基于Hadoop及出租车历史轨迹的乘客推荐算法

景维鹏[1,2],胡立坤[1,2]
JING Weipeng[1], HU Likun[1,2]

1.东北林业大学 信息与计算机工程学院,哈尔滨 150040
2.黑龙江省林业生态大数据存储与高性能(云)计算工程研究中心,哈尔滨 150040
1.College of Information and Computer Engineering, The Northeast Forestry University, Harbin 150040, China
2.Heilongjiang Province Engineering Technology Research Centre for Forestry Ecological Big Data Storage and High Performance (Cloud) Computing, Harbin 150040, China

JING Weipeng, HU Likun. Recommendation algorithm for passengers based on Hadoop and trajectory data. Computer Engineering and Applications, 2016, 52(7):264-270.

Abstract: In order to improve the efficiency of passenger recommendation algorithm for wisdom city, this paper uses classical probability to count the percentage of the empty taxi passing days in the total days to recommend the probability of passengers waiting for empty taxi, and uses least squares to fit arrival rate curve, namely, the relation curve between time and the number of empty taxis to predict the time passengers have to wait to take an empty taxi from the time that they reach the right road. To improve the efficiency of recommendation, the paper has done three more work: Select Hadoop as data storage and computing platform to improve the ability of data processing; put forward a new road network storage structure based on rasterizing maps to improve the search speed of the map; reform a map matching algorithm which based on computational geometry to improve the matching accuracy. Field testing experiments show that, the correct probability of empty taxi recommendation algorithm accuracy can reach 87% while the accuracy of waiting time recommendation algorithm is about 88.4 %, which shows the feasibility of mining trajectory data to provide recommendation service for passengers.

Key words: Hadoop; trajectory data; recommendation algorithm; recommendation service for passengers

摘 要: 针对智慧城市中乘客打车策略的推荐算法效率不高的问题,使用古典概率学统计历史轨迹中该时间该路段有空车的天数占数据集总天数比例,作为乘客等到空车概率;使用最小二乘法拟合时间与到达空车数曲线,预测乘客等到空车时间,以提高推荐效率。同时,使用Hadoop作为数据存储和计算平台以提高数据处理能力;提出一种基于地图栅格化的路网存储结构来提高搜索地图速度;改进一种基于计算几何的地图匹配算法提高匹配准确率。实验结果显示,空车概率推荐算法正确率约87%,等待时间推荐算法正确率达88.4%,表明挖掘轨迹数据为乘客提供推荐服务的可行性。

关键词: Hadoop; 轨迹数据; 推荐算法; 乘客推荐服务
文献标志码: A **中图分类号:** TP311 doi:10.3778/j.issn.1002-8331.1411-0170

1 引言

出租车是一种在城市中较为方便的交通工具,出租车历史轨迹数据可以记录并反映城市交通出行、人群移动情况,是智能交通研究中的重点研究对象,也是我国智慧城市的重要研究方向,文献[1]分析了移动轨迹数据在我国智慧城市发展中的作用及研究领域;文献[2]中通过对轨迹数据的分析,来预测和监控城市交通状况;文献[3]通过分析和统计轨迹数据,对结果多次细

基金项目: 黑龙省自然科学基金重点项目(No.ZD201403);哈尔滨市科技创新人才基金(No.2014RFQXJ132)。
作者简介: 景维鹏(1979—),男,博士,副教授,主要研究方向为云计算、容错计算,E-mail:nefujwp@gmail.com;胡立坤(1991—),男,硕士研究生,主要研究方向为云计算、智能推荐算法。
收稿日期: 2014-11-14 **修回日期:** 2014-12-30 **文章编号:** 1002-8331(2016)07-0264-07
CNKI网络优先出版: 2015-06-05, http://www.cnki.net/kcms/detail/11.2127.tp.20150605.1059.003.html

化，来获取路网的几何特性；文献[4]通过分析轨迹数据中出租车绕路、低速行驶等现象，来评价交通规划。

而国内城镇化在加速，一些城市规划的缺陷以及各城市经济发展的不平衡使得打车难的问题越来越显著，市场上有一些如嘀嘀打车之类的App，通过获取实时的乘客和出租车位置信息，将其显示在电子地图上，推送给出租车司机和乘客，这减轻了打车难的问题，但也存在隐私问题。研究人员尝试从出租车行驶轨迹中挖掘有用信息，并通过设计合适的推荐算法，为乘客推送相关服务。

推荐算法是基于内容的信息的推荐方法，它根据用户过去的浏览记录来向用户推荐用户没有接触过的推荐项。它主要是从两个方法来描述基于内容的推荐方法：启发式的方法[5]和基于模型的方法[6-7]。

文献[8]以非齐次泊松分布拟合出租车到达率曲线，设计推荐算法，但是该文只拟合一小时以内的空车到达情况，偶然性太强，同时，对处理轨迹数据的工具没有深入详细说明和研究；文献[9]使用MPI+openMP作为数据处理平台，在处理大文件这方面，优势并不是特别明显，单位计算成本昂贵。

Hadoop作为一款有着强大的可靠性、高效性、可伸缩性、高容错性的支持数据密集型分布式应用，在处理海量数据方面有着极强的优势。它的低廉的运算和存储成本也使之成为研究者们深入研究的对象[10]。

本文通过研究地图搜索和地图匹配，进一步对推荐效率进行提升。

传统的图存储方式邻接矩阵所需存储空间大，空间利用率低，邻接表查询速度慢，文献[11]中给出的图存储方法能有效减少图数据存储空间，但是查询复杂度高。针对地图匹配算法，文献[12]提出的轨迹点之间相互投票，实现精确匹配的算法计算量大，耗费时间长，实时性差；文献[13]中给出的缩小搜索域，使用几何距离来确定匹配路段简单，但该匹配准确度不佳。

综上所述，本文将使用Hadoop对出租车GPS轨迹数据进行挖掘处理，设计等待空车概率推荐算法和等待空车时间推荐算法，改良了一种基于计算几何的地图匹配算法，提出一种基于地图栅格化的路网数据存储结构，从而实现实时精确的推荐。

2　算法模型

本章中，空车概率推荐算法使用古典概率学推荐给乘客到达目标路段后3分钟时间段内等到空车的概率；等待时间推荐算法使用二分法拟合到达率曲线来预测乘客等待到第一辆空车的时间；地图路网数据存储结构借用地图栅格化和地图存储结构的的特点来实现低代价存储，高效查询；地图匹配算法通过改良一种基于计算几何的地图匹配算法，使之更高效，更具准确性。

2.1　路网数据存储结构模型

本文使用如文献[9]中的北京路网数据，有超过150万个路网节点，使用传统的邻接矩阵存储将会有超过百亿的矩阵元素，在存储过程中，将消耗大量内存，内存利用率低。本文要求对路网节点进行快速搜索和查找，而传统的邻接表不能灵活的针对固定区域进行查询，查询速度慢，影响定位效率，从而影响推荐算法的实时性。

本文综合地图栅格化及地图数据存储结构，利用GPS坐标的连续性，提出一种简捷的路网数据存储结构，取名栅条式路网存储，实现地图的高效查询，低代价存储，存储结构如图1所示。

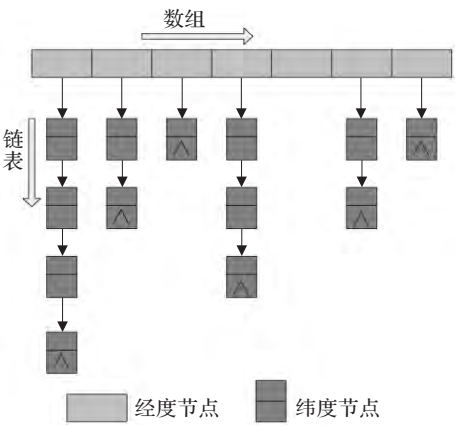


图1　路网存储结构

该结构横向为经度节点，以数组结构存储，将地图均匀切分为栅条状，把栅条图中纬度最大的路网节点存放在数组对应位置中，栅条图纵向路网节点以纬度降序方式链接在数组单元中的链表中。

此结构中，横向数组的下角标与各个经度区间相对应，纬度方向使用链表可动态增加节点的优点，每一个链表节点存储一个路网节点，减少了路网数据对内存的大量需求。查询中，通过横向数组的连续性能在经度方向对路网节点快速模糊定位，再通过纵向的查询，即能够实现目标点的准确定位，通过查找其前驱后继即可查找到相应的边，避免了类似邻接表方法中对所有节点的遍历，减小了搜索区域，减少了搜索时间。

2.2　地图匹配算法模型

文献[13]中，地图匹配算法以目标点（车辆或者乘客）周围边长为 x 矩形范围内（候选域）的路段为候选路段，即图2中的矩形 $ABCD$，该区域内的路段（ FG、GH、GJ、IK 等）作为候选路段，对每个路段分别进行检测验证，以目标点到路段的欧式距离作为比较标准，找出距离目标点 M 最近的路段，图2中目标点 M 到路段 GJ 的距离小于到 GH 的距离，故选取 GJ 作为预匹配路段，当该距离大于阈值 d 则抛弃该预选路段。

该算法使用固定边长的候选域，在数据密集区将极大增加候选路段的数量，花费验证的时间开销将等比增

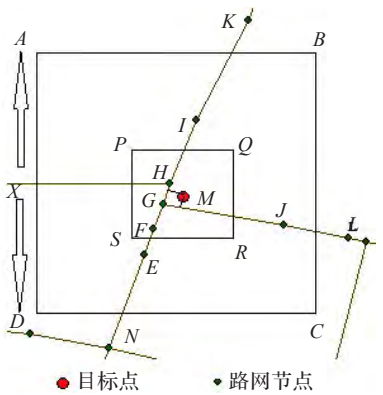


图2　地图匹配示意图

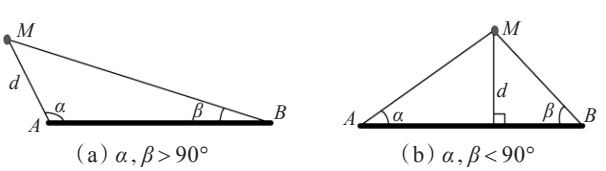
加。如图2,候选域PQRS中的候选路段即可定位目标路段 GJ ,使用固定候选域 $ABCD$ 则需要花费更多的时间去验证额外的候选路段,包括 EN、IK、JK 等路段。而在数据稀疏区,若固定候选域内没有路段,则该算法匹配将出错。

为解决以上问题,改进算法使用动态边长的候选域,即候选域边长从50 m递增300 m,逐步增大候选域的范围。相比比于原算法,在数据密集区,这种改进结构能减小候选路段的规模,故而减少搜索目标路段所需要的时间,即在很小边长的候选域内就能匹配到目标路段;在数据稀疏区,这种改进结构在找不到候选路段情况下,逐步增大候选域范围直到找到候选路段或者超过边长阈值,可以减少候选域中没有候选路段而匹配出错的情况。

为提高原算法的准确率,改良算法使用车辆运行速度方向与候选路段的夹角对匹配结果进行重验证,即车与候选路段的夹角 $a>\omega$ 则抛弃当前候选路段。这样可以减少路网密集区,路段之间距离太近造成的误匹配情况。

匹配过程中,认为人在地图上认为是一个静态的点,可以出现在地图上的任何位置(即可以不在路段上),故当人与预选路段的夹角是钝角时,以人到该路段端点的直线距离作为判断依据。

车作为一个动态的点,认为只能出现在路段上,故当车与预选路段夹角为钝角时,抛弃该预选路段。如图3所示。



(a) $\alpha,\beta>90°$　(b) $\alpha,\beta<90°$

图3　不同目标的匹配模型

AB 为预选路段, M 为目标点,即为待匹配的乘客或者出租车, α、β 为待匹配点和预选路段的夹角, d 在图3(a)中表示待匹配点到预选路段的最短直线距离,在图3(b)中表示目标的到预选路段的欧式距离。

当目标点 M 为乘客,如图3(a), $\alpha,\beta>90°$ 时,以线段的长度 d 作为判断根据;如图3(b), $\alpha,\beta<90°$ 则以 M 到路段 AB 的距离 d 为判断依据。

当待目标点 M 为出租车,如图3(a), $\alpha,\beta>90°$ 时,则认为该点不能匹配到当前当前候选路段;在图3(b), $\alpha,\beta<90°$,与乘客相同,以 d 为判断依据。

2.3　空车概率推荐算法模型

古典概率是假设随机事件发生有限次、事件之间发生互不相容、事件发生可能性相等,有无空车事件显然符合条件,故使用古典概率去估算空车到达概率是可行的。

算法的输入为乘客的位置信息(GPS)以及时间,即图4中的 (x,y,t) ,乘客被定位到路网上的某个路段 r 上。

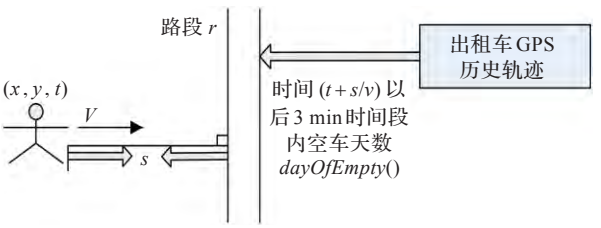


图4　空车概率推荐示意图

考虑到乘客不在路段旁边的情况,需要估算乘客到达路段的时间 $T=t+s/v$,如图4,其中 s 为乘客与匹配路段距离, v 为乘客速度, t 为乘客输入的时间。

通过统计所有出租车历史轨迹数据中,从乘客到达目标路段 r 时间开始的3 min时间段内 $[T, T+3\ \text{min}]$,有空车通过该路段的天数,该天数占总天数的比例,即为乘客等到空车的概率 $P(t)$:

$$P(t)=\frac{daysOfEmpty(r,t+s/v)}{daysOfAll}\times 100\% \tag{1}$$

其中 $daysOfEmpty()$ 为有空车总天数, $daysOfAll$ 为总天数。

2.4　等待时间推荐算法模型

算法将每天分割为3 min一段的区间,总计480个,统计每个区间到达空车的数量,并平均到每一天,使用基函数为多项式的最小二乘法拟合时间与到达空车数的关系曲线,即到达率曲线 $f(t)$,最后计算从乘客到达目标路段的时刻开始以后,至到达率函数的积分大于等于1的时间差即为预测到达第一辆空车的时间,认为预测结果小于等于15 min为可接受。

$$f(t)=a_0+a_1t+a_2t^2+a_3t^3+a_4t^4,\ 0<t<1\,440 \tag{2}$$

$f(t)$ 为到达率函数, t 为乘客到达目标路段的时间点(单位min)。

当 $\int_t^{t+\Delta t} f(t)\geqslant 1$ 且 $\Delta t\leqslant 15$ min , Δt 即为等到下一辆空车的预测时间。

3　算法实现

3.1　推荐算法流程

乘客输入自己的位置和时间信息,算法验证信息格

式的正确性,确认正确后,使用存储于内存中的路网数据,将乘客匹配到地图路段上,以该路段端点坐标、乘客时间信息和GPS历史轨迹数据作为空车概率推荐算法的输入,获取历史记录中,乘客到达路段时间点以后3 min的时间段内有空车的天数,它占总天数的比例即为乘客等到空车概率;等待时间预测以路段端点坐标和GPS历史轨迹为输入,统计每天中每3 min通过该路段的空车数,并以此计算出到达率函数,计算从乘客到达路段时间开始,到达率函数的积分为1的时间点,与乘客到达时间之差即为预测第一辆空车到达时间。如图5所示。

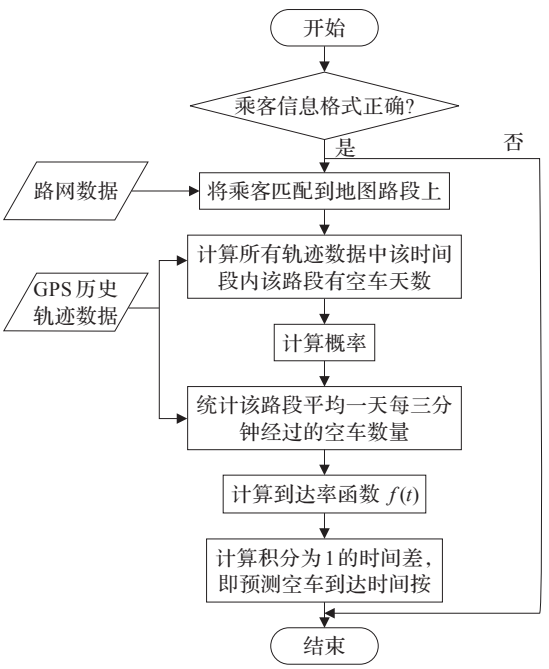


图5　流程图

3.2　推荐算法具体实现

3.2.1　路网存储结构实现

地图经度方向,以 x 为间隔,将地图分割成栅条状,存储于数组当中,本文中北京路网经度在区间[115.375 000, 117.500 000]中,故选取 x 为0.000 01°,横向数组约220 000个单元,数组单元的为RPoint类;纬度方向,将每个栅条图中的所有路网节点存放于RPoint的list中,纬度最大的点放入RPoint,list表中的路网节点存储结构为RearPoint。所有记录全部处理完毕,使用Java的Collection的sort静态方法对list中的节点按纬度降序排列,查询使用折半法加快搜索速度。具体结构如下:

经度节点(横向数组):

```
RPoint{
public int  pid=-1; //路段ID
public float  x=-1,y=-1; // x、y分别为经度和纬度
public Point pre,rear; //该节点的前后节点
public List<RearPoint> list;
}
```

纬度节点(纵向list链表节点):

```
RearPoint{
public int  pid=-1; //路段ID
public float  x=-1,y=-1; // x、y分别为经度和纬度
public Point pre,rear; //该节点的前后节点
}
```

3.2.2　地图匹配实现

输入乘客坐标 p,找到区间 $[p.x-50\ \text{m}, p.x+50\ \text{m}]$ 内包含的所有经度节点RPoint,遍历RPoint的链表list,将区间 $[p.y-50\ \text{m}, p.y+50\ \text{m}]$ 的纬度节点加入候选点集。

检测所有候选点,若它的邻节点属于候选点且该点和邻节点没有全部被访问过,则对该点与没被访问的邻节点组成的路段进行匹配检验,如图6中,若 A 点或 A 点的前一点 P_A 之前没有全部被访问过,则计算点 T 到路段 AP_A 的欧式距离,记录当前路段 AP_A 及距离大小;若当前点的邻节点在矩形外,只要当前点没有被访问过,则可对该点与邻节点之间的路段进行匹配检验。如图6中的 A 点与其矩形外的邻接点 R_A。当候选域没有候选点时,候选域边长递增50 m,重复以上步骤。

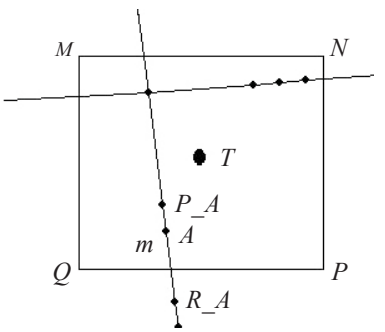


图6　候选路段检验图

如此反复,直到能得出匹配结果或者候选区的变长超过预定大小(本文设为300 m)。

最后选择距离最小的路段作为实际匹配路段。

对于车辆,在匹配检验过程中计算路段和汽车速度夹角,小于20°则认为匹配有效。

3.2.3　空车概率推荐算法实现

将出租车GPS历史轨迹上传到HDFS,将其作为mapreduce的输入。HDFS将文件按设置的block大小,将文件分成小块存储,每一块启动一个map任务,每个map对每条记录进行读取和解析。

map阶段:map对数据集进行筛选,提取value中轨迹数据的GPS坐标 (*getGPS*(*value*)),与乘客的所在的路段坐标进行匹配(fit);若能够匹配在该路段,获取该记录的时间 (*getTime*(*value*)),若时间处于乘客到达目标路段时间点以后的三分钟时间段内,则将数据写入临时文件,由reduce抓取,其中 *key* 是YYYY-MM-DD格式,表示具体某天,*value* 表示一条符合的记录数;

reduce阶段:使用一个计数器counter计算 *key* 的个数,即每执行一次reduce方法自增一次。

主函数:获取计数器的值,表示符合要求的天数,再根据公式(2)输出空车概率。

具体算法伪代码如下:

```
map(key,value){
if(fit(getGPS(value),road costumer stands on))
//坐标能匹配到乘客所在的路段
{
    time=getTime(value);
    if (time⊂[timeofCostomer, timeofCostomer+3])  //时间处于乘客预计到达目标路段时间点以后三分钟时间段内
        write(time.format(YYYY-MM-DD),1); //将时间格式修改为年月日
    }
}
reduce(key,value){
        Counter=getCounter();
        Counter++; //统计符合天数
}
Main{                    //主函数
        Print(Counter/daysOfAll×100%);
}
```

3.2.4　等待时间推荐算法实现

将24小时按3分钟一段,分成480段,时间段标号从零开始。

map阶段:从 *value* 中提取GPS坐标,匹配乘客所在路段,若成功匹配,则从 *value* 中获取时间,将时间的时分秒转换成该时间段的标号(timeToIndex),如0时10分12秒的标号是3,作为输出 $key(0<key<480)$,而输出 *value* 表示一条符合的记录数。

reduce阶段:将所有同标号的(同时间段)*value* 进行累加,写入输出文件作为最小二乘法的样本数据集。

计算到达率函数 $f(t)$:根据样本数据集,计算正规方程组的系数矩阵 A,获得正规方程组,用超松弛迭代法(*SOR*())计算系数,获得达率函数 $f(t)$。

主函数:计算从乘客给出的时间开始,积分等于1的时间点,时间差即为预测的等待时间。

具体算法伪代码如下:

```
map(key,value) {          //map阶段
    if(fit(getGPS(value),Road costumer stands on))
        {
            time=getTime(value);
            write (timeToIndex(time), 1);
        }
}
reduce(key,value) { //reduce阶段
        sum=Sum(key.list);
        write (key, sum);
}
f() //计算到达率函数 f(t):
{
        f(x)=SOR(the output of mapreduce);
}
main            //主函数
{
        if (sum(f(x to x+Δt)≥1)
        print (Δt);
}
```

4　实验结果

4.1　实验环境

实验虚拟软件XenServer 6.2将4台曙光I450-G10塔式服务器(InterXeon E5-2407四核2.2 GHz处理器,8 GB内存)虚拟成16台主机,一台HP Compaq dx 2308(Intel Pentium E2160 1.8 GHz处理器,1 GB内存)作为Master,在Cenos6.4_final(内核2.6.32)系统上搭建起Hadoop的cloud cluster,集群中包括16个DataNode和TaskTracker,HDFS总大小为3.34 TB;在Window7旗舰版上安装Microsoft SQL Server 2008,使用该数据库作为GPS轨迹数据载体,同时使用该主机的FAT32本地文件系统,对Hadoop的分布式文件系统的性能进行分析和验证。

4.2　实验数据集

实验使用的路网数据是北京市的路网数据,地图处于东经[115.375 000,117.500 000]北纬[39.416 667,41.083 333]区域内,有超过20万条完整路段,路网节点达到158万多个。轨迹数据是北京市30天内12 000辆出租车GPS历史轨迹,超过9亿条出租车记录,转换成文本格式大约50G。

4.3　路网存储结构比较实验

在Java环境下分别使用邻接矩阵和邻接表以及本文提出的路网存储结构实现北京路网数据的存储,并比较各方法在内存大小需求及节点查询速度进行比较。节点查询速度比较实验以查询1 000个随机节点的时间为标准。结果表明,邻接矩阵因为所需内存太大造成内存溢出,无法实现。栅条式路网存储结构在内存使用上只占邻接表的40.7%,而节点查询速度是邻接表的3.05倍。结果如表1所示。

表1　网存储结构比较

存储结构	内存占用	查询速度/s
邻接矩阵	内存溢出	—
邻接表	264.1 MB	371.7
栅条式路网存储结构	107.6 MB	121.9

4.4　改良地图匹配算法比较实验

随机在北京路网中选取1 000个点作为实验集,其中密集区的点和稀疏区的点各500个,分别用原算法和

改良后的算法进行匹配:经分析,在匹配效率上,改良后的地图匹配算法在数据密集区的匹配速率(公式(3))比原算法快73%;在数据稀疏区,改良后的算法匹配速率是原算法的91%,但是匹配成功率(公式(4))比原算法高27%。结果如表2所示。

表2　地图匹配算法比较

算法	数据稀疏区		数据密集区	
	v/s	P/%	v/s	P/%
原算法	2.51	71.2	2.81	93.7
改良算法	2.76	90.3	2.05	96.1

表中 v 为匹配速率, P 为成功匹配率:

$$v=\frac{T}{M} \tag{3}$$

$$P=\frac{m}{M}\times 100\% \tag{4}$$

T 为测试点总时间, M 为总匹配点数, m 为匹配正确点数。

4.5　运行时间比较实验

分别使用SQL Server 2008和本地文件系统(FAT32)作为数据载体串行进行计算的方式与HDFS+MapReduce进行比较,在16个计算节点的情况下,处理格式化后的130M路网数据时,串行运行时间超过了1 min,而使用HDFS+Mapreduce用时平均在12 s左右。各个方法处理结构的性能如表3所示。

表3　处理性能比较表

方法	130M路网数据	50G轨迹数据
HDFS+MapReduce	12.26 s	13 min 7 s
SQL Server 2008+串行	102.74 s	2 h 49 min
本地文件系统(FAT32)+串行	74.09 s	2 h 21 min

由该表分析可得到,130M数据处理实验中,Hadoop的HDFS+MapReduce相比于SQL Server 2008+串行,效率提高了7倍多,相比于本地文件系统(FAT32)+串行,效率提高了5倍以上;50G轨迹数据处理实验中HDFS+MapReduce相比于SQL Server 2008+串行,效率提高了11倍,相比于本地文件系统(FAT32)+串行,效率提高了7.5倍。

4.6　推荐算法实验

实验设置24组测试数据,每组表示以该组号为时间起点的测试,例如12表示正午十二点整测试,每组测试5个点。打车概率的测试,通过比较两个不同点在3 min间时间内等到空车的难度来估计空车到达概率的正确性,概率大的等到空车的难度更低,实验结果表明,空车概率推荐的正确率在87%左右;在等待时间上预测测试方面,结果正确率能达到88.4%,预测时间实验结果如图7所示。

以第一个测试点[116.363 203,39.900 215]为例,通过分析出折线图如图8所示。

测试点	116.363203　39.900215		116.386427　39.968173		116.266141　39.92401		116.334742　39.968961	
测试时间	预测	实际	预测	实际	预测	实际	预测	实际
1:00	over		over		over		over	
2:00	over		over		over	11:27	over	
3:00	over		15	over	over		13	over
4:00	12	11:37	13	12:14	over		13	6:57
5:00	11	6:20	13	11:50	14	12:15	12	10:19
6:00	11	11:39	8	8:55	11	6:04	10	7:21
7:00	5	4:29	5	3:10	6	5:39	4	5:37
8:00	5	3:21	5	4:20	5	3:52	6	5:32
9:00	3	3:01	3	2:42	4	0:19	4	3:39
10:00	4	3:14	4	3:05	4	2:08	4	3:51
11:00	2	2:11	3	2:11	4	2:10	3	1:42
12:00	3	2:06	4	3:26	4	3:34	3	3:21
13:00	3	1:54	4	2:24	3	2:51	3	2:41
14:00	5	3:51	4	3:43	5	3:40	1	2:41
15:00	4	2:08	6	4:53	1	2:01	4	3:26
16:00	5	3:33	5	4:03	5	5:09	5	0:42
17:00	5	4:28	6	4:40	6	5:46	5	5:11
18:00	4	2:44	4	2:22	4	3:42	3	2:10
19:00	5	3:45	5	3:41	5	3:09	5	4:40
20:00	6	3:26	8	7:33	6	5:40	6	4:40
21:00	8	5:35	7	6:09	8	7:00	9	7:37
22:00	13	13:29	13	13:29	13	11:57	12	12:33
23:00	14	over	over		over	13:51	over	
0:00:00	over		over		over		over	

图7　等待时间测试

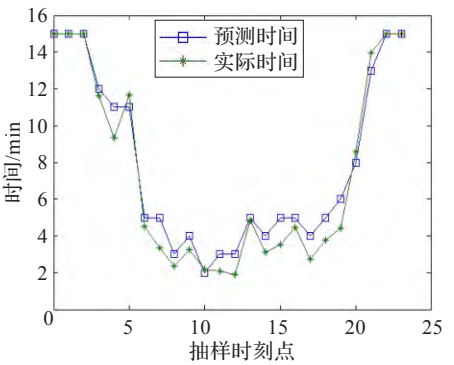


图8　第一个测试点预测与实际时间对比

5　结束语

本文针对当前城市里打车难,现有的打车软件有安全隐患等问题,使用Hadoop云平台挖掘出租车历史轨迹,提出一种以数组和链表结构组合的路网存储结构,以动态候选区和根据车辆速度方向重验证匹配的方式改良一种基于计算几何的地图匹配算法,来提高推荐效率;根据该时段该路段历史中有空车通过的天数占数据集总天数的比例来推荐空车概率,使用最小二乘法拟合达率曲线,来推荐空车等待时间。实验表明,两种推荐算法都有很高的准确率,通过使用Hadoop预先处理并将结果存储,能实现实时在线推荐。

本文实现的推荐算法在数据密集区,算法误差很小,随着数据密度减小,误差逐步增大,误差主要来源是概率统计法和拟合算法本身对稀疏数据的不可靠性,因此,今后在此方向有待深入研究。

参考文献:

[1] 潘纲.移动轨迹数据分析与智慧城市[J].中国计算机学会通讯,2012,8(5):31-36.

[2] Liu S,Liu Y,Ni L M,et al.Towards mobility-based clustering[C]//Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,2010:919-928.

[3] 蒋益娟,李响,李小杰,等.用车辆轨迹数据提取道路网络的几何特征与精度分析[J].地球信息科学学报,2012(2):165-170.
[4] Zheng Y,Liu Y,Yuan J,et al.Urban computing with taxicabs[C]//Proceedings of the 13th international Conference on Ubiquitous Computing,2011:89-98.
[5] Ramos J.Using tf-idf to determine word relevance in document queries[C]//Proceedings of the 1st Instructional Conference on Machine Learning,2003.
[6] Blei D M,Ng A Y,Jordan M I.Latent Dirichlet allocation[J].Journal of Machine Learning Research,2003,3:993-1022.
[7] Breese J S,Heckerman D,Kadie C.Empirical analysis of predictive algorithms for collaborative filtering[C]//Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence.[S.l.]:Morgan Kaufmann Publishers Inc,1998:43-52.
[8] 潘遥,李石坚,潘纲.基于出租车轨迹数据挖掘的乘客候车时间预测[J].软件学报,2013,24(S2):14-23.
[9] Yuan J,Zheng Y,Zhang L,et al.Where to find my next passenger[C]//Proceedings of the 13th International Conference on Ubiquitous Computing,2011:109-118.
[10] Shvachko K,Kuang H,Radia S,et al.The Hadoop distributed file system[C]//Proceedings of 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST),2010:1-10.
[11] Song zhiping.The trifurcate chain-table storage structure of huge directed graph[J].Computer Engineering and Applications,2002,38(21):39-41.
[12] Yuan J,Zheng Y,Zhang C,et al.An interactive-voting based map matching algorithm[C]//Proceedings of the 2010 Eleventh International Conference on Mobile Data Management.[S.l.]:IEEE Computer Society,2010:43-52.
[13] 陆文昌,张迎,陈龙,等.基于计算几何的地图匹配算法研究[J].机械设计与制造,2012(1):43-45.

(上接185页)

[13] Ho J M,Lin S Y,Fann C W,et al.A novel content based image retrieval system using *K*-means with feature extraction[C]//Proceedings of International Conference on Systems and Informatics (ICSAI),2012:785-790.
[14] Pass G,Zabih R.Histogram refinement for content-based image retrieval[C]//Proceedings of the 3rd IEEE Workshop on Applications of Computer Vision(WACV'96),1996:96-102.
[15] 常哲,侯榆青,李明俐,等.综合颜色和纹理特征的图像检索[J].小型微型计算机系统,2011,32(1):161-164.
[16] Yap P T,Paramesran R,Omg S H.Image analysis by Krawtchouck moments[J].IEEE Transactions on Image Processing,2003,12(11):1367-1377.
[17] Narasimhan H,Ramraj P.Contribution-based clustering algorithm for content-based Image retrieval[C]//Proceedings of International Conference on Industrial and Information Systems(ICIIS),Mangalore,Jul 29-Aug 01,2010:441-447.
[18] 程涛.基于颜色和形状特征的图像检索[D].西安:西北大学,2010.
[19] 苑丽红,孙爽滋,付丽.灰度共生矩阵检索纹理图像的算法研究[J].计算机科学,2009,36(11):300-303.
[20] Bounthanh M,Hamamoto K,Attachoo B,et al.Content-based image retrieval system based on combined and weighted multi-features[C]//Proceedings of the 13th International Symposium on Communications and Information Technologies(ISCIT),Surat Thani,Sept 4-6,2013:449-453.
[21] 吕明磊,刘冬梅一种改进的*K*-means聚类算法的图像检索算法[J].计算机科学,2013,40(8):285-288.

(上接258页)

[8] 陶胤强,牛惠民.带时间窗的多车型多费用车辆路径问题的模型和算法[J].交通运输系统工程与信息,2008,8(1):113-117.
[9] 张景玲,赵燕伟.多车型动态需求车辆路径问题建模及优化[J].计算机集成制造系统,2010,16(3):543-550.
[10] Guo H,He J.A colorfractal-based morphing algorithm[J].Computer Modelling and New Technologies,2013,17(3):63-68.
[11] 张海刚,吴燕翔,顾幸生.基于免疫遗传算法的双向车辆调度问题实现[J].系统工程学报,2007,22(6):649-654.
[12] Krainyukov A,Kutev V,Opolchenov D.Reconstruction of the roadway inner structure electro-physical characteristics[J].Transport and Telecommunication,2010,11(4):14-28.
[13] Xue Changjiang.The steel supply chain of forging stability—Rerecognition of current relationship for steel and steel production[J].China's Steel Industry,2007,1:33-35.
[14] Wang Wei,Wu Min,Chen Xiaofang,et al.Raw material stock optimization system based on multiple parallel genetic algorithm[J].Control Engineering of China,2003,10(1):33-37.
[15] Liu Guoli,Tang Lixin,Zhang Ming.A study on raw material inventory in lron and steel industry[J].Journal of Northeastern University:Natural Science,2007,28(2):172-175.