论文封面

/*模板可从“同济大学教务管理信息系统”（xuanke.tongji.edu.cn）下载“毕业设计内封面（声明）”内容无需填写，输入学号后即可自动生成，但声明必须手签名。*/

装

订

线

摘要

序列周期模式的挖掘是数据挖掘领域的一个重要问题，将用户的周期模式与个性化推荐服务结合可以满足用户在不同生活状态下的需求，这是未来个性化推荐领域的主要发展趋势。在实际生活中，用户的某些行为具有一定的规律，因此挖掘用户的行为周期模式可以帮助我们更好的了解用户的行为轨迹，从而进行地点的预测与推荐。本文设计了一种周期模式挖掘算法。该算法结合子序列位置信息以及 FP-Growth 算法的思想，能够较好的挖掘出时间序列中非严格多周期的周期模式。此外，本文还实现了一个个性化推荐系统——个性化日程管理软件。该软件应用了周期模式挖掘算法，可以挖掘出用户的周期性行为，从而预测用户未来可能到达的地点，并推送预测到达地点的周边商圈。该软件在普通日程管理软件上做出了创新与改进，为用户提供更加精准的推荐服务。

关键词:个性化推荐，日程管理软件，周期模式，多周期，数据挖掘

装

订

线

Abstract

Mining periodic patterns in sequence is an important problem in the field of data mining. The combination of periodic pattern and personalized recommendation fulfills people's requirement in different situations. It definitely will be the main developing tendency in the field of personalized recommendation service. In actual life, sometimes regularities are found in people's behaviors, so mining periodic pattern can help us learn people's behavior better. Consequently, we can predict people' schedule and recommend locations which are more suitable for them. This paper proposes an algorithm to mine periodic pattern. It uses the location information of subsequences and the idea of FP-Growth, which can mine the non-strict periodic pattern with multiple periods. Moreover, this paper designs and implementations a personalized recommendation system, a personalized schedule management software, which applies the periodic pattern mining algorithm. It can detect user's periodic behavior, and predict user's schedule, and push surrounding sites of the prediction. It is an innovation and improvement of the regular schedule management software, which provides users with more accurate recommendation service.

Keywords: personalized recommendation; schedule management software; periodic pattern; multiple periods; data mining

装 订 线

目 录

装 订 线

装 订 线

1 绪论

1.1 研究背景

在生活中，我们常常会面对这样的难题：假设你今天光顾一个从来没有去过的商场，想找一家味道不错的店吃晚饭。但当你面对琳琅满目的美食店，倘若你没有明确的需求，就会不知道该去哪一家，或者假设你今天想找部有趣的电影看看，但面对数不胜数的电影，如果你没有明确的目标，就会纠结该看哪一部。这便是“信息过载”问题。

随着互联网上的信息量井喷式地增长，人们逐渐从“信息匮乏”的时代步入“信息过载”的时代。虽然便捷又快速的互联网可以满足用户对信息的需求，但相对地，这也为用户的选择增添了不必要的麻烦。当用户想从茫茫信息中查询自己想要的信息时，如果无法进行一定的筛选，会降低对信息的使用效率。

个性化推荐系统是目前解决“信息过载”问题的有效手段之一。它可以结合用户的兴趣特点和历史行为，向用户推荐他们可能感兴趣的商品。个性化推荐系统属于信息过滤的一种应用，它能够预测用户可能喜爱的信息或物品（例如：电影、音乐、书籍、电视节目、新闻等），并推荐给用户。在文献[1]中，作者向我们介绍了多种个性化推荐算法，其中基于用户的协同过滤算法[2]和基于物品的协同过滤算法[3]是现在比较流行的算法。

目前，个性化推荐系统的应用领域非常广泛，涉及电子商务、电影、社交网络、音乐、新闻、阅读、基于位置的服务（Location-based Services，简称LBS）等等。它通过分析用户行为日志，给不同的用户提供个性化服务。电子商务无疑是个性化推荐系统的应用最为广泛的领域，著名的电商平台亚马逊就应用了个性化推荐系统，它根据用户的历史行为（历史购买的商品）为用户做出推荐。比如，我曾经在亚马逊网站上购买过一本《GRE词汇精选》，那么亚马逊就为我推荐有关GRE学习方面的书籍。

除了电子商务，LBS正逐渐受到人们的广泛关注。有研究报告显示，使用智能手机上网的用户中有58%的用户使用过基于位置的服务[4]。其中，使用导航和获得基于位置推荐的占55%，分享自身位置信息的占12%。所谓LBS[5]是指通过移动终端的定位系统，确定用户的实际地理位置，从而可以提供与位置相关的位置服务。它提供的主流服务有：手机导航、基于位置的社交网络（Location-based Social Network，简称LBSN）、智能交通等等。其中，LBSN已经成为现在必不可缺的服务了，Facebook、twitter、微博、大众点评等具有代表性的社交网络平台，都已经具备了位置分享、位置签到、位置评论等位置服务相关的功能。具体地讲，用户可以在某地点进行签到，公开他们的地理位置或者留下他们的评论。然后，商家对这些签到信息、评论内容和好友关系等处理过后，对用户相似度、用户感兴趣地点进行数据挖据，并利用合理的推荐算法为用户提供更具个性化的位置推荐。

然而，这样的推荐虽然结合了用户的历史记录以及他们的喜好达到了某种程度的“精确推销”，但这样设计出来的推荐系统并不能满足用户在不同生活状态下的需求，即它忽略了用户各自的行为模式。在日常生活中，不同的用户拥有不同的行为模式。例如，有的用户每周一到每周五去公司上班，周六周日休息；而有的用户每周六也需要上班。若某用户周一到周五的白天都会去公司，那么在这个时间段里为用户推荐距离较远的旅游地点显然是不合适的。这样的周期性行为的例子

在我们的生活中不胜枚举，它属于用户行为模式的一种，描述了用户日常生活中某些固定的行为模式。因此，通过挖掘用户的行为模式，对于个性化推荐是很有帮助的，并且满足了用户的现实需求。

周期性行为是用户行为模式的一种。挖掘用户行为的周期性可以有效地分析和总结用户的行为序列。周期性行为的定义中包含了三个主要的元素：重复性的活动、固定的地点、有规律的时间间隔，结合这三个元素我们可以更好的认识用户的行为。挖掘周期性行为可以用来压缩行为数据[6, 7, 8]，因为它是行为序列的总结，所以可以替换原始数据，节省空间。挖掘周期性行为还可以预测未来的行为[9]，因此，结合周期性行为可以有效地提高推荐的准确性的。本文通过研究用户行为周期模式挖掘算法，发现用户日常生活的周期性行为，实现一个个性化推荐系统，具有一定的现实意义。

1.2 研究现状

现在有许多应用都会基于 LBSN，例如：国外的 Foursquare、GeoLife，以及国内的新浪微博、大众点评等等。每个应用的功能都不完全相同：Foursquare 作为签到界的鼻祖，可以向用户提供某个地点的有价值的信息。比如，走进一家从未到过的咖啡厅，它会告诉你有个朋友也在场，或者这家店的美式咖啡不错。GeoLife 系统[10]从用户与位置的关系图中，获得用户与用户的关系、以及位置与位置的关系，从而实现了三个应用模块：基于 GPS 轨迹的生活经验的分享，例如用户出行方式的判断、路线查询等；行程推荐，例如热门景点、旅游地点的序列的推荐等；个性化好友和地点的推荐。新浪微博是将位置信息附在博文信息中，分享在网络上面。大众点评可以向用户推荐用户当前位置附近的餐饮、娱乐、电影院的信息，包括具体地址与信息、用户的打分与评论、优惠活动等。这些应用的功能不完全相同，但都可以以手机用户的签到信息或者评论信息为基础，为用户提供地点推荐、博文推荐等个性化功能。

通过调研，目前已经存在着大量有关挖掘用户行为模式以及基于LBSN推荐的研究成果。论文[11]中，郑宇等人通过对用户未知行驶方式的移动轨迹的研究，提出了基于图的预处理方法，使用概率线索进一步的提高了判断用户行驶方式的准确率，并将此功能融入到了“GeoLife”项目中。郑宇等人基于GPS轨迹，提出了User-Location-Activities三维张量模型[12]，能够为用户提供位置推荐、活动推荐等。用户偏好与事件位置的结合[13]，能够为用户的某一件事件推荐潜在的客人，通过贝叶斯模型能够提供准确的推荐。在文献[14]中，作者总结出了过去数十年挖掘周期模式的算法和应用，它定义了周期模式是一种在给定序列中以某一固定的周期重复出现的子序列模式。在文献[15]中，作者研究了挖掘运动对象周期性行为，提出了一个算法——Periodica，它研究了如何检测运动的周期性，以及如何挖掘周期性行为，并分别在人造数据集和真实数据集上进行实验，证明其算法的有效性。在文献[16]中，作者提出了结合周期图与自相关，能够挖掘运动对象的行为周期，并通过挖掘老鹰行为序列的周期发现了老鹰的迁徙周期，证明了这个方法。学习国内外在基于用户行为模式的推荐方面的现有成果，可以帮助我在挖掘用户行为周期模式方面的研究。

1.3 本文主要内容与结构安排

本文设计并实现了一个个性化推荐系统——个性化日程管理软件。考虑到在 LBSN 领域中，用户的日程总是会重复地发生，即用户的行为轨迹呈现出周期性，本文提出了在时间序列为非严

格多周期情况下的周期模式挖掘算法，其中包括了 Stay Point 算法、OPTICS 算法、以及基于 FP-Growth 的周期模式挖掘算法。本文最终将通过一个具体的个性化推荐系统，实现基于用户周期模式挖掘算法的地点推荐。具体的工作如下：

(1) 实现 Stay Point 算法：基于用户 GPS 坐标，获得用户的逗留点。

(2) 实现 OPTICS 算法：对逗留点进行聚类，并为每个逗留点贴上一个标签。

(3) 获得时间序列：将聚类的结果按照时间戳排序后，转换成时间序列，即“01”序列。‘0’表示用户未到该地点，‘1’表示用户到达该地点。

(4) 实现周期模式挖掘算法：结合子序列位置信息，以及 FP-Growth 算法，挖掘时间序列为非严格多周期情况下的周期模式。

(5) 建立预测模型：判断用户明日是否会到达某一个地点。

(6) 设计并实现个性化日程管理软件：Android 客户端和服务器端。

Android 客户端主要模块有：①基本日程管理功能模块（对日程进行增、删、改、查等操作）；②GPS 数据采集模块；③接收服务器预测到达地点的推送消息模块；4、基于预测到达地点的移动互联网 POI 推荐服务模块。

服务器端的主要模块有：①数据库处理模块（存储客户端发送的数据）；②数据预处理模块（获得用户逗留点、对逗留点进行聚类、获得时间序列）；③用户周期模式挖掘模块（挖掘用户行为的周期模式）；④消息推送模块（建立预测模型、推送消息）。

本文的具体组织结构如下：在第 2 章节中，概括地描述了个性化日程管理软件，并介绍了开发环境及相关技术。在第 3 章中，本文将具体介绍个性化管理软件的实现，包括整个系统框架、Android 客户端、服务器端。在第 4 章中，本文将详细介绍 Stay Point 算法、OPTICS 算法、周期模式挖掘算法。在第 5 章中，本文将展示软件的界面，以及软件测试结果。在第 6 章中，总结了所有工作，并展望了一下未来。

装 订 线

2 系统概述与开发环境

2.1 软件需求分析

现在上班族的工作量大，稍不留神就会把一些重要事情给忽略了。比如，领导交代你本周周五10点召开项目小组会议，要求你提前准备好所需的材料；或者你每月月底都会回家乡看望父母，那么在那之前就需要提前预定火车票或者飞机票。通常，我们把这类即将要做、又比较容易遗忘的事情写在日程管理软件中，方便提醒自己、不容易遗忘。目前市面上比较常见的手机日程管理软件分成两类，如图2.1。

图2.1（a）是便条式的日程管理软件。用户可以根据自己的行程安排，一条一条地写明即可，如“下班回家给孩子买本书”或者“晚上给XX回复邮件”等等。该类软件虽然操作简单、容易上手，但不附带提醒功能。图2.1（b）是具有提醒功能的日程管理软件，可以设置时间，到时间会自动提醒用户。

（a）便条式日程管理软件　　（b）提醒式日程管理软件

图 2.1 日程管理软件

总的来说，这样的日程管理软件通常都需要用户手动输入，免不了给用户带来一定的麻烦。如何为用户提供更加人性化的服务呢？通过对用户日程的调研，发现用户的行为轨迹往往呈现出某种周期性行为。这种周期性行为可以通过日程管理软件进行呈现。本文设计的日程管理软件不仅满足了用户对普通日程管理软件的需求，还结合用户的周期性行为模式为用户推送预测的日程，以及推送预测到达地点的周边商圈。可以说该软件在原先的日程管理软件上做出了创新与改进，

能够为用户提供更加精准的推荐服务。

2.2 系统概要设计

本系统保留了日程管理软件的基本功能。用户使用本系统时，可以根据日历表选择事件提醒的时间，输入事件提醒的内容，以及事件的重要程度；用户可以选择按时间删除某事件；用户可以自由地修改某事件的内容；用户可以查看今天、明天、本周、一周后的事件提醒；用户可以通过帮助，来了解本软件的功能以及操作步骤。

除了普通日程管理软件的基本功能外，本系统能为用户推送预测的日程，以及预测地点的周边商圈。GPS数据采集模块定时采集用户手机的GPS坐标，并在wifi环境下将用户ID、GPS坐标的经度、纬度、时间戳上传到服务器端。预测日程的推送消息模块会将明天的日程推送至用户的手机，明天的日程是由服务器端进行一系列计算得到的，结合了用户过往的GPS数据做出的预测。基于预测地点的移动互联网POI（points of interest）推荐服务模块提供了预测地点周边的商圈，用户可以对其进行查看。

服务器端在本系统中起着至关重要的作用。它需要处理用户发送的数据，对数据进行处理，推送预测的日程。其中，接收数据模块将用户上传的用户ID、GPS坐标的经度、纬度、时间戳存储进数据库。数据预处理模块包括了3个子过程，首先通过Stay Point算法获得用户的逗留点（详细过程见4.3）；其次通过OPTICS算法对逗留点进行聚类，并对每个类贴上标签（详细过程见4.4）；最后根据聚类的结果，计算每个类的时间序列，形成相应的“01”序列（详细过程见4.5）。对数据进行一定的处理后，才能继续进行周期模式的挖掘。用户周期模式挖掘模块也包括了3个子过程，首先假定用户的行为均以“周”或“周的倍数”为周期，即我们只挖掘以“周”或“周的倍数”为周期的周期模式，通过基于FP-Growth的周期模式挖掘算法获得用户行为的周期模式（详细过程见4.6）；其次根据用户的周期模式建立预测模型；最后将预测结果推送给对应用户ID的用户手机。按照这样的流程，完成了一次对用户日程的预测。

2.3 开发环境及相关技术

2.3.1 系统开发环境

(1) Java 环境配置

本系统采用的 jdk 的版本是 jdk1.8，目前 jdk 最高版本为 jdk1.8。本机为 win7 64 位操作系统，从 http://www.oracle.com/technetwork/java/javase/downloads/index.html 下载对应的 64 位 jdk，然后按照安装步骤完成安装。然后配置 Java 的环境变量，整个 jdk 配置就完成了。参考网址：http://jingyan.baidu.com/article/215817f7e3f2bd1eda1423f4.html。

(2) Eclipse 的安装

本系统采用的 Eclipse 版本是 eclipse luna。根据本机系统，从 http://www.eclipse.org/downloads/下载相应的 Eclipse，然后解压缩，直接启动 eclipse.exe 即可。安装过程参考网址：http://jingyan.baidu.com/article/154b46315136b028ca8f41f5.html

(3) 数据库

现如今数据库种类多样，如 oracle、sqlserver、mysql 等等。考虑到是开发手机应用，所以本系统采用的是一种轻量级数据库 mysql。

2.3.2 其他相关技术

(1) 极光推送

什么是消息推送？通过互联网，定期为用户传送所需信息的一项新技术。推送技术通过自动传送信息给用户，减少了用户在网络上的搜索时间，从而可以有效地避免信息过载的问题。它还可以结合用户的兴趣爱好来进行信息过滤，帮助用户高效率地发掘有价值的信息。

本系统需要利用消息推送服务，为用户提供日程提醒。本系统采用第三方推送平台——极光推送，来实现推送功能。它是一个面向普通开发者的，免费的第三方消息推送服务。因此，开发者只需在客户端集成极光推送 SDK，就可以轻松地添加 Push 功能到 App 中。

(2) 大众点评开发者

本系统可以推送给用户预测地点的周边商圈。该功能同样应用了第三方平台——大众点评开发者。它提供了 API 接口，通过传入位置坐标，获得周边地点的推荐。

3 详细设计

3.1 系统框架设计

3.1.1 系统框架

图 3.1 显示了整个软件的系统框架，分为 Android 客户端、服务器端、和数据库。其中，Android 客户端包含了 3 个模块：GPS 数据采集模块、基本功能模块、推送模块；服务器端包含了 3 个模块：Web Service 模块、数据预处理模块、周期挖掘模块；数据库包含了用户 GPS 数据及用户的账户信息。具体流程如图 3.1。

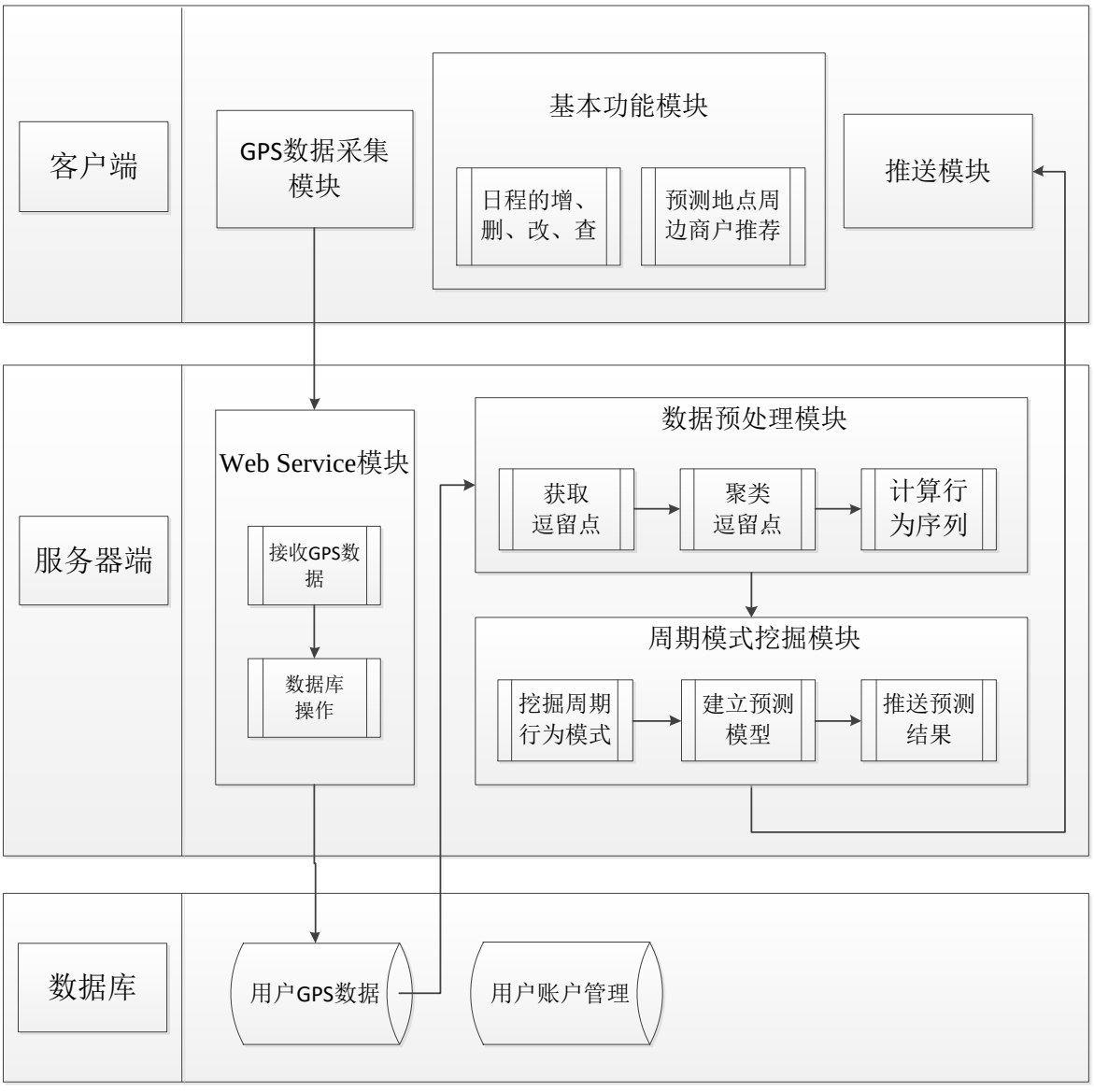


图 3.1 软件系统框架

3.1.2 系统功能

图 3.2 显示了该个性化日程管理软件的基本功能：添加事件提醒、删除事件提醒、修改事件提醒、查看事件提醒、查看帮助、预测并推送给用户明天可能的行程、推送预测地点的周边商圈。

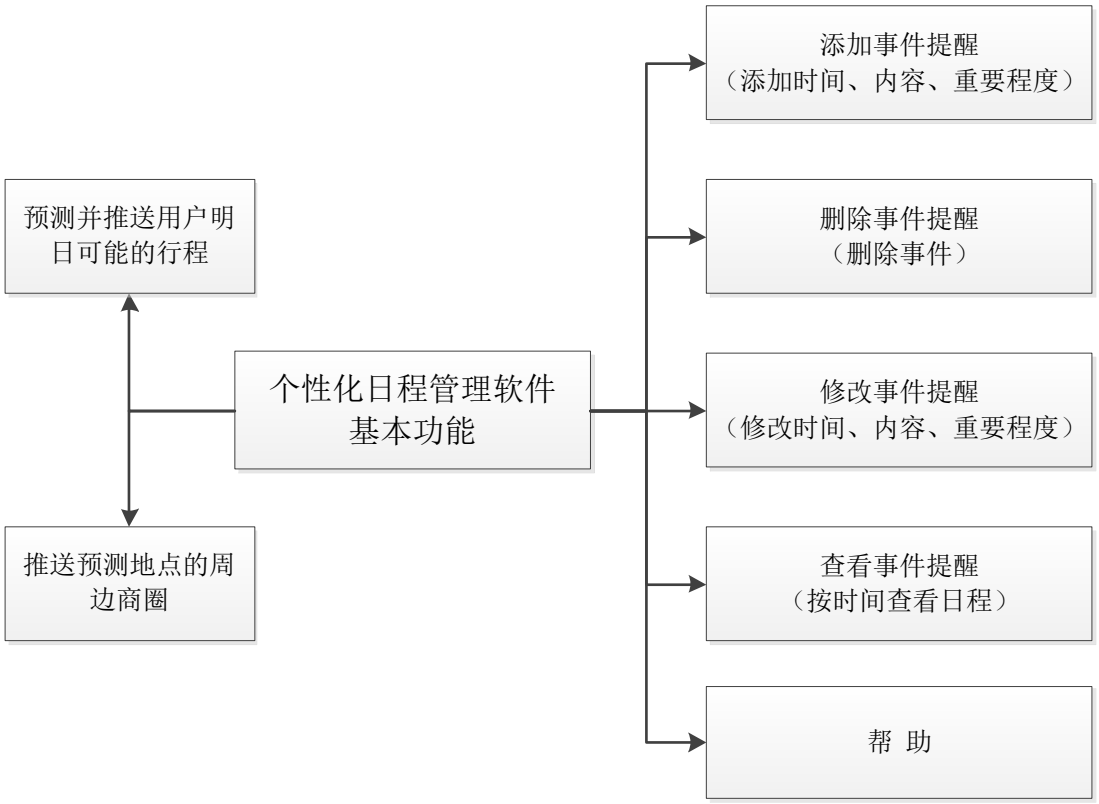


图 3.2 个性化日程管理软件的基本功能

3.2 Android 客户端

3.2.1 混合模式移动应用

目前有很多移动端应用都是基于混合模式移动应用（Hybrid App）的方式开发的，比如工商银行、百度搜索、还有一些游戏等等。Hybrid App 通常基于第三方跨平台移动应用引擎框架进行开发。这些引擎框架一般使用 HTML5 和 JavaScript 作为编程语言，调用引擎封装的底层功能如照相机、二维码等等。

HTML5 最初是用来编写互联网网页的，随着移动互联网的发展，HTML5 也可以应用在手机客户端的界面开发上了。目前，这种开发一般都是通过本地封装器完成的，例如 Cordova，它使得 HTML 和 JavaScript 可以在手机平台上驱动 app。本系统就是采用这种混合模式（HTML5 + Cordova）进行开发，其界面会更加美观，开发起来更简单。

3.2.2 数据库设计

数据库用的是 SQLite，它是一种轻量级的小型数据库。用来临时保存采集到的用户位置信息，

然后再在 wifi 环境下，将数据上传至数据库。如表 3.1 所示。

表 3.1 用户位置信息

字段名称	类型	大小	说明
userimei	varchar(20)	20	用户手机识别码
longitude	double	20	经度
latitude	double	20	纬度
time	Datetime	—	时间

3.2.3 Service

采集 GPS 数据是本系统比较主要的模块。本系统采用 Android 自带的函数获取位置。通过系统提供的 LOCATION_SERVICE 中的 GPS_PROVIDER 来进行获取。其中 Location 类中我们会用到的参数有：经度、维度、时间。

3.3 服务器端

3.3.1 数据库设计

服务器端的数据库需要存储用户账户信息，以及用户 GPS 数据。其中，userimei 为用户手机识别码，即唯一可以识别用户身份的用户 ID。IMEI（International Mobile Equipment Identity，移动设备国际识别码）是手机的唯一识别号码，我们利用它来定位用户的位置，并将 GPS 数据上传到数据库中。如图 3.3 所示，为该数据库设计的 E-R 图。

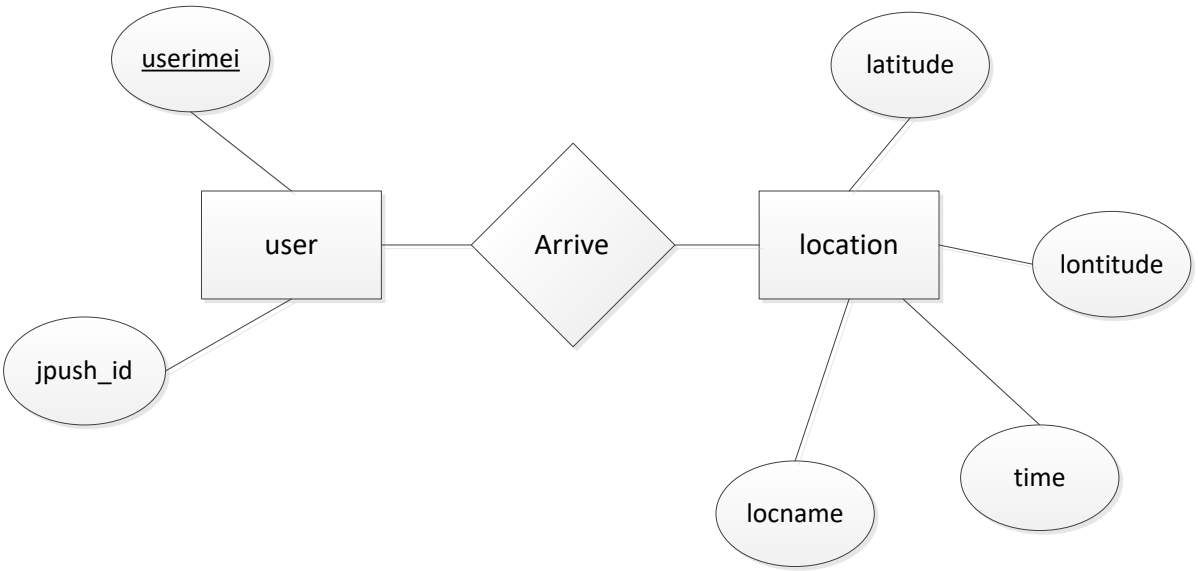


图 3.3 ER 图

根据图 3.3 的 ER 图来设计数据库。本系统共包含两张表：User 表和 Location 表，如表 3.2 和表 3.3 所示：

表 3.2 User 表（用户账户管理）

字段名称	类型	大小	说明
userimei	Varchar	20	用户手机识别码
jpush_id	Varchar	13	推送号

表 3.3 Location 表（用户 GPS 数据管理）

字段名称	类型	大小	说明
userimei	Varchar(20)	20	用户手机识别码
latitude	double	20	经度
longtitude	double	20	纬度
time	Datetime	—	纪录时间
locname	Varchar(30)	30	地点名

3.4.2 具体实现

首先，服务器端的 Web Service 模块需要接收客户端发送的 GPS 数据，并把它们存入数据库中。图 3.4 显示了实现该功能的所有类的列表。大致的流程是这样的：①用户填写表单（form 类）；②controller 类用来接受用户提交的表单；③接收后，调用 Service 类中的函数，进行相关的判断或数据库操作；④dao 类专门用于数据库操作（增删改查）。domain 类里的成员变量与数据库表一一对应。具体代码见附录。

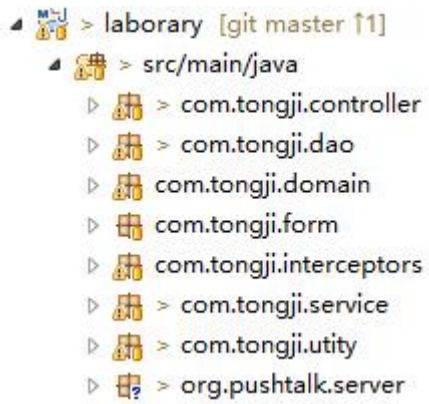


图 3.4 Web Service 模块

其次，服务器数据预处理模块需要处理用户 GPS 数据，周期模式挖掘模块应用了周期模式挖掘算法，预测用户的日程。图 3.5 显示了实现该功能的所有类的列表。大致的流程是这样的：①调用 staypointdetection 类计算用户的逗留点（详细过程见 4.3）；②调用 optics 类对逗留点进行聚类（详细过程见 4.4）；③调用 seq01 类将每个类转换成“01”序列（详细过程见 4.5）；④调用 fptree 类挖掘用户行为周期模式（详细过程见 4.6）。具体代码见附录。

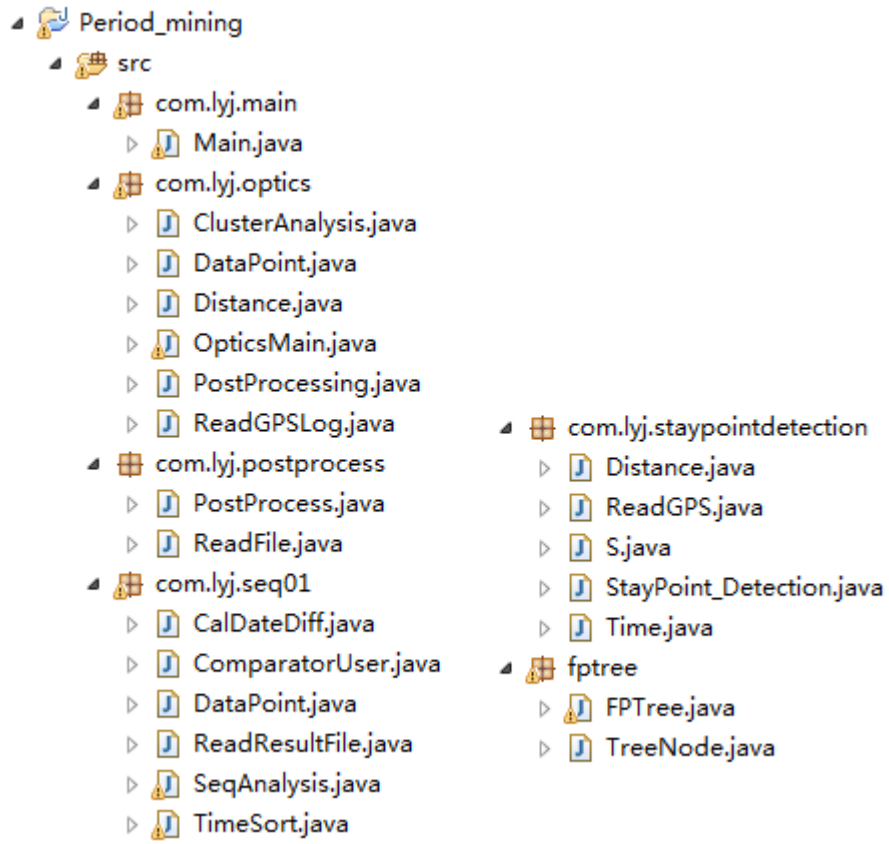


图 3.5

4 周期模式挖掘

4.1 数据集介绍

为了方便测试算法，本系统使用了微软提供的一个数据集。这个数据集中包含了 182 个用户超过五年（2007 年 4 月-2012 年 8 月）的 GPS 轨迹。数据集中的文件均为.PLT 格式。每个用户的 GPS 轨迹记录按照时间戳的顺序进行排列。每一个文件中的格式相同：第 1~6 行是无用信息，可以忽略。从第 7 行开始，每一个行表示一条 GPS 轨迹记录，每条记录包含 7 个参数，依次是：经度（十进制）、纬度（十进制）、0（所有记录中这个参数均为 0）、高度（英尺）、天数（距离 12/30/1899 的天数）、日期（字符串格式 yyyy-mm-dd）、时间（字符串格式 hh:mm:ss）。图 4.1 显示了部分数据集。

我们所需要的参数是经度、纬度、日期和时间。因此，首先需要从数据集中提取出这部分有用的信息，且保证每一个行依旧表示一条 GPS 轨迹记录，并按照时间戳顺序排列。如图 4.2 所示。

```
Geolife trajectory
WGS 84
Altitude is in Feet
Reserved 3
0,2,255,My Track,0,0,2,8421376
0
39.984702,116.318417,0,492,39744.1201851852,2008-10-23,02:53:04
39.984683,116.31845,0,492,39744.1202546296,2008-10-23,02:53:10
39.984686,116.318417,0,492,39744.1203125,2008-10-23,02:53:15
39.984688,116.318385,0,492,39744.1203703704,2008-10-23,02:53:20
39.984655,116.318263,0,492,39744.1204282407,2008-10-23,02:53:25
39.984611,116.318026,0,493,39744.1204861111,2008-10-23,02:53:30
39.984608,116.317761,0,493,39744.1205439815,2008-10-23,02:53:35
39.984563,116.317517,0,496,39744.1206018519,2008-10-23,02:53:40
```

图 4.1 部分 GPS 数据集

```
39.984702,116.318417,2008-10-23 02:53:04
39.984683,116.31845,2008-10-23 02:53:10
39.984686,116.318417,2008-10-23 02:53:15
39.984688,116.318385,2008-10-23 02:53:20
39.984655,116.318263,2008-10-23 02:53:25
39.984611,116.318026,2008-10-23 02:53:30
39.984608,116.317761,2008-10-23 02:53:35
39.984563,116.317517,2008-10-23 02:53:40
```

图 4.2 部分 GPS 数据集

4.2 数据预处理

数据预处理是指在主要的处理以前先对数据进行一些处理。在本文中，采集到的用户 GPS 数据量庞大，并且杂乱无规律，无法直接进行周期模式的挖掘。因此，需要通过一系列的方法将这些数据转换成可供挖掘的数据。其过程为：

第一步，通过 Stay Point 算法获得用户的逗留点（详细过程见 4.3）；

第二步，通过 OPTICS 算法对逗留点进行聚类，给每个类贴上标签（详细过程见 4.4）；

第三步，根据聚类的结果，计算每个类的时间序列，形成相应的 01 序列（详细过程见 4.5）。

4.3 Stay Point 算法

4.3.1 相关定义

在这个部分我们先定义一些名词，包括 GPS 日志、GPS 轨迹、逗留点。

GPS 日志：如图 4.2 所示，GPS 日志是指一系列 GPS 点，记为$P = \{p_1, p_2, \ldots, p_n,\}$。对每个 GPS 点$p_i \in P$，包含了三个参数——经度$(p_i.Lngt)$、纬度$(p_i.Lat)$、时间戳$(p_i.T)$。

GPS 轨迹：如图 4.3 所示，在一个二维空间中，按照时间戳的顺序，将 GPS 点连接起来形成的一个轨迹（Traj）。

	经度	纬度	时间
P1：	Lngt1	Lat1	T1
P2：	Lngt2	Lat2	T2
	……	……	……
Pn：	Lngtn	Latn	Tn

图 4.3 部分 GPS 日志

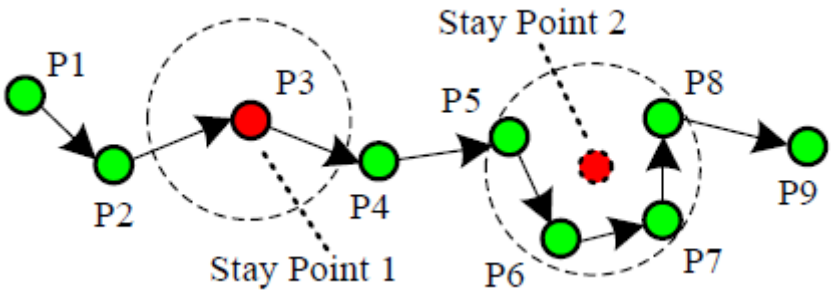


图 4.4 部分 GPS 轨迹和逗留点

逗留点（Stay Point）：一个逗留点表示用户逗留了一段时间的一个地理区域。相对于原始 GPS 数据，逗留点具有特定的语义，比如我们居住工作的地方、吃饭的地方或是旅游的景点等。图 4.4 中显示了两个逗留点（红色的点）。第一个逗留点中只包含了 GPS 点 P3，表示用户在 P3 逗留了一段超过某个阈值的时间。一般情况，当用户走进了一幢大楼，会与卫星信号失去联系，直到离开大楼才恢复定位。第二个逗留点中包含了 GPS 点 P5、P6、P7、P8，表示一个用户在一个特

定的空间区域徘徊了一段时间。因此，我们需要计算这个几个坐标点的平均值，作为逗留点的新坐标。这种情况一般发生在用户外出旅游，在某个景点来回进行参观。

4.3.2 算法概述

那么，为什么要获取逗留点呢？第一，如果我们直接聚合用户的 GPS 数据，计算量会非常大，因为用户的 GPS 数据相对于逗留点的数量要多的多；第二，由于 GPS 采集装置时常会在室内失效，所以会丢失用户在室内的 GPS 数据，这样会导致我们无法聚类像家、购物商场这样的室内地点；相对地，反而会提取像十字路口这样没有特定语义的室外地点，这样的结果对我们的推荐是没有任何意义的。

利用 Stay Point 算法[17]可以自动地根据用户的 GPS 数据检测出逗留点，即寻找用户逗留时间超过某个阈值的空间区域。比如，如果一个用户在某个 200 米范围的区域内，待的时间超过 30 分钟，就把这个区域看作是一个逗留点。每个逗留点 S 可以表示为：$(S.\bar{x}, S.\bar{y}, S.arvT, S.levT)$。其中，$(S.\bar{x}, S.\bar{y})$表示逗留点的平均坐标，$S.arvT$表示到达逗留点的时间，$S.levT$表示离开逗留点的时间。

4.3.3 具体实现

算法 4.1 Stay Point 算法
输入：用户 GPS 日志 P，距离阈值distThreh，时间阈值timeThreh 输出：Stay Point 集合 SP={S}
方法： 1：$i = 0, pointNum = \|P\|;$ //计算 GPS 点的个数 2：**while**$(i < pointNum)$ do 3：　　$j = i + 1;$ 4：　　**while**$(j < pointNum)$ do 5：　　$dist = Distance(p_i, p_j);$ //计算两点的距离 6：　　**if**$(dist > distThreh)$ 7：　　$\Delta T = p_j.T - p_i.T;$ //计算两点的时间差 8：　　**if**$(\Delta T > timeThreh)$ 9：　　$S.coord = ComputMeanCoord(\{p_k \mid i \le k \le j\});$ 10：　　$S.arvT = p_i.T;$ 11：　　$S.levT = p_j.T;$ 12：　　$SP.insert(S);$ 13：　　$i = j;$ break; 14：　　$j = j + 1;$ 15：**return** SP;

算法 4. 1 Stay Point 算法伪代码

算法 4.1 显示了 Stay Point 算法的伪代码。它的输入分别有：用户 GPS 数据、距离阈值、时间阈值。它的输出为 Stay Point 集合。首先，计算用户 GPS 日志中 GPS 点的个数（第 1 行）。从第一个 GPS 点开始，进行逐个扫描。对每两个 GPS 点p_i、p_j，先计算两点之间的距离（第 2~5 行）。由于是 GPS 坐标，计算距离时按照公式 4.1 进行转换：

$$p_i.x = p_i.lngt \times \pi \div 180.0，\ p_i.y = p_i.lat \times \pi \div 180.0$$
$$p_j.x = p_i.lngt \times \pi \div 180.0，\ p_j.y = p_i.lat \times \pi \div 180.0$$
$$dist = 2 \times earthRadius \times \sin^{-1}\sqrt{\left(\frac{\sin(p_i.x - p_j.x)}{2}\right)^2 + \cos(p_i.y) \times \cos(p_j.y) \times \left(\frac{\sin(p_i.y - p_j.y)}{2}\right)^2} \quad (4.1)$$

其中，地球半径$earthRadius = 6371009$。

如果距离小于distThreh，那么继续计算下一个 GPS 点p_{j+1}与p_i之间的距离，直到找到 GPS 点p_t满足 $Distance(p_i, p_t) > distThreh$。进一步，计算$p_i$与$p_t$之间的时间差（第 6~7 行），如果时间差大于timeThreh 时（第 8~12 行），表示p_i和p_t之间所有的 GPS 点可以合并成一个逗留点，计算逗留点的坐标，即取p_i到p_t之间所有的 GPS 点的坐标的平均值，用$(S.\bar{x}, S.\bar{y})$表示（第 9 行）。逗留点的起始时间为第一个 GPS 点p_i的时间戳（第 10 行），逗留点的结束时间为最后一个 GPS 点p_t的时间戳（第 11 行）。最后将该逗留点添加到逗留点集合中去（第 12 行）。这样就完成了一个逗留点的计算。下一次逗留点的计算从 GPS 点p_t和p_{t+1}开始，重复上述过程即可。

4.3.4 实验结果

为了测试算法，我使用了 4.1 节中介绍的数据集。首先，提取数据集中所需的参数，也就是经度、纬度和时间戳。对多个用户的 GPS 数据分别进行了测试。设定时间阈值为 30 分钟，距离阈值为 200 米，即求出用户在 200 米区域范围逗留超过 30 分钟以上的地点。下面呈现三组实验测试结果，具体如下：

（1）000 号用户的 GPS 数据的起始时间为 2008-10-23，结束时间为 2009-07-05。结果显示 000 号用户共有 589 个逗留点。图 4.5 为部分数据截图。每一条记录的格式如下：逗留点的名称（序号），经度，纬度，到达逗留点的时间，离开逗留点的时间。

```
585,39.99285017499997,116.32675525000005,2009-07-04 09:11:48.0,2009-07-04
09:43:23.0
586,40.00044121052631,116.32609115789472,2009-07-04 09:48:08.0,2009-07-05
02:53:52.0
587,39.95076512499999,116.32801787500004,2009-07-05 03:49:37.0,2009-07-05
04:44:25.0
588,40.080349276315765,116.23627809210531,2009-07-05 05:32:36.0,2009-07-05
06:24:21.0
589,40.080459255154594,116.23644316237106,2009-07-05 06:24:21.0,2009-07-05
07:08:56.0
```

图 4.5 部分数据截图

（2）003 号用户的 GPS 数据的起始时间为 2008-10-23，结束时间为 2009-07-05。结果显示 003 号用户共有 1279 个逗留点，图 4.6 为部分数据截图。

```
1269,40.000918,116.33513674999999,2009-07-03 02:56:50.0,2009-07-03
03:47:53.0
1270,40.00944350666666,116.32263904000007,2009-07-03 03:56:40.0,2009-07-03
04:23:40.0
1271,39.999853952380946,116.32669885714292,2009-07-03 04:39:10.0,2009-07-03
07:23:41.0
1272,40.007770902173924,116.31947022101451,2009-07-03 07:49:16.0,2009-07-03
08:30:43.0
1273,40.002883,116.3282385,2009-07-03 08:37:18.0,2009-07-04 04:26:34.0
1274,39.99971783720929,116.32711151162789,2009-07-04 04:48:10.0,2009-07-04
09:09:08.0
1275,39.99285017499997,116.32675525000005,2009-07-04 09:11:48.0,2009-07-04
09:43:23.0
1276,40.00044121052631,116.32609115789472,2009-07-04 09:48:08.0,2009-07-05
02:53:52.0
1277,39.95076512499999,116.32801787500004,2009-07-05 03:49:37.0,2009-07-05
04:44:25.0
1278,40.080349276315765,116.23627809210531,2009-07-05 05:32:36.0,2009-07-05
06:24:21.0
1279,40.080459255154594,116.23644316237106,2009-07-05 06:24:21.0,2009-07-05
07:08:56.0
```

图 4.6 部分数据截图

（3）010 号用户的 GPS 数据的起始时间为 2007-08-04，结束时间为 2009-03-21。结果显示 010 号用户共有 335 个逗留点，图 4.7 为部分数据截图。

```
330,39.978436046478905,116.34168154507003,2009-03-07 09:04:53.0,2009-03-07
09:50:27.0
331,39.99183693604649,116.32579738953483,2009-03-07 10:08:20.0,2009-03-15
09:31:33.0
332,39.90597620258491,116.44142341583148,2009-03-15 11:03:06.0,2009-03-15
13:52:28.0
333,39.99252022131148,116.32689297540982,2009-03-15 13:56:37.0,2009-03-21
03:21:56.0
334,39.939404122222236,116.3474520222222,2009-03-21 03:32:41.0,2009-03-21
04:00:24.0
335,39.86249027560971,116.37280756829267,2009-03-21 04:12:23.0,2009-03-21
04:35:06.0
```

图 4.7 部分数据截图

4.4 Optics 算法

4.4.1 相关定义

在这个部分我们先定义一些名词，包括邻域、邻域密度阈值、核心距离、可达距离。

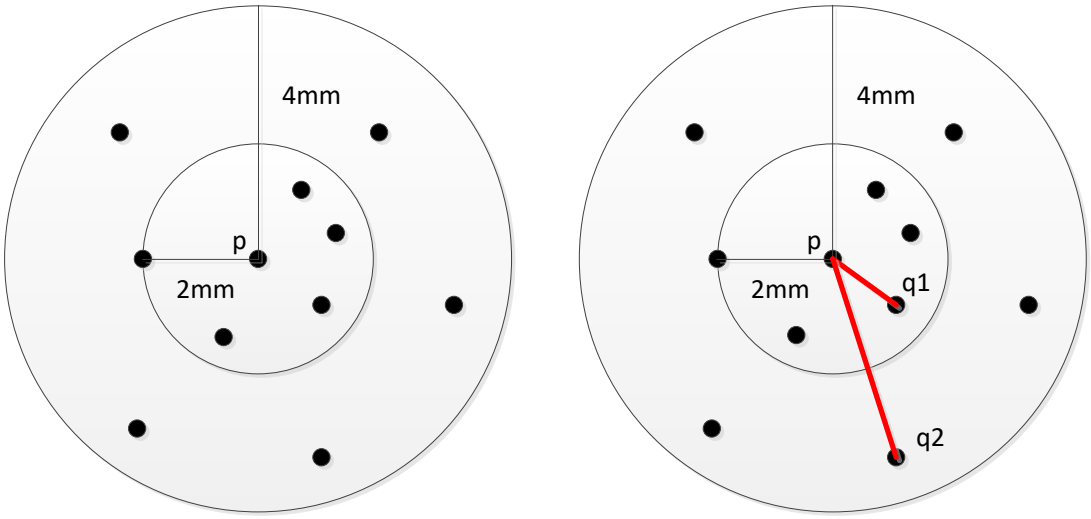
邻域ε：对象 o 的 ε-邻域是以 o 为中心、以ε为半径的空间。

邻域密度阈值（MinPts）：邻域内包含的对象数的阈值。

对象 p 的核心距离（core-distance）：最小的ε′，使得 p 的ε′-邻域内至少包含 Minpts 个对象。如果 p 不是核心对象，那么 p 的核心距离没有定义。

对象 q 到对象 p 的可达距离（reachability-distance）：使 p 从 q 密度可达的最小半径。因此，从 q 到 p 的可达距离是 max{$core\_distance(q), dist(p,q)$}。

例 4.4.1：如图 4.8 所示，进一步阐明“核心距离”与“可达距离”的定义。假设邻域半径为 4mm，密度阈值为 6。对象 p 的核心距离是 p 与 p 的第 5 个最近的数据对象之间的距离，假设这里是 2mm，如图 4.8（a）所示。那么 p 到 q1 的可达距离就是 2mm，p 到 q2 的可达距离等于 p 到 q2 的欧式距离，如图 4.8（b）所示。



（a）p的核心距离　　（b）q1、q2到p的可达距离

图 4.8 核心距离与可达距离的定义

4.4.2 算法概述

OPTICE 聚类[18]是一种基于密度的聚类算法。得到逗留点集合$SP = \{S_1, S_2, \ldots, S_m,\}$，其中，m 为逗留点的个数。在挖掘用户周期之前，要先对逗留点集合进行聚类。为什么要进行聚类呢？观察逗留点集合，可以发现一些逗留点的坐标非常接近，它们很可能处于了同一个地方或区域。聚类可以有效地将逗留点进行合并，从而获得一个地点集合。并且这更有利于我们获得用户的行为序列。

基于密度的聚类算法有 DBSCAN、OPTICS。为什么选择 OPTICS 算法进行聚类呢？因为 DBSCAN 算法的输出结果在极大程度地依赖于它的两个输入参数：领域半径和邻域密度阈值。然而，参数的设置通常基于用户的经验值，难以确定，特别是解决实际生活中的问题。OPTICS 聚

类分析克服了这个缺点。它不显示地产生数据聚类，而是对集中的对象进行排序，输出一个有序的列表。在这里参数的细微变化并不会影响到样本点的相对输出顺序，因此对聚类结果没有很大的影响。从这个角度考虑，最终选择了 OPTICS 算法进行聚类。

4.4.3 具体实现

算法 4.2 OPTICS，一种基于密度的聚类算法
输入：逗留点集合 SP，领域半径 ε，邻域密度阈值 MinPts 输出：有序逗留点集合 dpList
方法： 1：for each p in SP do 2：　　if ! p ∈ dpList 3：　　if p 是核心对象 4：　　计算 p 邻域半径内的所有对象; 5：　　dpQue.add(p); 6：　　dpQue.sort(ReachableDist); 7：　　while(dpQue ! = null) 8：　　从 dpQue 取$Min_{ReachableDist}$的对象p′来拓展，dpList.add(p′); 9：　　if p′是核心对象 10：　　计算p′邻域半径内的所有对象，记为集合 ArvObj; 11：　　for each q in ArvObj do 12：　　if q ∈ dpList break; 13：　　else if q ∈ dpQue 14：　　重新计算 ReachableDist; 15：　　dpQue.sort(ReachableDist); 16：　　else dpQue.add(q); 17：　　dpQue.sort(ReachableDist); 18：输出 dpList;

算法 4.2 OPTICS 算法伪代码

算法 4.2 显示了 OPTICS 算法的伪代码。它的输入分别为：逗留点集合、领域半径、邻域密度阈值。它的输出为有序逗留点集合。首先需要创建两个队列，有序队列 dpQue 和结果队列 dpList。其中，有序队列用来临时存放核心对象及其直接可达对象的，并且按照可达距离升序排列。结果队列按照输出次序存放逗留点。

算法开始从逗留点集合中的第一个逗留点开始，进行逐个扫描（第 1 行）。如果集合中所有的样本点都处理完毕，则算法结束。第二步，选择一个未处理的逗留点（即不在结果队列中），

判断它是否为核心对象（即是否能找到直接密度可达点）。如果是，则将其放入有序队列中，并将有序队列按照可达距离升序排序（第 2~6 行）。第三步，如果有序队列不为空（第 7 行），从有序队列中取出可达距离最小的逗留点进行拓展，并将其放入结果队列中（第 8 行）。判断该拓展点是否是核心对象，如果不是，继续拓展有序队列中的下一个逗留点，否则计算该拓展点的所有直接密度可达点（第 9~10 行）。接下来，对每一个直接密度可达点进行处理，如果它已经存在于结果队列中，则不处理（第 12 行）；如果它存在于有序队列中，则重新计算可达距离，如果新的可达距离小于旧的，则取代旧的，并对有序队列重新排序（第 13~15 行）；如果它既不存在于结果队列中，也不存在于有序队列中，则将其插入有序队列，并对有序队列重新排序（第 16~17 行）。最后，如果集合中所有的点都处理完毕，且有序队列为空，则算法结束，输出有序逗留点集合（第 18 行）。

4.4.4 实验结果

为了测试算法，沿用 4.3.4 中得到的结果，对多个用户的逗留点集合进行聚类。设定邻域半径 100 米，邻域密度阈值为 8。下面呈现三组实验测试结果，具体如下：

（1）000 号用户的逗留点集合最后聚成了 6 类，图 4.9 是部分数据截图。每一条记录的格式如下：逗留点的名称（序号），到达时间，离开时间，逗留点所属的类别名称（序号）。

```
111,2009-04-06 08:08:56.0,2009-04-06 09:48:02.0,5
287,2009-05-11 09:39:43.0,2009-05-11 10:05:52.0,5
212,2009-04-27 10:32:23.0,2009-04-27 12:43:04.0,6
299,2009-05-13 09:03:14.0,2009-05-13 10:29:17.0,6
359,2009-05-25 12:02:41.0,2009-05-25 12:36:24.0,6
419,2009-06-05 10:10:03.0,2009-06-05 13:08:56.0,6
444,2009-06-09 12:02:42.0,2009-06-09 14:05:52.0,6
487,2009-06-16 12:00:23.0,2009-06-16 14:02:45.0,6
```

图 4.9 部分数据截图

（2）003 号用户的逗留点集合最后聚成了 19 类，图 4.10 是部分数据截图。

```
920,2009-04-29 15:44:10.0,2009-04-30 03:23:57.0,17
933,2009-05-01 11:47:49.0,2009-05-02 08:05:22.0,17
686,2009-03-11 22:49:38.0,2009-03-12 00:51:48.0,18
639,2009-03-03 22:48:49.0,2009-03-04 00:53:58.0,18
806,2009-04-10 08:55:33.0,2009-04-10 13:06:03.0,18
811,2009-04-11 08:42:02.0,2009-04-11 13:06:49.0,18
815,2009-04-12 08:47:14.0,2009-04-12 13:06:20.0,18
820,2009-04-13 08:42:23.0,2009-04-13 13:06:08.0,18
823,2009-04-13 22:51:35.0,2009-04-14 04:43:25.0,18
824,2009-04-14 04:43:25.0,2009-04-14 05:05:45.0,18
798,2009-04-06 03:49:05.0,2009-04-06 04:43:25.0,19
261,2008-12-11 00:06:44.0,2008-12-11 00:29:04.0,19
```

图 4.10 部分数据截图

（3）010 号用户的逗留点集合最后聚成了 5 类，图 4.11 是部分数据截图。

```
258,2008-10-17 00:14:37.0,2008-10-17 00:42:10.0,4
275,2008-11-06 23:48:40.0,2008-11-07 00:14:22.0,4
264,2008-10-20 00:01:47.0,2008-10-20 00:29:48.0,4
230,2008-10-08 00:37:35.0,2008-10-08 01:05:18.0,4
238,2008-10-09 22:50:19.0,2008-10-09 23:10:30.0,5
146,2008-09-16 22:13:33.0,2008-09-16 22:40:44.0,5
183,2008-09-27 23:24:07.0,2008-09-27 23:45:35.0,5
257,2008-10-16 22:50:01.0,2008-10-16 23:10:06.0,5
263,2008-10-19 22:31:17.0,2008-10-19 22:55:17.0,5
285,2008-12-07 10:34:10.0,2008-12-07 11:00:01.0,5
305,2009-01-03 14:23:17.0,2009-01-03 15:09:19.0,5
312,2009-01-11 13:51:04.0,2009-01-11 14:41:31.0,5
161,2008-09-22 13:05:06.0,2008-09-22 13:48:12.0,5
144,2008-09-16 14:04:24.0,2008-09-16 14:56:27.0,5
```

图 4.11 部分数据截图

装

订

线

4.5 获得时间序列

时间序列是按照时间顺序排列的一组数据，它在人类社会中的各个领域广泛存在，如在金融证券市场中每天起伏变化的股票价格；在气象预报中，某个地区每天的气温与气压的度数；在生物医学中，某个病人在每个时刻的心跳变化等等。对时间序列进行周期模式的挖掘，可以有效地帮助我们获得事物运动、发展、变化的规律。

4.5.1 相关定义

时间序列：$S = s_1 s_2 \ldots s_n$，其中，S 为某个地点的时间序列，s_i表示在时间戳 i 时的状态，并按照时间戳的进行排序。

4.5.2 具体实现

得到聚类集合$\{C_1, C_2, \ldots, C_m,\}$，其中，m 为聚类个数。针对某一个类$C_i$，先要将类中的逗留点按照到达时间和离开时间进行排序。然后再以“天”为单位，将其转换成“01”序列，‘1’表示用户到达了该地，‘0’表示用户未到达该地。

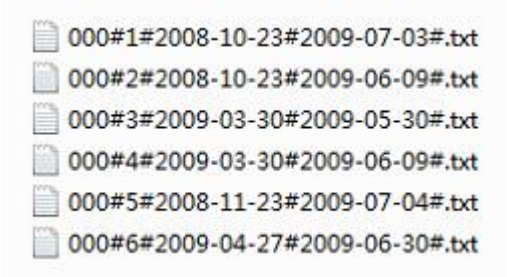
例 4.5.1：假设有两条连续的记录分别为“123,2015-10-12 00:14:37.0,2015-10-15 00:42:10.0,4”，“451,2015-10-17 00:24:57.0,2015-10-17 00:25:40.0,4”。前者表示用户从 2008-10-12 到 2008-10-15 这 4 天会到 4 号这个地方，后者表示用户 2008-10-17 这天会到 4 号这个地方，因此用户 2008-10-16 这天未到 4 号这个地方。那么转换为“01”序列为：111101。

4.5.3 实验结果

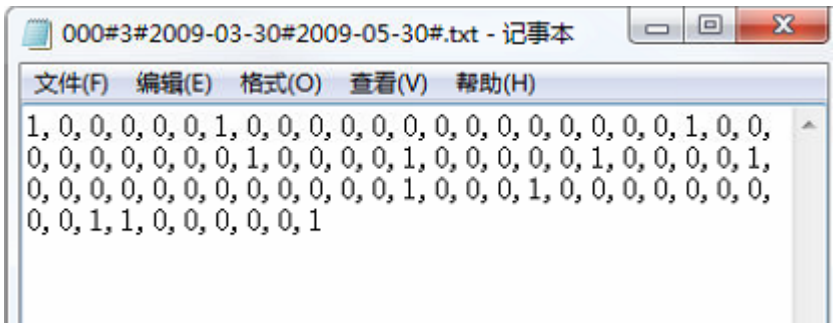
沿用 4.4.4 中得到的结果，计算多个用户的时间序列。为每一个用户所聚得的每一个类新建一个文件，文件名的格式为：用户 ID#类别名（序号）#到达时间#离开时间#.txt，文件内容为 01 序列，用‘,’隔开。下面呈现三组实验测试结果，具体如下：

（1）图 4.12 是 000 号用户的部分数据截图。图 4.12（a）是 000 号用户的 6 个类形成的所有

文件。图 4.12（b）是 000 号用户 3 号地点形成的 01 序列。



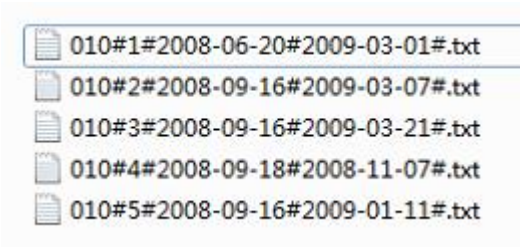
（a）000 号用户的所有文件



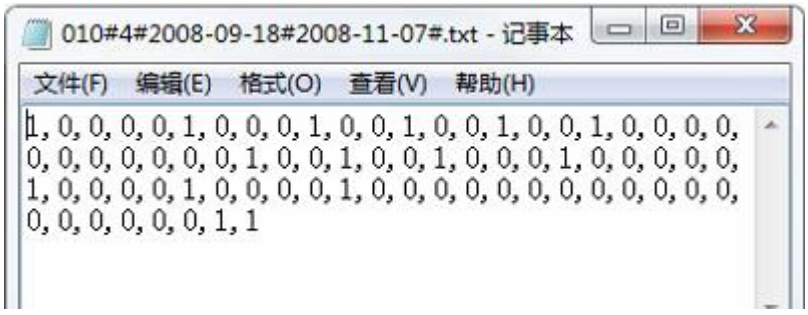
（b）000 号用户 3 号地点的 01 序列

图 4.12 部分数据截图

（2）图 4.13 是 010 号用户的部分数据截图。图 4.13（a）是 010 号用户的 5 个类形成的所有文件。图 4.13（b）是 010 号用户 4 号地点形成的 01 序列。



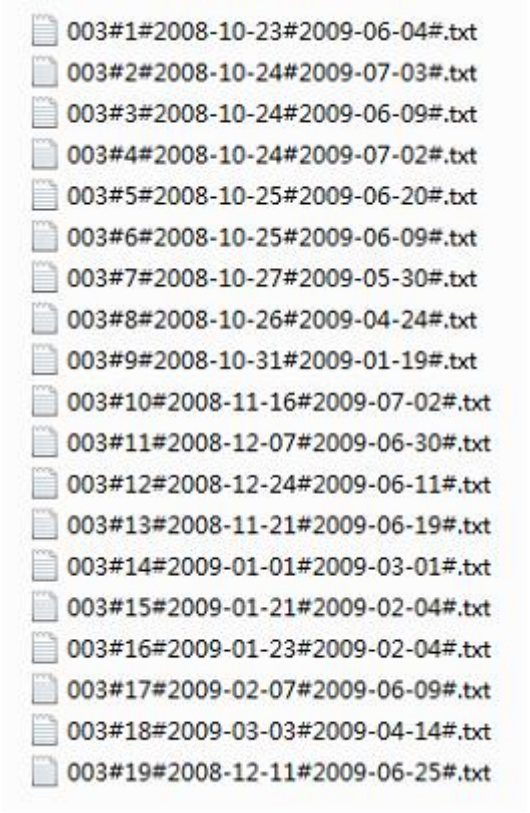
（a）010 号用户的所有文件



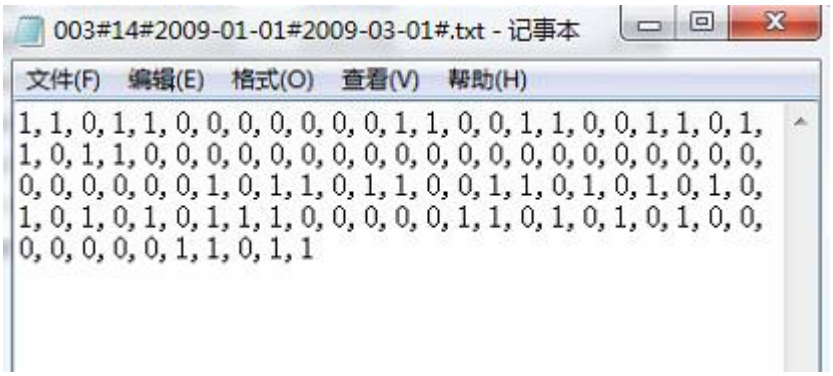
（b）010 号用户 3 号地点的 01 序列

图 4.13 部分数据截图

（3）图 4.14 是 003 号用户的部分数据截图。图 4.14（a）是 003 号用户的 19 个类形成的所有文件。图 4.14（b）是 003 号用户 14 号地点形成的 01 序列。



（a）003 号用户的所有文件



（b）003 号用户 3 号地点的 01 序列

图 4.14 部分数据截图

4.6 周期模式挖掘算法

4.6.1 相关定义

在这个部分我们先定义一些名词，包括周期模式、周期模式的置信度。

周期模式：假设时间序列$S = s_1 s_2 \ldots s_n$中某个状态s_i以周期 T 呈现出周期性的特征，周期模

式则是指某事件以周期 T、状态s_i的模式周期性地发生。

例 4.6.1：设时间序列S = abcadcabd。序列 S 中状态 a 存在以 3 为周期的周期模式为 P=a**（‘*’表示在该周期中，这个时间戳的状态不确定），表示周期 3 的第 1 个时间戳的确定状态是 a，而其他位置并不呈现出周期性规律。

周期模式的置信度：假设 P 为一个周期模式，P 的置信度$conf(P)$等于周期模式发生次数与周期段个数的商，如下公式 4.2 所示：

$$conf(P) = \frac{freq\_count(P)}{L} \quad (4.2)$$

其中，$freq_count(P)$是指周期模式发生的次数，即能够与该周期模式成功匹配的子序列的个数；L 是指周期段的数目。

例 4.6.2：对某教师去学校这个事件，其事件状态是{0,1}，其中‘1’表示该教师去学校了，‘0’表示该教师不去。若该教师去学校的时间序列为“0000100，1000100，0000100，1000100，0000100，1000100”，假设时间序列的周期为“周”或“周”的倍数，则存在两个周期模式：该教师每周五去学校和该教师双周周一也去学校。

4.6.2 算法概述

时间序列周期模式挖掘的基础是时间序列频繁模式的挖掘。频繁模式是指频繁地出现在数据集中的模式。比如，分析购物篮中的商品，通过挖掘啤酒、尿布这个项集，可以发现啤酒和尿布常常同时出现在购物篮中。那么，当一个顾客购买了啤酒就可以为他推荐尿布，这样顾客购买的可能性会比较大，从而提高销售额度。因此，频繁模式的挖掘已经广泛地应用于各个领域，它可以有效地分析用户的行为，并做出预测。

第一，频繁序列模式挖掘的核心就在于发现序列中存在的所有频繁项集，常见的频繁项集挖掘算法有 Apriori 算法和 FP-Growth 算法。Apriori 算法通过不断构造候选集、筛选候选集，来挖掘频繁项集。但该算法的计算量很大，特别是当原始数据量较大时，会产生大量的候选项集。相对地，FPGrowth 算法[18]只需扫描原始数据集两次，通过构造 FP-tree 对原始数据进行压缩，提高效率。因此，该周期模式挖掘算法将基于 FPGrowth 算法进行。

第二，在实际生活中，用户所产生的行为序列往往是非严格多周期的时间序列，其中包含了两个重要信息：非严格和多周期。一方面，例 4.6.2 展示的就是用户行为序列是多周期的情况，该教师的行为序列中包含了两个周期模式。另一方面，用户在某个时间段中的周期性行为可能不是 100%发生的，即用户的周期性行为往往不是严格按照周期频率发生的。例 4.6.3 展示了发生概率大于周期模式置信度的周期模式的挖掘。因此，该周期模式挖掘算法能够处理用户行为是非严格多周期的时间序列。

例 4.6.3：若某教师去学校的时间序列为“0001100，0001100，0100000，1000100，0000100，1000100”，假设时间序列的周期为“周”或“周”的倍数，周期模式的置信度为 60%，则存在一个周期模式：该教师每周五会去学校，其概率为 83.3。

第三，考虑到用户的行为通常是以“周”或“周”的倍数为周期的。相对于“月”、“日”，用户普遍更加偏向于以“周”为单位来规划自己的行程。比如，有的老师每周二、四去学校，有的老师则每双周周五去学校。因此，该周期模式挖掘算法挖掘以“周”为倍数的周期性行为，这样得到的结果更具有实际意义。

4.6.3 具体实现

算法 4.3 周期模式挖掘算法
输入：时间序列 S， 周期 T，周期模式置信度阈值 conf_thred 输出：周期模式

方法：
添加位置信息过程如下：
1：SubSeq = Split(S, T); //分割序列
2：for each Str in SubSeq
3： for each Item in Str
4： AddLocInfo(Item); //给每个子序列添加位置信息
 	
构造 FP-tree 过程如下：
5：CountAndSort(Item); //统计发生次数，去除小于置信度阈值的项，按降序排序
6：创建 FP-tree 根结点，记作 ROOT;
7：For each Item
8： if(N.name == Item.name)N 的计数增加 1;
9： else 创建新的结点 N;
10： N.name = Item.name;
11： N 计数等于 1;
 	
FP-tree 挖掘通过函数FP_growth(FP_tree, null)实现。函数实现如下：
12：function FP_growth(Tree, α)
13： if Tree 包含单个路径 P
14： for 路径 P 中的结点的每个组合(记作 β);
15： 产生模式β ∪ α，其计数等于β中结点的最小计数;
16： else for Tree 的头表中每个α_i
17： 产生一个模式 $\beta = \alpha_i \cup \alpha$，其计数等于$\alpha_i$的计数;
18： 构造 β 的条件模式基，然后构造 β 的条件 FP 树$Tree_\beta$;
19： if $Tree_\beta \neq null$
20： FP_growth($Tree_\beta$, β);
21：return 周期模式;

算法 4.3 周期模式挖掘算法

算法 4.3 显示了基于 FP-Growth 的周期模式挖掘算法，大体上分为 3 个部分，下面将集合一个例子，具体描述该算法的过程。

（1）第一次扫描时间序列

在第一次扫描时间序列的过程中，根据给定的周期长度（7、14、21、28）分割序列，形成多个子序列，给子序列中的每一项添加其在当前子序列中的位置信息（位置从 0 开始算），将其转换为“value[loc]”这样的形式，其中 value 表示状态值，loc 表示该状态在子序列中的位置（第 1~4 行）。统计各项的发生次数，在项集中去除小于置信度阈值的项，并按照发生次数从大到小排序（第 5 行）。具体操作见例 4.6.4（Part I）。

例 4.6.4（Part I）设时间序列为“0011110，1101110，0011111，1000110”，给定周期为 7，设周期模式的置信度为 60%。第一遍扫描时间序列，以长度 7 分割成 4 个子序列，分别为子序列中的每一项添加其位置信息，如表 4.1 所示。

表 4.1

子序列	项集
0011110	0[0], 0[1], 1[2], 1[3], 1[4], 1[5], 0[6]
1101110	1[0], 1[1], 0[2], 1[3], 1[4], 1[5], 0[6]
0011111	0[0], 0[1], 1[2], 1[3], 1[4], 1[5], 1[6]
1000110	1[0], 0[1], 0[2], 0[3], 1[4], 1[5], 0[6]

统计各项的发生次数，如表 4.2 所示。根据子序列的个数 4 以及周期模式置信度 60%，计算得到置信度阈值为 3，在项集中去除发生次数小于 3 的项，如表 4.3 所示；按照从大到小排序，如表 4.4 所示。

表 4.2

项	发生次数
0[0]	2
1[0]	2
0[1]	3
1[1]	1
0[2]	2
1[2]	2
0[3]	1
1[3]	3
1[4]	4
1[5]	4
0[6]	3
1[6]	1

表 4.3

项	发生次数
0[1]	3
1[3]	3
1[4]	4
1[5]	4
0[6]	3

表 4.4

序号	项集
0	1[4], 1[5], 0[1], 1[3], 0[6]
1	1[4], 1[5], 1[3], 0[6]
2	1[4], 1[5], 0[1], 1[3]
3	1[4], 1[5], 0[1], 0[6]

（2）第二次扫描时间序列

FP-tree 构造过程如下：首先，创建树的根结点，用“ROOT”标记。第二次扫描时间序列时，所有的项集以项的顺序沿着 ROOT 结点（不包含 ROOT 结点）往下匹配 FP-tree 中的结点。如果当前项与当前结点匹配，则该结点的计数加 1，并继续沿着该结点往下匹配项集中的下一个项（第 8 行）；若不匹配，则在其父结点下创建一个新的分枝结点，该结点的值等于项，结点的计数为 1（第 9~11 行）。

为了方便 FP-tree 的遍历，创建一个项头表，使得每项通过一个结点链指向它在树中的位置。其中，每一行包含项、项的发生次数以及指向 FP-tree 中最左边匹配该项的结点的指针。具体操作见例 4.6.4（Part II）。

例 4.6.4（Part II）基于表 4.4，构造如图 4.15 的 FP-tree 及项头表，FP-tree 中的结点表示为“value[loc]: count”，其中value[loc]表示项，count 表示发生次数。例如 0 号项集“1[4], 1[5], 0[1], 1[3], 0[6]”，共包含了 6 个项，因此在 ROOT 结点下构造依次包含 6 个结点的一个分枝：⟨1[4]：1⟩，⟨1[5]: 1⟩，⟨0[1]：1⟩，⟨1[3]：1⟩，⟨1[6]：1⟩。对于 1 号项集“1[4],1[5], 1[3], 0[6]”，其中结点⟨1[4]：1⟩，⟨1[5]: 1⟩与 0 号项集所产生的分枝匹配，因此它们共享同一个路径，并且结点的计数增加 1，即结点变为⟨1[4]：2⟩，⟨1[5]: 2⟩。而项 1[3]无法进行下一步匹配，因此创建一个新结点⟨1[3]：1⟩，它的父结点是⟨1[5]：2⟩。同样地，为 0[6]创建一个新结点⟨0[6]：1⟩，它的父结点是⟨1[3]：1⟩。然后，依次对 2 号项集、3 号项集继续进行相同的操作，最终可以得到如图 4.15 所示的 FP-tree 及项头表。

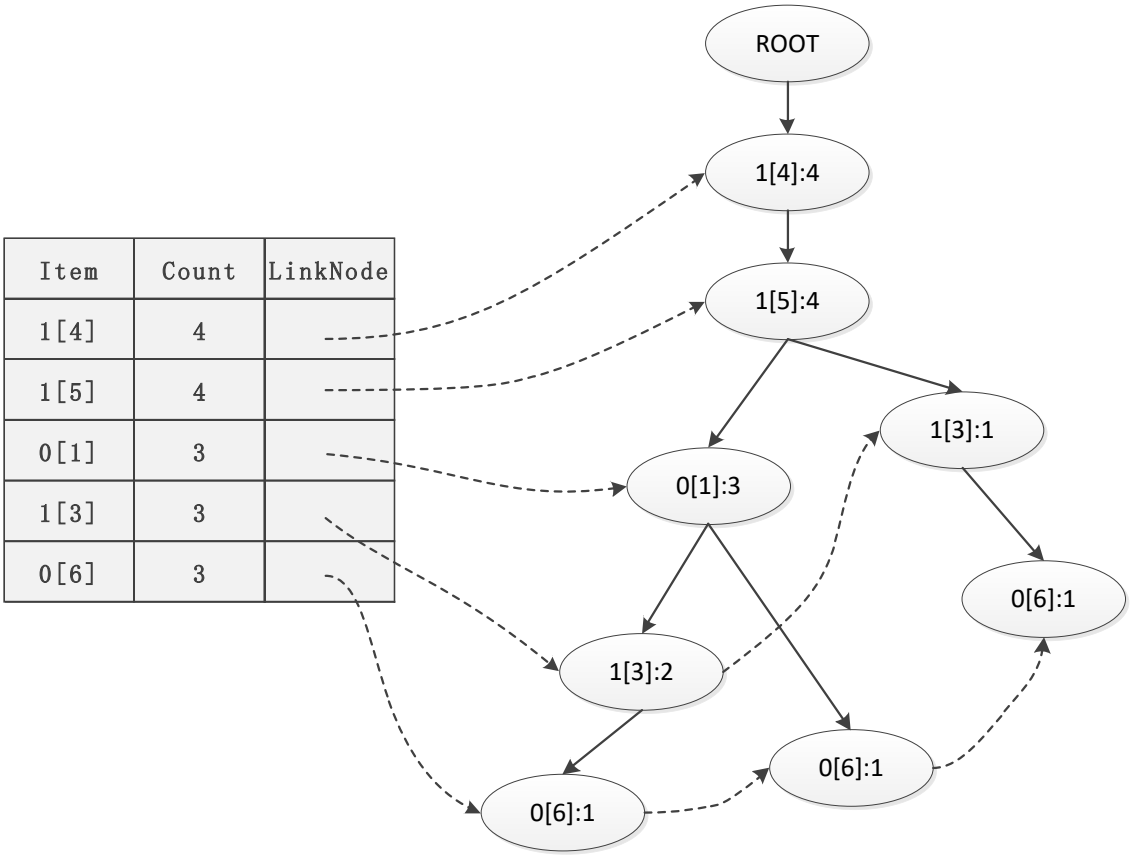


图 4.15 构造 FP-Tree

（3）挖掘 FP-tree

基于 FP-tree 以及项头表，采用分而治之的思想依次从项头表中的最后一项开始依次往上，寻找以该项为后缀的频繁模式。由长度为 1 的频繁模式开始，构造它的条件模式基。然后构造它的条件 FP-tree，并递归地在树上进行挖掘（第 12~21 行）。具体操作见例 4.6.4（Part III）。

例 4.6.4（Part III）图 4.13 的 FP-tree 的挖掘过程细节如下。首先考虑 0[6]，0[6]出现在 FP-tree 的三个分支中。这些分支形成的路径是⟨1[4], 1[5], 0[1], 1[3], 0[6]: 1⟩、⟨1[4], 1[5], 0[1], 0[6]: 1⟩、⟨1[4], 1[5], 1[3], 0[6]: 1⟩。因此，考虑以 0[6]为后缀，它的三个对象前缀路径是⟨1[4], 1[5], 0[1], 1[3]: 1⟩、⟨1[4], 1[5], 0[1]: 1⟩、⟨1[4], 1[5], 1[3]: 1⟩。它们形成了 0[6]的条件模式基。使用这些条件模式基构造 0[6]的条件 FP-tree，如图 4.16 所示。根据该条件 FP-tree，挖掘 0[6]的频繁模式。它应该只包含单个路径⟨1[4]: 3, 1[5]: 3⟩；而不包含 0[1]和 1[3]，因为它们的发生次数为二次，小于置信度阈值。该单个路径产生的频繁模式的组合有：{1[4], 0[6]: 3}、{1[5], 0[6]: 3}、{1[4], 1[5], 0[6]: 3}。

对于 1[3]，它的两个前缀形成的条件模式基为⟨1[4], 1[5], 0[1]: 2⟩、⟨1[4], 1[5]: 1⟩。因此，产生的频繁模式的组合有：{1[4], 1[3]: 3}、{1[5], 1[3]: 3}、{1[4], 1[5], 1[3]: 3}。

对于 0[1]，它的一个前缀形成的条件模式基为⟨1[4], 1[5]: 3⟩。因此，导出了三个频繁模式{1[4], 0[1]: 3}、{1[5], 0[1]: 3}、{1[4], 1[5], 0[1]: 3}。

对于 1[5]，它的一个前缀形成的条件模式基为⟨1[4]: 4⟩。因此，导出了一个频繁模式

{1[4],1[5]:4}。

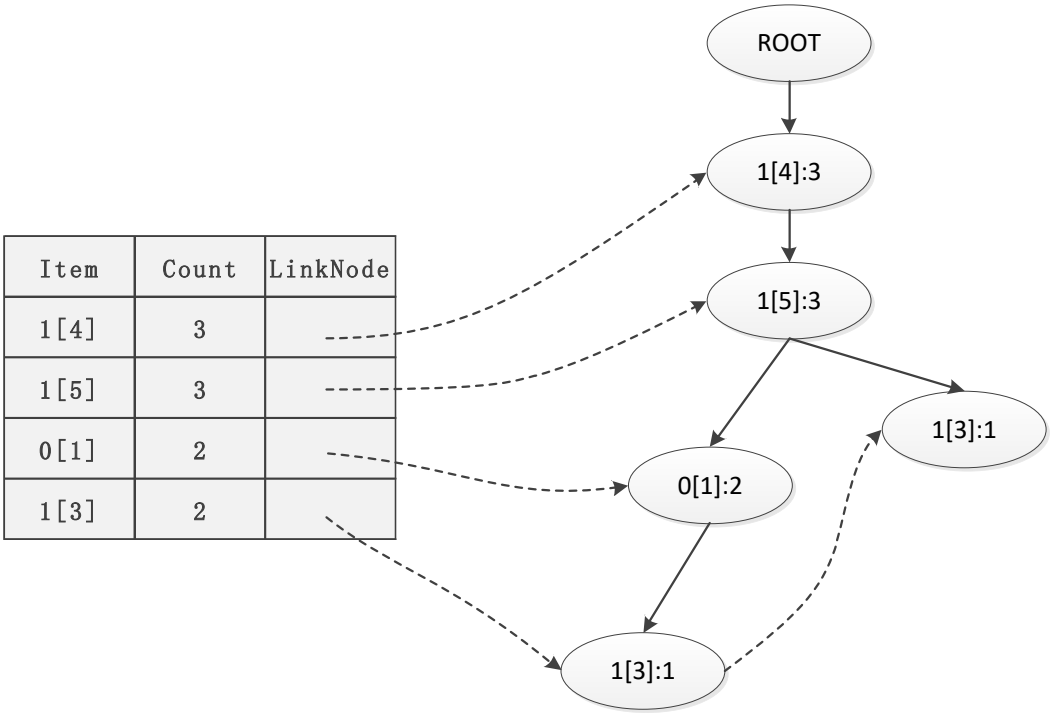


图 4.16 构造条件 FP-Tree

（4）获得周期模式

获得所有的频繁模式，结合周期 T 可以将频繁模式转成周期模式。其中*表示在此周期模式下，对应时间戳中的值不能确定。

例 4.6.4（Part IV）将表 4 所产生的频繁模式，转换成对应的周期模式，如表 4.5 所示。在实际应用中，我们只会关心用户什么时候去某个地方，而并不会关心用户什么时候不去某个地方。因此，挖掘状态为 0 的频繁模式是没有任何意义的。去除表 4.5 中，状态为 0 的项，得到表 4.6。

表 4.5

项	产生的频繁模式	周期模式
0[6]	1[4], 0[6]	****1*0
0[6]	1[5], 0[6]	*****10
0[6]	1[4], 1[5], 1[3]	****110
1[3]	1[4], 1[3]	***11**
1[3]	1[5], 1[3]	***1*1*
1[3]	1[4], 1[5], 1[3]	***111*
0[1]	1[4], 0[1]	*0**1**
0[1]	1[5], 0[1]	*0***1*

0[1]	1[4], 1[5], 0[1]	*0**11*
1[5]	1[4], 1[5]	****11*

表 4.6

项	产生的频繁模式	周期模式
1[3]	1[4], 1[3]	***11**
1[3]	1[5], 1[3]	***1*1*
1[3]	1[4], 1[5], 1[3]	***111*
1[5]	1[4], 1[5]	****11*

如表 4.6 所示，所得到的周期模式存在着冗余的周期模式。例如模式“***111*”包含着“***11**”、“***1*1*”、“****11*”。也就是说模式“***111*”涵盖了其他所有的模式，而其他的模式均为冗余模式。因而，例 4.6.4 中的教师的行为周期模式应该是“每周周四、周五、周六会去学校”。

4.6.4 实验结果

为了测试算法，沿用 4.5.3 中得到的结果，对多个用户的多个地点形成的时间序列进行挖掘。设定置信度阈值为 60%，周期为 7、14、21、28。

（1）000 号用户的 3 号地点有 3 个周期模式：21 天的第 1 天，其发生概率为 75%（红色圈出部分）；28 天的第 20 天，其发生概率为 66.6%（绿色圈出部分）；28 天的第 9 天，其发生概率为 66.6%（蓝色圈出部分）。图 4.17 为部分的数据截图。

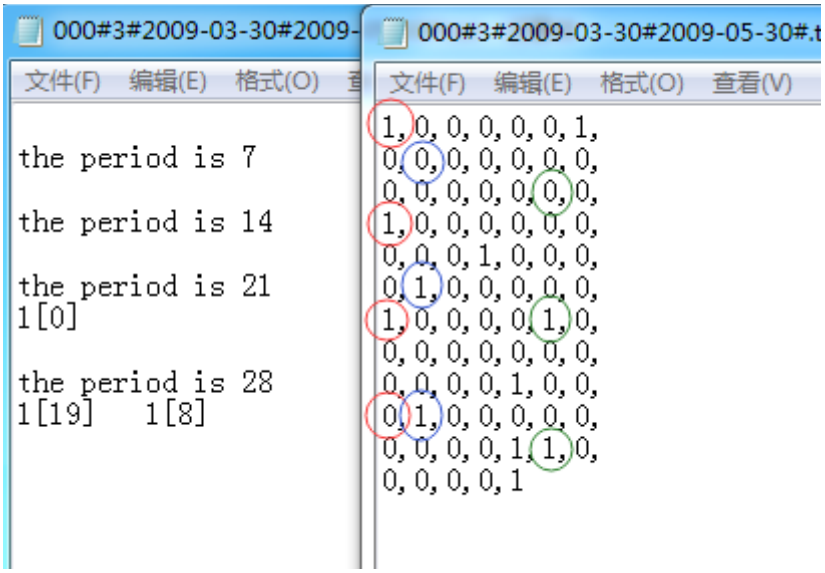


图 4.17 部分的数据截图

（2）003 号用户的 14 号地点有 2 个周期模式：28 天的第 4 天，其发生概率为 75%（红色圈出部分）；28 天的第 21 天，其发生概率为 75%（绿色圈出部分）。图 4.18 为部分的数据截图。

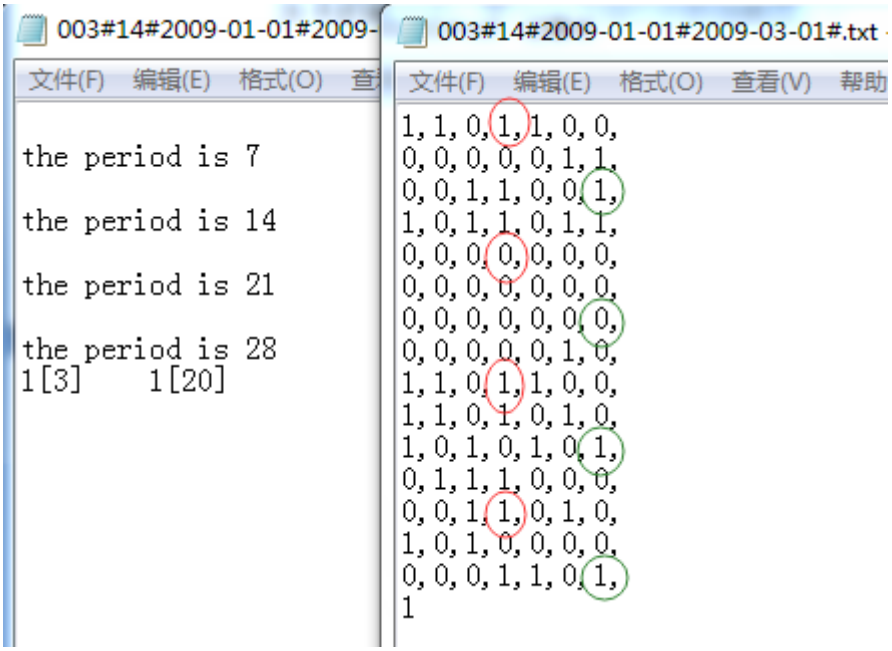


图 4. 18 部分的数据截图

（1）010 号用户的 4 号地点有 3 个周期模式：28 天的第 19 天，其发生概率为 66.6%（蓝色圈出部分）；28 天的第 6 天，其发生概率为 66.6%（红色圈出部分）；28 天的第 13 天，其发生概率为 66.6%（绿色圈出部分）。图 4.19 为部分的数据截图。

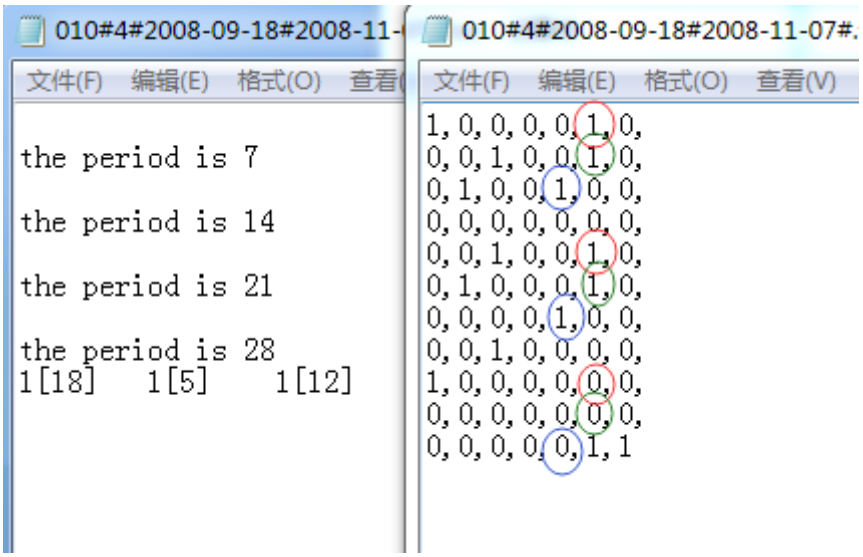


图 4. 19 部分的数据截图

5 系统运行界面

5.1 客户端主界面

客户端主页面采用目前流行的侧滑设计，既美观又方便，加强了用户的体验感。如图 5.1 所示，用户可以通过向右侧滑动的动作，获得相应的功能。侧滑栏中提供了查看今天、明天、本周、以及一周后的日程安排的功能。另外，还提供了预测地点的互联网地点推荐的功能。

图 5.1 侧滑栏

图 5.2 今天的日程

点击侧滑栏中的“今天”，用户可以查看今天有哪些日程，如图 5.2 所示。参考“四象限法则”，本系统将日程分为 4 种类型，分别用四种颜色进行标识：很重要很紧急（如 5.3（a）所示）、很重要不紧急（如 5.3（b）所示）、不重要很紧急（如 5.3（c）所示）、不重要不紧急（如 5.3（d）所示）。此外，颜色块会显示在日程标题的最前端，让用户有一种一目了然的感觉，

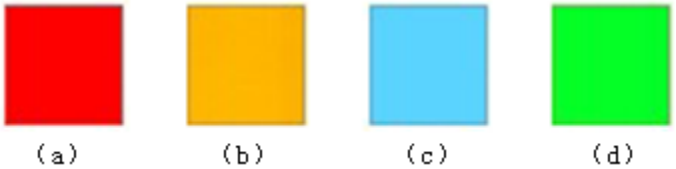


图 5.3 颜色块

5.2 添加日程界面

点击图 5.2 右上角的“+”，可以进入添加日程的界面，如图 5.4 所示，用户可以填写日程的内容，对该日程的重要程度进行分级，选择日程的开始时间和结束时间，输入一些附加的信息。然后点击“增加”，变会弹出“日程增加成功”窗口，如图 5.5 所示。这样就完成了一个日程的添加过程。

图 5.4 增加日程

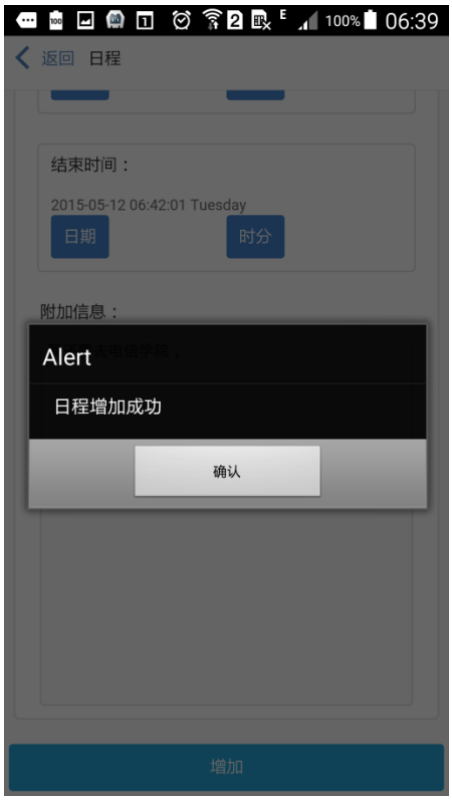


图 5.5 增加日程成功

5.3 日程提醒

当日程时间到了，手机自动提醒用户。如图 5.6 所示。

图 5.6 日程提醒

5.4 推送界面

由服务器后台分析用户的行为序列，并挖掘用户的周期性行为，若能预测用户第二天到达的地点，则会在前一天向用户推送出预测日程的消息，如图 5.7 所示。

若用户点击侧滑栏中的“预测地点商户推荐”，可以获得预测地点附近的地点推荐列表，如图 5.8 所示。


08:12 5月 12日, 星期二
WLAN
位置
移动数据
声音
屏幕旋转
自动
S 搜索
快速连接
calendar
06:36
预计明天将到达同济大学嘉定校区-铁道与..
清除
发送
图片
拍照
视频
文件

图 5.7


06:44
今 天
张记油条(这是一条测试商户数据，仅用于测试开发，开发完成后请申请正式数据...)
和萌牛肠烧烤(这是一条测试商户数据，仅用于测试开发，开发完成后请申请正式数据...)
天手古舞(这是一条测试商户数据，仅用于测试开发，开发完成后请申请正式数据...)
和良形象(这是一条测试商户数据，仅用于测试开发，开发完成后请申请正式数据...)
慢时光MYTIMECAFE(这是一条测试商户数据，仅用于测试开发，开发完成后请申请正式数据...)
炊锅家(这是一条测试商户数据，仅用于测试开发，开发完成后请申请正式数据...)
穆氏特色葱油饼(这是一条测试商户数据，仅用于测试开发，开发完成后请申请正式数据...)
羊掌柜手工面(这是一条测试商户数据，仅用于测试开发，开发完成后请申请正式数据...)

图 5.8

装

订

线

6 总结与展望

6.1 本文总结

考虑到在 LBSN 领域中，用户的日程会重复地发生，即用户的行为轨迹呈现出周期性。因此本文提出了时间序列为非严格多周期情况下的周期模式挖掘算法，其中包括了 Stay Point 算法、OPTICS 算法、以及基于 FP-Growth 的周期模式挖掘算法的实现。

此外，本文设计并实现了一个个性化推荐系统——个性化日程管理软件，来呈现周期模式挖掘算法的结果。将该周期模式挖掘算法应用到个性化日程管理软件中，可以实现基于用户周期性行为的日程推荐。该软件在普通的日程管理软件上做出了创新与改进，能够为用户提供更加精准的推荐服务。

6.2 未来发展

本文将用户周期模式挖掘算法与日程管理软件结合，在这个实践过程中，可以预见基于用户周期性行为的推荐系统非常有发展潜力，这样设计出来的推荐系统更加能满足用户在不同生活状态的需求，更加贴近生活实际，能为用户提供更加精准的推荐服务。未来的工作主要集中于个性化互联网地点推荐系统的研究，以及结合用户偏好和用户周期性行为的个性化地点推荐系统的研究。

参考文献

[1] 项亮. 推荐系统实践[M]. 人民邮电出版社, 2012: 1-64.

[2] Zhao Z D, Shang M S. User-based collaborative-filtering recommendation algorithms on hadoop[C]//Knowledge Discovery and Data Mining, 2010. WKDD'10. Third International Conference on. IEEE, 2010: 478-481.

[3] Jiang J, Lu J, Zhang G, et al. Scaling-up item-based collaborative filtering recommendation algorithm based on hadoop[C]//Services (SERVICES), 2011 IEEE World Congress on. IEEE, 2011: 490-497.

[4] 刘树栋，孟祥武. 一种基于移动用户位置的网络服务推荐方法[J]. 软件学报，2014, (11):2556-2574.

[5] Jensen C S, Friis-Christensen A, Pedersen T B, et al. Location-Based Services -- A Database Perspective[J]. Proceedings of Theghth Scandinavian Research Conference on Geographical Information Science, 2001:59-68.

[6] N. Mamoulis, H. Cao, G. Kollios, M. Hadjieleftheriou, Y. Tao, and D. W. Cheung. Mining, indexing, andquerying historical spatiotemporal data. In KDD, 2004.

[7] Y. Xia, Y. Tu, M. Atallah, and S. Prabhakar. Reducing data redundancy in location-based services. In GeoSensor, 2006.

[8] H. Cao, N. Mamoulis, and D. W. Cheung. Discovery of periodic patterns in spatiotemporal sequences. IEEE Trans. Knowl. Data Eng., 19(4), 2007.

[9] H. Jeung, Q. Liu, H. T. Shen, and X. Zhou. A hybrid prediction model for moving objects. In ICDE, 2008.

[10] Zheng Y, Xie X, Ma W Y. GeoLife: A Collaborative Social Networking Service among User, Location and Trajectory[J]. IEEE Data Eng. Bull., 2010, 33(2): 32-39.

[11] Zheng Y, Li Q, Chen Y, et al. Understanding mobility based on GPS data[C]//Proceedings of the 10th international conference on Ubiquitous computing. ACM, 2008: 312-321.

[12] Zheng V W, Cao B, Zheng Y, et al. Collaborative Filtering Meets Mobile Recommendation: A User-Centered Approach[C]//AAAI. 2010, 10: 236-241.

[13] Saez-Trumper D, Quercia D, Crowcroft J. Ads and the city: considering geographic distance goes a long way[C]//Proceedings of the sixth ACM conference on Recommender systems. ACM, 2012: 187-194.

[14] Sirisha M S, GV Padma Raju G. Periodic Pattern Mining–Algorithms and Applications[J]. Global Journal of Computer Science and Technology, 2014, 13(13).

[15] Li Z, Ding B, Han J, et al. Mining periodic behaviors for moving objects[C]//Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2010: 1099-1108.

[16] Vlachos M, Philip S Y, Castelli V. On Periodicity Detection and Structural Periodic Similarity[C]//SDM. 2005, 5: 449-460.

[17] Li Q, Zheng Y, Xie X, et al. Mining user similarity based on location history.[J]. Gis, 2008:1-10.

[18] 范明，孟小峰译. 数据挖掘概念与技术[M]. 机械工业出版社, 2012: 166-169, 309-311.

附录

装

订

线

谢辞

本论文的研究工作是在我的导师丁志军老师的悉心指导下完成的，我的点滴进步无不倾注着导师的心血。因此，首先我要由衷地感谢他，他在理论研究和实际论文撰写方面都给了我耐心的指导，以及很多重要的意见。导师高瞻远瞩的学识，严谨认真的治学态度，实事求是的研究作风，都给我留下了深刻的印象，这必将是我学习的榜样，使我终身受益。在整个毕业设计期间，他着力培养我独立思考的能力和探索创新的精神。当我的研究受阻时，他会悉心指导并鼓励我，让我重振信心。有了丁志军老师的帮助，使得我在整个过程中很少走弯路，事半功倍。

其次，感谢同济大学 2011 级信息安全班的班主任和我的同学们，在他们的热心帮助和关怀下，促使我顺利完成了本论文。

最后我要感谢我的父母对我学习生活上的支持和关心，是他们的无私奉献和真切关爱使得我能快乐地完成同济大学 4 年的求学之路。

在此也一并感谢所有帮助我完成毕业设计的人，虽然无法一一列出他们的名字，但还是送上我真诚的感谢。

装

订

线