任务一：案例数据标注

自动化用户轨迹标注与分析工具设计

葛钰峣*　　包启延　　刘涛　　赵士轲　　张子扬　　周镇涛

北方工业大学 20 级计算机实验班**码怪**项目组†

日期：2023 年 8 月 31 日

摘　　要

由于标准方法中存在诸多重复的机械性的工作，因此我团队考虑编写自动化脚本自动填写经纬信息并进行辅助分析和写入表格。

首先，工作者可以将用户 uid 输入程序，程序在数据库中搜索到 uid 对应的用户的路径文件。然后，程序将路径文件中的路径点、轨迹和时间标记在地图上，并在浏览器中自动弹出经处理后的地图。此时，工作者可以根据用户的轨迹和经计算得到的辅助标识填写程序程序输出的问题。最后，程序将结果分析汇总后，自动回填到表格。

本方法极大的提高了工作效率，适合大规模数据标识任务。由于使用了程序辅助计算和标识，可以提高数据的标注质量，为训练深度学习模型提供数据支持。

关键词： 数据标注, 瓦片地图, GIS

1　常规方法流程

1. 从文件地铁轨迹数据标注.*xlsx* 中找到未标记的用户，取其 uid
2. 到文件夹任务 *1-1000* 个案例地铁信令数据集中找到 uid 对应的文件
3. 将对应的文件的第一列 location 列复制
4. 打开打点网站https://gistool.mastercom.cn/location/#/map, 点击左上角“开始打点”按钮
5. 将 location 中的数据粘贴到“开始打点”文本框，打点类型选择折线，点确定
6. 观察地图上出现的折线
   (a) 是否与地铁线路重合，如果是则记录起始站及终点站
   (b) 是否有明显错误，如折线大部分贴合地铁线路但是有个别点显著离群
   (c) 是否中间缺点，如果缺点需要记录缺失的时段
   (d) 是否有明显的目的性，比如一看就知道去医院或者去学校
7. 将分析结果写回地铁轨迹数据标注.*xlsx*
   (a) 如果存在6a中 a 情况则在文件地铁轨迹数据标注.*xlsx* 用户对应的行的 is_metro_label 列标记为 1，表示该线路是用户乘坐地铁线路。
   (b) 如果存在6b情况则需要在备注说明列填写存在离群点

*码怪项目组组长, 北方工业大学本科在读, 中科院计算技术研究所客座学生, 百度飞桨深度学习框架开发组 (PFCC) 成员。研究方向: 机器学习理论, 人工智能安全-对抗样本生成技术, 大模型量化。邮箱: yuyao.ge.work@gmail.com

†码怪项目组 (Coding Monsters) 是来自北方工业大学的致力于以**高质量**, **高标准**完成课内代码实践任务的编码兴趣小组。小组由计算机实验班同学构成, 团队有**专利三项**, **SCI/EI 论文三篇**, **中文核心期刊一篇**, 曾以 98 分的成绩完成操作系统课内实践项目。

(c) 如果存在6c情况则需要在问题时段列中填写出现问题的时间段

(d) 如果存在6d情况，则在备注中填写“前往医院”/“前往学校”

2 本文方法: 自动化用户轨迹标注与分析工具

2.1 方法介绍

由于标准方法中存在诸多重复的机械性的工作，因此我团队考虑编写自动化脚本自动填写经纬信息并进行辅助分析和写入表格。

首先，工作者可以将用户 uid 输入程序，程序在数据库中搜索到 uid 对应的用户的路径文件。然后，程序将路径文件中的路径点、轨迹和时间标记在地图上，并在浏览器中自动弹出经处理后的地图。此时，工作者可以根据用户的轨迹和经计算得到的辅助标识填写程序程序输出的问题。最后，程序将结果分析汇总后，自动回填到表格。

本方法极大的提高了工作效率，适合大规模数据标识任务。由于使用了程序辅助计算和标识，可以提高数据的标注质量，为训练深度学习模型提供数据支持

2.2 本文方法流程

1. 首先使用 folium 库从高德地图读取瓦片地图。
2. 然后使用 pandas 库读取用户的 OD 数据并将路径点标注在地图上。
3. 计算用户时间上相邻的路径点间的平均物理距离 avg_{dist}。
4. 若用户两个用户时间上相邻的路径点间的物理距离大于 $10 \times avg_{dist}$ 则被判定为超远距离移动，有离群或信令丢失的可能。
5. 用户按照程序提示填写该用户的情况，程序自动填写表格。

2.3 本文方法优势

1. 不需要工作者在文件夹中手动搜索 uid 对应的文件
2. 不需要工作者复制经纬度并粘贴到网站
3. 可以将时间标注在地图上，如果存在信令丢失不需要工作者去翻看 uid 对应的文件，将地点与时间对应
4. 程序可以辅助**判定路径点是否靠近地铁站**和辅助判定是否为**离群路径**。

3 环境配置 & 使用说明

3.1 环境配置

1. 由于使用高德地图因此需要使用 VPN
   - 推荐使用 Clash: https://github.com/Fndroid/clash_for_windows_pkg/releases
   - 在代理一栏选择全局代理
   - 在常规一栏打开“系统代理”选项
2. 安装 pycharm 并配置 python 环境
   - *pandas == 2.0.3*: 用于数据处理和分析。
   - *webbrowser*: 用于在默认浏览器中打开地图文件。
   - *folium = 0.14.0*: 用于地图可视化。

- *datetime*: 用于处理时间数据。
- *os*: 用于系统操作

3.2 使用说明

1. 从文件任务 *1-地铁轨迹数据标注.csv* 中找到未标记的用户，取其 uid
2. 将 uid 输入程序
3. 查看地图，按照程序提示填写即可
4. 待全部完成后，将文件任务 *1-地铁轨迹数据标注 - 副本.csv* 中自己修改的部分替换在线文档中对应的部分

3.3 注意事项

1. 如何判断用户正在驾驶？
   用户的轨迹和公路重合，且间距较远
2. 如何判断用户正在步行？
   用户的路径点在商场聚集
3. 如何判断信令丢失？
   出现红色轨迹。此时，需要将红色轨迹的起始和终止时间点输入程序。
4. 程序运行的过程中一定要关闭文件任务 *1-地铁轨迹数据标注 - 副本*否则程序会报错
5. 如何判断离群点？
   用户瞬移，又瞬移回来。

4 效果展示

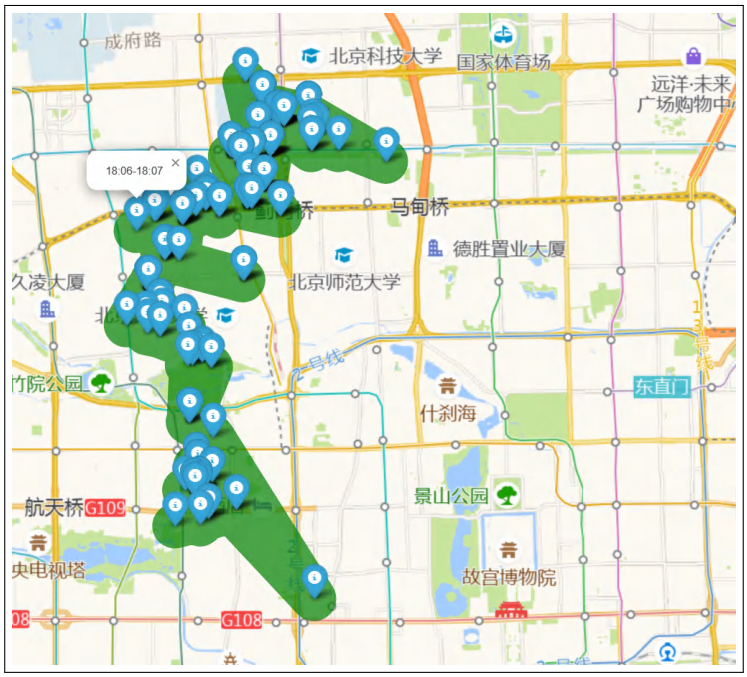

图 1: 以 UID 为 9898 的用户为例。当将 UID 输入本文程序之后，程序自动在数据库中搜索到该用户的 OD 数据并在地图上用标点和绿色线条自动标记用户经过的路径点和路径。

图 2: 以 UID 为 134270 的用户为例。当两个在时间上相邻的路径点在空间上距离显著大于常规距离的时候，本文方法会在地图上用红色标注该段路径。当出现被红色标注的路径时，就意味着可能存在离群点或信令丢失的情况。本实例为存在离群点而不是信令丢失，因为此时红色路径并不与其他路径的延长线上且从时间上考量存在瞬移。

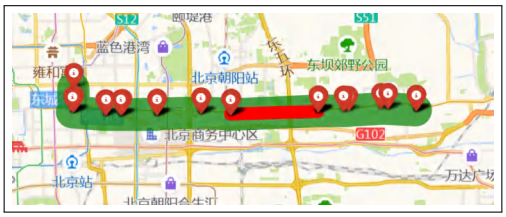

图 3: 以 UID 为 131244 的用户为例。当个别路径点显著靠近地铁站点时，路径点会显示为红色。当出现红色路径点时，就意味着用户此时有极大可能在乘坐地铁，工作者应复核路径与地铁线路是否重合，如果重合可以判定为乘坐地铁。

图 4: 以 UID 为 131244 的用户为例。本文方法不需要工作者在表格中手动输入地铁线路及备注，仅需要与程序交互即可。极大地减少了工作者的工作负荷和重复性，适合大量数据处理。

编号	UID	start_time	end_time	is_metro_algo	is_metro_label	地铁站点	备注说明
201	1897465	7:32:12	8:00:43	1	1	东大桥-平安里	
202	1901843	14:59:13	21:39:09	0	0		
203	1906653	8:22:32	8:53:32	1	1	望京-清华东路西口	
204	1919659	22:04:44	22:44:41	0	0		
205	1920103	17:02:40	17:44:54	0	0.5		
206	1930976	18:24:21	20:52:27	0	0		
207	1960324	6:16:39	6:42:31	1	0		路径中存在离群点,
208	1999388	6:55:28	8:21:55	0	0		路径中存在离群点,
209	2006891	13:48:49	22:47:41	0	0		
210	2038042	6:54:16	8:33:56	0	0		路径中存在离群点,
211	2039495	15:52:31	17:13:46	0	0		
212	2051877	7:32:51	8:28:27	0	0		
213	2055035	16:25:36	17:15:36	0	0		
214	2074756	8:10:38	9:00:40	1	0		
215	2076335	20:19:21	21:55:53	0	0		
216	2077307	12:08:39	12:54:53	1	0		存在长距离驾驶迹象,
217	2081829	11:14:55	12:09:46	0	0		
218	2084639	17:51:24	18:58:50	1	1	湾子-立水桥	
219	2087937	20:17:14	22:26:11	0	0		
220	2090062	17:19:27	20:32:12	0	0		

表 1: 部分工作成果展示

5 代码实践

5.1 核心代码分析

在测试了腾讯地图、谷歌地图、天地图、高德地图后，我们认为使用高德矢量地图最适合本次任务。此段代码通过调用 folium 库调用高德地图。

```
m = Map(location=[39.91, 116.40],
  zoom_start=12,
  tiles='http://wprd04.is.autonavi.com/appmaptile?lang=zh_cn&size=1&style=7&x={x}&y={y}&z={z
      }',
  attr='default')
```

此段代码将用户的路径点添加到地图上。其中路径点的标签设置为用户停留在该路径上的起止时间。当该路径点靠近地铁站时，路径点的颜色被设置为红色，否则默认为蓝色。

```
for i in range(len(data)):
  x, y = float(data.loc[i, 'location'].split(',')[1]), float(data.loc[i, 'location'].split
      (',')[0])
  Marker(
  location=[x, y],
  popup=Popup("{}-{}".format(datetime.strptime(data.loc[i, 'start_time'], "%H:%M:%S").strftime
      ("%H:%M")
  , datetime.strptime(data.loc[i, 'end_time'], "%H:%M:%S").strftime("%H:%M")),
  parse_html=True,
  max_width=100),
  icon=Icon(color='red' if data.loc[i, 'is_metro'] == 1 else 'blue')
```

```
  ).add_to(m)
```

本段代码用于在地图上用 α 为 0.7 的绿色线段标记用户的移动轨迹。

```
  tp = [[float(i[0]), float(i[1])] for i in trajectory_points]
  PolyLine(tp, color="green", opacity=0.7, weight=50).add_to(m)
```

当两个时间上相邻的路径点间的物理距离超过两点间平均距离的十倍时，会被本方法判定为距离过远从而用红色线段在地图上标注。

```
for i in trajectory_points:
  x, y = float(i[0]), float(i[1])
  if last_point == [0, 0]:
  last_point = [x, y]
  continue
  t_dist = ((x - last_point[0]) ** 2 + (y - last_point[1]) ** 2) ** 0.5
  if t_dist > 10 * avg_dist:
  filtered_df = data[data['location'] == i[1] + ',' + i[0]]

  PolyLine([last_point, i], color="red", opacity=1, weight=20).add_to(m)

  last_point = [x, y]
```

5.2 技术文档

本文档介绍了一个 Python 脚本即自动化用户轨迹标注与分析工具，用于处理地铁轨迹数据，并在地图上进行可视化展示以及标注用户出行方式、信令丢失、离群点等信息。

本方法可以自动化处理地铁轨迹数据，进行可视化展示和分析，同时可以辅助分析出行方式、信令丢失、离群点，最终可以将分析结果写回表格。本方法适合大规模标注，可以有效减少重复和机械性工作，为提高工作者效率、提高办公自动化水平、减少标注者标注错误、提高数据集质量做出了贡献。

以下是脚本的功能和使用方法的详细说明。

5.2.1 功能概述

该脚本用于读取用户的地铁轨迹数据，绘制相关路径点在地图上，并根据用户的输入进行标注。脚本的主要功能包括：

- 读取用户的地铁轨迹数据。
- 将路径点添加到地图上，根据是否为地铁轨迹进行不同的标记。
- 蓝色表示路径点距离地铁站较远
- 红色表示路径点距离地铁站较近或几乎重合
- 计算路径点之间的距离，并用 α 为 0.7 的绿色线段绘制连接线。
- 当两个在时间上相邻的路径点在空间上距离显著大于常规距离的时候，本文方法会在地图上用红色标注该段路径。当出现被红色标注的路径时，就意味着可能存在离群点或信令丢失的情况。
- 根据用户输入，判断出行方式、信令丢失、离群点等，并将结果填写到表格中。

5.2.2 使用方法

导入依赖库

安装以下第三方库：

- *pandas == 2.0.3*: 用于数据处理和分析。
- *webbrowser*: 用于在默认浏览器中打开地图文件。
- *folium = 0.14.0*: 用于地图可视化。
- *datetime*: 用于处理时间数据。
- *os*: 用于系统操作

主要步骤

以下是脚本的主要步骤：

- 用户输入要处理的用户 ID（uid），直到输入-1 结束程序。
- 从高德地图读取地图矢量数据，并将天安门设置为地图中心。
- 读取用户的地铁轨迹数据文件，并处理可能的路径错误。
- 将路径点添加到地图上，根据是否为地铁轨迹分别使用红色和蓝色标记。
- 计算路径点之间的平均距离，并绘制路径线。
- 判断路径中是否存在距离过远的路径点，标记为红色线段。
- 将地图保存为 HTML 文件，并在默认浏览器中打开。
- 工作者输入关于出行方式、信令丢失和离群点的信息，并更新数据文件。

用户输入

脚本会要求用户输入以下信息：

- 出行方式: 用户从多个选项中选择，包括驾驶、地铁、与地铁部分重合、步行、不确定。
- 信令丢失: 用户从是或否中选择。
- 离群点: 用户从是或否中选择。

数据标注

根据用户的输入，脚本会将相关信息标注在数据文件中，包括出行方式、地铁站点、信令丢失时段、离群点等。

5.2.3 注意事项

- 用户需确保已安装所需的依赖库，可使用以下命令安装：‘pip install pandas pip installfolium‘
- 脚本中使用的数据文件路径需要根据实际情况进行修改，确保文件存在且格式正确。
- 由于该脚本针对特定任务进行数据标注，使用前需要理解数据处理和标注的背景。

5.2.4 文件树

```
Traffic analysis
   main.py
   NCUT.html
   任务1-地铁轨迹数据标注 - 副本.csv
   任务1-地铁轨迹数据标注.csv

  .idea
      .gitignore
      encodings.xml
      misc.xml
      modules.xml
      Traffic analysis.iml
      workspace.xml

     inspectionProfiles
```

```
            profiles_settings.xml
            Project_Default.xml

    metro_signalings_bj20230625
    10008060.csv
    10053099.csv
    10103979.csv
    1014575.csv
    10233093.csv
    10281430.csv
    ........
```

6 总结

6.1 自动检测信令丢失模块设计失败

通过程序判断是否存在信令丢失这是一个非常自然的想法。计算用户单位时间内的位移记为 avg_dist。当单位时间内两个位置的间隔超过 avg_dist 的十倍，即被标记为信令丢失。但是在实际实验中出现了诸多的问题。

1. 用户长时间在一个位置
2. 用户从步行转换为乘坐交通工具
3. 交通工具的切换导致用户移动速度改变

上述情况的存在导致信令丢失检测只能使用人工方法。

为了方便工作者衡量是否存在可能的信令丢失，程序将长距离的移动标记为红色路径，阈值设置为 avg_dist 的十倍。当出现红色路径出现时，如果路径间不存在交通工具或者交通主干道则可认定为信令丢失。

6.2 实验感悟

在本次实验中，我们了解了 OD 数据类型和交通方面的知识。本次实验对我们团队而言收获很大，我们学习了数据挖掘领域的知识、对 python 语言有了进一步的掌握。我们意识到自动化办公的重要性。除了编程和知识之外，我们还意识到软件测试和文档的重要性。在将来的学习和工作中我们将运用本次实验学习到的知识。

7 参考文献

8 附录

8.1 工作文件

笔记及手稿: https://flowus.cn/share/161555d8-3ebb-4d64-b1fd-b3f1affb9a46
在线文档: https://docs.qq.com/sheet/DQVRZUnNCbU16U3RE?tab=g633cv

8.2 源代码

```
import pandas as pd
import webbrowser
from folium import Map, Marker, Popup, Icon, PolyLine
from datetime import datetime
from os import system

while (True):
    idx = int(input("请输入用户uid"))
    if idx == -1:
        break

    # 读取地图
    m = Map(location=[39.91, 116.40],
            zoom_start=12,
            tiles='http://wprd04.is.autonavi.com/appmaptile?lang=zh_cn&size=1&style=7&x={x}&y
                ={y}&z={z}',
            attr='default')
    csv_path = r"metro_signalings_bj20230625\{}.csv".format(idx)

    # 读取用户文件
    try:
        data = pd.read_csv(csv_path)  # 读取文件
        system('cls')
        print("========现处理uid为{}的用户的文件========".format(idx))
    except:
        print("路径错误,请检查")
        continue
    trajectory_points = []
    last_point = [0, 0]
    avg_dist = 0
    lack_points_times = []

    # 将用户路径点添加到地图上
    for i in range(len(data)):
        x, y = float(data.loc[i, 'location'].split(',')[1]), float(data.loc[i, 'location'].
            split(',')[0])
        Marker(
            location=[x, y],
            popup=Popup("{}-{}".format(datetime.strptime(data.loc[i, 'start_time'], "%H:%M:%S
                ").strftime("%H:%M")
                                       , datetime.strptime(data.loc[i, 'end_time'], "%H:%M:%S
                                           ").strftime("%H:%M")),
                        parse_html=True,
                        max_width=100),
            icon=Icon(color='red' if data.loc[i, 'is_metro'] == 1 else 'blue')
        ).add_to(m)
        trajectory_points.append([data.loc[i, 'location'].split(',')[1], data.loc[i, 'location
            '].split(',')[0]])

        # 计算每两个路径点间的距离并添加到avg_dist上
        if i == 0:
            last_point = [x, y]
        else:
            dist = ((x - last_point[0]) ** 2 + (y - last_point[1]) ** 2) ** 0.5
```

```
        avg_dist += dist
        last_point = [x, y]
tp = [[float(i[0]), float(i[1])] for i in trajectory_points]
PolyLine(tp, color="green", opacity=0.7, weight=50).add_to(m)

# 计算平均距离
avg_dist /= len(data)

last_point = [0, 0]

# 当两个时间上相邻的路径点间的物理距离超过两点间平均距离的十倍时，会被本方法判定为距离过远
    从而用红色线段在地图上标注
for i in trajectory_points:
    x, y = float(i[0]), float(i[1])
    if last_point == [0, 0]:
        last_point = [x, y]
        continue
    t_dist = ((x - last_point[0]) ** 2 + (y - last_point[1]) ** 2) ** 0.5
    if t_dist > 10 * avg_dist:
        filtered_df = data[data['location'] == i[1] + ',' + i[0]]

        PolyLine([last_point, i], color="red", opacity=1, weight=20).add_to(m)

    last_point = [x, y]

m.save('NCUT.html')
webbrowser.open('NCUT.html')

print("请猜测用户出行方式:")
print("1.驾驶")
print("2.地铁")
print("3.与地铁部分重合或其他")
print("4.步行")
print("5.不确定")
idx_0 = int(input())
if idx_0 == 2:
    print("请输入起始站:")
    m_st_0 = input()
    print("请输入终点站:")
    m_en_0 = input()

print("是否存在信令丢失?")
print("1.是")
print("2.否")
idx_1 = int(input())
if idx_1 == 1:
    print("请输入丢失起始时间:")
    t_st_0 = input()
    print("请输入丢失终止时间:")
    t_en_0 = input()

print("是否存在离群点?")
print("1.是")
print("2.否")
```

```
idx_2 = int(input())

xlsx_path = r"任务1-地铁轨迹数据标注 - 副本.csv"
xlsx = pd.read_csv(xlsx_path, encoding="gbk")

if idx_1 == 1:
    xlsx.loc[xlsx['UID'] == idx, '备注说明'] = "" if str(xlsx.loc[xlsx['UID'] == idx]['备
        注说明'].values[0])=='nan' else str(xlsx.loc[xlsx['UID'] == idx]['备注说明'].
        values[0]) + "存在信令丢失,"
    xlsx.loc[xlsx['UID'] == idx, '问题时段'] = "{}-{}".format(t_st_0, t_en_0)


# print("" if str(xlsx.loc[xlsx['UID'] == idx]['备注说明'].values[0])=='nan' else str(xlsx
    .loc[xlsx['UID'] == idx]['备注说明'].values[0]))

if idx_0 == 2:
    xlsx.loc[xlsx['UID'] == idx, 'is_metro_label'] = 1
    xlsx.loc[xlsx['UID'] == idx, '地铁站点'] = "{}-{}".format(m_st_0, m_en_0)
elif idx_0 == 1:
    xlsx.loc[xlsx['UID'] == idx, 'is_metro_label'] = 0
    xlsx.loc[xlsx['UID'] == idx, '备注说明'] = "" if str(xlsx.loc[xlsx['UID'] == idx]['备
        注说明'].values[0])=='nan' else str(xlsx.loc[xlsx['UID'] == idx]['备注说明'].
        values[0]) + "存在长距离驾驶迹象,"
elif idx_0 == 3:
    xlsx.loc[xlsx['UID'] == idx, 'is_metro_label'] = 0.5
    xlsx.loc[xlsx['UID'] == idx, '备注说明'] = "" if str(xlsx.loc[xlsx['UID'] == idx]['备
        注说明'].values[0])=='nan' else str(xlsx.loc[xlsx['UID'] == idx]['备注说明'].
        values[0]) + "路径与地铁线路部分重合,"
elif idx_0 == 4:
    xlsx.loc[xlsx['UID'] == idx, 'is_metro_label'] = 0
    xlsx.loc[xlsx['UID'] == idx, '备注说明'] = "" if str(xlsx.loc[xlsx['UID'] == idx]['备
        注说明'].values[0])=='nan' else str(xlsx.loc[xlsx['UID'] == idx]['备注说明'].
        values[0]) + "存在步行迹象,"
elif idx_0 == 5:
    xlsx.loc[xlsx['UID'] == idx, 'is_metro_label'] = 0

if idx_2 == 1:
    xlsx.loc[xlsx['UID'] == idx, '备注说明'] = "" if str(xlsx.loc[xlsx['UID'] == idx]['备
        注说明'].values[0])=='nan' else str(xlsx.loc[xlsx['UID'] == idx]['备注说明'].
        values[0]) + "路径中存在离群点,"

xlsx.to_csv("任务1-地铁轨迹数据标注 - 副本.csv", encoding="gbk", index=False)
print("更改已保存!")
```