HiTaxi 的设计与实现

《无线网络与移动计算》课程项目结题报告

张灵箫（学号：1201111304）

陈秀招（学号：1201214069）

北京大学信息科学与技术学院

目　录

1 选题依据

都说出租车是一个城市的名片，但现在国内一线城市的出租车服务却很难让人满意，尤其是打车难的问题，平时打车等上一刻钟或是半小时不算什么，刮风下雨天就更不用说了。移动互联网时代的来临，让传统的打车方式面临挑战。移动智能终端随时随身的特性，实在能够给我们带来便捷的体验。借助智能机固有的便携性和标配的 GPS 定位功能，一批十分有特色的交通出行类 APP，如雨后春笋般迅速出现。风投、互联网公司、创业者、通信运营商大佬们似乎都看到了交通出行类 APP 的光明前途，而使一时百花齐放，都梦想着移动互联网能给自己带来的丰厚的回报。

针对这种情况，打车应用应运而生。据相关数据显示，目前国内出租车保有量 120 万辆，与滴滴打车合作的出租车保有量有 6.7 万辆，摇摇招手透露将达到 2.6 万辆，快的打车近五万辆。由此可见打车应用在市场上已经初获成绩，甚至大量的乘客对于这类可以更快地在打车难问题无法彻底解决的城市更快地打到车应用深感不错，甚至还愿意花费更高的价格更快地打到车。而司机方面，在资源明显得到优化的情况下，司机白天使用打车软件接单工作所获得的月收入竟然和以前普通日子努力拉客得到的月收入差不多，这在一定程度上就明显证明了打车应用的存在价值。若没有行政干预，市场格局基本尘埃落定。

1.1 打车应用的机遇

叫车市场期待共赢：无论是乘客还是司机师傅，都有各自的叫车难题。乘客若遇到偏僻地方或者高峰期，想要打上车十分困难。而司机师傅的“空驶率”问题也一直很难解决。打车应用基于这类用户的需求而诞生，在一定程度上缓解了这类问题，实现了各方信息对称。

O2O 最佳路径：利用移动终端普及的优势，快速将网络技术融入人们的线下生活一直是 O2O 所追求的目标。从这个层面来看，打车应用是 O2O 模式的完美案例。改变传统的线下模式，利用手机应用砍掉中间环节，节省乘客和司机的对接成本，同时又拉近了双方的心理距离。

带动创新技术运用：小小的打车应用里其实蕴含着很多可以拥有更大发挥空间的技术，比如语音识别、在线支付、后台资源调度、用户信用管理等。这几方面的技术目前还并未发展成熟。伴随打车应用的体验优化，它们将进一步带动该类技术的发展。

1.2 打车应用的挑战

恶意竞争愈演愈烈：从线下推广人员恶意卸载对方软件，到快的打车与摇摇招车爆发公关口水战，这个行业在没有做大之前，已经陷入恶意竞争的漩涡。太多企业纷纷涌入这一行业，目前国内已有超过 30 款在线叫车 App 应用，在市场未进一步壮大之前，已出现饱和。

盈利模式并不清晰：有 VC 总结，这个行业不缺用户群，但未来如果发展壮大，无论是向用户收费还是向出租车司机收取提成恐怕都会遭到反对。而广告和数据挖掘模式，无论从体验还是实际投入来看未必见效。因此，对于仍处在烧钱阶段的应用，这是一个还没有答案的问题。

面临资源垄断风险：打车市场的特殊之处在于，每个城市都有自己独特的资源，无论是线下车队资源、用户习惯、市场监管特点还是交通状况。掌握了这些资源的机构往往更容易快速形成竞争壁垒。对于打车应用而言，一方面要形成自己的资源，一方面又要破除别人的垄断难题。

一个新产品能同时获得供求双方欢迎的原因很简单，那就是其满足了用户最本质的需求（解决问题、提升效率、收获快乐、降低不确定性等），而移动打车应用满足用户最核心的需求就是提升了用户的效率：出租车师傅的拉客效率提高了，能获得更多的时间支配权;乘客叫车的等待时间与不确定性降低了，所以我们觉得设计一款拥有优质用户体验的打车应用非常有需求而且也是一件很有挑战性的事情，所以我们选 HiTaxi 打车应用作为我们的无线网络与计算课程的工程实践项目。

2 相关工作

2.1 Hailo

Hailo 是专为伦敦黑色出租车（Black Cab）打造的一款智能手机应用，有了它，不管你在伦敦哪个角落，想什么时候打车，你都可以打到出租车。而你要做的就是在手机上点两下，然后选择用现金支付还是刷信用卡，没有任何额外费用。

缺点：虽然 Hailo 目前和伦敦的 2.3 万家持有出租许可证的黑色出租车司机有合作关系，并且已经得到政府许可。但是目前还是仅支持大伦敦地区。并且 Hailo 会从出租车四级那里收取一部分佣金。而且目前只有 iOS 和 Android 版客户端，不支持黑莓和 Windows Phone 版本。

2.2 滴滴打车

全国第一款能打到正规出租车的叫车应用。 拥有全国最多数量的出租车公司鼎力合作。无需拨打人工电话，打开手机就能看到周围行驶的出租车，对着手机说出所在位置及目的地就可以叫来车；即时看到将要接你的司机车牌、电话及所属公司，不用担心黑车的“坑蒙拐骗”。 嘀嘀打车最特别的是它的信用机制，如果乘客有一次爽约，被司机举报，该名乘客的电话号码将被禁止使用嘀嘀打车一周；如果司机未能如约而至，乘客给予的差评会直接通报给承运公司。

缺点：分为司机版和乘客版，存在既拼路程又拼价格的现象，司机经常被放鸽子，维权难，垃圾短信多。虽然现在没有被相关部门强制要求司机卸载，但也因为加价打车服务的缘故成为相关监管部门重点攻击区域。

3 目标和内容

本 app 主要包括两个部分，一个是乘客角色的功能，一个是司机角色的功能，可以分别有不同的操作。

3.1 乘客

预约用车：输入手机、目的地、预约时间等信息，点击“上传信息”按钮；

现在打车：输入手机、目的地等信息，点击“上传信息”按钮。

3.2 司机

显示地图：根据司机当前位置，系统显示地图，地图上有各地的打车热点，司机可以放大缩小地图，

找约车乘客：系统根据司机当前位置，给司机推荐附近的约车乘客；

找打车乘客：司机根据地图上推荐的热点，选择一个地方，然后系统会返回司机这个热点的乘客信息，司机呼应一个乘客后可拨打乘客电话联系。

4 完成情况

4.1 聚类算法实现

本应用的一个核心任务是向司机提供打车热点信息，所谓打车热点，就是附近有较多等车乘客的一个位置坐标。对于海量的上传到服务器的位置信息（同一城市中），我们希望在其中找出十几到几十个数量不定的打车热点，就需要使用聚类算法。

目前流行的聚类算法主要有三种：层次聚类算法，该算法反复对数据进行聚合，不同层次的聚合度不同，它的问题时效率较低，对密度不敏感；划分聚类方法（K-means），该算法反复迭代运算逐步降低目标函数的误差值，但是需要预先指定聚类数目或聚类中心；基于网格和密度的聚类算法，它通过数据密度来发现任意形状的类簇。根据我们的需求来看，在实现不是到类簇数目和类簇大小的情况下，为了发现密度较高的热点，我们选择密度聚类算法。

我们选择了效率较高 DBScan 算法进行聚类。DBSCAN(density-based spatial clustering of applications with noise)算法具有较好的抗噪性能，同时性能适中，符合我们的需求。一下是 DBSCAN 算法的伪码描述:

输入: 包含 n 个对象的数据库，半径 e，最少数目 MinPts;
输出:所有生成的簇，
Repeat：
    从数据库中抽出一个未处理的点；
    IF 抽出的点是核心点 THEN ：
        找出所有从该点密度可达的对象，形成一个簇；
    ELSE ：
        抽出的点是边缘点(非核心对象)，跳出本次循环，寻找 下一个点；
UNTIL 所有的点都被处理。

在真实数据集上测试上述算法过程中，我们发现的主要问题在于，由于该算法以密度为聚类依据，不对类簇的大小做限制，可能导致一个覆盖面过大的类簇。距离来说，海淀区整个区域内打车密度都很大，那么聚类结果中可能就出现一个异常庞大的类簇，覆盖了海淀区一片的区域。这是我们给司机的热点就会是海淀区都某一个中心点，包含的确是海淀区的一大片区域，这样造成热点精确度非常低，对司机也就没有信息量了。

对于该问题，我们采取了限制类簇大小的解决方案。由于聚类完成后的结果中类簇的形状是不规则的，所以后期分割非常困难。所以我们在聚类过程中就对类簇大小进行限制。具体方法是在计算密度可达点时，同时计算到中心点的距离，超过一定距离的店被认为不属于该类簇。修改后的算法描述如下：

Repeat：

从数据库中抽出一个未处理的点；

IF 抽出的点是核心点 THEN ：

找出所有从该点密度可达的对象，形成一个簇；

去掉可达点中距离中心点大于阈值的店；

将这些点放回数据库；

ELSE ：

抽出的点是边缘点(非核心对象)，跳出本次循环，寻找 下一个点；

UNTIL 所有的点都被处理。

上述方案能够较好的解决类簇过大的问题，同时会带来一个将高密度点搂抱的副作用。搂抱的点占总数比例较小，所以我们没有对其进行进一步处理。

我们用上述修改后的算法，在北京市方圆 15 公里内随机生成 1000 个位置坐标进行测试，最后得到 39 个类簇，平均每个类簇含有 8 个点，精确度上能够符合我们的预期。

4.2 服务器端的实现

服务器端主要完成以下三个功能：

- 数据存储与维护

用户通过手机客户端上传位置信息之后，服务器端对其进行存储，并且需要定期对数据进行维护。在用户上传数据后，并没有对其后续处理，所以如果服务器端不对数据进行维护，那么过期数据将会永远留在数据库中。

我们将每个用户上传的数据打上时间戳，服务器端定期对超过一个时间阈值的数据进行删除，这样就解决了上述问题。

我们用 mysql 数据库作为后台数据存储，数据库中只需要包含一个表，里面存储了用户姓名、联系方式，以及所处位置的经度、纬度。

- 提供远程数据接口

本应用中，手机端不能完成 CPU 和 IO 密集的聚类算法，需要和服务器进行数据从而得到结果。由于通行频率和每次通讯的数据量都不打，我们采取最普适的 HTTP 请求方式进行通讯。

用 HTTP 协议通讯的主要好处有：

- ✓ 实现简单，错误处理等逻辑封装良好。
- ✓ 客户端服务器端耦合度降到最低。
- ✓ 普适性强。客户端的数据接口写成网络服务模式，可以很方便的给其他应用

提供接口。

实现时我们采用 Java 的 Servlet 框架，将每个服务封装成一个 Servlet。手机端发送请求后，servlet 处理数据，并将结果封装为统一的 json 格式返回给手机端，手机处理返回数据后显示。

- 计算聚类结果

处于效率和能耗的考虑，我们不能让手机端完成 CPU 和 IO 密集的聚类算法，所以我们把这个工作交给服务器端完成。

计算聚类结果主要分为以下几个步骤：

1. 读取乘客位置数据
2. 数据维护，删除过期数据
3. 执行聚类算法
4. 依次将类簇中心坐标通过百度地图 API 得到附近的 POI 作为类簇名称

以上步骤消耗较多的 CPU 和网络资源，不可能对每一个请求都单独执行一遍该过程。我们给出的结局方案是：在内存中维护一个存放聚类的结果的对象，每个 http 请求从该对象中得到数据；同时运行一个线程定期重复上述过程，每一次得到结果后，就更型该变量，这样就达到了结果实时更新，又不必耗费太多硬件资源的目的。

以上是服务器端功能的具体描述，下面给出项目详细的代码清单和各个类的功能。

服务器端程序主要分成 4 个包，每个包内容的详细描述如下：

- Data

  主要用于存放表示数据模型的类

  - class CustomLocation

    该类用于存放表示客户信息的数据模型，模型包括客户姓名，联系方式，以及通过 GPS 传感器获得的坐标位置信息。

  - class LocationCluster

    该类用于存放表示一个类簇结果的数据模型，其中包括了一个包含于这个类簇的客户信息列表，以及该类簇的名字。该类簇的名字为一个 POI（point of interest）地点的名称，该名称通过将类簇中心点坐标提交给百度地图 API 而得到。

- Dbscan

  主要用于存放和 DBscan 聚类算法相关的类。

- class Dbscan

  实现聚类算法的泪。输入一个用户信息的列表，该类给出聚类结果。

- class Utility

  存放和聚类算法相关的一些工具方法。

- Service

  属于业务逻辑层，存放实现主要业务逻辑的类。

  - class MainService

    实现最主要的业务逻辑，包括读取数据，调用 DBscan 类完成聚类算法，得到结果，访问百度 API，维护数据等。该类中的主要方法以线程的方式被异步调用，从而实现在后台维护数据的功能。

  - class Utils

    主要存放和业务逻辑相关的公共工具类。

- Servlet

  存放 servlet 类，实现所有网络服务（http）接口

  - class GetCustomDetail

    得到打车热点具体信息的网络服务接口。手机端提供热点的 index 后，该方法查询内存中维护的聚类结果列表，将 index 指向的热点中的用户列表返回给手机端。

  - class GetHotSpotList

    得到打车热点列表的服务。向手机端返回所有的打车热点名称和人数信息。

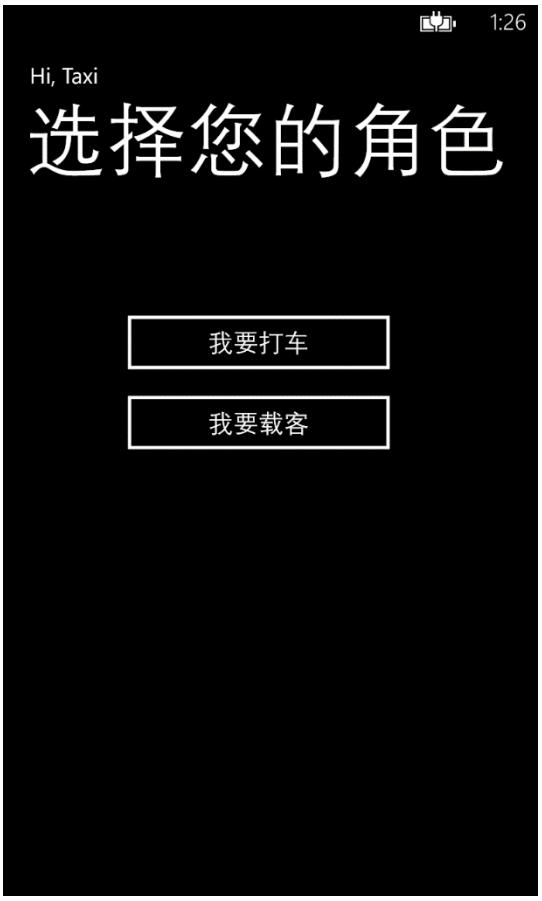
  - class MainServlet

    启动维护进程的服务。服务器启动后会自动调用该服务，该服务启动一个维护线程，在其中调用 mainService 的相关方法来维护内存中的结果数据。

  - class UploadUserInfo

    上传用户信息的服务。用户信息和个人位置通过该服务从手机端上传，并存到数据库中。

4.3 手机端展示

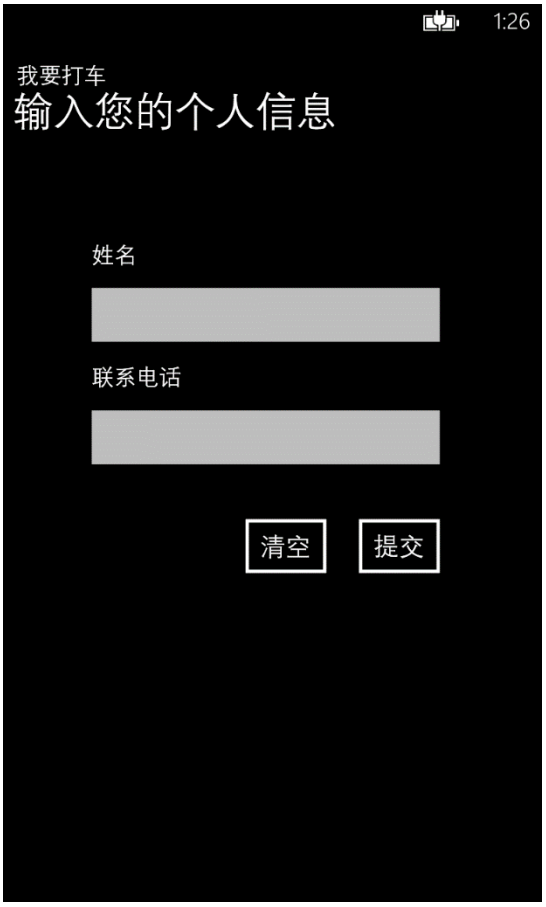


相比服务器端，手机端的功能较为简单，主要的任务就是发送请求，得到数据，处理后向用户展示数据。

手机端代码结构较为清晰。每一个手机上的页面都对应一个 XAML 文件，其中包含了控件的属性和布局等内容。每一个 XAML 文件对应一个 CS 文件，其中包含了 C#代码，用户相应的数据处理和页面跳转逻辑的实现。

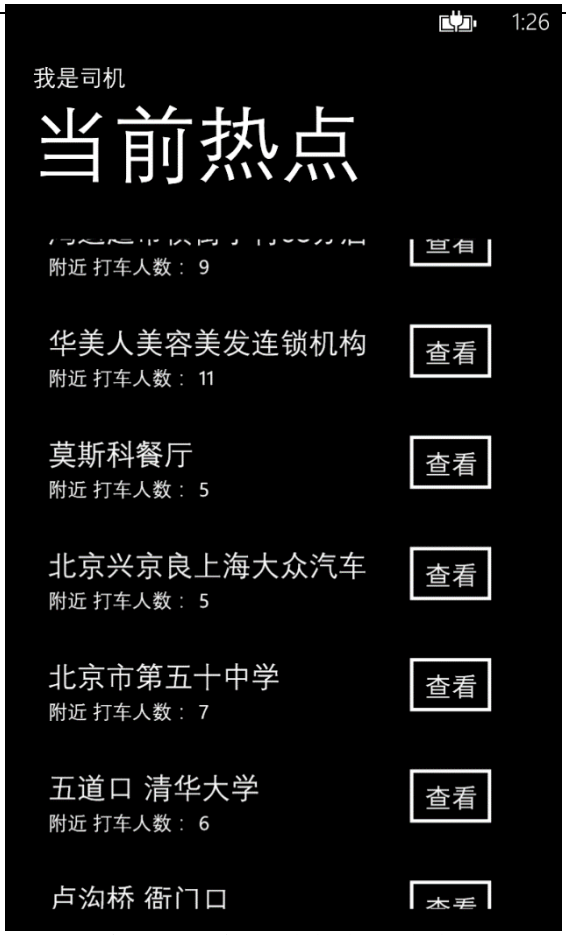
以下是手机端运行效果的展示：

首先是用户进入主界面，包含简单的两个按钮：

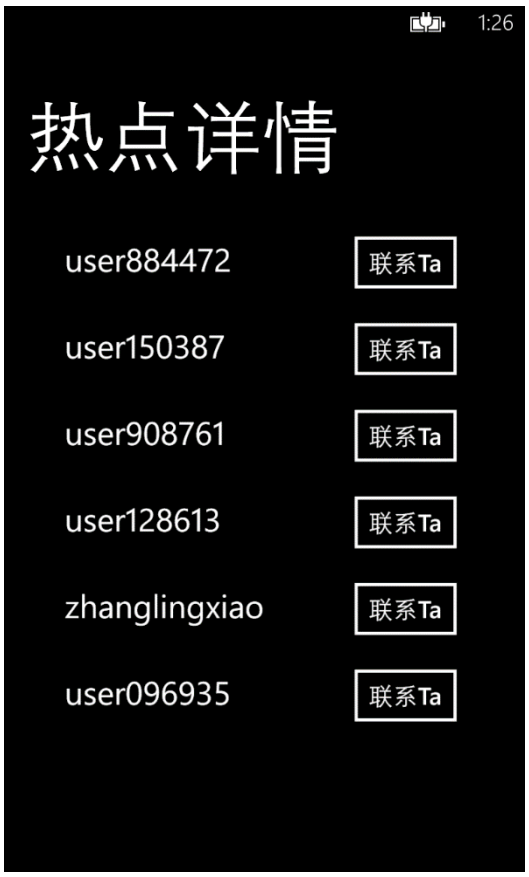


用户为打车客户，那么点击我要打车按钮，进入右图页面，可在其中填写个人信息后，上传到服务器。手机会自动将 GPS 坐标一同上传。

如果用户是司机，那么点击我要载客按钮，界面显示所有打车热点的 POI 名称，以及附近的打车人数：



用户点击查看，给出热点附近用户姓名，并且司机可以直接通过旁边的联系按钮拨打乘客的电话。



5 结果分析及结论

经过一学期的学习与实践，终于完成了 HiTaxi 打车应用项目，最终完成的 HiTaxi 应用能够实现预想的功能并有不错的效果和性能。

通过《无线网络与移动计算》这门课的学习，我们将课堂讲授的知识与实际工程项目结合起来，既丰富了理论储备，又锻炼了工程能力，收获颇多。非常感谢严伟老师一学期的辛勤耕耘，最后再向老师表达我们衷心的感谢！

6 附：参考文献

[1] Programming Windows Phone 7, Charles Petzold

[2] Windows Phone 7 程序设计 = Programming Windows Phone 7 microsoft silverlight edition, Charles Petzold

[3] Windows Phone 程序设计 XNA 框架 = Programming Windows Phone 7 Microsoft XNA framework edition, Charles Petzold

[4] Windows Phone 3G 手机软件开发, 杨云