## 第一步：

运行SortTrunk.py文件

统计文件信息：该文件中包含多少辆车及其对应的坐标点数。

运行程序后会生成两个json文件：" All20170901\_Trunk\_Sort "," Top20Sort "，

All20170901\_Trunk\_Sort文件：此文件统计出所有的车牌号及其对应的坐标数；如图：



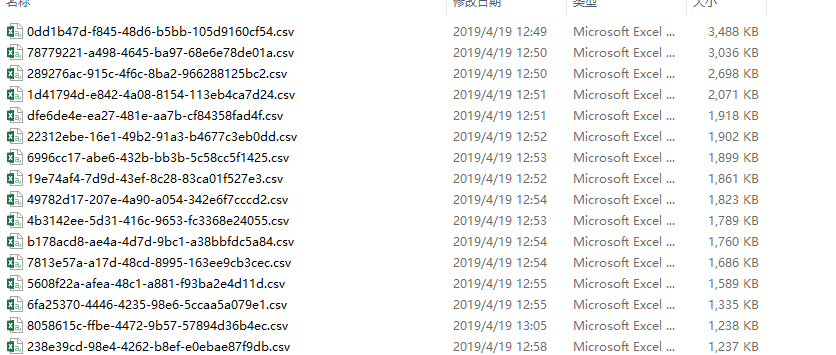
Top20Sort文件：此文件提取All20170901\_Trunk\_Sort文件中的前20%，并将其作为之后程序中的源数据。

## 第二步：

运行 CSV2Mongodb.py 文件

实现将csv文件转存到Mongo dB数据库，并提取前20%（默认）的车辆数据（每个车辆的坐标点数据均保存为单个csv文件）

如图：

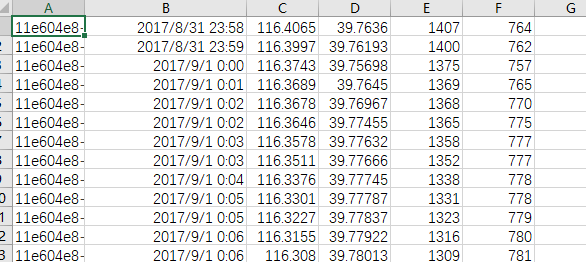


## 第三步：

运行Meshing.py文件

数据网格化，此部分只选取原csv文件中的前4列：车牌号；时间；经度；维度。网格化并未采取GeoHash算法网格化，程序运行后生成文件数可能会与第二步中提取的前20%的文件数量不一致，原因是网格化的过程剔除了明显的异常值（时间异常和经纬度异常，如果一个车辆的中的坐标均不在北京市内，则该车辆的坐标不会被网格化）

网格化后的文件如图：



## 第四步：

（1）提取待补点路段

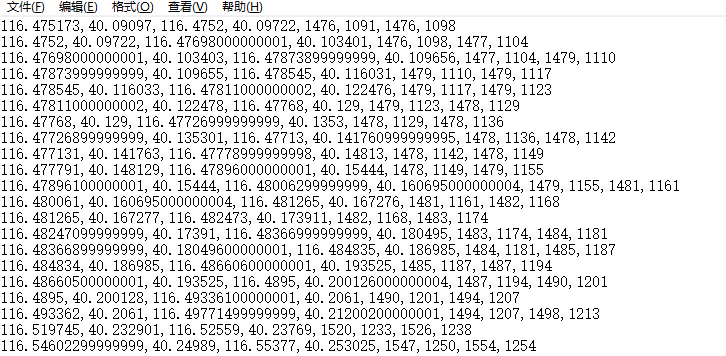
运行FillTrack.py文件（需要注意的的此部分程序包含了待补路段提取，相似区域提取，和迭代补点策略三部分）

演示示例：

FindAllRoute('H:\GPS\_Data\\20170901\Top20\Meshed','H:\GPS\_Data\\20170901\Top20\Trajectory')

第一个参数为网格化后的路径，第二个参数为文件保存路径

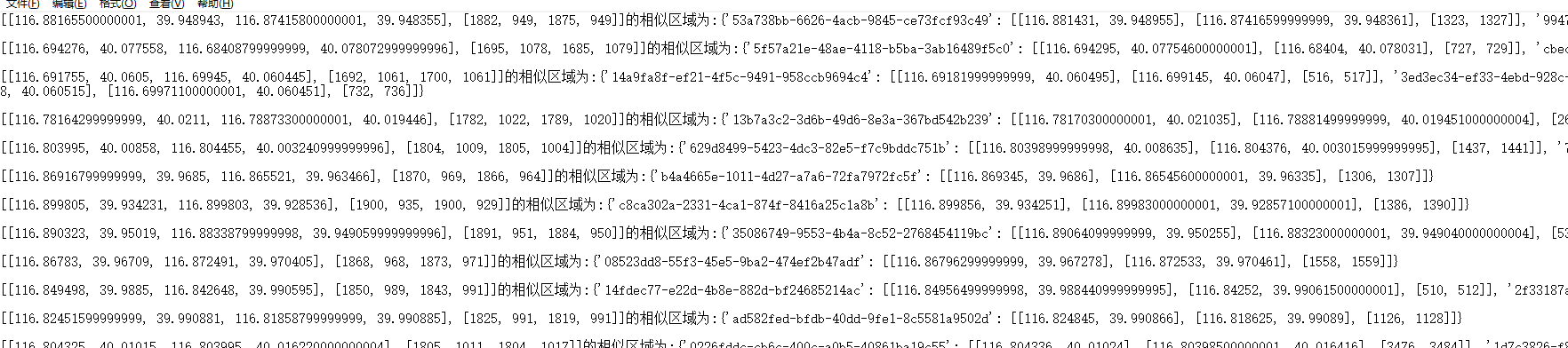
此部分生成的文件为txt文件，内容示例如图：



一个txt文件为一个车辆，每一行为一个待补路段，包含了待补路段的起终点坐标及其所属的网格编号。

（2）查找相似路段

此部分生成txt文件，每一个文件代表一个车辆，每一行代表一个路段的相似路段，如图所示（图片放大即可更加清楚的查看）：



相似路段为字典格式：{相似路段所在车牌号：[[起点经纬度],[终点经纬度],[起点终点所在的csv文件中的行数]] }

（3） 补点：

此部分会生成两个文件，补点文件（只含有补点坐标），完整补点文件（包含原始迹）。

补点文件只包含三列，经度坐标，维度坐标，坐标标记，0代表轨迹原始点，2代表补充点，1代表待补路段的起终点坐标。

## 第五步：补点策略二

通过统计相似路段的类别，选取包含轨迹最多的类别来补点，生成的文件同第四步中的（3）相同。

## 第六步：可视化

1、matplotlib可视化，运行Project.py文件即可

2、有路网的可视化，将csv文件转换为kml文件，即可通过GPXSee软件打开kml文件。

注：每一个文件下均有演示示例，程序中的文件名，保存路径均可修改。

以上步骤需依次完成，每一步运行需注意文件路径，只需要修改路径即可运行