基于密度聚类的签到轨迹大数据分层预处理研究

文若晴[1,2] 马 昂[1,2] 潘 晓[1,2*] 杨伟伟[1]

[1](石家庄铁道大学 河北 石家庄 050043)

[2](河北省高校人文社会科学重点研究基地(石家庄铁道大学) 河北 石家庄 050043)

摘 要 随着基于位置的社交网络的发展,时空-文本等轨迹数据量呈指数式增长,与此同时数据低质的问题日益显著。高质的签到数据可以使研究人员更好地从中挖掘丰富且有意义的知识,因此为了更有效地使用签到大数据,数据预处理必不可少。签到数据具有冗余度高、同时签到、时空签到跨度大等低质问题,导致不能直接使用现有的数据预处理流程和方法。针对签到数据特性,提出一套具有针对性的数据预处理流程。通过平均化处理消除了签到轨迹中存在的同时签到数据;通过学习基于熵的时间戳间隔阈值划分签到轨迹,解决签到轨迹时间跨度大的问题;利用基于密度聚类的方法实现签到轨迹分层,解决空间跨度大的问题。实验采用真实的签到轨迹数据,从离群点和分层效果两个方法对预处理效果进行评价,实现不同空间粒度的签到轨迹分离预处理,为后续的轨迹分析与挖掘奠定基础。

关键词 签到轨迹 预处理 轨迹相似性 聚类 分层

中图分类号 TP311 **文献标识码** A DOI:10.3969/j.issn.1000-386x.2019.03.005

TAXONOMIC PREPROCESSING OF CHECK-IN TRAJECTORY BIG DATA BASED ON DENSITY CLUSTERING

Wen Ruoqing[1,2] Ma Ang[1,2] Pan Xiao[1,2*] Yang Weiwei[1]

[1](*Shijiazhuang Tiedao University*, *Shijiazhuang 050043*, *Hebei*, *China*)

[2](*Key Research Base for Humanities and Social Sciences in Hebei Province*(*Shijiazhuang Tiedao University*), *Shijiazhuang 050043*, *Hebei*, *China*)

Abstract With the development of location-based social networks, the amount of trajectory data such as space-time and text has grown exponentially. Meanwhile, the problem of low quality data has become increasingly prominent. High-quality check-in data allows researchers to better extract rich and meaningful knowledge. Data preprocessing is essential to use check-in big data more effectively. The check-in data has low quality issues: high redundancy, simultaneous check-in, and large spatio-temporal check-in span. The result is that existing data preprocessing process and methods cannot be used directly. According to the characteristics of check-in data, we proposed a set of targeted data preprocessing process. We applied the averaging process to eliminate the presence of the simultaneous check-in data in the check-in trajectory. By learning the threshold of time stamp interval based on entropy to divide the check-in trajectory, we solved the problem of long time span of the check-in trajectory. Using density-based clustering method, the problem of long-span multi-level space of check-in trajectory was solved. The experiment used the real check-in trajectory data, and evaluated the preprocessing effect from the two methods of outliers and taxonomy effects. The results show that the preprocessing of check-in trajectory separation with different spatial granularity is realized, which lays a foundation for subsequent trajectory analysis and mining.

Keywords Check-in trajectory Preprocessing Trajectory similarity Clustering Taxonomic

收稿日期:2018-09-23。国家自然科学基金项目(61303017);河北省自然科学基金项目(F2018210109);河北省教育厅青年基金项目(ZD2018040);石家庄铁道大学第四届优秀青年科学基金项目(Z661250444);石家庄铁道大学研究生创新资助项目(YC201718);国家级大学生创新创业训练计划项目(201710107007)。**文若晴**,本科生,主研领域:移动计算,数据挖掘。**马昂**,硕士生。**潘晓**,副教授。**杨伟伟**,硕士生。

0　引　言

受数据获取手段、数据传输、数据存储等诸多因素的限制,数据“脏”的问题难以根治。数据质量优化的问题已成为分析数据、挖掘知识等研究的瓶颈,为研究人员带来了新的挑战。基于位置的社交网络 LBSN (Location based social networks)广受欢迎,随之产生的时空-文本等高维数据呈指数式增长[1]。通过签到数据预处理技术,可以挖掘人类活动规律、兴趣爱好和行为特征等信息,研究人员可以据此为用户做推荐[2]。然而轨迹数据的数据量大、高维异构、多粒度性、不确定性、高冗余等特点[1],不可避免地对数据分析工作造成了困扰,因此亟待有一套完整的针对轨迹数据的数据预处理流程和方法。我们分析了一组由知名移动社交网站四方(Foursquare)抓取的真实签到数据集[3-4]。在分析过程中发现,签到数据除了具有上述特点外,还具有同时签到和时空跨度大等问题,导致不能直接使用现有的数据预处理流程和方法。

本文对于以上问题,提出了针对签到数据的基于密度聚类的数据分层预处理框架,能够使签到轨迹更好地为后续的挖掘工作所使用。具体来说,首先,对数据集中的属性值进行筛选,并滤除空数据,将签到数据依照用户编号和时间戳进行排列生成用户的签到轨迹;其次,通过平均化处理消除了签到轨迹中存在的同时签到数据;再次,学习基于熵的时间戳间隔阈值划分签到轨迹,解决了签到轨迹时间跨度大的问题;最后,使用基于密度聚类的方法对签到轨迹分层处理,解决了空间跨度大的问题。

1　相关工作

在轨迹数据上通用的数据预处理的典型流程包含异常点检测,轨迹压缩[5]以及轨迹分段等步骤。

轨迹数据的异常点检测是为了找出由于传感器错误或用户特殊行为导致的轨迹中明显不同的点,已经有许多的国内外学者对异常点检测技术进行综述,如 Hodgw[6]、Chandola[7]、Aggarwal[8]、Zhang[9]及 Gupta[10]等。典型检测方法有:均值平滑异常点方法,该方法规定一个尺寸为 n 的滑动窗口,将待测量点的前 $n-1$ 个点的均值作为该点的估计值。粒子滤波方法[11],该方法首先初始化粒子 $x_i^{(j)}$,$j=1,2,\cdots,P$,使得粒子具有 0 初始速度并且以高斯分布聚集在初始位置周围,然后以动态模型 $P(x_i|x_{i-1})$ 来概率地模拟粒子在一个时间步长内的变化。再根据测量模型计算所有粒子的权重 $\omega_i^{(j)}$,权重越大则说明该粒子对测量的支持度越大。将这些重要粒子的权重归一化后,选择与归一化后权重成比例的一组粒子重复进行初始化,最终计算这些重要粒子的权重和,得到被修正后的轨迹。基于启发式的异常点检测方法[12],根据邻近轨迹点间的距离或根据轨迹点的行进速度,判断该距离或行进速度是否满足距离阈值 d 或速度阈值 v,找到轨迹中的异常点并删除。异常点检测方法也可以分为对静态轨迹的检测和对动态轨迹的检测。对静态轨迹的检测可以对轨迹进行特征建模[13],从而将轨迹片段与特征库进行匹配,比较判决阈值。对动态轨迹的检测通常是基于历史轨迹挖掘所有的频繁模式,从而据此识别异常轨迹,Lei 等[14]提出的 MT-MAD 方法可用于海量航海船只轨迹数据的异常检测,Zhu 等[15]提出的 TPRO 方法可以应用于出租车轨迹数据,从而检测是否有司机欺诈绕路的异常轨迹。本文采用一种基于统计的四分位异常点检测方法处理签到数据。

轨迹压缩的目的是节省不必要的存储空间、通信、计算开销。现有的两种轨迹压缩策略为离线压缩和在线压缩。离线压缩(批量模式)是在轨迹完全生成后,减小轨迹大小的方法。该方法通过丢弃具有可忽略误差的点得到与原始轨迹近似的轨迹,这与计算机图形学和制图学研究领域的线简化问题相似[16]。利用垂直的欧氏距离,以首尾轨迹点作为压缩点,通过其连线与其他各个轨迹点的垂线距离,找到误差最大的轨迹点作为压缩点,并使用新的压缩点集合递归上述方法,直到找到的最大误差值小于指定的阈值则停止递归[17]。在线压缩,即在对象移动时实时进行轨迹压缩[18]。不同于离线压缩方法,在线压缩方法不需要得到整条轨迹,而是在轨迹增长的过程中,计算垂线距离,得到最大误差值,当轨迹增长到出现某个最大误差值超过指定阈值时,将尾轨迹点作为压缩点,并将其作为新的首压缩点,继续寻找新的压缩点,最终在轨迹完全生成后得到以所有压缩点形成的压缩轨迹。

轨迹分段的提出是由于整体轨迹的计算复杂性高,轨迹分段后能减小计算量,并且可以挖掘出更丰富的知识。现有的轨迹分段方法大致可分为三种:基于时间间隔的轨迹分段[19],该方法根据给定的时间间隔经验阈值划分轨迹;基于轨迹形状的轨迹分段,该方法通过轨迹中的转向点来划分轨迹,或使用线条简化算法(如 Douglas-Peucker 算法)确定保持轨迹形状的关键点;基于语义的轨迹划分,该方法考虑数据的应用场景[20]或者不同的交通模式[21],根据轨迹点的速度和加速度判断移动物体的运动方式(步行/骑行/公交/

驾车……),从而据此划分轨迹,旅游者的经验和 POI(Point Of Interest)停留点的学习也可以作为语义的一种,从而划分原始轨迹[22-23]。本文采用基于时间间隔的轨迹分段方法,其中轨迹点间的时间间隔的划分阈值通过基于熵的学习获得。

2 签到轨迹的生成

在本节中,我们对 Foursquare 数据集的格式与内容,以及签到轨迹的生成方法进行介绍。

2.1 Foursquare 数据集

Foursquare 网站是目前流行的 LBSN 之一。本文使用的实验数据集是微软研究院发布的从 Foursquare 网站上抓取的真实签到数据[3-4]。该数据集分别包括洛杉矶和纽约用户的签到数据,统计信息如表 1 所示。

表 1 Foursquare 数据集的统计信息

统计信息	洛杉矶	纽约
用户数	31 539	49 005
签到地点数	215 614	206 412
签到数据时间范围	2009/02/06 -2011/07/12	2008/05/06 -2011/07/04
地点类别总数	320	320
地点类别个数取值范围(单条)	[0,18]	[0,18]

数据集中共包含 5 个文件:

(1) Venue(地点) Venue 文件中描述的是用户签到地点的相关信息,文件中包括地点编号、名称、创建时间、经纬度和地点类别编号,我们在生成签到轨迹时使用了该文件中"地点编号"、"经度"和"纬度",形式化地表示为关系 venue(v_id,lat,lon)。

(2) Categories(类别) Categories 文件是地点所属类别的信息,包括类别编号和类别名称。我们使用各个类别名称,作为签到位置的语义,形式化地表示为关系 categories(c_id,c_name)。

(3) Tips Tips 文件中记录的是每个用户的签到数据。Tips 文件中的一条记录无时序地包含一个用户的所有签到数据,其中每一条签到数据包含签到时间、地点编号、地点类别编号等信息。其中一条签到数据形式化地表示为关系 tips(u_id,v_id,createdTime,c_id_0,c_id_1,…,c_id_n)。

(4) User(用户)和 Friendship(朋友) 用户文件包含用户编号、名字、姓氏、头像、性别和籍贯。朋友关系文件中存储了用户间的好友关系,包含 u_id_1 和 u_id_2 两列。本文对签到轨迹的预处理不涉及这两个文件。

2.2 签到数据特点

通过对上述数据集分析发现,签到数据具备数据冗余、数据缺失、同时签到、时空跨度大等特点。

(1) 数据冗余 数据冗余是指数据重复存储的现象。例如,Foursquare 数据集中的 Venue 文件记录了洛杉矶和纽约用户访问过的地点编号、名称、经纬度、地点类别等信息。Tips 文件是用户的签到数据文件,包含用户编号、地点编号、签到文本、签到时间与地点类别等信息。很明显,两个文件都重复存储了地点类别信息,可能会导致数据的不一致。

(2) 数据缺失 数据集中存在值缺失的现象。例如,Venue 文件中经度和纬度(longitude,latitude)两列,如图 1 所示,缺失值占 0.87%。Users 数据文件中的姓名、性别和家乡等列缺失值占 1.94%。

Venue id	latitude	longitude
4ccda07faa25a35d071f1c0f	40.65404342	-73.95978928
4ccda0e7aa25a35dc8211c0f	28.057729	-82.502819
4ccda22f72106dcbf310a499	40.724705	-73.949196
4ccda23da92595214cb86e29		
4ccda27454f0b1f7e7c21dca	40.6745217	-73.9629344

图 1 Venue 文件中地点编号及其对应经纬度

(3) 同时签到 签到服务允许用户同时在多个地点连续签到。因此,存在用户在相同时间在多个地点签到的现象。如图 2 所示,u_id 是用户编号,时间戳(createdTime)是从 1970 年 1 月 1 日 00:00:00 开始以秒计算的时间偏移量。图 2 中共有 5 条签到记录,这 5 条记录中地点编号(v_id)是不同的,但是后三条签到数据的时间戳是相同的。用户的实际位置可能仅是其中的某一个签到地点,或是与签到地点临近的某一个点。

u_id	v_id	createdTime ▲ 1
169961	4b9f87e5f964a5204a2937e3	1300166544
169961	4b50f75bf964a520f73a27e3	1300166545
169961	4bc66b12bf29c9b643e2f92a	1300235683
169961	4c25a8b6f1272d7f854785c5	1300235683
169961	4b81b5def964a52034b930e3	1300235683

图 2 用户 169961 的同时签到数据

(4) 时空跨度大 签到数据跨度大主要表现在时间和空间两个方面。时间方面,数据集中包含了 2008 年 5 月 6 日 05:47:27 至 2011 年 7 月 12 日 06:07:30 的用户签到记录。我们将同一用户相邻签到数据的时间戳跨度分布进行统计,见表 2(LA:洛杉矶 NYC:纽约)。可以发现将近 3/4 的相邻签到数据时间跨度在半个月内,剩余 1/4 的相邻签到数据时间跨度在半个

月至两年半不等。

表 2　相邻签到数据时间戳跨度分布表

时间戳跨度	签到数据个数(LA)	累积百分比(LA)	签到数据个数(NYC)	累积百分比(NYC)
0 – 0.5 个月	164 490 个	74.76%	278 598 个	74.17%
0.5 – 1 个月	20 632 个	84.13%	35 645 个	83.65%
1 – 6 个月	29 365 个	97.48%	51 144 个	97.27%
6 – 12 个月	5 057 个	99.78%	9 302 个	99.75%
>12 个月	487 个	100.00%	955 个	100.00%

空间方面,数据集中的签到数据几乎覆盖全球,在以这些签到数据中最小和最大的经纬度所形成的矩形范围内,对角跨度约为 398.20 度,约 4.4 万公里。而根据不同用户形成的签到轨迹的对角跨度分布不均,统计签到轨迹空间跨度分布见表 3。图 3 中记录了某用户的 27 条签到数据,图中左侧的签到数据共 3 条与其余的签到数据距离较远。放大图 3 中右侧的签到数据到图 4,其中有 24 条签到数据,可以看到签到轨迹中被圈出来的 1 条签到数据与其余 23 条签到数据不在同一个城市而且相距很远。

表 3　签到轨迹空间跨度分布表

签到轨迹空间跨度/km	签到轨迹数(LA)	累积百分比(LA)	签到轨迹数(NYC)	累积百分比(NYC)
0 ~ 5 550	28 877 条	91.56%	43 609 条	88.99%
5 550 ~ 11 100	793 条	94.07%	2 510 条	94.11%
11 100 ~ 16 650	742 条	96.43%	948 条	96.05%
16 650 ~ 22 200	178 条	96.99%	988 条	98.06%
22 200 ~ 27 750	605 条	98.91%	681 条	99.45%
27 750 ~ 33 300	279 条	99.79%	241 条	99.94%
33 300 ~ 38 850	62 条	99.99%	27 条	100.00%
38 850 ~ 44 400	3 条	100.00%	1 条	100.00%

图 3　用户 100140 的签到轨迹(共 27 条签到数据)

图 4　用户 100140 的部分签到轨迹(共 24 条签到数据)

2.3　签到轨迹

将关系 categories、venue 和 tips,通过 v_id 连接得到签到数据的经纬度信息,通过 c_id 连接得到签到数据的文本信息以便于生成签到轨迹后计算轨迹的文本相似性,可获得用户的一条签到数据,定义如下。

定义 1　签到数据。一条签到数据 $p = (u\_id, createdTime, lon, lat, c\_name_0, c\_name_1, \cdots, c\_name_n)$,其中,根据表 1 统计信息,$n = 0, 1, \cdots, 18$。

利用用户签到数据生成用户签到轨迹,即,将同一用户的签到数据按时间戳排序形成一个序列,以用户编号命名该用户的签到轨迹,定义如下。

定义 2　签到轨迹。一条签到轨迹 $T_{u\_id} = (u\_id, loc\_list, word\_list)$,其中 $loc\_list = \{(p_1.lon, p_1.lat), (p_2.lon, p_2.lat), \cdots, (p_m.lon, p_m.lat)\}$ 是同一个用户的所有签到数据按时间顺序排列的地理坐标对序列,且 $p_i.createdTime - p_{i-1}.createdTime \geqslant 0$。签到轨迹的 word_list 是所有签到位置的类别名称的并集,并以空格隔开形成的一个字符串。

3　签到数据预处理

3.1　签到数据预处理总流程

签到数据的预处理主要包括:数据清理与数据转换、签到轨迹划分、签到轨迹分层,其流程见算法 1。

算法 1　签到数据预处理流程

输入:原始签到数据

输出:分层的签到轨迹集

```
选择有用属性,减少冗余
选择完整信息,滤除含有缺失值的数据
生成签到轨迹
FOR every 签到轨迹{
    平均化同时签到点
    检测并删除每条轨迹的离群点
}END FOR
学习时间戳阈值,划分签到轨迹
```

9 计算签到轨迹间的相似性,聚类并分层签到轨迹

对于签到数据因被存储在多个文件所造成的数据冗余以及数据缺失的问题,首先进行初步数据清理,然后将签到数据转换为签到轨迹,并对每一条轨迹中同时签到数据和离群签到数据进行清理。由于签到轨迹数据时间、空间跨度大的特性,还要对签到轨迹进行划分和分层,采用基于熵的方法学习用于划分签到轨迹的时间间隔阈值,分层采用基于密度的 DBSCAN 聚类算法聚类签到轨迹。

(1) 选择有用属性和完整信息 Foursquare 数据集中存在数据冗余和缺失。对于冗余,选择各个文件中的有用维度生成签到轨迹,避免同一信息多处存储;对于缺失值,选择使用包含完整信息的数据。

(2) 生成签到轨迹 将 Tips 文件中各用户的签到数据按签到时间排序,并与 venue 连接得到签到数据的经纬度信息、与 categories 连接得到签到数据的文本信息,以方便后续计算签到轨迹的时空和文本相似性。

(3) 平均化同时签到 数据中存在很多如图 2 所示的同时签到数据。具有多条同时签到数据的用户可能只在其中一个地点或者都不是。为了使签到轨迹更合理,本文将这些相同时间戳的签到数据进行平均化处理。

具体来讲,对于每一个用户的签到轨迹,若用户的 n 个签到数据的时间戳相同,则将这 n 个签到数据的经度和纬度分别取均值,以得到的新经纬度作为该时间戳的签到地点。对于 n 个签到数据的文本,将其去重后以空格分隔合为一个字符串,作为该时间戳的文本。例如用户 169961 在同一时刻对 6 个地点进行签到,基本信息如表 4 所示,各个地点的经纬度都不相同。计算得到经度和纬度的平均值分别为 -79.028 9 和 36.146 3,最终得到该用户在这一时刻的签到数据如表 5 所示。

表 4 用户 169961 的部分签到数据

地点经度	地点纬度	签到时间	签到文本类别
-73.620 7	41.020 1	1300166537	Park
-80.119 5	26.785 6	1300166537	Playground
-80.175 2	26.933 9	1300166537	Other Great Outdoors
-78.782 7	35.723 4	1300166537	Hiking Trail Park Other Great Outdoors
-74.084 9	39.871 5	1300166537	Park Beach
-87.390 6	46.543 0	1300166537	Park

表 5 用户 169961 的部分签到数据平均化结果

地点经度	地点纬度	签到时间	签到文本类别
-79.028 9	36.146 3	1300166537	Park Playground Other Great Outdoors Hiking Trail Beach

(4) 检测并处理签到轨迹离群点 根据统计知识,一个序列上如果有值在序列上所有值的上下四分位的 1.5 倍四分位距以外的范围,这样的值是离群的。将用户签到轨迹可视化后,我们发现用户的轨迹存在如图 3、图 4 的情况。对于这样的签到轨迹我们提出一种基于统计的四分位检测离群签到数据的方法处理用户的签到轨迹,算法基本思想是将每条待处理签到轨迹的经度和纬度分别排序并计算分别的四分位数 (Q_1,Q_2,Q_3)、四分位距 $(IQR = Q_3 - Q_1)$ 和内限 $(Q_3 + 1.5 \times IQR, Q_1 - 1.5 \times IQR)$,并分别找到并标注经度和纬度上的离群值,最后将签到轨迹上被标注为离群的签到数据删除。

(5) 划分签到轨迹 签到轨迹的相邻签到数据中约 1/4 的时间戳跨度分布在半个月至两年半不等,这使得在原始签到轨迹上直接计算的结果没有意义。基于时间阈值的轨迹划分将一条轨迹划分为了几条轨迹使相邻签到数据的时间戳跨度在一定合理范围内。我们通过基于熵的方法学习适用于划分签到轨迹的时间间隔阈值,从而划分签到轨迹,具体方法见 3.2 节。

(6) 分层签到轨迹 签到轨迹分层用以解决签到轨迹空间跨度大的问题。通过计算签到轨迹间的相似性,对签到轨迹根据轨迹的时空文本相似性进行聚类,并通过调整聚类参数,得到不同层次粒度的签到轨迹集合。

3.2 基于熵的签到轨迹划分

轨迹划分中最常用的方法是按时间阈值将轨迹分段,但该方法中时间间隔阈值难以确定,这也是签到数据预处理流程中的一个难点。我们使用 Cui[24] 提出的基于熵的轨迹时间阈值学习方法,确定一个较客观的轨迹划分阈值,具体方法如下。

统计一条轨迹上连续签到点的时间间隔,设有 n 个不重复的时间间隔 $(s_1,s_2,\cdots,s_n)$,其中 s_i = T. p_{i+1}. createdTime − T. p_i. createdTime,令 $f_1,f_2,\cdots,f_n$ 是 $s_1,s_2,\cdots,s_n$ 的频数,即 $N = \sum_{i=1}^{n} f_i$ 为所有轨迹中相邻签到数据的间隔总数。任意一个时间间隔 s_i 的频率为 q_i,公式如下。

$$q_i = \frac{f_i}{N} \quad \sum_{i=1}^{n} q_i = 1, i = 1,2,\cdots,n \tag{1}$$

设 t 是时间间隔阈值,则小于 t 的间隔分布的熵的定义如下:

$$E_1 = -\sum_{i=1}^{t} \frac{q_i}{\sum_{j=1}^{t} q_j} \ln \frac{q_i}{\sum_{j=1}^{t} q_j} \tag{2}$$

大于 t 的时间间隔分布的熵的定义如下:

$$E_2 = -\sum_{i=t+1}^{n} \frac{q_i}{\sum_{j=t+1}^{n} q_j} \ln \frac{q_i}{\sum_{j=t+1}^{n} q_j} \tag{3}$$

签到轨迹划分的目标是在多个 t 中找到使得熵 E_1 和熵 E_2 的和最大的阈值 θ,公式如下,当两个连续的签到轨迹时间间隔大于阈值 θ 时,则将划分该签到轨迹为两条签到轨迹。

$$\theta = \mathrm{Arg} \max_{t=1,2,\cdots,n} (E_1 + E_2) \tag{4}$$

4　基于密度聚类的签到轨迹分层

在本节中,将对签到轨迹的分层处理做详细说明,包括对用户签到轨迹进行相似性定义和基于时空文本相似性的聚类方法。

4.1　签到轨迹的时空-文本相似性

签到轨迹相似性包含三个方面:空间、时序和关键字(即类别名称)。离散弗雷歇距离是目前被广泛用于计算不同多边形轨迹曲线间距离的方法,它动态地考虑了轨迹点的时序和空间相似性,因此本文选用离散弗雷歇距离计算用户签到轨迹间的时空距离。同时,在关键字相似度方面,Jaccard 系数是基于字符串语义相似性度量中常用的方法之一,其对于两个用于计算相似性的向量的维度没有限制。因此,本文结合二者优点提出了签到轨迹的时空文本相似性度量方法。

定义 3　离散弗雷歇距离。设任两条签到轨迹 T_1、T_2,T_1 有 m 个签到点,T_2 有 n 个签到点生成。T_1 和 T_2 的离散弗雷歇距离为:

$$\mathrm{DFD}(T_1, T_2) = \max \begin{cases} \mathrm{Dist}(T_1.p_m, T_2.p_n) \\ \min \begin{cases} \mathrm{DFD}(T_1.p_{m-1}, T_2.p_n) \\ \mathrm{DFD}(T_1.p_m, T_2.p_{n-1}) \\ \mathrm{DFD}(T_1.p_{m-1}, T_2.p_{n-1}) \end{cases} \end{cases} \tag{5}$$

当 $m=0, n=0$ 时,DFD 计算方法如下:

$$\mathrm{DFD}(T_1.p_0, T_2.p_0) = \mathrm{Dist}(T_1.p_0, T_2.p_0) \tag{6}$$

当 $m>0, n=0$ 时,DFD 计算如下:

$$\mathrm{DFD}(T_1.p_0, T_2.p_j) = \max \begin{cases} \mathrm{Dist}(T_1.p_0, T_2.p_j) \\ \mathrm{DFD}(T_1.p_0, T_2.p_{j-1}) \end{cases} \tag{7}$$

当 $m=0, n>0$ 时,DFD 计算如下:

$$\mathrm{DFD}(T_1.p_i, T_2.p_0) = \max \begin{cases} \mathrm{Dist}(T_1.p_i, T_2.p_0) \\ \mathrm{DFD}(T_1.p_{i-1}, T_2.p_0) \end{cases} \tag{8}$$

式中:$\mathrm{Dist}(p_i, p_j)$ 是任意两个签到数据 p_i 与 p_j 间的欧氏距离。

下面用一个简单的例子解释离散弗雷歇距离的定义。现有两条签到轨迹 $T_1 = \{(0,0),(1,0),(2,-10),(3,0),(4,0),(5,0)\}$,$T_2 = \{(0,0),(1,0),(2,0),(3,0)\}$。轨迹 T_1 和 T_2 的 DFD 值根据定义 3 可以得到一个如图 5 所示的二维矩阵,签到轨迹 T_1 和 T_2 的离散弗雷歇距离为 10。若两条签到轨迹的离散弗雷歇距离越大则说明它们在时空上距离越远。

T1\T2	$T_2.p_0$	$T_2.p_1$	$T_2.p_2$	$T_2.p_3$
$T_1.p_0$	0	1	2	3
$T_1.p_1$	1	0	1	2
$T_1.p_2$	√102	√101	10	√101
$T_1.p_3$	√102	√101	10	10
$T_1.p_4$	√102	√101	10	10
$T_1.p_5$	√102	√101	10	DFD 10

图 5　签到轨迹 T_1 和 T_2 的 DFD 二维矩阵

定义 4　基于 Jaccard 系数的文本相似性。签到轨迹 T_1 和 T_2 的类别集合的文本相似性定义如下:

$$\mathrm{SimJCD}(T_1, T_2) = \frac{|T_1.\mathrm{word\_list} \cap T_2.\mathrm{word\_list}|}{|T_1.\mathrm{word\_list} \cup T_2.\mathrm{word\_list}|} \tag{9}$$

可见 $\mathrm{SimJCD} \in [0,1]$,越接近于 1 表示两条轨迹的文本方面越相似。

定义 5　签到轨迹相似性。对于给定的两条签到轨迹 T_1、T_2,签到轨迹相似性形式化的表示为:

$$\mathrm{SOT}(T_1, T_2) = \alpha \times \mathrm{SimDFD} + (1-\alpha) \times \mathrm{SimJCD} \tag{10}$$

$$\mathrm{SimDFD}(T_1, T_2) = 1 - \frac{\mathrm{DFD}}{\mathrm{MaxDFD}} \tag{11}$$

式中:SimDFD 是基于离散弗雷歇距离的时空相似性,$\mathrm{SimDFD} \in [0,1]$,越趋近于 1 表示两条轨迹的时空方面越相似;参数 $\alpha \in [0,1]$,为轨迹的时空相似性的权重。$\mathrm{SOT} \in [0,1]$,越趋近于 1,则两条签到轨迹在时空-文本方面越相似。在同一个用户签到轨迹集 D 内,不同的 α,可以得到不同的 SOT 值和轨迹相似性矩阵 $\boldsymbol{M}$。

4.2　基于 DBSCAN 的签到轨迹分层

4.1 中已经得到签到轨迹间的相似性,现在要据此对签到轨迹进行聚类,从而得到分层的签到轨迹簇。算法 2 的输入包括签到轨迹集 D,任两条用户签到轨迹的轨迹相似性矩阵 $\boldsymbol{M}$,最小簇轨迹数阈值 *MinTrs* 和

聚簇半径 *Eps*。

如算法 2 所示,初始状态下,任选一个未被访问的签到轨迹 *T*,找出与 *T* 间的签到轨迹相似性 SOT 大于等于阈值 *Eps* 的所有签到轨迹,记为 *N*。如果 *N* 中签到轨迹数量小于阈值 *MinTrs*,则 *T* 被标记作为噪声舍弃(第 2 ~ 6 行)。如果 *N* 中签到轨迹数量大于等于阈值 *MinTrs*,则 *T* 与其 *N* 中的签到轨迹形成一个簇 *c*,并且 *T* 被标记为已访问(第 7 ~ 12 行)。以相同的方法处理该簇内所有未被标记为已访问的签到轨迹,从而对簇进行扩展(第 13 ~ 16 行)。如果簇充分地被扩展,即簇内的所有签到轨迹被标记为已访问,则用同样的算法去处理未被访问的签到轨迹。

算法 2 签到轨迹的 DBSCAN 聚类

```
输入:D,M,MinTrs,Eps
输出:簇集合 C
1  初始化簇集合 C = null
2  FOR every 没有被访问的轨迹 T in D{
3    标记 T 为已访问
4    N = {T'|SOT(T,T') ≥ Eps}
5    IF(Size(N) < MinTrs){
6      将轨迹 T 标记为噪声
7    } ELSE {
8      新建一个簇 c 加入到簇集合 C 中
9      扩展以 T 为核心对象的簇 c{
10       将 T 加入到 c 中
11       FOR every T' in N{
12         标记 T'为已访问
13         N' = {T''|SOT(T',T'') ≥ Eps}
14         IF(Size(N') ≥ MinTrs){
15           N = N + N'
16         }
17         IF T'不是任何簇的成员{
18           将 T'加入到簇 c 中
19         }
20       }ENDFOR
21     }
22   }
23 }ENDFOR
```

在一个给定的 α 下,通过对聚类参数 *MinTrs* 或 *Eps* 的调整可以改变签到轨迹的层次粒度。逐渐减小参数 *Eps*,形成的簇的层次范围将逐渐扩大(由街区到城市到国家等)。若 *Eps* 较小,则有更多轨迹被认为是相似的,使得聚簇的结果可能是一个洲上的所有签到轨迹,层次粒度很大。反之,高相似性要求使得很少有签到轨迹被认为相似,甚至难以成簇。参数 *MinTrs* 的变化往往影响同一个层次内的签到轨迹簇数,但随着 *MinTrs* 的增大,簇的扩展效果被放大,使得簇的层次粒度提升。当寻找城市层次的相似签到轨迹时,小的 *MinTrs* 会使得到的结果出现很多个簇,且一个簇中的签到轨迹很少,反之由于没有足够多的签到轨迹满足 *Eps*,将会无法成簇。*Eps* 和 *MinTrs* 都对分层效果有一定的影响,好的分层结果需要通过对两个参数进行多次调节。

5 实验结果与分析

实验运行于 Inter(R) Core(TM) i7-7500U @ 2.7 GHz 处理器、12 GB 内存的 Windows10 计算机上。实验程序采用 Java 语言进行编写。由于基于时空-文本相似性的密度聚类的签到轨迹分层方法主要完成了签到轨迹中离群签到数据的去除以及签到轨迹的分层,所以实验中对离群签到数据和签到轨迹分层分别进行了评价。

5.1 离群点检测

在签到数据预处理流程中,要进行两次的离群点检测:一次是在生成签到轨迹后,对每一条签到轨迹进行基于四分位的离群点检测;另一次是在签到轨迹分层阶段,使用 DBSCAN 算法聚类时,在不同的参数下得到的噪声轨迹。下式计算了签到轨迹集中的离群率和分层处理过程的签到轨迹离群率。

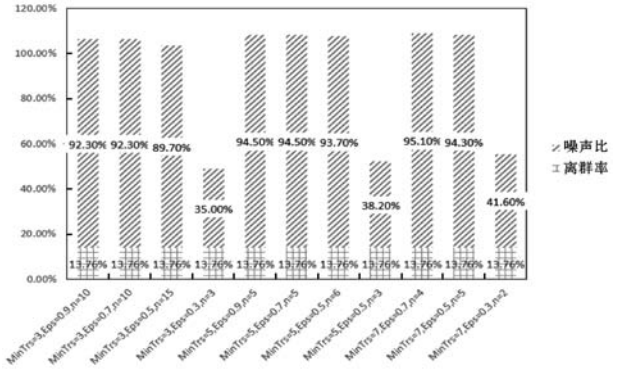
$$\text{离群率} = \frac{\text{离群点个数}}{\text{原签到数据总数量}} \times 100\% + \left(1 - \frac{\text{聚类结果中所有的签到轨迹数}}{\text{待聚类签到轨迹数}} \times 100\%\right) \tag{12}$$

其中,一个签到数据集生成的签到轨迹是固定的,对于每条签到轨迹检测出的基于四分位的离群点也是不变的,因此针对一个签到数据集,其基于四分位离群率是固定的。图 6 展示了 1 000 条签到轨迹的原始分布和经过基于四分位的离群签到数据检测过滤了离群签到数据的签到轨迹分布,不同深浅颜色点表示不同的签到轨迹,整体可见经过处理的签到轨迹集中少了很多孤立的点,该部分的离群率(即根据式(12)的前半部分计算所得)为 13.76%,如图 7 网状阴影部分。分层阶段中 DBSCAN 算法的参数变化时,得到的噪声签到轨迹数量也相应变化,该部分的离群率如图 7 斜线阴影部分,一般情况下,满足参数条件的签到轨迹数并不多,因此由分层处理过程得到的离群率会较高。

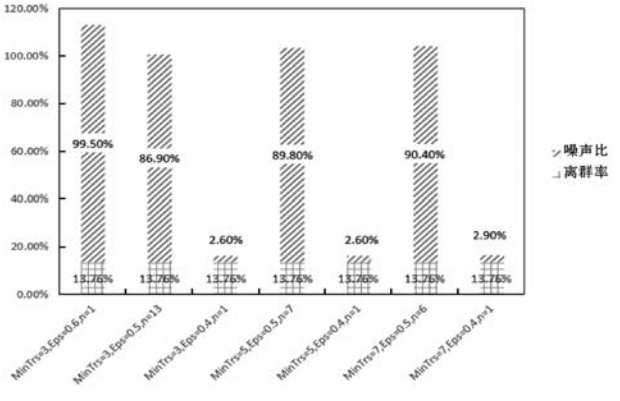
(a) 未处理离群点的签到轨迹

(b) 去除离群点后的签到轨迹

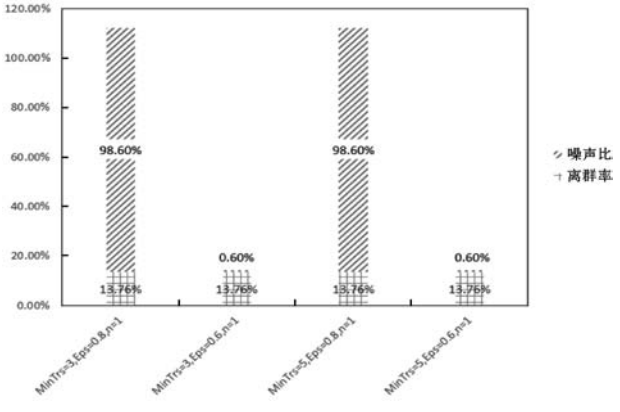
图 6　1 000 条签到轨迹分布图



(a) $\alpha=0.1$



(b) $\alpha=0.5$



(c) $\alpha=0.9$

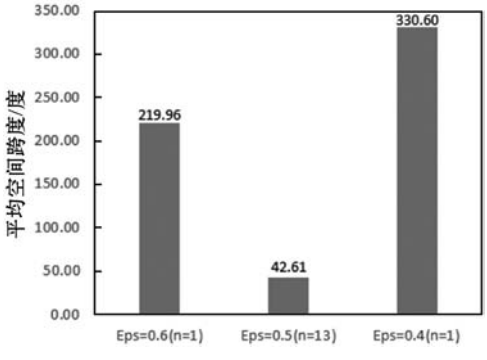
图 7　签到轨迹数据预处理的离群率

5.2　轨迹分层

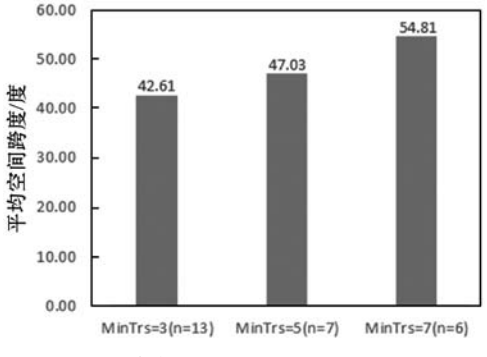
该部分对三个参数 α,*MinTrs* 和 *Eps* 进行调节,我们研究了不同参数在时空-文本的影响下对轨迹分层结果所带来的变化,以下式计算签到轨迹分层的平均跨度,从空间方面来评价分层效果。

$$\text{平均空间跨度} = \frac{\sum_{1}^{n} \text{每个簇的空间跨度 } k_i}{\text{簇的个数 } n} \tag{13}$$

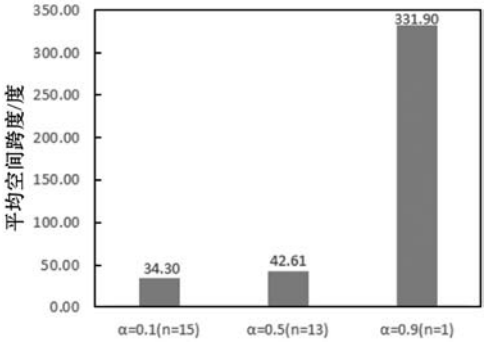
图 8 展示了不同参数(签到轨迹的时空相似性权重 α,最小簇轨迹数阈值 *MinTrs* 和聚簇半径 *Eps*)对签到轨迹的平均空间跨度的影响,平均跨度越大,所划分的层次就越高(如国家),平均跨度越小,则所划分的层次越低(如城市)。如图 8(a)所示,时空和文本的权重各占一半,聚簇的最小成簇轨迹数为 3 条,随着聚簇半径 *Eps* 逐渐减小,同一个簇内的签到轨迹在时空-文本上越来越不相似,使得该条件下的平均空间跨度增大。但实际上当 *Eps* 分别为 0.6 和 0.4 时,所得到的簇中包含了一条本身空间跨度就十分大的签到轨迹,使得在这两个参数下的簇个数 n 只有一个,其平均空间跨度被高度影响。如图 8(b)所示,在时空和文本的权重各占一半,聚簇半径 *Eps* 为 0.5 时,随着最小簇轨迹数的增多,聚簇的最小范围变大,簇的个数减少,使得平均空间跨度有增大的趋势。如图 8(c)所示,当时空相似性占签到轨迹相似性比重提高时,得到的签到轨迹间的相似性不同,因此计算得出的平均空间跨度没有相关性。



(a) $\alpha=0.5$, *MinTrs* = 3



(b) $\alpha=0.5$, *Eps* = 0.5



(c) $MinTrs=3, Eps=0.5$

图 8 不同参数对签到轨迹的平均空间跨度的影响

6 结 语

随着定位技术的精准与社交软件的普及,人们更加频繁地在网络上留下自己行走过或者停留过的地点痕迹,并赋予文本描述。然而,海量数据的质量问题是所有研究工作的重要前提。本文针对 Foursquare 签到数据集提出了一套数据预处理流程和方法,对签到轨迹进行基于四分位的离群签到数据检测,通过学习时间间隔阈值对签到轨迹划分解决了签到轨迹时间跨度大的问题,通过基于签到轨迹时空文本相似性的密度聚类方法对签到轨迹进行了分层,解决了空间跨度大的问题。通过对聚类参数的调整可以将签到轨迹集分层到洲、国家甚至城市,为将来更富有意义的签到数据挖掘工作提供了便利。

参 考 文 献

[1] 潘晓, 马昂, 郭景峰, 等. 基于时间序列的轨迹数据相似性度量方法研究及应用综述[R]. 石家庄:石家庄铁道大学, 2018.

[2] 高强, 张凤荔, 王瑞锦, 等. 轨迹大数据:数据处理关键技术研究综述[J]. 软件学报, 2017, 28(4):959 - 992.

[3] Bao J, Zheng Y, Mokbel M F. Location-based and preference-aware recommendation using sparse geo-social networking data[C]//International Conference on Advances in Geographic Information Systems. ACM, 2012:199 - 208.

[4] Wei L Y, Zheng Y, Peng W C. Constructing popular routes from uncertain trajectories[C]//ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2012:195 - 203.

[5] Lee W C, Krumm J. Trajectory preprocessing[M]//Computing with Spatial Trajectories. Springer New York, 2011:3 - 33.

[6] Hodge V, Austin J. A survey of outlier detection methodologies[J]. Artificial Intelligence Review, 2004, 22(2):85 - 126.

[7] Chandola V, Banerjee A, Kumar V. Anomaly detection: A survey[M]. ACM Computing Surveys, 2009, 41(3):75 - 79.

[8] Aggarwal C C. Outlier analysis[M]. Springer Publishing Company, Incorporated, 2015.

[9] Zhang Y, Meratnia N, Havinga P. Outlier detection techniques for wireless sensor networks:a survey[J]. IEEE Communications Surveys & Tutorials, 2010, 12(2):159 - 170.

[10] Gupta M, Gao J, Aggarwal C C, et al. Outlier detection for temporal data: a survey[J]. IEEE Trans. on Knowledge and Data Engineering, 2014, 26(9):2250 - 2267.

[11] Krumm J. Trajectory analysis for driving[M]//Computing with Spatial Trajectories. Springer New York, 2011:213 - 241.

[12] Yuan J, Zheng Y, Xie X, et al. T-Drive: Enhancing driving directions with taxi drivers' intelligence[J]. IEEE Transactions on Knowledge & Data Engineering, 2012, 25(1):220 - 232.

[13] 王洋, 单征, 赵炳麟, 等. 基于静态行为轨迹的异常特征检测技术[J]. 计算机应用研究, 2017, 34(8):2434 - 2438.

[14] Lei P R. A framework for anomaly detection in maritime trajectory behavior[J]. Knowledge and Information Systems, 2016, 47(1): 189 - 214.

[15] Zhu J, Jiang W, Liu A, et al. Time-dependent popular routes based trajectory outlier detection[C]//International Conference on Web Information Systems Engineering. Springer International Publishing, 2015:16 - 30.

[16] Mcmaster R B. A statistical analysis of mathematical measures for linear simplification[J]. American Cartographer, 1986, 13(2):103 - 116.

[17] Hershberger J, Snoeyink J. Speeding up the Douglas-Peucker line-simplification algorithm[J]. Proceedings of the International Symposium on Spatial Data Handling, 1992:134 - 143.

[18] Meratnia N, By R A D. Spatiotemporal Compression Techniques for Moving Point Objects[C]//Advances in Database Technology—EDBT 2004, 9th International Conference on Extending Database Technology, Heraklion, Crete, Greece, March 14 - 18, 2004, Proceedings. DBLP, 2004:765 - 782.

[19] 张晓滨, 杨东山. 基于时间约束的 Hausdorff 距离的时空轨迹相似度量[J]. 计算机应用研究, 2017, 34(7):2077 - 2079.

[20] Yuan N J, Zheng Y, Zhang L, et al. T-Finder: A recommender system for finding passengers and vacant taxis[J]. IEEE Transactions on Knowledge and Data Engineering, 2013, 25(10):2390 - 2403.

[21] Zheng Y, Chen Y, Li Q, et al. Understanding transportation modes based on GPS data for web applications[J]. Acm Transactions on the Web, 2010, 4(1):1 - 36.

[22] Zheng Y, Xie X. Learning travel recommendations from user-generated GPS traces[J]. ACM Transactions on Intelligent Systems and Technology, 2011, 2(1):1 - 29.

(下转第 56 页)

[2] Tschorsch F, Scheuermann B. Bitcoin and beyond: A technical survey on decentralized digital currencies[J]. IEEE Communications Surveys and Tutorials, 2015, 18(3): 2084-2123.

[3] Narayanan A, Bonneau J, Felten E. Bitcoin and cryptocurrency technologies[M]. Princeton: Princeton University Press, 2016.

[4] 秦波,陈李昌豪,伍前红,等. 比特币与法定数字货币[J]. 密码学报, 2017, 4(2): 176-186.

[5] Pappalardo G, Di T, Caldarelli G, et al. Blockchain inefficiency in the Bitcoin peers network[DB/OL]. [2018-05-06]. https://arxiv.org/pdf/1704.01414.pdf.

[6] Bonneau J, Miller A, Clark J, et al. Sok: Research prespectives and challenges for Bitcoin and cryptocurrencies[C]//Proceedings of the 36th IEEE Symp on Security and Privacy (SP 2015). Piscataway, NJ: IEEE, 2015:104-121.

[7] Wikipedia. The DAO (organization)[OL]. [2018-09-01] https://en.wikipedia.org/wiki/The_DAO_(organization).

[8] Bitcoin Cash Website[OL]. [2018-09-10] https://www.bitcoincash.com.

[9] Decker C, Wattenhofer R. A fast and scalable payment network with bitcoin duplex micropayment channels[G]//LNCS 9212: Symposium on Self Stabilizing Systems (SSS2015). Berlin: Springer, 2015: 3-18.

[10] Bip 68,[OL]. [2018-09-01] https://github.com/bitcoin/bips/blob/master/bip-0068.mediawiki.

[11] Bip 112[OL]. [2018-09-01] https://github.com/bitcoin/bips/blob/master/bip-0112.mediawiki.

[12] Poon J, Dryja T. The bitcoin lightning network: Scalable off-chain instant payments[OL]. [2018-09-02] https://lightning.network/lightning-network-paper.pdf.

[13] Raiden network[OL]. [2018-09-02] http://raiden.network/.

[14] Miller A, Bentov I, Kumaresan R, et al. Sprites: Payment channels that go faster than lightning[DB/OL]. [2018-05-02] https://arxiv.org/pdf/1702.05812.pdf.

[15] Bip 141[OL]. [2018-09-02] https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki.

[16] Bitcoin Transaction Malleability[OL]. [2018-09-04] https://eklitzke.org/bitcoin-transaction-malleability.

[17] Decker C, Wattenhofer R. Bitcoin transaction malleability and Mtgox[C]//LNCS 8713: Proceedings of European Symp on Research in Computer Security (ESORICS 2014). Berlin: Spinger, 2014: 313-326.

[18] Andrychowicz M, Dziembowski S, Malinowski D, et al. On the malleability of Bitcoin transactions[C]//LNCS 8976: Proceedings of the 19th International Conference on Financial Cryptography and Data Security (FC 2015). Berlin: Springer, 2015: 1-18.

[19] Bitcoin Script[OL]. [2018-09-09] https://en.bitcoin.it/wiki/script.

[20] Wikipedia. Coinbase[OL]. [2018-09-04] https://en.wikipedia.org/wiki/Coinbase.

[21] Wikipedia. Merkle tree[OL]. [2018-09-06] https://en.wikipedia.org/wiki/Merkle_tree.

[22] Bitcoin development: fix openssl linkage problems[OL]. [2018-05-03] https://github.com/bitcoin/bitcoin/commit/a30b56eb.

[23] Bitcoin development: Don't count or spend payments until they have one confirmation[OL]. [2018-09-03] https://github.com/bitcoin/bitcoin/commit/a790fa46f40.

[24] Wikipedia. Schnorr signature[OL]. [2018-09-06] https://en.wikipedia.org/wiki/Schnorr_signature.

[25] Maxwell G, Poelstra A, Seurin Y, et al. Simple schnorr multi-signatures with applications to Bitcoin[DB/OL]. [2018-05-03] https://eprint.iacr.org/2018/068.pdf.

[26] Zhao Y L. Aggregation of gamma-signatures and applications to Bitcoin[DB/OL]. [2018-09-04] https://eprint.iacr.org/2018/414.pdf.

[27] Andrychowicz M, Dziembowski S, Malinowski D, et al. How to deal with malleability of Bitcoin transactions[DB/OL]. [2018-09-08]. https://arxiv.org/pdf/1312.3230.

[28] Croman K, Decker C, Eyal I, et al. On scaling decentralized blockchains[C]//LNCS 9604: Proceedings of the 20th International Conference on Financial Cryptography and Data Security (FC 2016). Berlin: Springer, 2016: 106-125.

(上接第 28 页)

[23] Zheng Y, Zhang L Z, Xie X, et al. Mining interesting locations and travel sequences from GPS trajectories[C]//International Conference on World Wide Web. ACM, 2009:791-800.

[24] Cui G, Luo J, Wang X. Personalized travel route recommendation using collaborative filtering based on GPS trajectories[J]. International Journal of Digital Earth, 2018, 11(12): 1-24.

(上接第 41 页)

[17] Lin M, Hsu W J, Lee Z Q. Detecting modes of transport from unlabelled positioning sensor data[J]. Journal of Location Based Services, 2013, 7(4):19.

[18] ChaoSimple. 常用聚类算法(一) DBSCAN 算法[EB/OL]. 2013-07-01. https://www.cnblogs.com/chaosimple/archive/2013/07/01/3164775.html.