
                if(    PcdAuthState(    PICC_AUTHENT1A,    4,    CardKeyABuf,
&CardRevBuf[2] ) != MI_OK )// 验证密码
                    return;
                memset( CardWriteBuf, 0, 16 );
                memcpy( CardWriteBuf, "1010001", 7 );
                if( PcdWrite( 7, CardWriteBuf ) != MI_OK )// 写数据
                    return;
                bPass = 1;
                break;
            case OPT_READ_MODE:     //读卡方式
                if(    PcdAuthState(    PICC_AUTHENT1A,    4,    CardKeyABuf,
&CardRevBuf[2] ) != MI_OK )// 验证密码
                    return;
                if( PcdRead( 7, CardReadBuf ) != MI_OK )// 读数据
                    return;
                bPass = 1;
                break;
            default:
                break;
        }
        PcdHalt();
    }
```

当读操作完成后，此时我们已经获取到 RFID 卡标签的数据了，然后通过串口发送函数 send_byte()对该数据按照预先定义好的格式进行封装,然后从串口发送出去。

```
    void send_byte( INT8U SendData[] )
    {
        INT8U sendBuf[30];
```

```
        strncpy(sendBuf,"begin#RFIDDATA#", 15);
        strcat(sendBuf, SendData);
        strcat(sendBuf, "#end")
        ES = 0;
        TI = 0;
        SBUF = sendBuf;
        while( TI == 0 );
        TI = 0;
        ES = 1;
    }
```

4.2 数据库 Python 接口实现过程

1.插入数据接口的实现

```
def DAInsert(table, data, type, safe_on = False):
    tag = DO_ERROR_INVALID
    mask = ((type & DEVICE_TYPE.DEVICE_SYS) == DEVICE_TYPE.DEVICE_SYS) and
(argv.DEVICE | (type & 0x7fffffff)) or type
    if (mask & DEVICE_TYPE.DEVICE_MONGO) == DEVICE_TYPE.DEVICE_MONGO:
        tag = mongo_interface.insert_into_mongo(table, data, safe_on)
    return tag
```

输入：

table：str 类型，表名

data：dict 类型，插入数据

type：DEVIDE_TYPE 类型，设备类型

safe_on：bool 类型，安全插入标志，默认 False

返回值：

tag：opcode 类型，>=-2 成功，-2 失败

2.查询数据接口的实现

```
def DAQuery(table, con, sort, field, skip, limit, type):
    return mongo_interface.find_from_mongo(table, con, sort, field, skip,
limit)
```

输入：

table：str 类型，表名

con：dict 类型，查询条件

sort：list 类型，list 元素为 dict 类型，排序规则

field：dict 类型，查询列

skip：int 类型，起始记录行

limit：int 类型，返回记录最大数

type：DEVIDE_TYPE 类型，设备类型

返回值：

tag：opcode 类型，>=-2 成功，-2 失败

cursor：dict 类型，结果集

3.更新数据接口实现

```
def DAUpdate(table, con, value, type):
```

```
    return mongo_interface.update_from_mongo(table, con, value, False)
```

输入：

table：str 类型，表名

con：dict 类型，更新条件

value：dict 类型，更新值

type：DEVIDE_TYPE 类型，设备类型

返回值：

tag：opcode 类型，>=-2 成功，-2 失败

4.3 Web 服务器主要功能实现过程

4.3.1 用户登录

本网站是购物网站，所以很多功能必须要登录过后才能使用，消费者和管理员都在此处登录，在系统实现中使用 session 变量存储用户的登录状态。

登录流程如图 4-1：

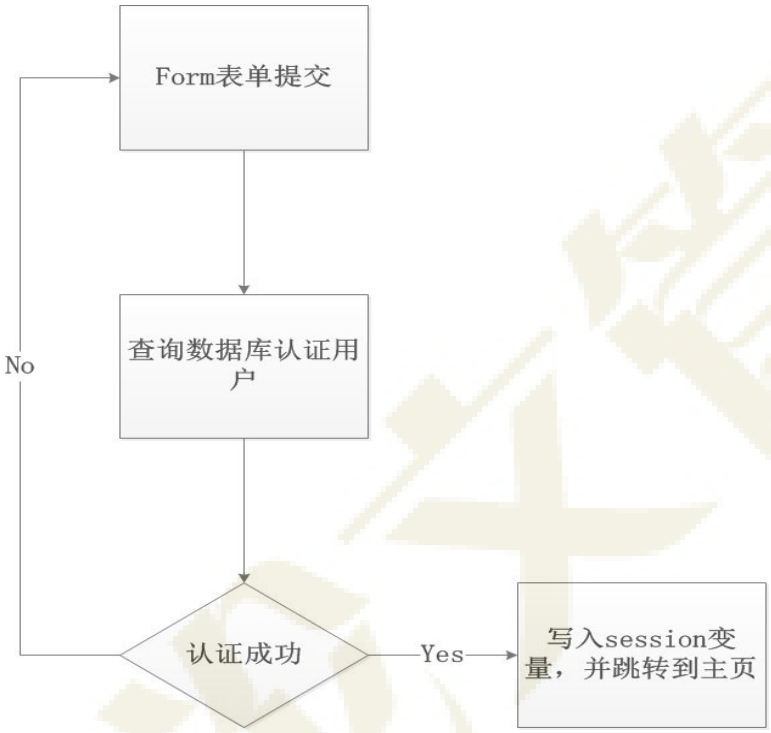

图 4-1 登录流程

后台验证代码：

```
session_start();
$manager = new MongoDB\Driver\Manager("mongodb://localhost:27017");
$query =
 new
MongoDB\Driver\Query(["USER_ID"=>$_POST["user"],"PASSWORD"=>$_POST["pwd"]]);
$cursor = $manager->executeQuery("shop.TAccount", $query);
$count=0;
foreach( $cursor as $document)
{
    foreach($document as $seagment=>$value)
```

```
        if($seagment=="USER_NAME")
            $name=$value;
    $count++;
}
```

4.3.2 商品显示功能

展示商品是本网站最重要的功能之一，如何能展示出详细的信息又让用户感到舒服，是本网站的目标。本网站在显示商品时主要显示了商品图片，商品价格以及商品的名字。我们将上述的三个属性包含在一个超链接标签内，如：

```
echo "<li>
<a href='detail.php?code=$code' target='_parent'>
<img src='$img' height='80' width='80'/>
<br>
<center>$type<br>";
echo "￥$price
</center>
</a>
</li>
```

由于商品的数量不确定，所以我们采用的是边查询数据库边显示的方式：

```
$manager = new MongoDB\Driver\Manager("mongodb://localhost:27017");
if($goodtype&&$goodtype!="all")
{
$query = new MongoDB\Driver\Query(["OWNED_PROPERTY"=>intval($goodtype)]);
}
else
{
    $query = new MongoDB\Driver\Query([]);
}
$cursor = $manager->executeQuery("shop.TGoods_List", $query);

foreach( $cursor as $document)
{
    foreach($document as $seagment=>$value)
    {
        if($seagment=="CODE")
            $code=$value;
        if($seagment=="CODE_TYPE")
            $type=$value;
        if($seagment=="PRICE")
            $price=$value;
        if($seagment=="IMG_URL")
```

```
            $img=$value;
        }
    }
```

4.3.3 购物车的实现

由于网站后台程序的特性，采用普通程序的方式存储购物车是不现实的，PHP 提供了 session 的方式，可以让用户开始进入网站到结束都一个存在的变量，我们通过往 session 变量中写入数据实现购物车。PHP 中要使用 session 变量必须在 HTML 标签之前声明一下：

```
<?php
    session_start();
?>
```

这样，在这个页面中就可以直接调用 session 变量了。同时，由于购物车要具有可增删的功能，我们使用了一个 php 数组用来存储购物车中的商品以及数量。往购物车添加商品：

```
if($_SESSION["car"])
{
    $car=$_SESSION["car"];
    if($car[$code])
        $car[$code]++;
    else
        $car[$code]=1;
        $_SESSION["car"]=$car;
    }
    else
    {
        $car[$code]=1;
        $_SESSION["car"]=$car;
    }
}
```

4.3.4 创建订单

订单一旦创建成功便不可更改，而且在关闭浏览器后，下次登录还能看到。所以，必须将订单写入数据库。MongoDB 的存储方式类似于 json，所以在 MongoDB 中存储可以直接存储一个 json 数组。订单创建代码如下：

```
session_start();
$order = [
"USER_ID"=>$_SESSION["id"],
"GOODS_AMOUNT"=>$_SESSION["car"],
"PURCHASE_DATE"=>date("Y-m-d h:i:s",time()),
];
$bulk = new MongoDB\Driver\BulkWrite(['ordered' => true]);
```

```
$order_id  = $bulk->insert($order);
$wc = new MongoDB\Driver\WriteConcern(
    MongoDB\Driver\WriteConcern::MAJORITY,
    1000
);
$time=date('Ym',time());
$collection="shop.TOrder_$time";
$manager = new MongoDB\Driver\Manager("mongodb://localhost:27017");
try{
    $result  = $manager->executeBulkWrite("$collection", $bulk, $wc);
}catch (MongoDB\Driver\Exception\Exception $e) {}
$_SESSION["car"]=null;
header("Refresh:1;url=/index.php");
```

为了防止一个集合存储过多条目，我们按月存储订单。

4.4 数据分析服务器实现过程

4.4.1 使用 Apriori 算法来发现频繁集的算法实现

```
def createC1(dataSet):
    C1 = []
    for transaction in dataSet:
        for item in transaction:
            if not [item] in C1:
                C1.append([item])

    C1.sort()
    return map(frozenset, C1)   #use frozen set so we
                            #can use it as a key in a dict

def scanD(D, Ck, minSupport):
    ssCnt = {}
    for tid in D:
        for can in Ck:
            if can.issubset(tid):
                if not ssCnt.has_key(can): ssCnt[can]=1
                else: ssCnt[can] += 1
    numItems = float(len(D))
    retList = []
    supportData = {}
    for key in ssCnt:
        support = ssCnt[key]/numItems
        if support >= minSupport:
            retList.insert(0,key)
        supportData[key] = support
    return retList, supportData
```

```
def aprioriGen(Lk, k): #creates Ck
    retList = []
    lenLk = len(Lk)
    for i in range(lenLk):
        for j in range(i+1, lenLk):
            L1 = list(Lk[i])[:k-2]; L2 = list(Lk[j])[:k-2]
            L1.sort(); L2.sort()
            if L1==L2: #if first k-2 elements are equal
                retList.append(Lk[i] | Lk[j]) #set union
    return retList

def apriori(dataSet, minSupport = 0.5):
    C1 = createC1(dataSet)
    D = map(set, dataSet)
    L1, supportData = scanD(D, C1, minSupport)
    L = [L1]
    k = 2
    while (len(L[k-2]) > 0):
        Ck = aprioriGen(L[k-2], k)
        Lk, supK = scanD(D, Ck, minSupport)#scan DB to get Lk
        supportData.update(supK)
        L.append(Lk)
        k += 1
    return L, supportData
```

函数执行及作用详解：

1）函数 createC1()输入参数为原始数据集 dataset，该函数生成大小为 1 的所有候选集的集合 C1，然后将 C1 调用 map(set,dataset)生成集合 D[9]

2）函数 scanD()，输入参数为集合 D,Ck 为候选集（k 为每一个项集的元素个数),该函数将满足最小支持度的数据重新组成集合 LK,并计算出支持度 supK

3）函数 aprioriGen(LK, k),输入参数为 scanD 生成的 LK，以及项集元素个数 k,将输出 Ck,然后一直重复（2），（3）步骤直到所有的组合情况都过滤完毕。

以上（1）-（3）步骤均是由 apriori()函数来进行逻辑控制，该函数的输入为原始数据集以及最小支持度。该函数最后返回一个所有满足最小支持度数据组合的列表，以及各自的支持度组成的列表。

4.4.2 从频繁项集中挖掘关联规则算法实现

从频繁项集中挖掘关联规则，我们会用到 apriori 原理来对不满足最小置信度的规则的超集不进行计算。

```
def generateRules(L, supportData, minConf=0.7):  #supportData is a dict
```

```
coming from scanD
        bigRuleList = []
        for i in range(1, len(L)):#only get the sets with two or more items
            for freqSet in L[i]:
                H1 = [frozenset([item]) for item in freqSet]
                if (i > 1):
                    rulesFromConseq(freqSet,  H1,  supportData,  bigRuleList,
minConf)
                else:
                    calcConf(freqSet, H1, supportData, bigRuleList, minConf)
        return bigRuleList

    def calcConf(freqSet, H, supportData, brl, minConf=0.7):
        prunedH = [] #create new list to return
        for conseq in H:
            conf  =  supportData[freqSet]/supportData[freqSet-conseq]  #calc
confidence
            if conf >= minConf:
                print freqSet-conseq,'-->',conseq,'conf:',conf
                brl.append((freqSet-conseq, conseq, conf))
                prunedH.append(conseq)
        return prunedH

    def rulesFromConseq(freqSet, H, supportData, brl, minConf=0.7):
        m = len(H[0])
        if (len(freqSet) > (m + 1)): #try further merging
            Hmp1 = aprioriGen(H, m+1)#create Hm+1 new candidates
            Hmp1 = calcConf(freqSet, Hmp1, supportData, brl, minConf)
            if (len(Hmp1) > 1):    #need at least two sets to merge
                rulesFromConseq(freqSet, Hmp1, supportData, brl, minConf)
```

函数执行及作用详解：

1）根据前面所计算出来的频繁项集，然后从一个频繁项集开始，创建一个规则列表，该规则被定义为 generateRules(L, supportData, minConf)，输入参数为频繁项集列表、包含频繁项集支持数据的字典、最小置信度。该函数用于遍历 L 中的每一个频繁项集并对每个频繁项集创建只包含单个元素集合的列表 H1。

2）calcConf(freqSet, H1, supportData, bigRuleList, minConf)是对于项集中只有两个元素的计算置信度的值

3）由于无法从单元素项集中构建关联规则，所以要从包含两个或以上元素的项集开始构建规则，因此对于元素在两个及以上的，需要我们用 rulesFromConseq()函数来调用 aprioriGen()函数，以此来对项集合并，并对

合并后的项集调用 calcConf()来测试是否能得到正确的置信度，如果没得到，则递归调用该函数直到可以得出正确的置信度。

4.4.3 基于协同过滤的推荐引擎算法实现

在协同过滤算法中，需要用到的数学方法是相似度的计算。已有的相似度计算方法有：

1.欧式距离计算方法。通过对需要进行相似度计算的两个物品的数量进行距离计算，然后运用“相似度=1/1+距离”来计算相似度[10]。由此可以从商品被购买的数量中得出商品之间的相似度。

```
def ecludSim(inA,inB):
    return 1.0/(1.0 + la.norm(inA - inB))
```

2.皮尔逊相关系数计算方法。该计算方法相对于欧式距离算法的优势在于它对用户喜爱程度不敏感，例如某消费者对所以商品都买相同多，但某消费者又对所以商品都只买很少，皮尔逊相关系数会认为这两个向量是相等的。在 numPy 中皮尔逊相关系数的计算是由函数 corrcoef()进行的，因此需要在算法中使用该函数来计算相似度[10]。

```
def pearsSim(inA,inB):
    if len(inA) < 3 : return 1.0
    return 0.5+0.5*corrcoef(inA, inB, rowvar = 0)[0][1]
```

3.余弦相似度计算方法。计算的是两个向量之间夹角的余弦值，夹角=90 度，则相似度为 0；向量方向相同，则相似度为 1。两个向量夹角余弦相似度定义为：cos θ =向量 A 与 B 的乘积/向量 A 与的 B 的 2 范数的乘积。NumPy 的线性代数工具箱提供了范数的计算方法 linalg.norm()。因此需要在算法中使用该函数来计算相似度[10]。

```
def cosSim(inA,inB):
    num = float(inA.T*inB)
    denom = la.norm(inA)*la.norm(inB)
    return 0.5+0.5*(num/denom)
```

指定了相似度计算方法，函数 standEst()就会使用该方法来对计算相似度。输入参数包括数据矩阵、消费者 ID、相似度计算方法、消费者未购买的商品 ID，该函数将先对数据中消费者购买过的商品进行遍历，并将购买过的商品所在列和消费者自己未购买的商品所在列比较，从中找到其他消费者都购买过的商品，然后对这些都被购买的商品进行相似度计算，并且对相似度进行累加，每次计算时还应考虑相似度和当前消费者的购买总量。最后通过除以所有购买总量来对相似度乘积进行归一化，并将结果用于预测值排序[10]。

有了相似度值，使用函数 recommend()来对指定的消费者进行推荐。输入参数分别为数据矩阵、要推荐的消费者 ID、最高 N 个的推荐结果、相似度计算

方法和相似度计算函数。该函数主要是先获取消费者未购买的商品 ID，然后依次对未购买的商品调用 standEst()函数来计算相似度，以此来实现对消费者未购买过的商品的进行最高 N 个结果的推荐。

实现推荐的源代码如下：

```
def standEst(dataMat, user, simMeas, item):
    n = shape(dataMat)[1] #计算有多少个项目
    simTotal = 0.0; ratSimTotal = 0.0
    for j in range(n): #遍历该用户所在行的所有项目
        userRating = dataMat[user,j]
        if userRating == 0: continue
        overLap = nonzero(logical_and(dataMat[:,item].A>0, \
                                      dataMat[:,j].A>0))[0]
        if len(overLap) == 0: similarity = 0
        else: similarity = simMeas(dataMat[overLap,item], \
                                   dataMat[overLap,j])
        print 'the %d and %d similarity is: %f' % (item, j, similarity)
        simTotal += similarity
        ratSimTotal += similarity * userRating
    if simTotal == 0: return 0
    else: return ratSimTotal/simTotal

def recommend(dataMat, user, N=3, simMeas=cosSim, estMethod=standEst):
    unratedItems = nonzero(dataMat[user,:].A==0)[1]#find unrated items
    if len(unratedItems) == 0: return 'you rated everything'
    itemScores = []
    for item in unratedItems:
        estimatedScore = estMethod(dataMat, user, simMeas, item)
        itemScores.append((item, estimatedScore))
    return sorted(itemScores, key=lambda jj: jj[1], reverse=True)[:N]
```

5 系统测试与改进

表 5-1 系统要测试的功能点

功测试能点	测试结果
登录/注册是否成功	可以成功注册和登录
主页面显示是否正常	显示正常
能否查看订单	显示正常
Apriori 算法分析结果	由于数据数量不够，导致支持度和置信度都比较低，所以在指定最小支持度和置信度时需要指定较低的
协同过滤分析算法分析结果	同样是由于数据量不够，导致相似度的值都偏小
销量统计结果	能正确统计各个商品的销量并且能按数量排序

5.1 系统页面展示

图 5-1 展示的是页面登录功能测试结果：

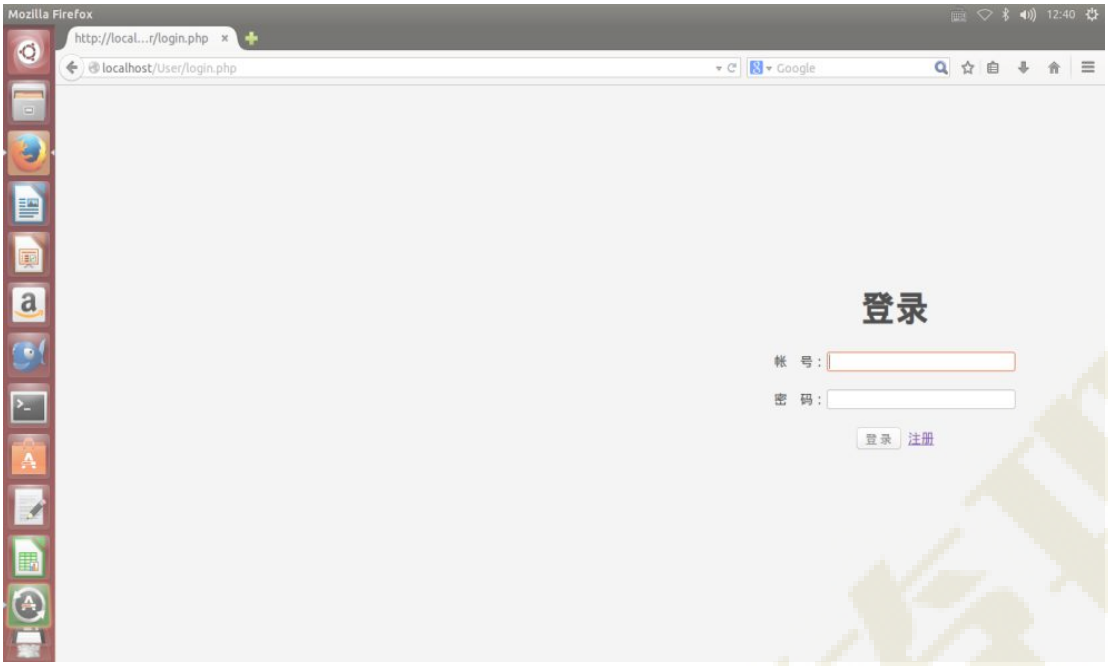

图 5-1 页面登录/注册

图 5-2 所示为主页面（部分商品还未添加图片）：

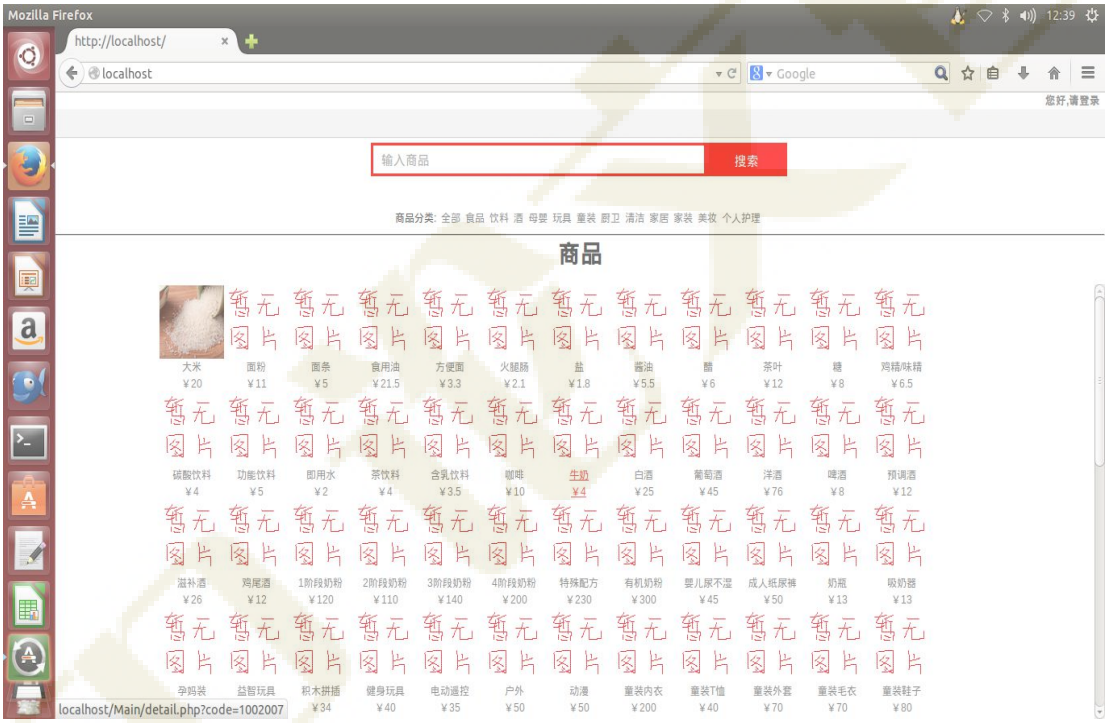

图 5-2 主页面展示

图 5-3 是查看订单详情的页面：

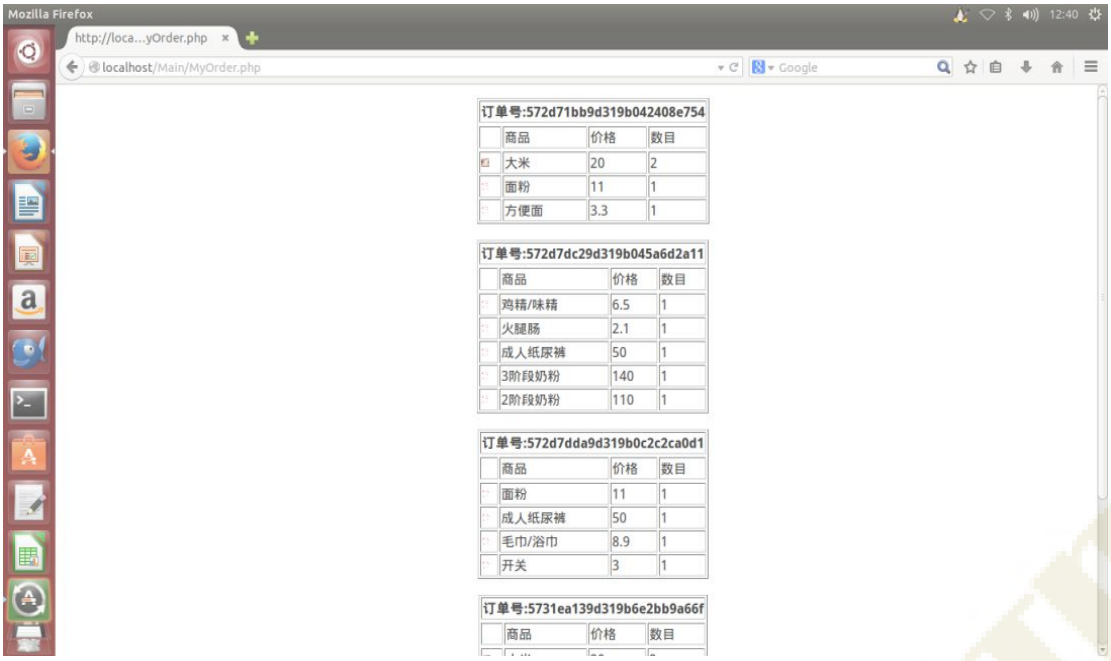

图 5-3 查看订单详情

5.2 系统主要功能测试

测试 Apriori 算法对关联分析的结果：

使用的测试数据如图 5-4 显示，此处只展示部分数据，测试数据总共约有 2000 条。对于本系统，我们需要对该数据进行多次处理以得到不同的数据，对于 Apriori 算法的输入数据，则只需要提取每次购买的商品 ID 所组成的 List 即可；对于协同推荐算法则需要先统计和合并当月各个用户所购买的所有商品及数量，然后转换成矩阵形式传递给协同过滤算法实现的接口；而对于销售量的统计又需要对当月所有物品的销售量进行统计和排序。

{u'_id': {u'$oid': u'572d7dc29d319b045a6d2a11'}, u'USER_ID': u'000000001', u'GOODS_AMOUNT': {u'1011008': 1, u'1011003': 1, u'1001012': 1, u'1011002': 1, u'1001006': 1}, u'PURCHASE_DATE': u'2016-05-07 01:31:46'}
{u'_id': {u'$oid': u'572d7dda9d319b0c2c2ca0d1'}, u'USER_ID': u'000000001', u'GOODS_AMOUNT': {u'1032002': 1, u'1001002': 1, u'1011008': 1, u'1031004': 1}, u'PURCHASE_DATE': u'2016-05-07 01:32:10'}
{u'_id': {u'$oid': u'572d71bb9d319b042408e754'}, u'USER_ID': u'000000001', u'GOODS_AMOUNT': {u'1001001': 2, u'1001002': 1, u'1001005': 1}, u'PURCHASE_DATE': u'2016-05-07 12:40:27'}
{u'_id': {u'$oid': u'572d7dda9d319b0c2c2ca0d1'}, u'USER_ID': u'000000001', u'GOODS_AMOUNT': {u'1032002': 1, u'1001002': 1, u'1011008': 1, u'1031004': 1}, u'PURCHASE_DATE': u'2016-05-07 01:32:10'}
{u'_id': {u'$oid': u'572d71bb9d319b042408e754'}, u'USER_ID': u'000000001', u'GOODS_AMOUNT': {u'1001001': 2, u'1001002': 1, u'1001005': 1}, u'PURCHASE_DATE': u'2016-05-07 12:40:27'}
{u'_id': {u'$oid': u'572d7dc29d319b045a6d2a11'}, u'USER_ID': u'000000001', u'GOODS_AMOUNT': {u'1011008': 1, u'1011003': 1, u'1001012': 1, u'1011002': 1, u'1001006': 1}, u'PURCHASE_DATE': u'2016-05-07 01:31:46'}
{u'_id': {u'$oid': u'572d7dda9d319b0c2c2ca0d1'}, u'USER_ID': u'000000001', u'GOODS_AMOUNT': {u'1032002': 1, u'1001002': 1, u'1011008': 1, u'1031004': 1}, u'PURCHASE_DATE': u'2016-05-07 01:32:10'}
{u'_id': {u'$oid': u'572d71bb9d319b042408e754'}, u'USER_ID': u'000000001', u'GOODS_AMOUNT': {u'1001001': 2, u'1001002': 1, u'1001005': 1}, u'PURCHASE_DATE': u'2016-05-07 12:40:27'}
{u'_id': {u'$oid': u'572d7dc29d319b045a6d2a11'}, u'USER_ID': u'000000001', u'GOODS_AMOUNT': {u'1011008': 1, u'1011003': 1, u'1001012': 1, u'1011002': 1, u'1001006': 1}, u'PURCHASE_DATE': u'2016-05-07 01:31:46'}
{u'_id': {u'$oid': u'572d7dda9d319b0c2c2ca0d1'}, u'USER_ID': u'000000001', u'GOODS_AMOUNT': {u'1032002': 1, u'1001002': 1, u'1011008': 1, u'1031004': 1}, u'PURCHASE_DATE': u'2016-05-07 01:32:10'}
{u'_id': {u'$oid': u'572d71bb9d319b042408e754'}, u'USER_ID': u'000000012', u'GOODS_AMOUNT': {u'1001001': 2, u'1001002': 1, u'1001005': 1}, u'PURCHASE_DATE': u'2016-05-07 12:40:27'}
{u'_id': {u'$oid': u'572d7dc29d319b045a6d2a11'}, u'USER_ID': u'000000014', u'GOODS_AMOUNT': {u'1011008': 1, u'1011003': 1, u'1001012': 1, u'1011002': 1, u'1001006': 1}, u'PURCHASE_DATE': u'2016-05-07 01:31:46'}

图 5-4 原始测试数据（部分）

从原始数据提取出适用 apriori 算法的数据格式，如图 5-5 所示(只展示部分

数据)：

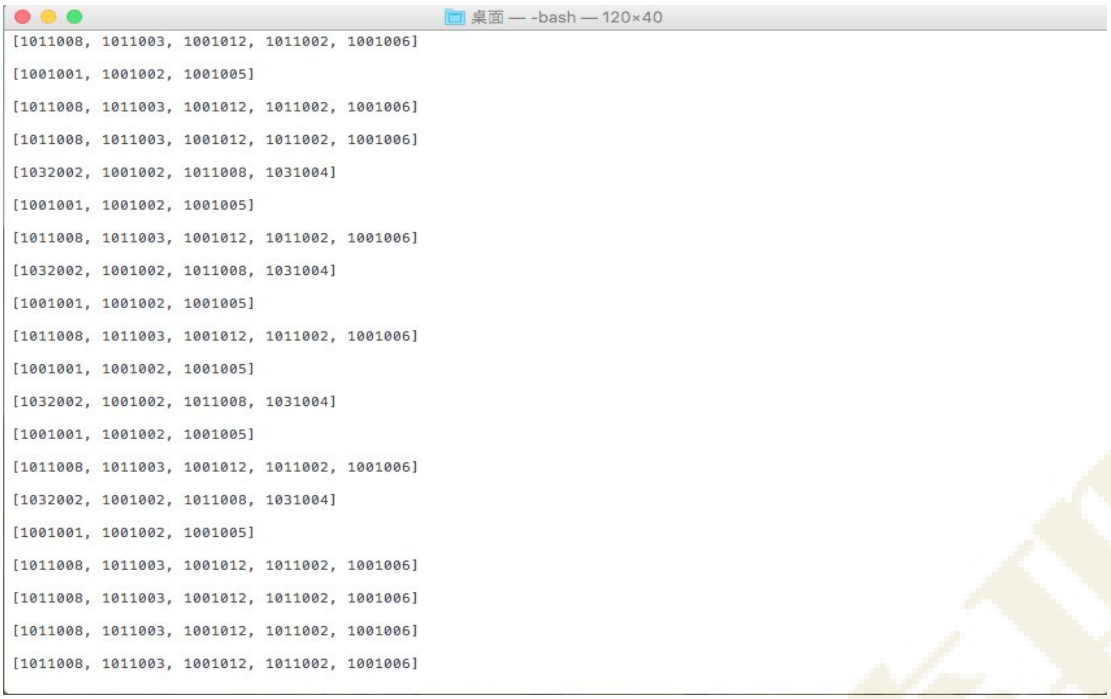

图 5-5 Apriori 输入数据

使用 apriori 算法对图 5-5 所示数据进行分析得出一定的关联度，如图 5-6 所示，显示的是使用最小支持度为 0.3，最小置信度为 0.5 时的结果：

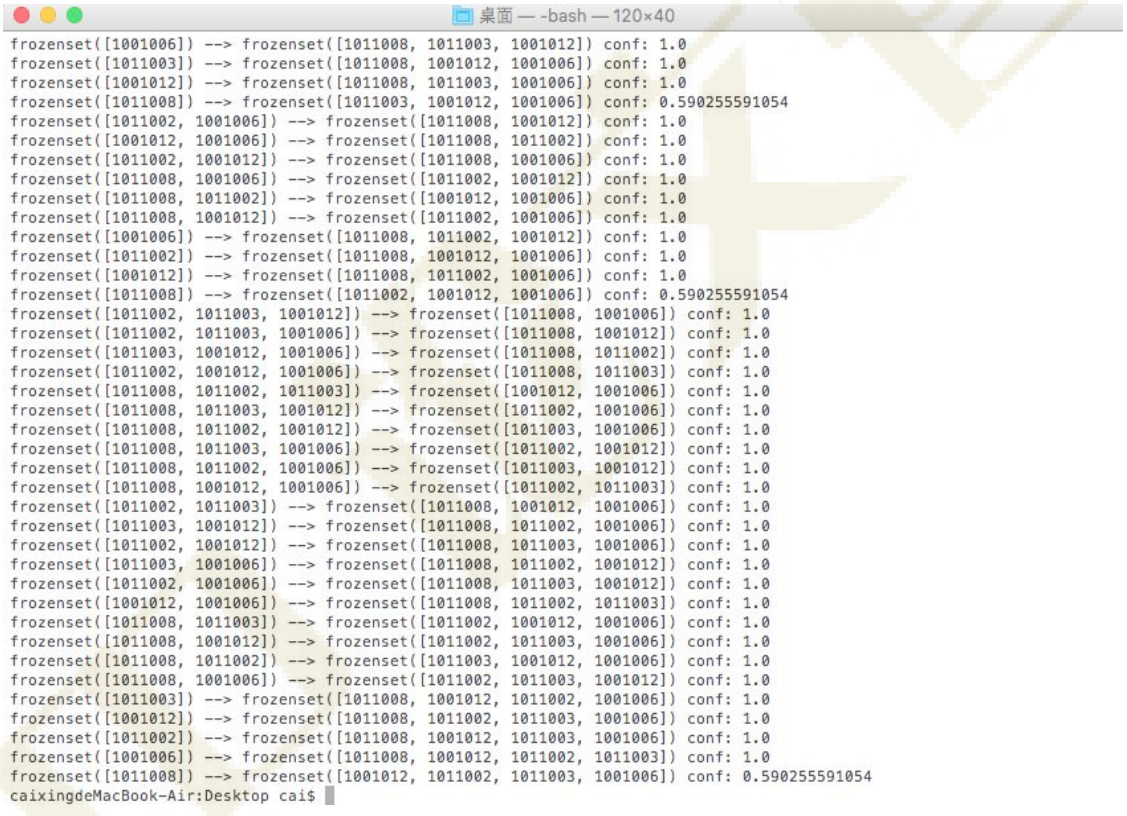

图 5-6 关联分析结果

从原始数据中整理出协同过滤算法的输入数据如图 5-7 所示：

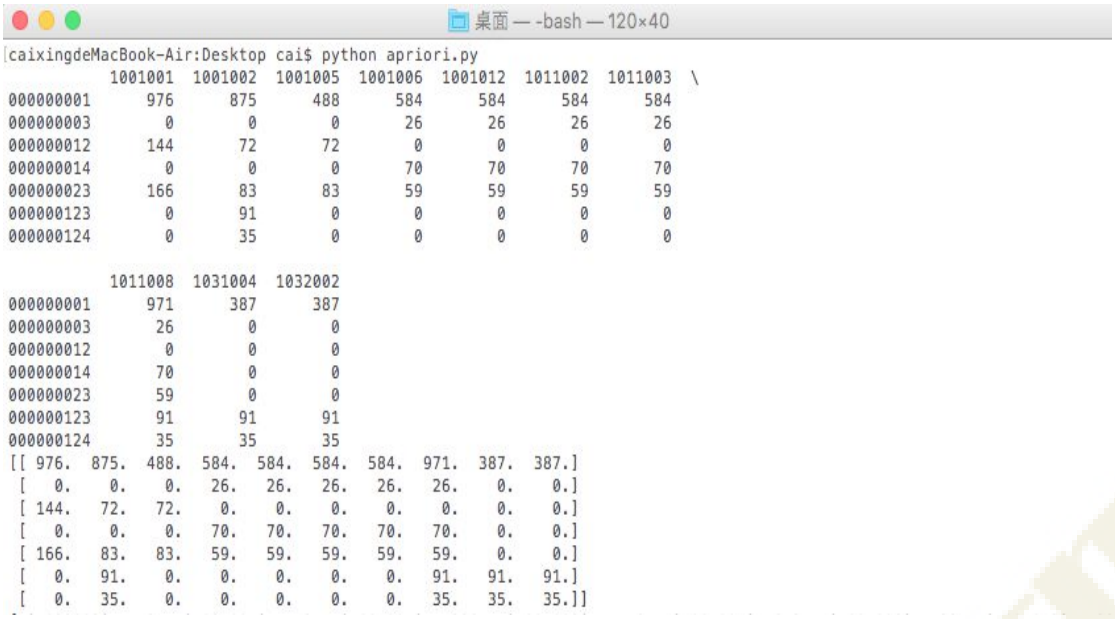

图 5-7 协同过滤算法输入数据

其中带有行和列名的数据是使用 pandas 的 DataFrame 来对数据进行第一次转换，所得到的数据为一个表格形式，行名代表用户 ID，列名代表商品 ID，数据则是该用户对该商品所购买的数量。然后通过 DataFrame.fillna(0)对缺失数据进行清理，然后再使用 mat()函数对 DataFrame 数据进行矩阵转换，得到协同过滤算法的输入数据。

使用协同过滤推荐算法对用户 4（对应用户 ID 为 000000023）进行推荐的结果如图 5-8 所示，所使用的是余弦相似度计算方法：

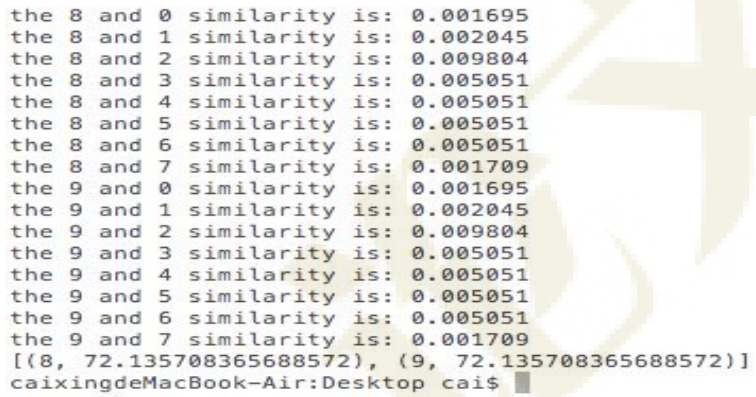

图 5-8 协同过滤推荐结果

从推荐结果可以看出用户 4 对商品 8 和 9（对应商品 ID 为 1011008 和 1031004）的购买力度约为 72.14。

最后从原始数据中对当月销量进行统计和排序结果如图 5-9 所示：

```
{u'1032002': 513, u'1031004': 513, u'1001001': 1286, u'1001002': 1156, u'1001005': 643, u'1001006': 739, u'1001012': 739
, u'1011002': 739, u'1011003': 739, u'1011008': 1252}
1031004     513
1032002     513
1001005     643
1001006     739
1001012     739
1011002     739
1011003     739
1001002    1156
1011008    1252
1001001    1286
```

图 5-9 销量排行结果

5.2 改进方案

存在的问题：从以上数据分析的结果可以看出，由于数据是通过人为构造的，导致分析的结果与预计的分析结果差别较大，当使用关联规则进行分析时，由于数据是人为构造所以相似度较高，导致支持度普遍较低，所以须降低最小支持度和最小置信度来查看结果，同时由于数据量偏少，也会导致分析的结果不是很准确。

改进方案：继续增加更多的数据，并且降低数据的相似度来重新进行测试，并多次改变最小支持度和最小置信度，改变不同的相似度算法，来对分析的结果进行比价，选出最适合的支持度和置信度以及相似度算法来进行分析，尽量得出更可靠的分析结果。

结　　语

通过本次课题的研究，经过大量的阅读资料与咨询老师同学，完成了基于数据分析的物联网超市系统设计与实现，基本实现了系统的基本功能与要求。

在本次课题研究刚开始时，好多知识点都没有接触过，尤其是数据分析以及机器学习算法这部分更是从未了解过。但幸好身边有良师好友的帮助，给我指导学习方法，推荐学习路径，怎么去构建系统框架，然后将框架细分，逐步攻破难点，感谢他们的帮助，也感谢自己一路的坚持，通过这次课题设计，真的学到了很多解决问题的新思路和新想法，也学到了很多新的知识以及新的领域。但是由于知识及能力的匮乏，也导致在系统包括设计与实现等多个方面均有不足。在网页端没有更好的对网页布局进行设计，同时由于系统硬件资源的缺乏，不能搭建分布式环境，这有可能会导致当数据增多时，单台服务器计算能力有限而导致资源消耗过多，以及计算时间较长等问题。但是最重要的是有了这些收获，对我来说还是意义很大的，这不仅仅是对大学四年的成果的一个考核，更是考验我们将所学致以所用的能力，也是让我们对时间规划、项目计划等能力得到锻炼，我希望在以后的人生中，会继续努力学习，努力前进。

参考文献

[1] 刘云浩.物联网导论[M].科学出版社,2011.15-40

[2] David Hows. The Definitive Guide to MongoDB: A Complete Guide[M]. 清华大学出版社, 2013.5-11

[3] Wes McKinney. Python for Data Analysis[M].机械工业出版社, 2015

[4] 莫桂江.一种基于51单片机的物联网信息采集器的设计与实现[J].制造业自动化,2010,33（12）:9-11

[5] 苏变萍, 金维兴, 董丽丽, 侯筱婷.基于关联规则挖掘技术挖掘建设法规领域数据的方法[P].中国:CN 200910023991,2013年5月8日

[6] 姚平平, 邹东升, 牛宝君.基于用户偏好和项目属性的协同过滤推荐算法[J].计算机系统应用,2014,24（7）：15-16

[7] 李建军，周晓中，桂卫华.Mifare系列射频卡读写器的研制[J].《电气应用,2006,25(1):117-121

[8] mongodb 笔记安装启动mongodb[OL].http://howiefh.github.io/2014/04/26/mongodb-note-1-install-mongodb/, 2014-4-26

[9] 使用Apriori算法进行关联分析[OL].http://www.zyh1690.org/the-machine-learning-12-using-apriori-algorithm-for-correlation-analysis/, 2015-06-26

[10] Peter Harrington.Machine Learning in Action[M].人民邮电出版社,2013

致　　谢

本文是在指导老师石磊的帮助和指导下完成的，他严谨的治学态度和渊博的知识使我受益颇多，这也对我顺利完成本课题起到了很大的作用。在此我向石磊导师表示最衷心的感谢！

在论文完成过程中，我还得到了张浩曦老师和陈泰良、倪杰同学的热心帮助，我也在此向他们表示最真挚的谢意！

最后向在百忙之中评审本文的各位专家、老师表示衷心的感谢！谢谢！

作者简介：

姓　　名：蔡行　　性别：男

出生年月：1994.04　　民族：汉族

E-mail:ch_snail@163.com

声　明

本论文的工作是 2015 年 12 月至 2016 年 5 月在成都信息工程大学信息安全工程学院完成的。文中除了特别加以标注地方外，不包含他人已经发表或撰写过的研究成果，也不包含为获得成都信息工程大学或其他教学机构的学位或证书而使用过的材料。

关于学位论文使用权和研究成果知识产权的说明：

本人完全了解成都信息工程大学有关保管使用学位论文的规定，其中包括：

（1）学校有权保管并向有关部门递交学位论文的原件与复印件。

（2）学校可以采用影印、缩印或其他复制方式保存学位论文。

（3）学校可以学术交流为目的复制、赠送和交换学位论文。

（4）学校可允许学位论文被查阅或借阅。

（5）学校可以公布学位论文的全部或部分内容（保密学位论文在解密后遵守此规定）。

除非另有科研合同和其他法律文书的制约，本论文的科研成果属于成都信息工程大学。

特此声明！

作者签名：

2016 年 06 月 17 日