浙 江 大 學

本 科 生 毕 业 设 计 报 告

项目名称:	基于微博签到数据的 个性化场所推荐系统开发
姓名:	陶 源
学号:	3120101138
指导教师:	陈 岭
专业:	计算机科学与技术
学院:	计算机科学与技术

A Dissertation Submitted to Zhejiang University for the Degree of Bachelor of Engineering

TITLE: Weibo's check-in data based POIs Recommendation System Development

Author: Yuan Tao

StudentID: 3120101138

Mentor: Ling Chen

Major: Computer Science and Technology

College: Computer Science and Technology

Submitted Date: 05/30/2016

浙江大学本科生毕业论文（设计）诚信承诺书

1.本人郑重地承诺所呈交的毕业论文（设计），是在指导教师的指导下严格按照学校和学院有关规定完成的。

2.本人在毕业论文（设计）中引用他人的观点和参考资料均加以注释和说明。

3. 本人承诺在毕业论文（设计）选题和研究内容过程中没有抄袭他人研究成果和伪造相关数据等行为。

4. 在毕业论文（设计）中对侵犯任何方面知识产权的行为，由本人承担相应的法律责任。

毕业论文（设计）作者签名：

_____年_____月_____日

摘要

随着智能终端设备的普及和无线通讯网络的快速发展，传统位置服务与移动社交网络逐渐融合，推进了虚拟与现实产生关联，在提高信息质量的同时，也为互联网带来了更多纷繁的信息与新兴的业务。其中，推荐系统作为信息过滤的重要手段，深入研究用户模型，为用户提供个性化的信息推荐服务，在位置服务中发挥着重要作用。基于协同过滤的推荐系统是当下主流的推荐方法，然而，数据稀疏性和用户冷启动一直是传统算法亟待解决的难点。

本文以实现个性化场所推荐算法为目标，搭建了基于微博签到数据的个性化场所推荐应用。本项目的整体实现主要分为数据处理和系统开发两个部分。数据处理部分将微博签到数据规格化，建立起用户、场所和签到数据模型。利用 python 实现的随机梯度下降的矩阵分解来探索用户对于场所类别的偏好。在 HITS 算法的基础上加入了用户偏好的权重，然后进行排序，实现了个性化的推荐。系统开发采用 Django 后端框架实现核心算法的调用以及用户信息的管理，使用前端安卓应用完成用户的交互，并通过列表方式和百度地图来完成个性化推荐的功能。通过前端安卓与后端 Django 的交互实现了功能完整的个性化场所推荐系统。

测试表明目前实现的个性化场所推荐系统功能完整，满足用户需求。在实现用户信息系统和推荐算法的基础上，本系统预留了一些可扩展使用的数据和模块，方便了后期的维护和算法的进一步改进。

关键词 位置服务 推荐系统 HITS 矩阵分解 Django Android

Abstract

With the popularity of intelligent terminal devices and the rapid development of wireless communication networks, the traditional location-based services and mobile social networks gradually joined together, which brought the location-based social network(LBSNs). LBSNs provide numerous and complicated information and emerging business. Traditional recommendation systems based on collaborative filtering play an increasing role in filtering information in an overloaded information system. However, the user-point of interest(POI) matrix is usually very sparse and the cold start problem is always the challenge for the system.

To implement the personalized location recommendation algorithm, this paper builds the weibo's check-in data based POIs recommendation system. The overall system implementation is divided into two modules: data processing module and system development module. The data processing module normalizes the weibo's check-in data and establishes the user-POI category matrix. The algorithm uses matrix decomposition to explore the preference of the user towards the categories of the POIs. Based on the HITS algorithm, the user's preference matrix is added to implement the personalized recommendation. The system development module uses Django as the back-end framework for the deployment of applications, and displays the data with the front-end Android application by the formats as list and on map. It has achieved a fully functional personalized POI recommendation system.

Tests were made to ensure that the Recommendation System has completed functionality and users' requirements are met. It has met the actual needs of development plan and reserved some maintainability and extendibility.

Keywords Recommendation System, matrix decomposition, HITS, Django, Android

目录

第1章 项目背景

1.1 项目背景和意义

最近皮尤调查数据显示，2015 年全球智能手机占有率从 2014 年的 28%增长到了 37%，而在发达国家中的占有率中位数达到了 68%。智能手机等移动智能终端设备已逐步成为人们日常生活中非常重要的组成，在不同方面潜移默化地影响人们的行为。随着无线通讯网络的覆盖率提高和基础设施的普及，移动社交网络（Social Network Service）成为人们社会交往和获取信息的重要渠道。而与此同时，全球定位系统等位置信息获得技术的高速发展，也给用户带来了便捷。绝大多数用户开始通过智能手机等移动终端设备获取基于位置的信息，特别是众多移动社交网络中的位置签到和位置标签应用。因而，传统位置服务（Location Based Sevice，LBS）与社交网络逐渐融合，形成了一些基于位置的社交网络（Location-Based Social Networks，LBSNs）。[1]

通过移动智能终端设备对位置信息的传递和分享，基于位置的社交网络使得虚拟的线上社交关系和真实的线下交往产生关联，在提高信息质量、加固虚拟世界与物质世界联系的同时，也给互联网带来了更多纷繁复杂的信息和一些新兴服务。[2] 在学术界，自 2009 年以来，ACM SIGSPATIAL GIS 已连续几年举办基于位置的社交网络主题研讨会，就 LBSN 中的热点问题，如位置隐私、位置预测、事件发现、用户移动行为模式挖掘以及相关的预测推荐问题进行研究和探讨。其中，推荐系统作为信息过滤的重要手段，能够深入研究用户模型，为用户提供个人化的信息推荐服务，在位置服务中发挥着重要的作用。

相比传统在线社交网络推荐系统，引入了地理位置信息的推荐系统可以更好地发现用户真实喜好和线下行为模式。目前，基于位置的社交网络推荐系统中，地点推荐满足了人们在特定情境下的地理兴趣需求，已经成为了最近研究的重点。[3] 然而，因为个人用户访问的地理位置有限，基于位置的社交网络中大部分地点访问量稀缺，而新用户访问历史也有限，地点推荐不可避免地面临数据稀疏和冷启动的问题。[4]

因此，本项目主要基于微博上的签到数据，为用户作个性化的地点推荐，实现了一个完整的地点推荐系统，尝试解决了数据稀疏、冷启动等各种推荐系统中

的常见问题，针对不同用户提供了个性化的地点推荐，拓展了用户的社交渠道。

1.2 项目概述

本项目全称为基于微博签到数据的个性化场所推荐系统。该项目是依据从网络上爬下的微博签到数据，从实际的地理位置（POI）个性化推荐需求出发，利用现有较为成熟的基于矩阵分解和 HITS 的社交网络数据挖掘相关算法与模型的研究成果，以 Java、XML 等技术开发安卓客户端，以 Python、Django、Django-REST-framework、 MySQL 等技术开发后台处理所构建出来的系统。

系统主要分为以下四个模块：

用户系统管理模块：主要处理用户信息的建立，认证与管理

地点信息管理模块：主要处理地点信息的导入与管理

数据预处理和个性化推荐模块：对现有签到数据进行分析处理，生成中间数据，加速用户推荐速度，提高用户体验

推荐结果展示模块：对指定用户使用不同的过滤器进行推荐，同时也对推荐结果进行列表和地图上可视化的展示

1.3 本文的组织结构

本报告的组织结构如下：

第一章介绍个性化地理位置推荐系统的背景与内容，主要包括本项目所要解决的实际需求，项目意义和创新点以及本项目需要完成的工作内容等。

第二章描述项目的设计框架，并介绍本人在其中采用的关键技术与核心算法。

第三章详细介绍服务器端的需求、设计和最后实现。

第四章详细介绍客户端的需求、设计和最后实现

第五章展示对客户端和服务器端的功能测试过程。

第六章为项目总结，包括对目前成果总结，对项目不足之处的分析和相关改进思路以及系统可能发展的讨论。

其中第三章到第五章为本文作者的重点工作。

第2章 项目实施方案

2.1 项目目标规划

2.1.1 项目目标

基于上述项目背景，本系统总体目标是设计和实现一个基于 Python 服务器端并由安卓客户端 APP 呈现的个性化地点推荐系统，同时实现较为完整的用户信息管理和维护。具体而言，提出以下几点功能上的目标：

2.1.1.1 服务器端

- 使用现有微博签到数据、系统新用户的签到数据和地点信息，完成数据预处理和分析
- 实现推荐算法，并将功能接口对接到 APP 应用平台
- 建立用户系统，管理用户的创建、认证、更新，维护用户基本信息及历史记录
- 录入用户新的签到、喜爱、想去的数据，更新数据库并定期更新模型

2.1.1.2 安卓客户端

- 定位用户所在地点，对附近用户可能感兴趣的地点进行个性化推荐并展示
- 用户查询功能，对用户分类查询的推荐结果展示
- 用户登录登出功能，以及用户个人信息和相关地点关系的管理功能
- 提供良好的用户交互体验，以及对于大部分安卓版本的支持

2.1.2 项目任务

该项目是在已有算法和微博签到数据基础之上开发一个用户个性化场所推荐系统。具体而言，本人的主要任务可以按时间序分为以下四个部分：

- 前期对现有地理位置推荐应用的调研与分析，了解用户需求和项目技术难点，可行性
- 项目初期对前沿技术的学习和对项目具体实施架构的规划，详细设计，为之后代码编写的环节做准备
- 项目过程中服务器端与安卓客户端的具体开发和调整

- 项目后期对整个系统的测试和完善，同时编写对应的用户手册

2.1.3 项目计划

表 2-1 项目计划进度表

目录	任务	开始时间	结束时间	备注
需求分析	需求分析与整理	2015/11	2016/2	查找和阅读相关文献，了解项目背景
概要设计与详细设计	系统架构，详细设计	2016/3/5	2016/3/20	写完开题报告和文献翻译，系统性地规划整个工程，安排项目进度
	数据库设计	2016/3/21	2016/3/24	
	界面设计	2016/3/25	016/3/30	
系统实现与测试	代码编写	2016/3/31	2016/4/30	撰写中期报告
	系统测试	2016/5/1	2016/5/10	发布应用，获取反馈
毕业设计报告撰写	整合资料，编写报告	2016/5/11	2016/5/30	准备答辩

2.2 项目框架

本项目主要包括四个模块：用户系统管理模块，地点信息管理模块，数据预处理和个性化推荐模块，推荐结果展示模块。这四个模块的功能相对独立，下面本人将分别阐述这四个模块的模型建立和功能设计。

2.2.1 用户系统管理模块

该模块主要负责维护系统中注册的新用户信息，保存并管理用户的用户名、密码、性别、邮箱、头像等基础信息，同时同步更新推荐模块中对应数据库中的用户信息。因为推荐模块原始推荐采取的用户信息并非系统中注册的用户信息，而是从微博数据中获得的，因而规划分为两个模块中的不同用户表储存信息。

用户系统管理模块计划由 Django 内置的用户认证系统扩展而成，Django 内置的认证系统已经覆盖了很多认证、权限和授权系统，可以方便地由用户进行自定义扩展，在接下来 2.3.1 节中会有阐述。

2.2.2 地点信息管理模块

该模块主要负责维护地点信息，保存地点名称，位置，签到人数和分类等信息。目前暂时不对外提供更改的接口，只是在签到、喜爱和想去的数据变更时同步更新地点的被喜爱数目，作为对象的属性在推荐系统展示时被调用。

地点信息管理模块是最基础的 Django Model 模块，View 中封装返回展示模块所需要的地点属性。

2.2.3 数据处理和个性化推荐模块

该模块主要负责原始数据的预处理，签到、喜爱和想去的数据更新，具体的推荐化算法实现。

具体而言，预处理时过滤噪点地点并维护地点和分类的一对多关系数据表。用户签到时，插入包括签到用户、地点和时间的签到数据，同时同步更新用户模块中用户签到次数与地点模块中地点签到总次数和签到人数。用户喜爱和想去操作是相仿的，先检索数据表中是否已存在用户与该地点的关联，不存在则插入该条记录，存在则删去该条记录，最后同步更新用户喜爱条目数与地点被喜爱数。算法的具体实现在接下来的 2.4 节中会作详细论述。

2.2.4 推荐结果展示模块

该模块主要负责推荐结果的展示，同时作为用户交互层，提供用户查看和更新用户档案的接口，用户签到等操作的实施。

与其它模块不同，推荐结果展示模块主要是安卓客户端，因而与其他模块间的信息交换主要通过 HTTP 协议操作。同时在具体考虑展示效果下，该模块提供了两种展现方式：条目式的列表展示和调用百度地图 API 的地图展示。

2.3 关键技术

2.3.1 Django

Django 是一个由 Python 写成的开放源代码的 Web 应用框架，广泛用于快速开发轻量级网站，它鼓励快速开发，并采用了 MVC 的软件设计模式。Django 注重组件的重用性和“可插拔性”，敏捷开发和 DRY 法则(Don't Repeat Yourself)。在一个 project 下会有很多可插拔的 app 应用。它有很多功能十分强大的第三方插

件，比如在接下来 2.3.2 节中即将介绍的 Django-Rest-framework，因此具有很强的可扩展性。本节将详细介绍 Django 的 MVC 设计和可扩展的认证系统应用。

2.3.1.1 Django MVC 设计（URLconf + MTV）

MVC 是常见的设计模式，即将应用程序分解为 Model(模型)，View(视图)和 Controller(控制器)三个模块。Model 与数据库相连接，并处理业务规则。View 负责把数据格式化后呈现给用户。 Controller 接受外部用户的操作，根据操作访问 Model 获取数据，并调用 View 来显示数据。通过 Controller 将 Model 与 View 隔离开，形成松耦合的设计模式，便于开发。

而 Django 中的 MVC 框架设计中，Controller 中接收用户输入的部分由框架内部承担了，于是更关注的是 Model(模型)，Template(模板)，View(视图)三个模块，称为 MTV。具体而言，对于每个 app 默认有以下几个文件：models.py, views.py, urls.py, templates.py。其中，models.py 文件使用一个称作模型的类来描述数据库并封装了关于数据库存取查询等操作。负责数据从数据库中存入取出。views.py 文件包含了页面的业务逻辑，负责存取模型以及调用 templates.py 等合适的模板渲染展示。从而，Django 将传统 MVC 中的视图进一步分解为视图和模板两部分，分别决定“展现什么数据”和“如何展现”，达到了更好地解耦和的目的。urls.py 则对应着传统 MVC 架构中的 Controller，用正则表达式匹配对应的 url，然后调用 views.py 中合适的 Python 函数。[5][6][7]

这样的框架设计，使得 Django 中各模块之间的耦合十分松散，非常便于独立更改。

2.3.1.2 Django Auth 模块

Auth 模块是 Django 自带的用户认证模块，它提供了封装完整的用户管理功能和一些认证和授权功能。 Auth 内置的配置已经逐步可以满足大部分常见项目的需要，处理范围非常广泛的任务，且具有一套细致的密码和权限实现机制。[8]

Auth 模块基本已经提供了本项目所需的登入登出以及限制访问功能，对于 Anonymous 匿名用户的访问处理。同时对于需要与默认配置不同的需求，Django 1.5 版本后的该模块也支持扩展和自定义认证，这解决了之前为人所诟病的模块内部紧耦合难以改变的问题。比如本项目中的 User 继承了 AbstractUser 类，扩展了一些项目所需的属性。

表 2-2 继承 AbstractUser 的自定义 MyUser

```
# -*- coding: utf-8 -*-
from __future__ import unicode_literals
from django.contrib.auth.models import AbstractUser, UserManager
from django.db import models

class MyUser(AbstractUser):
    city = models.IntegerField(default=0)
    province = models.IntegerField(default=0)
    desc = models.CharField(max_length=400, null=True)      # intro
    gender = models.CharField(max_length=2, choices=[('m', '男'), ('f', '女')],
null=True, blank=True, verbose_name='性别')
    date_of_birth = models.DateField(blank=True, null=True, verbose_name='出
生日期')
    avatar = models.ImageField(upload_to='avatar/%Y/%m/%d',
default='avatar/default.png', blank=True, verbose_name='头像')
    likeNum = models.IntegerField(default=0)
    todoNum = models.IntegerField(default=0)
    doneNum = models.IntegerField(default=0)

    objects = UserManager()

    def __unicode__(self):
        return unicode(self.username)
```

2.3.2 Django-Rest-framework

Django REST 框架可以轻松部署 web APIs，是一个集鲁棒性与弹性为一体的 web 工具包。它提供了封装好的序列化和反序列化函数，如可以直接序列化模型中属性的 ModelSerializer，更为友好地，使用 HyperlinkedModelSerializer 是

一个好的 RESTful 设计，自动封装 API 下别的链接地址。[9]

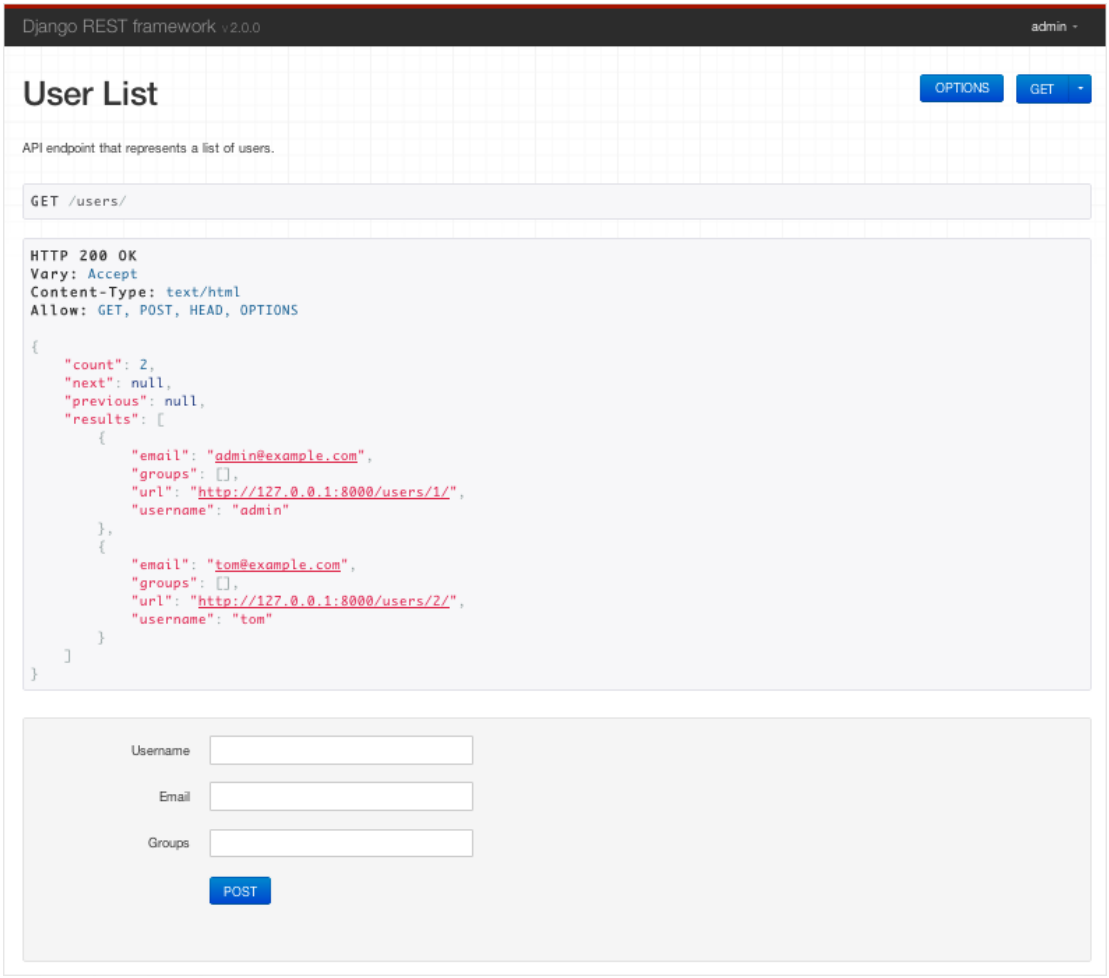

图 2-1 Django REST framework 界面实例

同时，django-rest-framework 为每一个状态码都提供了打包好的，非常精确的状态码完整描述供我们使用，比如 HTTP_400_BAD_REQUEST。这极大地方便了本项目的调试。[10]

2.3.3 uWSGI + nginx 服务器部署

Django 框架方便开发，然而并不适合在生产环境下使用，因此需要进一步将后端代码部署到服务器上。具体生产环境配置上，我选择了 uWSGI + nginx 的组合。nginx 作为服务器最前端，接收 WEB 的所有请求，统一管理请求。nginx 把所有静态请求自己来处理。然后，nginx 将所有非静态请求通过 uwsgi 传递给 Django，由 Django 来进行处理，从而完成一次 WEB 请求。

其实就本项目而言，目前并不是很需要 nginx 作负载均衡，不过因为配置相对简便，而且考虑到以后系统的扩展性，先配置了 nginx。

2.3.4 MySQL

MySQL 是一个开源的体积小、速度快、多线程的轻量级关系型数据库管理系统。它支持常规的 SQL 数据库查询并有自己的扩展，能提供非常复杂查询结果。MySQL 的核心程序采用完全的多线程编程，可以灵活地同时为多用户提供服务，而不过多占用系统资源。

因此，出于性能和开发考虑，本人选用了 MySQL 作为开发时的数据库选择。

2.3.5 Android Studio + Gradle

Android Studio 是 Google 于 2013 I/O 大会针对 Android 开发新推出的开发工具，是 Google 大力支持的一款基于 IntelliJ IDEA 改造的 IDE。同时整合了 Gradle 构建工具，更完善的插件系统和各大版本控制系统。

Gradle 是一个基于 Apache Ant 和 Apache Maven 概念的项目开源自动化建构工具。它使用一种基于 Groovy 的特定领域语言（DSL）来声明项目设置，抛弃了基于 XML 的各种烦琐配置。它具有强大的依赖管理和可切换的，基于约定的构建框架，能够为项目加入依赖，并轻松移植。

2.3.6 百度地图 API

百度地图 API 是为开发者免费提供的一套基于百度地图服务的应用接口，包括 JavaScript API、Web 服务 API、Android SDK、iOS SDK、定位 SDK、车联网 API、LBS 云等多种开发工具与服务，提供基本地图展现、搜索、定位、逆/地理编码、路线规划、LBS 云存储与检索等功能，适用于 PC 端、移动端、服务器等多种设备，多种操作系统下的地图应用开发。

本项目中，我采用了百度地图 API 提供的定位服务、基础地图和计算工具。基于 GPS、WiFi、基站的综合定位服务，具有定位精度高、覆盖率广、网络定位请求流量小、定位速度快等优势。而基础地图包括最基础的带有手势放大、缩小、移动等交互的地图展示功能，还有自定义的地图覆盖物，很好地解决了本项目对于用户个性化推荐场所的自定义展示。

2.4 关键算法

2.4.1 矩阵分解

针对个性化推荐系统中协同过滤算法所面对的矩阵稀疏和新用户冷启动问题，我们选择采取矩阵分解算法来对原始用户 – 项目评分矩阵进行补全， 从而减小矩阵稀疏和数据量大的影响，提高推荐系统的准确性。矩阵分解目标就是把用户 - 项目评分矩阵 R 分解成用户因子矩阵和项目因子矩阵乘的形式，即 R=PQ，这里 R 是 n×m，则 P 是 n×k，Q 是 k×m，提取出 k 维的隐因子。[11]

具体到本系统，矩阵分解的对象是将用户签到矩阵按类别统计后的用户 – 类别矩阵，其中评分对应的是用户签到次数。矩阵分解后得到经过补全后各用户对于不同类别的偏好矩阵。[12]

我采用了随机梯度下降法来对矩阵分解进行求解，核心代码如下：

表 2-3 矩阵分解核心代码

```
def matrix_factorisation(S,R, P, Q, K, steps=25, alpha=0.002, beta=0.02):
    Q = Q.T
    for step in xrange(steps):
        for (i,j) in S:
            if R[i][j] > 0:
                eij = R[i][j] - numpy.dot(P[i,:],Q[:,j])
                for k in xrange(K):
                    P[i][k] = P[i][k] + alpha * (2 * eij * Q[k][j] - beta * P[i][k])
                    Q[k][j] = Q[k][j] + alpha * (2 * eij * P[i][k] - beta * Q[k][j])
        e = 0
        for (i,j) in S:
            if R[i][j] > 0:
                e = e + pow(R[i][j] - numpy.dot(P[i,:],Q[:,j]), 2)
                for k in xrange(K):
                    e = e + (beta/2) * (pow(P[i][k],2) + pow(Q[k][j],2))
        if e < 0.001:
            break
        e = e/(len(R)*len(R[0]))
        print "Step:%d Error:%f"%(step,e)
    return P, Q.T
```

2.4.2 HITS 链接分析算法

HITS 算法是 Web 结构挖掘中最具有权威性和使用最广泛的算法，根据一个

网页的入度(指向此网页的超链接)和出度(从此网页指向别的网页)来衡量网页的重要性。Hub 页面和 Authority 页面是 HITS 算法最基本的两个定义。Authority 页面，是指与某个领域或者某个话题相关的高质量网页。Hub 页面，是指包含了很多指向高质量“Authority”页面链接的网页。HITS 算法的核心思想在于 Authority 和 Hub 两个属性是互相增强的。因而通过两个属性的不断相互迭代，更新每个页面的 Authority 权值和 Hub 权值，可以得到较为稳定的 Ranking 结果。

具体到本系统，用户和地点之间的关系也可以类比为 HITS 模型。更高质量(Popular)的地点会被更多的权威用户访问，而权威(Authority)的用户会访问更多高质量的地点。因此也是可以进行迭代计算的互相增强关系。[13]

考虑到针对特定用户的个性化推荐，结合上一节 2.4.1 中矩阵分解所得到的用户偏好，我们对 HITS 算法中用户指向 POI 的边赋上对应代表用户-类型偏好的权值（当一个地点有多个类别属性时选用用户偏好值最大的那个作为地点的权值），从而不断迭代得到协同过滤后的针对特定用户的个性化场所推荐。

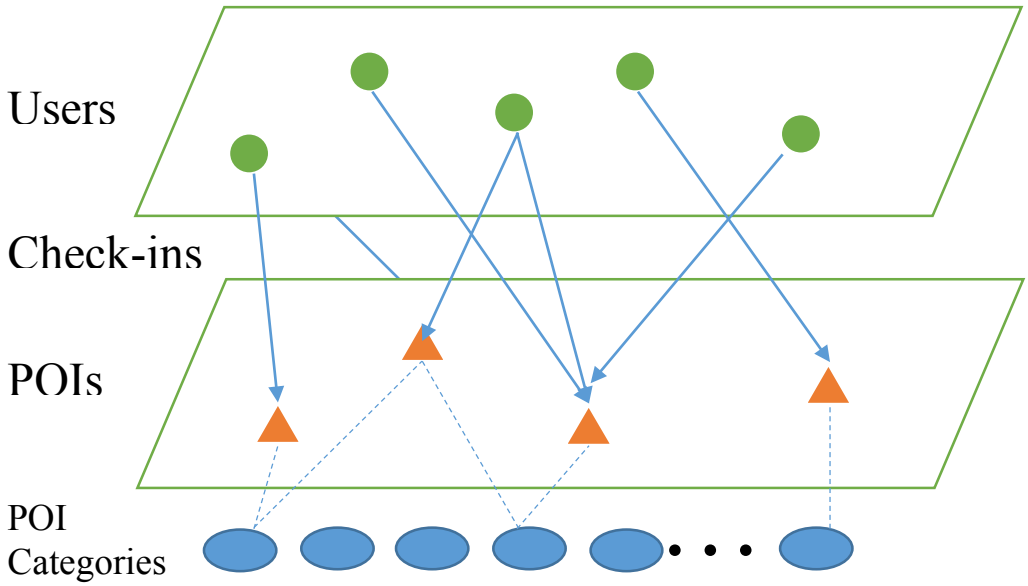

图 2-2 HITS-based 地点推荐算法

2.5 设计方案

2.5.1 微博签到数据预处理

基于微博签到数据的个性化场所推荐系统的数据处理流程主要包括：原始数据录入、有效信息过滤，和数据解释三步预处理 。

2.5.1.1 原始数据录入

本系统所使用的原始数据来自新浪微博平台中爬取的用户发表状态时的签到信息，获得数据后，将数据按一定格式存入 users_hz，pois_hz 和 checkin_hz 三个表中的结果。

2.5.1.2 有效信息过滤

原始数据中有一部分内容不合适，表达不规范的数据，因此，我们需要在正式使用前对数据进行梳理，以便于后续使用，最后将过滤后的数据存入新的表中。

主要问题有：

- 地理语义重复：如“吉木咖啡”、“吉木 coffee”，其实是同一个场所的不同表示
- 地理数据不准确或语义含糊：如“泰国”、“启真湖底”、“来自 iPhone6 Plus 客户端”，这分别代表了错误定位、用词模糊和错误数据这三种不同错误的地理数据。
- 不同地理范围语义：如“杭州市”、“杭州 西湖”、“杭州 西湖 黄龙沿线”

2.5.1.3 数据解释

原始数据中有些数据采取了便于存储，却不便理解的格式。在向用户呈现的时候和存入数据库的时候都要做一次转换。

checkin_hz 表中 checkin_time 是时间戳格式，而 django 模块 check 模型中设置的 auto_now 字段储存的是当前时间。

pois_hz 表中 Categorys 格式是 varchar(10)，存储了地理位置对应的场所类型，是用逗号隔开的整数。

pois_hz 表中 province, city 存成了 int 属性的编号。具体与省市的对应关系可以在新浪微博开发平台上的省份城市编码表里找到。

2.5.2 数据库设计

个性化地理位置推荐系统的数据主要由四部分构成：用户信息，地点信息，用户和地点之间的关系信息和算法产生的中间数据。数据库中表如下：

表 2-4 数据库中的表

Tables_in_v1	Description
auth_group	Django 内置模块表，维护定义的用户组
auth_group_permissions	Django 内置模块表，维护用户组和对应权限的关系
auth_permission	Django 内置模块表，维护不同模型的不同权限
django_admin_log	Django 内置模块表，存放 admin 的操作记录
django_content_type	Django 内置模块表，维护 project 中所有用户模型
django_migrations	Django 内置模块表，存放数据库修改记录
django_session	Django 内置模块表，维护 session 的 id 和属性
poisManage_poi	poiManage 模块表，维护 POI 信息
poisManage_cat	poiManage 模块表，维护 POI 的类别信息
poisRecommend_check	poisRecommend 模块表，存放用户签到记录
poisRecommend_data_user	poisRecommend 模块表，存放数据库中的用户信息
poisRecommend_like	poisRecommend 模块表，存放用户与 POI 之间的喜爱关系
poisRecommend_todo	poisRecommend 模块表，存放用户与 POI 之间想去的关系
userManage_myuser	userManage 模块表，维护客户端注册用户信息
userManage_myuser_groups	userManage 模块表，维护用户组的对应关系
userManage_myuser_user_permissions	userManage 模块表，存放特定用户的权限

个性化场所推荐系统的用户信息由 MyUser 模型维护，对应表如下：

表 2-5 userManage 模块下模型 myuser

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	自增主键
password	varchar(128)	NO	\	NULL	用户密码
last_login	datetime(6)	YES	\	NULL	用户最后登陆时间
is_superuser	tinyint(1)	NO	\	NULL	用户权限
username	varchar(30)	NO	UNI	NULL	用户名
email	varchar(254)	NO	\	NULL	用户注册邮箱
is_staff	tinyint(1)	NO	\	NULL	用户是否可以登陆 admin
is_active	tinyint(1)	NO	\	NULL	用户是否被禁用
date_joined	datetime(6)	NO	\	NULL	用户注册时间
uid	varchar(50)	YES	\	NULL	用户 ID
access_token	varchar(100)	YES	\	NULL	访问时的 Token
city	int(11)	NO	\	NULL	用户所在城市
province	int(11)	NO	\	NULL	用户所在省份
desc	varchar(400)	YES	\	NULL	用户签名档
gender	varchar(2)	YES	\	NULL	用户性别
date_of_birth	date	YES	\	NULL	用户出生日期
avatar	varchar(100)	NO	\	NULL	用户头像
likeNum	int(11)	NO	\	NULL	用户喜爱地点数目
todoNum	int(11)	NO	\	NULL	用户想去地点数目
doneNum	int(11)	NO	\	NULL	用户签到数

个性化场所推荐系统中的地点信息由 poi 模型和 cat 模型维护，其中 cat 表是用来扩展 poi 表中 category 列（见 2.5.1.2 数据解释）

表 2-6 poisManage 模块下模型 poi

Field	Type	Key	Description
pid	bigint(20)	PRI	地点 id,主键
name	varchar(100)	/	地点名称
category	varchar(20)	/	地点类别
lat	varchar(30)	/	地点纬度
lon	varchar(30)	/	地点经度
address	varchar(100)	/	地点地址
desc	varchar(500)	/	地点描述
checkinNum	int(11)	/	地点总签到数
checkinUserNum	int(11)	/	地点签到总用户数
todoNum	int(11)	/	想去该地点用户数
likeNum	int(11)	/	喜欢该地点用户数

cat 表中 id 一列是 Django Models 自动生成的模型主键，pid_id 为对应 poi 模型的外键。

表 2-7 poisManage 模块下模型 cat

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	自增主键
pid_id	bigint(20)	NO	MUL	NULL	地点 id
category	int(11)	NO	/	NULL	地点类别

个性化场所推荐系统的推荐模块下有数据库用户表、签到记录表、喜爱记录表和想去地点记录表。数据库用户表中存储有微博上爬下来的用户数据和该系统内新建的用户，因而注册用户时依据 id 字段在表 myuser 和表 data_user 中同步插入新数据。签到记录表中 time 字段为 django Model 字段 auto_now 属性，在创建时自动添加当前时间。

表 2-8 poisRecommend 模块下模型 data_user

Field	Type	Null	Key	Default	Extra
id	bigint(20)	NO	PRI	0	与 myuser 中的主键关联
username	varchar(50)	NO	/	NULL	用户名
city	int(11)	NO	/	NULL	用户所在城市
province	int(11)	NO	/	NULL	用户所在省份

表 2-9 poisRecommend 模块下模型 check

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	自增主键
time	datetime(6)	NO	/	NULL	签到的时间
pid_id	bigint(20)	NO	MUL	NULL	签到地点 id
uid_id	bigint(20)	NO	MUL	NULL	签到用户 id

表 2-10 poisRecommend 模块下模型 like

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	自增主键
pid_id	bigint(20)	NO	MUL	NULL	喜爱地点 id
uid_id	bigint(20)	NO	MUL	NULL	喜爱用户 id

表 2-11 poisRecommend 模块下模型 todo

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	自增主键
pid_id	bigint(20)	NO	MUL	NULL	想去的地点 id
uid_id	bigint(20)	NO	MUL	NULL	用户 id

表 2-12 poisRecommend 模块下模型 data_user

Field	Type	Null	Key	Default	Extra
id	bigint(20)	NO	PRI	0	与 myuser 中的主键关联
username	varchar(50)	NO	/	NULL	用户名
city	int(11)	NO	/	NULL	用户所在城市
province	int(11)	NO	/	NULL	用户所在省份

2.6 本章总结

本章主要对于基于微博签到数据的个性化场所推荐系统的目标和任务，需求分析，系统中采用的主要技术，重要算法以及具体的工程设计做了详细阐述。在项目目标部分，列出了本设计解决的主要问题以及功能上的要求。对于项目框架部分，阐述了项目的四大主要模块：专利数据集管理模块、核心专利评估模块和核心专利选取算法管理模块。对于关键技术和算法，同时阐述了他们本身的用途与在本系统中使用的理由以及与本系统结合的方式。最后从数据处理、数据库设计、接口设计和界面设计四个角度对该系统的具体设计进行了介绍。

第3章 服务器端的设计与实现

服务器端为整个个性化场所推荐系统的逻辑层和模型层，连接数据库，处理数据并对外提供 API 接口。

3.1 需求分析

需求分析分为功能需求和非功能需求两部分。功能需求指对用户对项目的具体的功能要求，而非功能需求包括该模块与其他模块的调用关系和数据转换等的要求以及内在的对该模块可修改性、可维护性等性能的需求。

3.1.1 功能需求

从用户的角度而言，服务器端具体应满足以下几点功能需求：

1. 提供用户注册、用户档案修改和存储功能
2. 提供用户登录登出和相应的权限审核功能
3. 提供个性化场所推荐功能
4. 提供用户历史记录查询，用户喜爱和想去的地点查询功能

3.1.2 非功能需求

为了满足 3.1.1 节中用户对项目的具体功能性需求，服务器端也有一些非功能需求：

1. 灵活可扩展的用户模型，不同权限的分类
2. 存储数据和返回客户端数据格式之间的转换
3. 满足一定规范，易于理解的 API 接口

3.2 系统设计

本项目的服务器端由 Django 架设而成，具体层次结构如下，APP 包括用户管理模块(userManage)、地理位置管理模块(poisManage)、场所推荐模块(poisRecommend)和工具类(utils)，GoHere 为主 project，media 文件夹存放服务器中需要存放的静态文件，如用户头像。

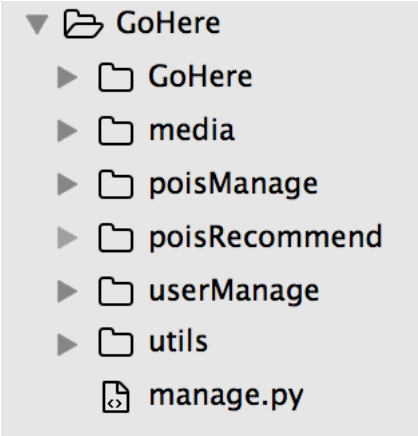

图 3-1 服务器端文件架构图

3.2.1 应用模型设计

3.2.1.1 userManage 应用

该应用主要负责管理用户信息，处理用户注册、登陆登出和对用户档案信息的操作。

MyUser 模型继承了 Django 内置认证系统中 AbstractUser 模型，在项目主文件夹中 GoHere / setting.py 里设置：

AUTH_USER_MODEL = "userManage.MyUser"

于是 MyUser 模型便成为系统默认的用户模型，获得了 Auth 认证系统的支持，用户管理系统的具体接口在 views.py 中定义，实施见 3.3 节。

3.2.1.2 poisManage 应用

该应用主要负责管理地点信息，储存并及时更新地点的一些动态属性。

虽然目前地点管理还只提供查询功能和在插入签到、喜爱或想去记录时同步更新对应属性的功能。考虑到项目的可扩展性和模块应用之间尽量的去耦合，于是将地点管理模块单独分了一个可插拔应用。

3.2.1.3 poisRecommend 应用

该应用主要负责存储并管理用户签到、喜爱和想去等操作信息，并且调用 utils 模块中相应的算法函数提供推荐。从数据的角度而言，该应用负责维护签到、喜爱和想去操作的用户与场所对应数据，同时存储矩阵分解后获得的用户偏好矩阵。

出于数据量的缘故，矩阵分解目前所需时间远超于用户期望等待时间，所以本

项目目前采取了由管理员定期依据签到数据更新矩阵分解的数据，从而每次用户推荐请求都只需要利用当前储存在服务器端的用户偏好矩阵和 HITS 算法，大大提高了推荐速度。

3.2.1.4 utils 应用

该应用中包括矩阵分解和 HITS 作推荐时的具体算法等算法函数，提供给其它需要的应用调用。

3.2.2 后端接口设计

后端接口设计使用 HTTP 并尽量遵循 REST 架构的 CRUD 原则，以资源为核心使用 HTTP GET、PUT、POST 等方法设计 API 接口。同时 Request 和 Response 的数据格式均为 Json 格式。具体接口设计如下：

表 3-1 登陆 API 接口

Address	**http://.../login**
HTTP 方法	[POST]
Request:	{“username”:”XXX”,”password”:”XXXXXXX”}
Response:	{“id”:”XXX”,”avatar”:”avatar/default.png”,”email:XXX@XXX.com”}
Error:	“用户名或密码不对！”

表 3-2 用户登出 API 接口

Address	**http://.../logout**
HTTP 方法	[POST]
Request:	null
Response:	“Bye~”

用户注册时只需要提供符合规范的用户名、邮箱和密码即可，如果用户名重复会返回“用户名已存在！”的报错。

表 3-3 用户注册 API 接口

Address	**http://…/register**
HTTP 方法	[POST]
Request:	{“username”:”XXXX”,”email”:”XXX@XXX.XXX”,”password”:”XXXXXXX”}
Response:	“SUCCESS”
Error:	“用户名已存在！”

用户注册时不提交用户档案信息，而是在用户档案 API 接口进行编辑。

表 3-4 用户档案 API 接口[GET]

Address	**http://…/profile/<pk>**
HTTP 方法	[GET]
Request:	null
Response:	{“email”:“daisylivinghere@gmail.com”,“city”:0,“province”:0,“desc”:null, “gender”:null,“date_of_birth”:null}
Error	404 Not Found

表 3-5 用户档案 API 接口[POST]

Address	**http://…/profile/<pk>**
HTTP 方法	[PUT]
Request:	{"email":"daisylivinghere@gmail\.com","city":0,"province":0,"desc":"only english","gender":"m","date_of_birth":"2016-05-23"}
Response:	“修改成功”
Error1:	{"email":["Enter a valid email address."]}
Error2:	{"gender":["\"django\" is not a valid choice."]}

表 3-6 用户推荐 API 接口

Address	**http://…/recommend/<pk>**
HTTP 方法	[POST]
Request:	{“lat”:””,”lon”:””,”range”:””}
Response:	[{“pid”:7,”pid”:176,”pid”:”25”}]
Error:	404 Not found

表 3-7 用户签到 API 接口

Address	**http://…/checkin/<pk>**
HTTP 方法	[GET] //列出该用户所有签到记录
Request:	null
Response:	[{"id":13,"time":"2016-05-27T10:40:40.960050Z","uid":38,"pid":9}, {"id":14,"time":"2016-05-27T10:43:00.511324Z","uid":38,"pid":9}, {"id":15,"time":"2016-05-27T10:44:35.388388Z","uid":38,"pid":9}, {"id":16,"time":"2016-05-27T10:50:48.107609Z","uid":38,"pid":9}]
Error:	404 Not Found
HTTP 方法	[PUT] //记录该用户的某次签到记录
Request:	{"pid":9}
Response:	{"checkin":14,"checkinUser":2}

用户喜爱操作与用户想去操作的 API 接口实现是相似的，只列其一。

表 3-8 用户喜爱操作 API 接口

Address	**http://…/like/<pk>**
HTTP 方法	[GET] //列出该用户所有喜爱地点
Request:	null

Address	**http://…/like/<pk>**
Response:	[{"id":3,"uid":38,"pid":7},{"id":4,"uid":38,"pid":8},{"id":5,"uid":38,"pid":10},{"id":6,"uid":38,"pid":9}]
Error	404 Not Found
HTTP 方法	[PUT] //建立或取消喜爱关系
Request:	{"pid":9}
Response:	{"user_like":1,"poi_like":1}

表 3-9 场所 API 接口

Address	**http://…/poi/<pk>**
HTTP 方法	[GET]
Request:	null
Response:	{"pid":9,"name":"dsadassa","category":"1","lat":"1","lon":"1","address":"","desc":"","checkinNum":0,"checkinUserNum":0,"likeNum":0,"todoNum":0}
Error	404 Not Found

3.3 应用开发

具体应用开发过程，以 userManage 为例讲解，其余就不再一一赘述。userManage 应用下文件组成如下图。包括了 admin.py, models.py, serializer.py, test.py, urls.py 和 views.py 等文件，下面将对各文件作详细阐述。

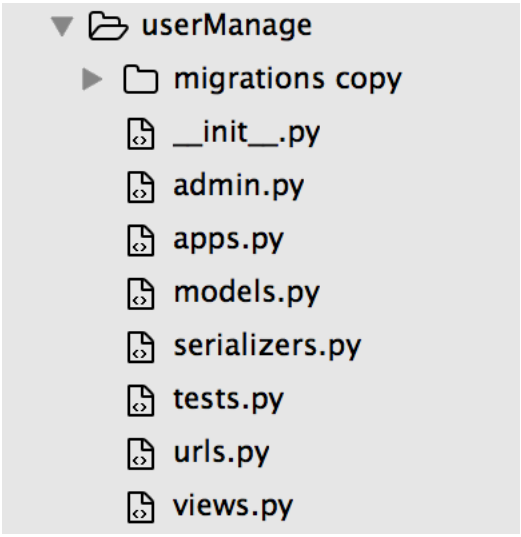

图 3-2 Django APP 文件架构图

3.3.1 建立模型

设计好应用模型后，先在 models.py 中创建对应和数据库连接的模型对象。本例中即是创建了继承 AbstractUser 的 MyUser 类，具体代码可见 2.3.1.2 节。在 GoHere / setting.py 中加入 MySQL 数据库支持，在 INSTALLED_APPS 中注册所有自定义应用。然后在项目目录下运行 python manage.py migrate 指令，可以自动在数据库中依据 models.py 中的模型依规则建立对应的数据表，当项目模块模型改动后，可以运行 python manage.py makemigrations UserManage 指令来更新单个应用的数据表。这些指令会在应用目录下建立 migrations 的文件夹，里面放着每次变化对应的数据库更改记录。

3.3.2 编写视图

在 class-based view 和 function-based view 之间，本人采取的是相对易于理解的 FBVs，每个函数对应着不同的视图。以一个验证用户是否授权的简单视图为例：

表 3-10 修饰符样例

```
@api_view(['GET','POST'])
@permission_classes((IsAuthenticated, ))
def example_view(request, format=None):
    content = {
        'status': 'request was permitted'
    }
return Response(content)
```

其中@api_view 修饰符后面的参数描述了该视图接受的 HTTP 方法，@permission_classes 修饰符后面的参数描述了该视图调用的权限。于是，该视图在授权用户使用正确 HTTP 方法的情况下会返回'status': 'request was permitted'。

serializers.py 为视图提供了可以直接关联 models 或者自定义的序列化函数。

依据 2.5.3 节中的后端接口设计表中的功能，借助于 simplejson 和 rest_framework 两个第三方库以及 django 内置函数，可以很方便地完成视图模块 views.py 的代码编写。

3.3.3 建立 URL 对应及其他

在 urls.py 中依据 2.5.3 节中的后端接口设计，将 URL 解析与视图函数产生一一对应。至此，一个可以本地测试的应用就基本完成了。

因为本项目中 presentation 层在安卓 app，所以并不需要 templates.py 来渲染 views.py 规格化后的数据。

而 tests.py 测试将在第五章测试中进行讲解。

3.4 本章小结

本章主要阐述了本人在该项目开发中在服务器端 Django 下的具体工作，包括应用的设计、建立，每个应用内模型的建立、视图的编写、URL 的对应和服务器部署。后端由 userManage、poisManage、poisRecommend 和 utils 四个应用模块组成，满足了功能和非功能需求需求，实现了整个系统的模型层和控制层逻辑。

第4章 安卓客户端设计与实现

安卓客户端是整个个性化场所推荐系统的展示层，它向用户展示界面，处理用户交互操作并调用服务器端的 API 接口提供服务。

4.1 需求分析

4.1.1 功能需求

从用户角度分析，安卓客户端具体应满足以下几点功能需求：

1.与后端功能所对应的展现界面

2.定位用户当前地点并返回参数的功能

3.调用百度地图 API，将推荐结果在地图上呈现的功能。

4.1.2 非功能需求

安卓客户端主要向服务器端请求信息并接收展示对应信息，遵循 2.5.3 节中的接口调用规范。同时因为暂时没有写本地缓存数据，所以界面跳转时，需要用 Bundle 在界面之间传递数据。

此外，由于数据在界面上的名字与呈现方式常与后台数据有出入，比如历史签到记录中的时间参数、个人档案信息中的省市信息。故而当展现是，需要实现界面数据和后端接口规范数据格式的转换。

4.2 界面设计

4.2.1 页面设计

在考量过用户需求和用户体验之后，本人接触学习了 Material Design 和 Sketch 界面设计，简单地定下了用户侧边栏和主界面滑动选项的界面模块设计。

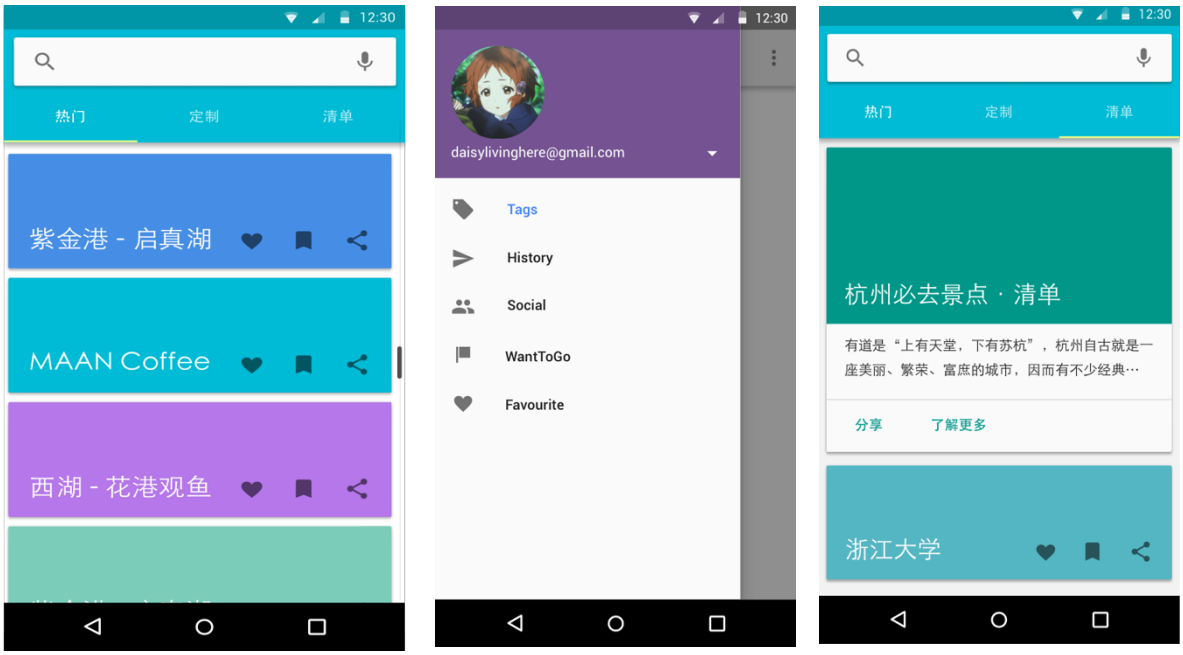

图 4-1 用户地点推荐条目和侧边栏设计页面

在具体实现界面时，遇到了一些困难，同时也做出了一些改变（比如加入了百度地图 API），最终界面见第五章的测试模块。

4.2.2 界面逻辑

界面跳转关系逻辑如下图所示，其中所有页面均可通过返回键返回上一页。

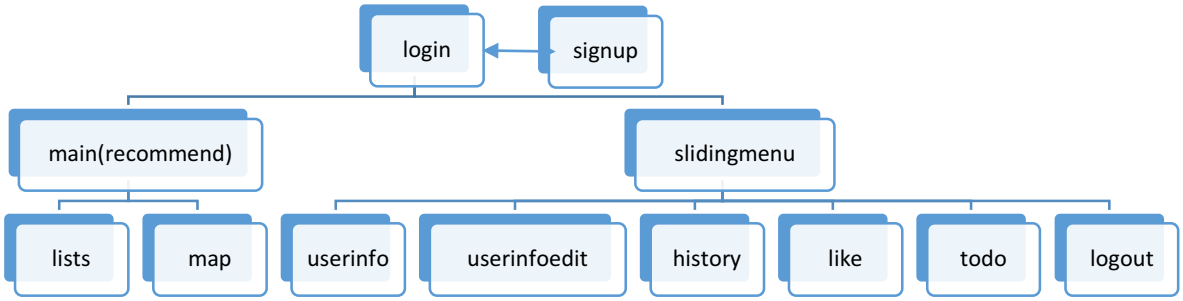

图 4-2 界面跳转关系图

4.3 界面开发

安卓端主要实现界面和值传递，本节将重点介绍界面开发中的一些技术难点，对于复杂界面的绘制，具体互动效果的呈现就不过多着墨了。

4.3.1 界面交互设计

本项目主要实现了以用户为主体的信息更新查询功能和以地点信息为另一主体的地点推荐功能。因而，界面设计左侧滑出导航栏为以用户为核心的用户 profile，主界面主要元素为地点信息。

具体实施上，在安卓 5.0 以上的版本中，Action Bar 被 Deprecate 了。于是本项目最终选用了 Material Design 提供的左侧滑出导航栏 Navigation Drawer 和主界面上方的水平滑动栏 TabLayout + ViewPager 来呈现。

4.3.2 Activity 之间值传递

采用 Bundle 的方法，在启动 Activity 时通过在 Intent 内绑定数据的方法进行 Activity 之间的值传递。

表 4-1 Intent 绑定值传递样例

```
    private void gogogo(int id,String email){
        Toast.makeText(getApplicationContext(), "Login Success!",
Toast.LENGTH_LONG).show();
        Intent intent = new Intent();
        intent.setAction("Main");
        intent.putExtra("id", id);
        intent.putExtra("username",usertext.getText().toString());
        intent.putExtra("email",email);
        startActivity(intent);
    }
```

4.3.3 Activity 和服务器端通讯

URLConnection 是个抽象类，它有两个直接子类分别是 HttpURLConnection 和 JarURLConnection。本项目中使用 HttpURLConnection 来完成与服务器端的通讯交流。 实际开发中需要注意的是，在 Android 4.0 后所有网络方面的请求都不能在主线程中执行，需要另开线程完成通讯。用 Thread + Handle 的方式来避免使用 Thread.join 函数所造成的阻塞

表 4-2 完整的服务器通讯样例

```
new Thread(new Runnable() {
          @Override
          public void run() {
              try{
                  URL url = new URL(baseurl+"login");
                  HttpURLConnection httpURLConnection =
(HttpURLConnection)url.openConnection();
                  httpURLConnection.setDoOutput(true);
                  httpURLConnection.setDoInput(true);
                  httpURLConnection.setRequestMethod("POST");
                  httpURLConnection.setRequestProperty("Connection", "Keep-
Alive");
                  JSONObject params = new JSONObject();
                  params.put("username",username);
                  params.put("password",password);
                  httpURLConnection.connect();
                  DataOutputStream dataOutputStream = new
DataOutputStream(httpURLConnection.getOutputStream());
                  dataOutputStream.writeBytes(params.toString());
                  dataOutputStream.flush();
                  dataOutputStream.close();
                  int resultcode = httpURLConnection.getResponseCode();
                  if (resultcode == HttpURLConnection.HTTP_OK) {
                      InputStream is = httpURLConnection.getInputStream();
                      BufferedReader br = new BufferedReader(new
InputStreamReader(is));
                      String line = null;
                      StringBuffer sb = new StringBuffer();
                      while ((line = br.readLine()) != null) {
                          sb.append(line);
                      }
                      Message message = new Message();
                      message.what = 0;
                      message.obj = sb.toString();
                      handler.sendMessage(message);
                  }else{
                      Message message = new Message();
                      message.what = -1;
                      handler.sendMessage(message);
                  }
              }
              catch (Exception e){
                  e.printStackTrace();
              }
          }
       }).start();
```

4.3.4 百度 API 调用

百度地图 API 所提供的接口功能十分强大，具体调用方法如下：

先在项目的 build.gradle 文件中添加包依赖，然后再在 Manifest.xml 中进行基本配置：

表 4-3 百度 API APP 权限添加

```
<uses-permission android:name="android.permission.BAIDU_LOCATION_SERVICE"></uses-permission>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"></uses-permission>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"></uses-permission>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"></uses-permission>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"></uses-permission>
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"></uses-permission>
<uses-permission android:name="android.permission.READ_PHONE_STATE"></uses-permission>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"></uses-permission>
<uses-permission android:name="android.permission.INTERNET"></uses-permission>
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"></uses-permission>
<uses-permission android:name="android.permission.READ_LOGS"></uses-permission>
```

表 4-4 百度 API key 配置

```
<meta-data
    android:name="com.baidu.lbsapi.API_KEY"
    android:value="zEPhhaeHMANc0MQ6KXMmdwqNm5MGAtBw" />
```

接下来就可以直接调用百度提供的地图控件与一些附加的功能了。

4.4 本章小结

本章主要叙述了安卓客户端的功能需求和实现方式。通过对各界面之间跳转关系和值传递关系的介绍，清楚展示了本项目中安卓应用的界面逻辑。界面开发一小节结合具体技术和项目工作，向读者展现了界面背后的具体实现。具体界面在第五章中会进行展示。

第5章 系统测试

本项目完成了服务器端和安卓客户端的代码编写，成功实现了个性化场所推荐算法，并在安卓端进行展示。本章将对服务器端和客户端分别进行功能测试，以验证其功能模块的正确性和完整性。

5.1 Django 本地测试

一方面，Django 本身为我们提供了单元测试，在每个模块 app 下都有对应的 test.py。另一方面，我们也可以直接借助 Django-Rest-framework 提供的 routers 界面，Django 提供的 Admin 管理界面和命令行下的 curl 命令进行本地测试。

5.1.1 Django 单元测试

在 3.3.3 节中，我们曾经提到在 Django 应用模块下的 test.py 文件。下面举例进行单元测试的介绍。

Django 内置单元测试使用 python 的 unittest 模块，这个模块使用基于类的方法来定义测试。类名为 django.test.TestCase，继承于 python 的 unittest.TestCase。运行测试指令时，Django 测试程序会在项目目录中所有以 test 开头的 Python 文件中查找所有 testcase 并建立测试集。它会为测试单独生成数据库，并在所有测试执行过后销毁，所以不用担心单元测试对项目数据库产生任何影响。

下图是一个简单的例子，分别是最基本的测试样例和访问网址，检测网页状态的功能。

表 5-1 Django 单元测试样例

```
from django.test import TestCase, Client

class SimpleTest(TestCase):
    def test_basic_addition(self):
        self.assertEqual(1 + 1, 2)

    def test_pois_404(self):
        response = self.client.get('/poi/3')
        self.assertEqual(response.status_code, 404)
    def test_pois_200(self):
        response = self.client.get('/POI')
        self.assertEqual(response.status_code, 200)
```

单元测试报错如下表所示，原因是未登录没有权限访问，所以状态码是 301 而非正确的 200。

表 5-2 Django 单元测试错误样例

```
taoyuandeMacBook-Pro:GoHere daisy$ python manage.py test
Creating test database for alias 'default'....F.
======================================================================
FAIL: test_pois_200 (poisRecommend.tests.SimpleTest)
----------------------------------------------------------------------
Traceback (most recent call last):
  File
"/Users/daisy/Desktop/TheseDays/GraduateDesign/Gohere_Django/GoHere/poisRe
commend/tests.py", line 16, in test_pois_200
    self.assertEqual(response.status_code, 200)
AssertionError: 301 != 200

----------------------------------------------------------------------
Ran 3 tests in 0.087s

FAILED (failures=1)
Destroying test database for alias 'default'...
```

下表列出了主要的单元测试类：

表 5-3 Django 单元测试类

TestCase	TestObject
test_register_201	SignUp Success
test_register_409	Duplicate Username
test_login_200	Login Success
test_login_403	Be Frozen
test_login_406	Wrong Username or Password
test_logout_200	Logout Success
test_userinfo_200	Update Success
test_userinfo_404	User Not Found
test_userinfo_400	Wrong Info Format
test_like_show_200	Show Like Info Success
test_like_action_200	Like Success
test_like_twice_200	Unlike Success
test_togo_show_200	Show Togo Info Success
test_togo_action_200	Togo Success
test_check_show_200	Show Check Info Success
test_check_action_200	Check Success

5.1.2 curl 指令测试

curl 指令格式:

curl –X [HTTP 方法] [地址] -d [传输的数据]

表 5-4 命令行 curl 指令测试样例

```
taoyuandeMacBook-Pro:~ daisy$ curl -X GET http://127.0.0.1:8000
{"users":"http://127.0.0.1:8000/users/","checks":"http://127.0.0.1:8000/checks/",
"POI":"http://127.0.0.1:8000/POI/","data_users":"http://127.0.0.1:8000/data_users/"}
taoyuandeMacBook-Pro:~ daisy$ curl -X PUT http://127.0.0.1:8000/todo/38 -d '{"pid":9}'
{"poi_todo":1,"user_todo":0}
taoyuandeMacBook-Pro:~ daisy$ curl -X GET http://127.0.0.1:8000/todo/38 -d '{"pid":9}'
taoyuandeMacBook-Pro:~ daisy$ curl -X POST http://127.0.0.1:8000/login -d
'{"username":"test3","password":"test2+1s"}'
{"id":6,"avatar":"avatar/default.png","email":"fd"}
taoyuandeMacBook-Pro:~ daisy$ curl -X GET http://127.0.0.1:8000/POI/
{"detail":"Authentication credentials were not provided."}
taoyuandeMacBook-Pro:~ daisy$ curl -X POST http://127.0.0.1:8000/logout
```

表 5-5 Django 客户端对应状态样例

```
taoyuandeMacBook-Pro:GoHere daisy$ python manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).
May 30, 2016 - 20:44:36
Django version 1.9.5, using settings 'GoHere.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
[27/May/2016 20:46:11] "GET / HTTP/1.1" 200 165
[28/May/2016 01:49:20] "PUT /todo/38 HTTP/1.1" 200 28
[28/May/2016 01:49:43] "GET /todo/38 HTTP/1.1" 200 2
[28/May/2016 01:50:40] "POST /login HTTP/1.1" 200 51
[28/May/2016 01:51:02] "GET /POI/ HTTP/1.1" 403 58
[28/May/2016 01:52:14] "POST /logout HTTP/1.1" 200 19
```

经测试，所有功能都成功地实现了。

5.2 安卓客户端测试

Android Studio 为本地测试提供了不同屏幕尺寸的虚拟机，可以本地连接本地服务器测试系统。同时，在成功部署到服务器上之后也可以用本人的安卓手机直接下载 apk 包进行调试。

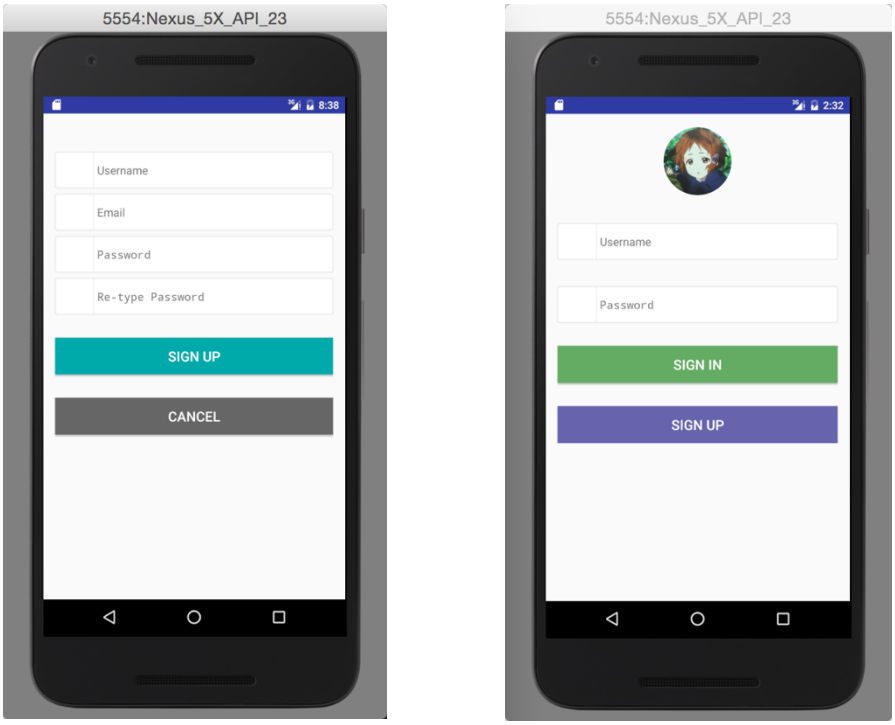

图 5-1 注册 / 登陆界面

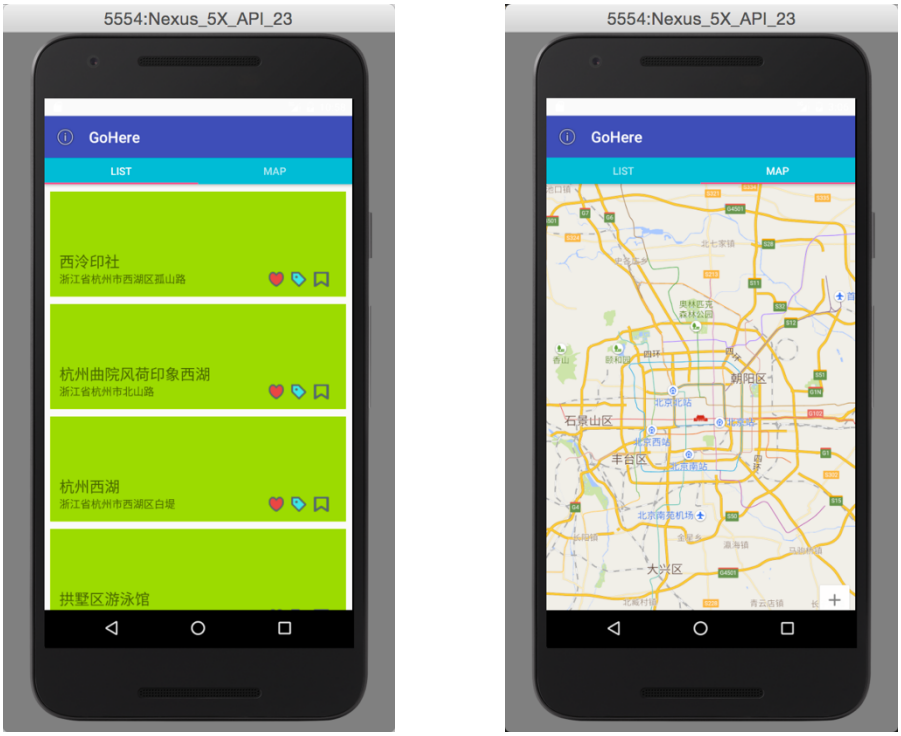

图 5-2 首页（条目/地图展现推荐结果）

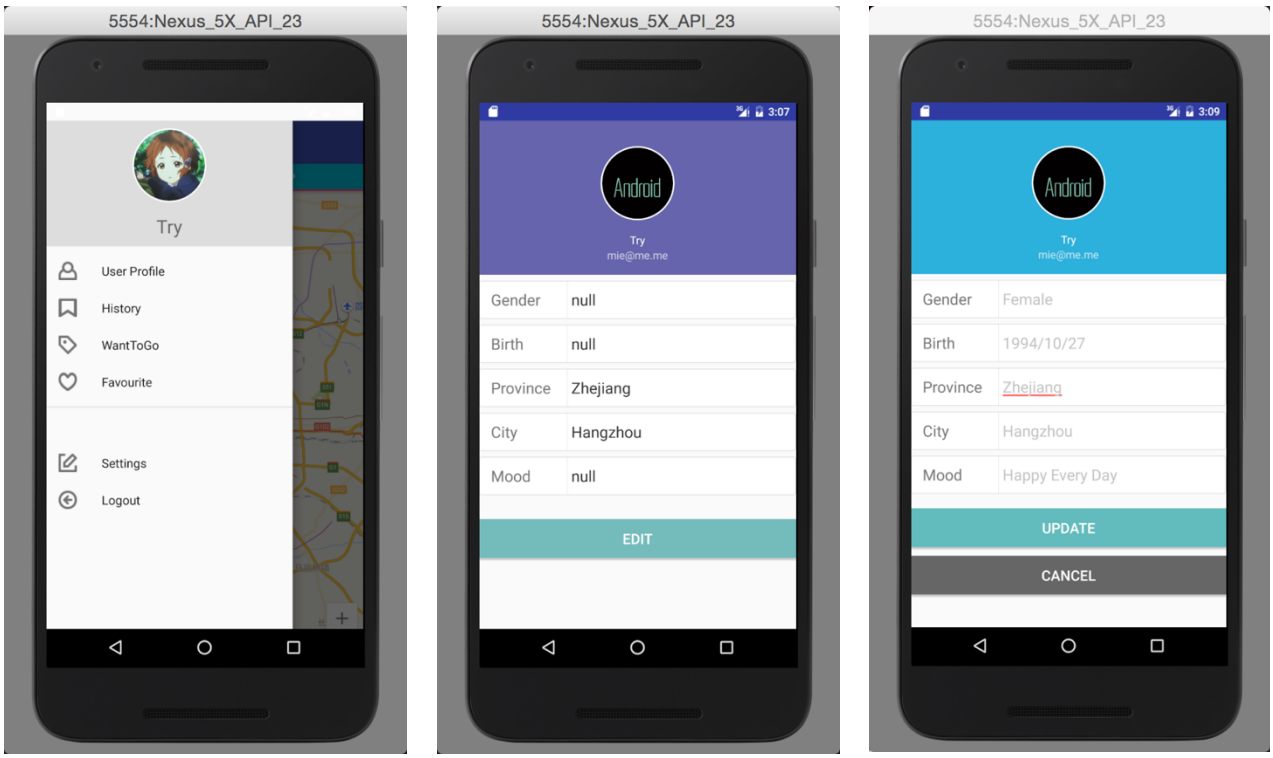

图 5-3 侧边栏（包括用户的档案属性）、用户档案页面

5.3 本章小结

在这一章中，我们对服务器端和安卓客户端进行测试。结果证明两个模块都能完成正确地完成既定的功能，和预期的行为一致，满足用户需求。且系统能够适时地对用户操作做相应的检查和约束，如对当前选择页面的权限验证等，以保证用户具有相应的权限。

第6章 总结与展望

6.1 工作成果总结

整个项目包括从后台到前端的所有功能，实现了个性化场所推荐算法并加以展示。开发过程中尝试使用了安卓开发与 Python 网络开发的一些较新的技术，最终成功地实现了基于微博签到数据的个性化场所系统。

后台 Django 项目完成了本系统用户管理系统和推荐算法的实现，按照模型实体和视图尝试对各应用模块进行解耦和，以 DRY 原则为设计要求。用户模型和场所模型的建立均为后续开发留下了可扩展的接口。

前端安卓项目完整地绘制了所有界面，实现了设计好的界面逻辑。TAB 界面和侧边栏的界面设计，使得安卓界面的扩展调整也非常方便。

综上所述，本项目基本完成了预期的项目功能，同时也确保了较好的可扩展性和解耦合。

6.2 未来工作

在基于微博签到数据的个性化场所推荐系统中，还可以继续完善的工作如下：

1.提供微博第三方接口登陆，取得用户授权后同步导入处理该用户微博上已有的签到信息。可以更好地扩展数据库的数据量，并且为从微博登陆的用户提供基于其过往记录的更加个性化的推荐，从根源上避免了新用户冷启动问题。

2.算法速度的改进。目前的矩阵分解由于数据量巨大，程序运行时间过长，因此只能周期性地更新矩阵分解后的用户偏好矩阵，从而难以做到根据用户近期更新的数据及时更新推荐结果。

3.算法本身的调整。目前本项目提供了用户喜爱和用户想去这两个用户与地点之间的关系操作，从而为用户和地点之间的关系提供了更多的信息，可以适当调参后加入到算法当中。该调整由于开发者经历有限未能加以尝试，可以考虑之后添加到算法中。

4.用户间建立社交。目前本项目只是用户和服务器之间的交互，如果加入了用户间社交的功能，可以获取用户和用户之间的关联信息，从而为个性化推荐作出更好的结果。

参考文献

[1] Bao, Jie, et al. Recommendations in location-based social networks: a survey. GeoInformatica : 525-565. 2015.

[2] Zhiyuan Cheng, James Caverlee, Kyumin Lee, and Daniel Z Sui. Exploring millions of footprints in location sharing services. ICWSM, 2011:81-88, 2011.

[3] Ye, Mao, et al. "Exploiting geographical influence for collaborative point-of-interest recommendation." Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval. ACM, 2011.

[4] Yehuda Koren. Factor in the Neighbors: Scalable and Accurate Collaborative Filtering, ACM Transactions on Knowledge Discovery from Data, Vol.4, No.1, Article 1, 1-24. 2010.

[5] http://python.usyiyi.cn/django_182/topics/db/models.html

[6] http://python.usyiyi.cn/django_182/topics/http/urls.html

[7] http://python.usyiyi.cn/django_182/topics/http/views.html

[8] http://python.usyiyi.cn/django_182/topics/auth/

[9] http://www.weiguda.com/blog/categories/12/

[10] http://www.django-rest-framework.org/

[11] Yehuda Koren, Robert Bell and Chris Volinsky. Matrix Factorization Techniques for Recommender, computer, Aug-09, 30-39. 2009.

[12] Cheng, Chen, et al. "Fused matrix factorization with geographical and social influence in location-based social networks." Twenty-Sixth AAAI Conference on Artificial Intelligence. 2012.

[13] Kleinberg, Jon M. "Authoritative sources in a hyperlinked environment." Journal of the ACM (JACM) : 604-632, 1999.

致谢

首先要感谢我的导师陈岭老师，谢谢您在项目选题上对我的耐心指点，在工程实践中对我的批评指正。在您的督促和指导之下，我得以顺利的完成这次毕业设计，并从中学会了很多知识和技能。

感谢实验室的应鸳恺师兄对我的毕设具体进展进行实时的指导，对我的报告和 demo 都进行了认真的指正。

我还要感谢我的父母，感谢他们一直尊重我的想法，以及多年来对我的激励与支持。

感谢 MSTC 社团，在这里我认识了很多真诚努力的学长学姐，感谢瑜佳学姐一直以来学业生活上的指点，感谢圈圈经常鞭策激励我，感谢萍学姐的关心帮助，感谢在办公室里努力完成毕设的日子。

感谢我的小伙伴周天和程吉锐，感谢你们一直以来给我带来的理性的思辨，感谢那些一起谈天说地，胡吃海喝的日子。

感谢室友们，谢谢你们两年来的陪伴帮助和对我缺点的包容，我很珍惜我们一起度过的时光。

最后，谨向所有拨冗审阅论文和参加答辩的评委老师，表示由衷的感谢。

本科生毕业设计 任务书

一、题目：基于微博签到数据的个性化场所推荐系统开发

二、指导教师对毕业设计的进度安排及任务要求：

进度安排：

3 月底前完成相关文献查阅及研发工具、方法学习；

4-5 月进行方法、原型系统的设计及开发；

5 月 15 日前完成相应测试及评价；

5 月 30 日前完成毕业设计初稿；

6 月 6 日前毕业设计定稿并完成答辩准备工作。

任务要求：

1 查阅毕业设计题目涉及领域内的代表性文献，掌握领域内的常用研究方法和开发技术；

2 能实现毕业设计相关方法、原型系统，实现其稳定运行，并进行必要的测试和评价；

3 毕业设计要结构清晰，重点突出，表达简练、准确。

起讫日期 2016 年 3 月 1 日 至 2016 年 6 月 6 日

指导教师（签名）_____职称_____

三、系或研究所审核意见：

负责人（签名）_____

年 月 日

毕 业 设 计　　考 核

一、指导教师对毕业论文（设计）的评语：

陶源同学的毕业设计“基于微博签到数据的个性化场所推荐系统开发”能对现有方法和技术进行分类介绍和分析，明确给出工作目标和内容，采用的方法和技术合理，结果翔实，并能实现原型系统，文章结构合理、表达流畅，符合毕业设计任务要求，是一篇优秀的本科毕业设计。

指导教师(签名)

年　月　日

二、答辩小组对毕业设计的答辩评语及总评成绩：

成绩比例	中期报告占（10%）	开题报告占（20%）	外文翻译占（10%）	毕业设计质量及答辩占（60%）	总评成绩
分值	9	17	9		

答辩小组负责人（签名）

年　月　日