OSpider v3.0.0 开发者手册

OSpider是GPL v3.0协议下的开源桌面软件及python库，致力于提供便捷的矢量地理数据获取和预处理体验。项目主页为：https://skytruine.github.io/OSpider/

OSpider v3.0.0桌面版的核心功能为**按行政区划名称、矩形框、圆形区和自定义面文件**四种方式**抓取POI**(暂仅支持百度POI,高德POI将再下一次更新中加入)，支持通过csv批处理文件批量执行POI抓取任务,且提供了分城市获取POI总量的实用工具。OSpider v3.0.0也集成了**WGS84/BD09/GCJ02坐标互转**工具与**地址解析**工具。

OSpider v3.0.0源码的功能模块包括POI抓取模块、坐标转换模块、行政区划获取模块、地址解析模块。

1 运行环境与依赖库

OSpider v3.0.0的开发环境为python3.7，依赖关系存于requirements.txt中。主要依赖geopandas (基于pandas, 进一步依赖pyproj,GDAL,Fiona和Shapely)和requests, 此外应用程序打包使用pyinstaller。由于GDAL等库的特殊性，直接采用`pip install -r requirements.txt`可能会报错，建议先行在Unofficial Windows Binaries for Python Extension Packages或其镜像站(速度快)下载pyproj,GDAL,Fiona和Shapely的whl文件进行手动安装。本处也提供了百度云的下载版本（适合python3.7版本）：https://pan.baidu.com/s/1OH2Pn1ohjn4Sm3OZ6xSJng 提取码：nn1s。下载完成后,以GDAL为例，安装对应whl的命令为`pip install YourPath\GDAL-3.0.4-cp37-cp37m-win_amd64.whl`。

2 功能模块与文件说明

2.1 功能模块

- **POISpider.py**

  POI抓取模块，内含不同数据源的POI爬虫类。以百度地图数据源POI爬虫类BaiduPOISpider为例。对外暴露8个功能函数

  1. set_key (设置key池)
  2. set_dispStatus (设置是否持续显示抓取进度)
  3. getPOI_byAD(按行政区划名称抓取POI)
  4. getPOI_byBounds(按矩形框抓取POI)
  5. getPOI_byCircle(按圆形区抓取POI)
  6. getPOI_byFile(按自定义面文件抓取POI)
  7. getPOI_byBatch(按批处理文件批量抓取POI)
  8. getPOI_CityNum(分城市获取POI总量)

在使用时应该先调用set_key设置key池，再执行6种功能函数。按批处理文件批量抓取POI无返回值，直接把抓取结果存入批处理文件声明的路径中；分城市获取POI总量执行成功返回pandas.DataFrame，失败返回None；其余四个抓取函数执行成功返回geopandas.GeoDataFrame, 失败返回None。此外在服务器端执行时建议设置set_dispStatus(False)关闭持续显示抓取进度功能从而提高运行速度。该类的抓取功能基于百度地图的Place API-按矩形抓取，在实现上比较值得注意的点在于Key池和线程关系的处理，POI抓取算法（初始格网阈值四分递归）。

- **CoordTrans.py**
  坐标转换模块，基于numpy，未封装成类，提供WGS84/GCJ02/BD09坐标互转的8个功能函数

1. gcj02_to_bd09(lng, lat)
2. bd09_to_gcj02(lng, lat)
3. wgs84_to_gcj02(lng, lat)
4. gcj02_to_wgs84(lng, lat)
5. bd09_to_wgs84(lng, lat)
6. wgs84_to_bd09(lng, lat)
7. coordtrans(lng,lat,origin_crs,target_crs)
8. coordtrans_byFile(inputFilePath,outFilePath,origin_crs,target_crs)

lng,lat分别为经纬度，可以是单独的数字，也可以是array-like的一维数组，如list。1-7返回numpy.narray或np.float64形式的结果: [result_lng, result_lat]。1-6是特定类型的坐标转换函数，coordtrans对1-6进行了封装，其中origin_crs，target_crs分别表示输入输出坐标系，可取'WGS84','GCJ02','BD09'。此外coordtrans_byFile则执行基于文件的坐标转换。

- **ADSpider.py**
  行政区划抓取模块，内含ChinaADSpider类，对外暴露出两个功能函数：

1. getADblur(模糊查询行政区划名称，返回指定长度的list)
2. getAD_byName(模糊查询行政区划的空间数据，返回geopandas.GeoDataFrame)
   该类模糊查询的实现基于python内置的difflib.get_close_matches，行政区划空间数据的获取基于高德的DATAV.Geoatlas

- **Geocoder.py**
  地址解析模块，写这个模块时作者偷懒了，单线程单Key且未考虑服务器部署的问题，只暴露了根据单key和待解析地址的CSV文件，进行地址解析并直接输出为指定目录下定名CSV解析结果文件的函数Geocoder。暂不建议调用。
- **OSpider_GUI.py**
  OSpider的用户界面文件，功能实现基于上述四个功能模块，而界面基于tkinter，各窗口封装成了类，包括一个继承tkinter.Tk的主窗口类和5个继承tkinter.Toplevel的子窗口类。具体为**OSpider_Main**(主窗口)、**OSpider_KeyPool**(Key池窗口)、**OSpider_CityNum**(分城市获取POI总量窗口)、**OSpider_POIBatch**(批量抓取POI窗口)、**OSpider_CoordTrans**(坐标转换窗口)、**OSpider_Geocoder**(地址解析窗口)。此外类**TextRedirector**用于将流输出重定向至widget，从而实现运行状态及报错信息在GUI界面中的即时显示。另一个值得注意的点是在每个窗口类中定义了**@staticmethod: thread_it**, 用于将耗时操作打包进独立线程从而防止用户界面卡死。

2.2 其余文件

- Demo

  文件输入样例，如对OSpider_GUI进行打包，请将该文件夹复制至打包结果的根目录中。

- icon.ico

  OSpider图标，运行OSpider_GUI.py所需资源文件，如对OSpider_GUI进行打包，请将该文件夹复制至打包结果的根目录中。

- property.ini

  OSpider配置文件，运行OSpider_GUI.py所需资源文件，如对OSpider_GUI进行打包，请将该文件夹复制至打包结果的根目录中。

- help.pdf

  样例用户手册，如对OSpider_GUI进行打包，请将该文件夹复制至打包结果的根目录中。

- requirements.txt

  OSpider依赖结构，通过`pip install -r requirements.txt`配置环境。

- LICENSE

GPL v3.0许可，不论在何种情况下使用OSpider或基于OSpider进一步开发，请保留该许可。

3 打包与调用样例

3.1 打包OSpider_GUI.py

由于调用了Geopandas库，故需要额外的操作才能用pyinstaller打包成功，具体操作如下：

1. 在shapely库文件目录下，找到geos_c.dll文件，将其复制一份重命名为geos.dll
2. 找到geopandas库文件下的`__init__.py`,将`import geopandas.datasets`这句注释掉
3. 在rtree库文件目录下，找到spatialindex_c.dll和spatialindex-64.dll并将其复制到待打包.py文件所在目录下
4. 在待打包.py文件头部加入如下的额外引用（OSpider_GUI中我已加入，故直接打包OSpider_GUI.py无需该步）

```
from pyproj import _datadir, datadir
from osgeo import ogr
from osgeo import gdal
from fiona import _shim, schema
```

完成上述操作后可通过如下命令进行打包。如果依然失败请参考博客。

```
pyinstaller --clean -y -F -w --add-binary=spatialindex_c.dll;. --add-
binary=spatialindex-64.dll;. -i icon.ico -n YourAppName OSpider_GUI.py
```

此外，不建议在Anoconda环境下进行打包（会一并打包进大量无用文件，造成速度慢，结果体积大），建议使用独立纯净环境进行打包（仅包含标准库和所需库）。

3.2 功能模块调用样例

3.2.1 按区域抓取POI

```
from POISpider import BaiduPOISpider

keylist=['YourBAIDUKeySample1','YouRbaiDuKEYsaMPlE2']
spider=BaiduPOISpider()
# key池设置一次就够了,thread_protect是用来限制并发的，表示一个Key最多可以被多少个线程同时占有,只要key不超并发，这个值就可以设置的比较大
# 想速度快，就大key池，多线程，同时在不触发并发限制的情况下调大并发保护数thread_protect
spider.set_key(keylist,thread_protect=3)

#如需在服务器部署，建议取消持续性状态输出用以加快速度,客户端调用建议保留状态输出
#spider.set_dispStatus(False)

#根据行政区划名称抓取POI-抓取广州市内的酒吧
gdf1 = spider.getPOI_byAD('酒吧', '酒吧', '广州', grid_num=4, threshold=100,
thread_num=6)

#根据矩形区域抓取POI-抓取西安市外接矩形内的高中(check)
gdf2 = spider.getPOI_byBounds('高中','中
学',118.351915,29.192178,120.724682,30.569969,
grid_num=4,threshold=100,thread_num=6)

#根据圆形区域抓取POI-抓取上海市政府周边5km范围内的咖啡厅
```

```
gdf3 = spider.getPOI_byCircle('咖啡厅','',121.480248,31.236276,5000,
grid_num=4,threshold=100,thread_num=6)

#根据自定义面文件抓取POI-抓取福州市内的KTV
gdf4 = spider.getPOI_byFile('KTV', '',
'https://geo.datav.aliyun.com/areas_v2/bound/350100.json',
grid_num=4,threshold=100,thread_num=6)

#将结果分别保存为CSV,TXT,Shapefile,GeoJSON
gdf1.to_csv('广州酒吧.csv', encoding='utf-8-sig')
gdf2.to_csv('西安高中.txt', encoding='utf-8-sig')
gdf3.to_file('上海市政府周边5km咖啡厅.shp', encoding='utf-8')
gdf4.to_file('福州KTV', driver='GeoJSON', encoding='utf-8')
```

抓取结果属性说明

列名	说明
uid	唯一标识符
name	POI具体名称
address	POI地址
province	POI所属省份
city	POI所属城市
area	POI所属区县
tag	POI标签（类型）
telephone	POI电话，可能为空
overall_rating	POI总体评分，-1表无，5最高
wgs84_lng,lat	WGS84经纬度
bd09_lng,lat	BD09经纬度
gcj02_lat	GCJ02经纬度
geometry	几何属性

3.2.2 批量抓取POI

```
from POISpider import BaiduPOISpider

keylist=['YourBAIDUKeySample1','YouRbaiDuKEYsaMPlE2']
spider=BaiduPOISpider()
spider.set_key(keylist,thread_protect=3)

#批量抓取POI
spider.getPOI_byBatch('Demo/批量抓取POI输入_Demo.csv')
```

批处理文件为列名确定的CSV文件，建议用户复制批处理Demo文件后进一步编辑，***批处理文件各列说明***如下：

列名	说明
id	任务唯一ID
query	检索关键字。支持多个关键字并集检索，不同关键字间以$符号分隔，最多支持10个关键字检索。如:"银行$酒店"。如果需要按POI分类进行检索，请将分类通过query参数进行设置，如"query=咖啡厅"，此时tag建议留空。
tag	检索分类偏好，可为空，如果需要严格按分类检索，请留空并设置query为分类名。典型的tag使用场景为特定类型下按名称检索，如抓取名称中包括"鑫源"的便利店（POI名称=鑫源，POI类型=便利店）
region	抓取区域，抓取区域，支持文件路径、行政区划名称、矩形定义、圆形定义，其中矩形和圆形定义的**分隔符为英文分号";"**，定义方式参见"按区域抓取POI"
grid_num	初始网格。抓取时首先将抓取区域的外接矩形划分为n*n个切片，n即为"初始网格"。**默认值"4"能满足绝大多数抓取需求。**
threshold	四分阈值。当切片返回的POI量大于四分阈值的时候，将对当前切片进一步四分。对于**百度地图而言设为"100"即可获得极高的POI查全率**，对于高德地图而言设为"850"即可获得极高的POI查全率。（批处理目前仅支持百度地图，故取默认值100即可）
thread_num	多线程抓取中启用的线程数，如果线程数>key池大小x并发保护数(在key池中设置)，实际执行时的线程数将被消减至"key池大小x并发保护数"。实际可用线程数越大抓取速度越快。
outFilePath	结果文件的保存路径，支持.shp/.csv/.txt/.json，无后缀则默认生成.shp

批处理结果文件保存在指定路径中（outFilePath），结果文件的属性与"按区域抓取POI"结果文件一致，本处不再赘述。

3.2.3 分城市获取POI总量

```
from POISpider import BaiduPOISpider

keylist=['YourBAIDUKeySample1','YouRbaiDuKEYsaMPlE2']
spider=BaiduPOISpider()
spider.set_key(keylist,thread_protect=3)

#测试分城市获取POI总量(check)
df=spider.getPOI_CityNum('高中','中学')

#输出结果至CSV文件
df.to_csv('分城POI总量_高中.csv', encoding='utf-8-sig')
```

抓取结果属性说明

列名	说明
provice	省份名
city	城市名
num	该城市特定POI总量

3.2.4 模糊查询行政区划名称

```
from ADSpider import ChinaADSpider

ads=ChinaADSpider()

#模糊查询行政区划名称
#参数分别为输入模糊查询，返回结果数量
r1=ads.getADblur('西安',5)
r2=ads.getADblur('洪山区',1)

#进一步应用
print(r1,r2)
```

[out]:

```
['西安市', '辽源市西安区', '西安市高陵区', '西安市雁塔区', '西安市阎良区'] ['武汉市洪山区']
```

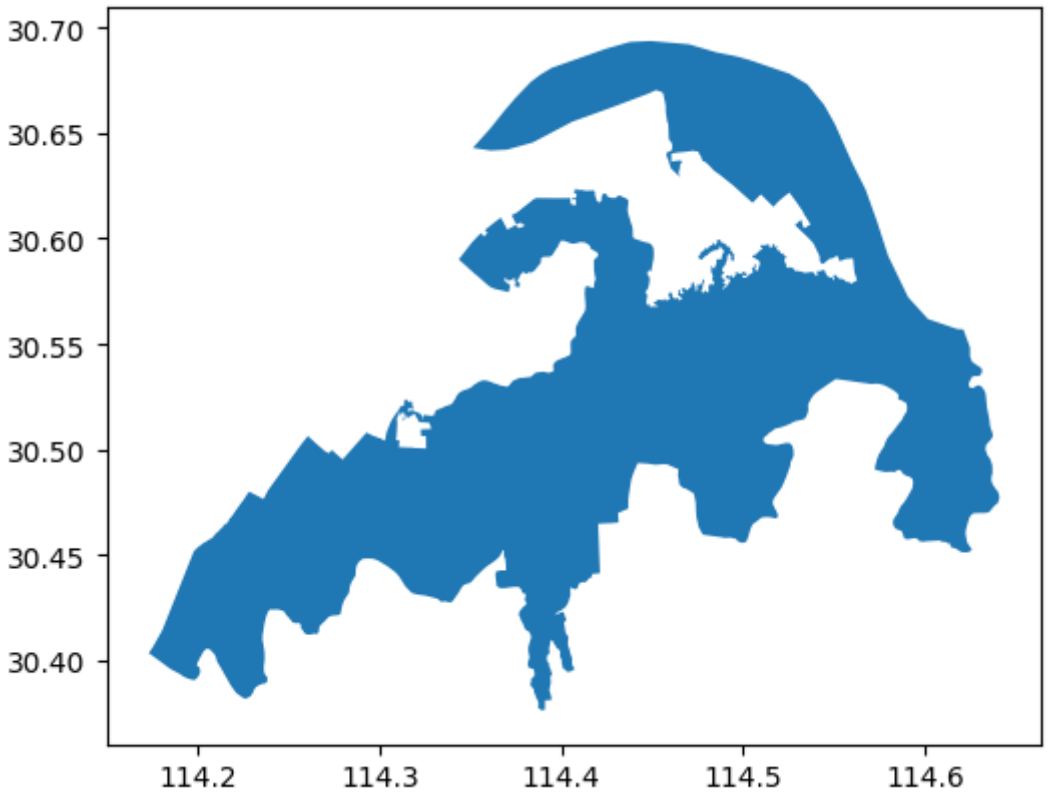
3.2.5 获取行政区划空间范围

```
from ADSpider import ChinaADSpider
ads=ChinaADSpider()

#模糊查询并获取行政区划空间范围
rgn=ads.getAD_byName('洪山区')

#进一步应用
rgn.plot()
print(rgn.head())
print(rgn.columns)
```

[out]:



```
adcode  ...                                           geometry
0  420111  ...  MULTIPOLYGON (((114.36329 30.60769, 114.36662 ...
[1 rows x 6 columns]
Index(['adcode', 'name', 'childrenNum', 'level', 'parent', 'geometry'],
dtype='object')
```

3.2.6 WGS84/BD09/GCJ02坐标互转

```
import CoordTrans as coord

#单点特定坐标转换（返回数值）
lng=121.234325
lat=30.7283713
rlng,rlat=coord.bd09_to_wgs84(lng,lat)
print(rlng,rlat)

#多点特定坐标转换（返回numpy.array）
lng = [122.23523, 120.2359]
lat = [31.2312432, 29.123142]
rlng,rlat=coord.wgs84_to_gcj02(lng,lat)
print(rlng,rlat)

#含参数坐标转换（对特定坐标转换的进一步封装，同样支持单点和多点，输入输出坐标可
取‘WGS84’,'GCJ02','BD09'）
lng = [122.23523, 120.2359]
lat = [31.2312432, 29.123142]
#BD09->GCJ02
rlng,rlat=coord.coordtrans(lng,lat,'BD09','GCJ02')
print(rlng,rlat)

#基于文件进行批量坐标转换(无返回值)
coord.coordtrans_byFile('Demo/坐标转换输入_Demo.csv', '坐标转换_Results.csv',
'WGS84', 'BD09')
```

[out]:

```
121.22344220207596 30.724624961384063
[122.23945144 120.2402979 ] [31.22923977 29.12035132]
[122.22862716 120.22948944] [31.22556744 29.11683913]
读取批量转换文件：Demo/坐标转换输入_Demo.csv
正执行从WGS84到BD09的转换...
转换中...
从WGS84到BD09的转换已完成
已将结果写入：坐标转换_Results.csv
```

注意：批量转换的输入csv文件，**包括id,lng,lat三个必须列**，其中lng指输入经度，lat指输入纬度。输入文件中允许包含其他用户自定义列，这些列将在输出文件中得到保留。

3.2.7 地址解析

```
from Geocoder import Geocoder

key='YourBAIDUKeySample'
inputFilePath='D:/地址解析输入Demo.csv'
outputDirPath='D:/'

#执行地址解析，输出文件自动保存为输出目录下的“输入文件名+地址解析结果.csv“
Geocoder(key,inputFilePath,outputDirPath)
```

待解析文件说明

列名	说明
id	地址唯一标识符
city	地址所在的城市名。用于指定上述地址所在的城市，当多个城市都有上述地址时，该参数起到过滤作用，但不限制坐标召回城市。**该参数可为空**，为空时不限制坐标召回城市。
address	待解析的地址。最多支持84个字节。 可以输入两种样式的值，分别是： 1、标准的结构化地址信息，如北京市海淀区上地十街十号 【推荐，地址结构越完整，解析精度越高】 2、支持“*路与*路交叉口”描述方式，如北一环路和阜阳路的交叉路口 第二种方式并不总是有返回结果，只有当地址库中存在该地址描述时才有返回。

地址解析结果文件列说明

列名	说明
id	唯一标识符
city	POI具体名称
address	POI地址
status	抓取状态码，0为成功抓取，非0为抓取失败。建议对所有非0结果重新抓取
wgs84_lng,lat	解析结果，地址对应的WGS84坐标
bd09_lng,lat	解析结果，地址对应的BD09坐标
precise	位置的附加信息，是否精确查找。1为精确查找，即准确打点；0为不精确，即模糊打点。
confidence	描述打点绝对精度（即坐标点的误差范围）。 confidence=100，解析误差绝对精度小于20m； confidence≥90，解析误差绝对精度小于50m； confidence≥80，解析误差绝对精度小于100m； confidence≥75，解析误差绝对精度小于200m； confidence≥70，解析误差绝对精度小于300m； confidence≥60，解析误差绝对精度小于500m； confidence≥50，解析误差绝对精度小于1000m； confidence≥40，解析误差绝对精度小于2000m； confidence≥30，解析误差绝对精度小于5000m； confidence≥25，解析误差绝对精度小于8000m； confidence≥20，解析误差绝对精度小于10000m；
comprehension	描述地址理解程度。分值范围0-100，分值越大，服务对地址理解程度越高**（建议以该字段作为解析结果判断标准）**； 当comprehension值为以下值时，对应的准确率如下： c=100，解析误差100m内概率为91%，误差500m内概率为96%； c≥90，解析误差100m内概率为89%，误差500m内概率为96%； c≥80，解析误差100m内概率为88%，误差500m内概率为95%； c≥70，解析误差100m内概率为84%，误差500m内概率为93%； c≥60，解析误差100m内概率为81%，误差500m内概率为91%； c≥50，解析误差100m内概率为79%，误差500m内概率为90%； //解析误差：地理编码服务解析地址得到的坐标位置，与地址对应的真实位置间的距离。
level	能精确理解的地址类型，包含：UNKNOWN、国家、省、城市、区县、乡镇、村庄、道路、地产小区、商务大厦、政府机构、交叉路口、商圈、生活服务、休闲娱乐、餐饮、宾馆、购物、金融、教育、医疗 、工业园区 、旅游景点 、汽车服务、火车站、长途汽车站、桥 、停车场/停车区、港口/码头、收费区/收费站、飞机场 、机场 、收费处/收费站 、加油站、绿地、门址

4 关于

4.1 主要开发人员

OSpider项目由小O发起和负责，当前开发团队包括：

- 华盛顿大学HGIS Lab | 小O

- 西南交通大学 | 苟强

4.2 加入我们

我们需要对POI、AOI、Land use、路网及其他GIS/规划相关数据获取及预处理有一定了解和实践经验，并希望为开源社区做贡献的开发者小伙伴-Talk is cheap. Show me the code. 如果你认为OSpider的文档编写、Web主页存在不足，且有能力进行改进，我们也非常欢迎你的加入。有加入意向的小伙伴请发送邮件至ospider_org@163.com

4.3 Bug报告，建议与代码贡献

如果你在使用OSpider的过程中发现Bug或对OSpider的有什么建议，请通过GitHub直接发起issue，或向ospider_org@163.com发送邮件。对于微小Bug的修正可以直接在GitHub中发pr，希望主导Feature的进一步开发，请提issue后向ospider_org@163.com发送邮件。
另外，非常欢迎加入OSpider用户群（QQ）：939504570。

4.4 支持我们

GitHub Star是对我们的最大肯定，而您的赞助支持将为项目的平稳发展保驾护航