第 7 讲　标注文件格式转换

PASCAL VOC 公开数据集的标注信息是用 XML 文件组织的，COCO 数据集是用 JSON 文件存储的。通过 LabelMe 标注的结果数据转换成 csv 或者 VOC、COCO 格式才能进行训练。

7.1 标注文件常用格式

7.1.1 JSON 与 XML

1.JSON 数据格式

使用键值对来表达信息组织形式，可用 Python 的 JSON 库来解析。

JSON 值可以是：

- 数字（整数或浮点数）
- 字符串（在双引号中）
- 逻辑值（true 或 false）
- 数组（在中括号中）
- 对象（在大括号中）
- null

（1）加载 JSON 文件

json.loads()：解析一个有效的 JSON 字符串并将其转换为 Python 字典

```
import json
f=open('1185.json',encoding='utf-8')
content=f.read()
text = json.loads(content)
print(text)
print(type(text))
```

```
{'version': '4.4.0', 'flags': {'__ignore__': False, '荷花': False, '梅花': False, '牡丹': False, '蔷薇': False, '樱花': True}, 'shapes': [], 'imagePath': '1185.jpg', 'imageData': None, 'imageHeight': 1062, 'imageWidth': 1590}
<class 'dict'>
```

json.load()：从一个文件读取 JSON 类型的数据，然后转换成 Python 字典

```
import json
f=open('1185.json',encoding='utf-8')
test1=json.load(f)
print(test1)
print(type(text))
```

```
{'version': '4.4.0', 'flags': {'__ignore__': False, '荷花': False, '梅花': False, '牡丹': False, '蔷薇': False, '樱花': True}, 'shapes': [], 'imagePath': '1185.jpg', 'imageData': None, 'imageHeight': 1062, 'imageWidth': 1590}
<class 'dict'>
```

（2）将字典转换成 JSON

json.dumps():用于将 dict 类型的数据转成 str，因为如果直接将 dict 类型的数据写入 json 文件中会发生报错，因此在将数据写入时需要用到该函数。

```
import json
data={'version': '4.4.0', 'flags': {'__ignore__': False, '荷花': False, '梅花': False, '牡丹': False, '蔷薇': False, '樱花': True}, 'shapes': [], 'imagePath': '1185.jpg', 'imageData': None, 'imageHeight': 1062, 'imageWidth': 1590}
json1=json.dumps(data)
print(json1)
print(type(json1))
f1=open("test1.json",'w')
f1.write(json1)
f1.close()
```

```
{"version": "4.4.0", "flags": {"__ignore__": false, "\u8377\u82b1": false, "\u6885\u82b1": false, "\u7261\u4e39": false, "\u8537\u8587": false, "\u6a31\u82b1": true}, "shapes": [], "imagePath": "1185.jpg", "imageData": null, "imageHeight": 1062, "imageWidth": 1590}
<class 'str'>
```

json.dump()用于将 str 类型的数据转成 dict。

```
with open('test2.json','w') as f:
    json.dump(data,f)
print(json)
```

2.XML 数据格式

XML 数据可用 beautifulsoup 库解析

```
from bs4 import BeautifulSoup
file=open('voc.xml','r')
soup=BeautifulSoup(file,'xml')
objs=soup.find_all('object')
#print(objs)
for obj in objs:
    objname=obj.name1.text
    xmin=int(obj.bndbox.xmin.string)
    ymin=int(obj.bndbox.ymin.string)
    xmax=int(obj.bndbox.xmax.string)
    ymax=int(obj.bndbox.ymax.string)
    print("{:^10}{:>5d}{:>5d}{:>5d}{:>5d}".format(objname,xmin,ymin,xmax,ymax))
```

```
   dog        48   240   195   371
 person        8    12   352   498
```

7.1.2 CSV 与 EXCEL 文件

csv 文件文本文件，字符分隔符

7.2 pandas 文件操作

pandas 是基于 numpy 的数据处理工具，擅长数据分析。

7.2.1 数据结构

1.Series 数据类型

一维数组，可以存储任意数据类型，类似于 numpy，但有索引。

不同于列表，numpy 数组和 pandas 的 Series 数据类型只允许存储相同的数据类型

```
import pandas as pd
obj=pd.Series([2,4,-5,8,3])
print(obj)
```

```
0    2
1    4
2   -5
3    8
4    3
dtype: int64
```

2.DataFrame 数据类型

二维数组，类似电子表格，由行名、列名和数据组成。

```
data={' 姓 名 ':['Peter','Tom','John','Smith'],' 英 语 成 绩 ':[85,98,56,89],' 数 学 成 绩 ':[67,79,88,78]}
frame=pd.DataFrame(data)
print(frame)
```

```
      姓名  英语成绩  数学成绩
0  Peter    85    67
1    Tom    98    79
2   John    56    88
3  Smith    89    78
```

7.2.2 Pandas 常用读取 csv 文件函数

1.CSV 文件读取 read_csv,read_table

```
read_csv(filepath_or_buffer,  sep='',  delimiter=None,  header='infer', names=None, index_col=None, usecols=None, **kwds)
```

参数：

filepath_or_buffer：字符串型，读取的文件对象路径，必填。

sep：字符串型，分隔符，选填，默认″,″。

delimiter：字符串型。定界符（备选分隔符），指定该参数，sep 失效。

header：指定第几行作为列名(忽略注解行)，如果没有指定列名，默认 header=0；如果指定了列名 header=None。

names：类数组，列名。默认为空。

dtype：每列数据类型。如：{'a': np.flat64, 'b': np.int32}。

skipinitialspace：忽略分隔符后的空白（默认 False，即不忽略）。

skiprows：类字典或整数，要跳过的行或行数，默认为空。

nrows：整数型，要读取的前记录总数，选填，默认为空，常用来在大型数据集下做初步探索之用。

thousands：字符串型，千位符符号，，默认为空。

decimal：字符串型，小数点符号，默认为(.)。

index_col：行索引的列表号或列名，如果给定一个序列则有多个行索引。

squeeze：布尔型，当为 True，如果数据仅有一列，返回 Series。默认 False，即只有一列也返回 DataFrame。

#读入 csv 文件

```
import pandas as pd
f1=open(r'./点名册.csv')
DBName=pd.read_csv(f1)
print(DBName.shape)
print(DBName.describe())
print(DBName)
```

2.CSV 文件写入 to_csv

DataFrame.to_csv(**path**=None, sep=',', **na_rep**='', float_format=None, **columns**=None, **header**=True, **index**=True, index_label=None, mode='w', encoding=None, compression='infer', quoting=None, quotechar='"', line_terminator=None, chunksize=None,

date_format=None, doublequote=True, escapechar=None, decimal='.', errors='strict')

参数：

pat：字符串或文件句柄，文件路径或对象，如果没有提供，结果将返回为字符串。

sep：默认字符‘,’，输出文件的字段分隔符。

na_rep：字符串，默认为 ‘’，缺失数据填充。

float_format：字符串，默认为 None，小数点保留几位。

columns：序列，数组，可选列写入，要写入的字段列表。

header：字符串或布尔列表，默认为 true，写出列名。如果给定字符串列表，则作为列名的别名。

index：布尔值，默认为 Ture，写入行名称（索引）。

index_label：字符串或序列，或 False,默认为 None。如果需要，可以使用索引列的列标签。如果没有给出，且标题和索引为 True，则使用索引名称。

```
from pandas import DataFrame as df
data=df({'long':[4.3,7,3],'width':[5,8,4],'height':[6,9,5]})
data.to_csv('zzz1.csv',sep=',')
```

```
from pandas import DataFrame as df
data=df({'long':[4.3,7,3,np.nan],'width':[5,8,4,1],'height':[6,9,5,2]})
```

```
data.to_csv('zzz2.csv', na_rep='null')

from pandas import DataFrame as df
data=df({'long':[4.3,7,3,np.nan],'width':[5,8,4,1],'height':[6,9,5,2]})
data.to_csv('zzz3.csv', index=False,header=False)

from pandas import DataFrame as df
data=df({'long':[4.3,7,3,np.nan],'width':[5,8,4,1],'height':[6,9,5,2]})
data.to_csv('zzz4.csv', column=['long','width'])
```

8.3 常用数据集格式

8.3.1 PASCAL VOC

PASCAL VOC 数据集下载地址：
链接：https://pan.baidu.com/s/1JI4BzE0KTxpiP9DNnXrcHQ
提取码：4zov
数据集分为 20 类，包括背景为 21 类，分别如下：

- 人：人
- 动物：鸟、猫、牛、狗、马、羊
- 车辆：飞机、自行车、船、巴士、汽车、摩托车、火车
- 室内：瓶、椅子、餐桌、盆栽植物、沙发、电视/监视器

```
.
└── VOCdevkit     #根目录
    └── VOC2012   #不同年份的数据集，这里只下载了2012的，还有2007等其它年份的
        ├── Annotations        #存放xml文件，与JPEGImages中的图片一一对应，解释图片的内容等等
        ├── ImageSets          #该目录下存放的都是txt文件，txt文件中每一行包含一个图片的名称，末尾会加上±1表示正负样
        │   ├── Action
        │   ├── Layout
        │   ├── Main
        │   └── Segmentation
        ├── JPEGImages         #存放源图片
        ├── SegmentationClass  #存放的是图片，语义分割相关
        └── SegmentationObject #存放的是图片，实例分割相关
```

每张图片对应一个 XML 格式标注文件：

```
<annotation>
    <folder>VOC2012</folder>    #表明图片来源
    <filename>2007_000027.jpg</filename> #图片名称
    <source>                              #图片来源相关信息
        <database>The VOC2007 Database</database>
        <annotation>PASCAL VOC2007</annotation>
        <image>flickr</image>
    </source>
    <size>        #图像尺寸
        <width>486</width>
```

```
        <height>500</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented> #是否用于分割
    <object>    #包含的物体
        <name>person</name> #物体类别
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>    #物体的 bbox
            <xmin>174</xmin>
            <ymin>101</ymin>
            <xmax>349</xmax>
            <ymax>351</ymax>
        </bndbox>
        <part> #物体的头
            <name>head</name>
            <bndbox>
                <xmin>169</xmin>
                <ymin>104</ymin>
                <xmax>209</xmax>
                <ymax>146</ymax>
            </bndbox>
        </part>
        <part>     #物体的手
            <name>hand</name>
            <bndbox>
                <xmin>278</xmin>
                <ymin>210</ymin>
                <xmax>297</xmax>
                <ymax>233</ymax>
            </bndbox>
        </part>
        <part>
            <name>foot</name>
            <bndbox>
                <xmin>273</xmin>
                <ymin>333</ymin>
                <xmax>297</xmax>
                <ymax>354</ymax>
            </bndbox>
        </part>
        <part>
            <name>foot</name>
```

```
                <bndbox>
                    <xmin>319</xmin>
                    <ymin>307</ymin>
                    <xmax>340</xmax>
                    <ymax>326</ymax>
                </bndbox>
            </part>
        </object>
</annotation>
```

8.3.2 COCO 数据集

1.80 个类别
person(人)

交通工具：bicycle(自行车) car(汽车) motorbike(摩托车) aeroplane(飞机) bus(公共汽车) train(火车) truck(卡车) boat(船)

公共设施：traffic light(信号灯) fire hydrant(消防栓) stop sign(停车标志) parking meter(停车计费器) bench(长凳)

动物：bird(鸟) cat(猫) dog(狗) horse(马) sheep(羊) cow(牛) elephant(大象) bear(熊) zebra(斑马) giraffe(长颈鹿)

生活用品：backpack(背包) umbrella(雨伞) handbag(手提包) tie(领带) suitcase(手提箱)

运动装备：frisbee(飞盘) skis(滑雪板双脚) snowboard(滑雪板) sports ball(运动球) kite(风筝) baseball bat(棒球棒) baseball glove(棒球手套) skateboard(滑板) surfboard(冲浪板) tennis racket(网球拍)

餐具：bottle(瓶子) wine glass(高脚杯) cup(茶杯) fork(叉子) knife(刀) spoon(勺子) bowl(碗)

水果：banana(香蕉) apple(苹果) sandwich(三明治) orange(橘子) broccoli(西兰花) carrot(胡萝卜) hot dog(热狗) pizza(披萨) donut(甜甜圈) cake(蛋糕)

家居：chair(椅子) sofa(沙发) pottedplant(盆栽植物) bed(床) diningtable(餐桌) toilet(厕所) tvmonitor(电视机)

电子产品：laptop(笔记本) mouse(鼠标) remote(遥控器) keyboard(键盘) cell phone(电话)
家用电器：microwave(微波炉) oven(烤箱) toaster(烤面包器) sink(水槽) refrigerator(冰箱)
家用产品：book(书) clock(闹钟) vase(花瓶) scissors(剪刀) teddy bear(泰迪熊) hair drier(吹风机) toothbrush(牙刷)
2.标注格式：json 文件
{
info{

"year": int, "version": str, "description": str, "contributor": str, "url": str, "date_created": datetime,
}

image{
"id": int, "width": int, "height": int, "file_name": str, "license": int, "flickr_url": str, "coco_url": str, "date_captured": datetime,
}

license{
"id": int, "name": str, "url": str,
}

8.4 项目任务

8.4.1 分类标注结果转换

LabelMe 进行分类标注后，完成标注数据 JSON 文件的保存。要求解析出 imagePath.flags 字段数据，使用 csv 文件保存所有图片文件名和对应的类别标签。

1.读入标注 json 文件

首先读取标注 json 文件，并解析出其中的 flags 字段数据，需要调用处理文件的库 os，解析 json 的库 json 和生成 csv 的库 pandas。

```
import json
import os
import pandas as pd
```

因为 labelme 生成的 json 文件都保存在图片相同的目录下，因此直接去读取这个目录。设置 ROOT_PATH 指向目录名，并使用 os.listdir 获取目录下所有文件名。

```
#设定读取 json 文件的目录
ROOT_PATH = './flowers'

#读取所有文件名
files = os.listdir(ROOT_PATH)
```

对于所有的文件，根据后缀判断是否是 json 文件

```
for file in files:
    #判断是否是 json 文件，
    if not file.endswith('.json'):
        continue
```

如果是 json 文件，则打开改文件，并使用 json 库来加载文件内容到 annotation 变量中。

```
    #获取文件全路径
    filepath = os.path.join(ROOT_PATH, file)
    print(filepath)
    #读取文件，并用 json 解析
    with open(filepath, 'r') as f:
        annotation = json.load(f)
```

2.解析 json 文件中的标注信息

因为是分类任务的标注结果，因此分类标签在 flags 对象中，同时 imagePath 中记录了对应的图像文件名。这些都是生成 csv 需要的标注信息。

```
#读取其中的 imagePath，获取图像路径
imgpath = annotation.get("imagePath")
```

然后再读取 flags 的值

```
#读取其中的 flags 作为分类标签
flags = annotation.get("flags")
```

读取到的 flags 是字典数据类型，其中 key 对应花朵的类型，value 对应 boolean 值标注图片是否是这种花朵类型。因为所有的花朵类型的顺序都和 flags.txt 中的一致。因此采用花朵类型的序列号作为类别的标签。下面把字典 flags 中的 boolean 值转换成数组类型。

```
#读取所有标签
values = list(flags.values())
```

获得的数组类似[False, False, True, False,False, False]。

遍历数组，将对应为 true 的序列号作为 label 保存。

```
label =0
  for i in range(len(values)):
     #如果为 true，则取当前序号作为标签
     if values[i]:
        label = i
  print("label:",label)
```

可以得到类似 label: 2

将解析出来的文件名称和标签组成一个数组。

```
#生成文件名和标签组成的数组
data = []
data.append(imgpath)
data.append(label)
```

得到类似['1185.jpg', 2]。

然后将每个 json 文件解析出来的数组汇总成一个二维数组。

```
#汇总到大数组中
alldata.append(data)
```

3.保存成 csv 文件格式

查看一下 alldata 数组的长度，应该是和标注的文件个数相同。

```
print(len(alldata))
```

为了方便通过 pandas 转成 csv 文件格式，先将数组类型的 alldata 转换成 pandas 的 dataframe 格式

```
#转换成 dataframe 格式
flowers_data = pd.DataFrame(alldata)
```

获得类似如下格式：

```
     0 1
0  1185.jpg 2
1  1195.jpg 2
```

2 1223.jpg 1
3 1346.jpg 4
…

然后 pandas 可以方便的把 dataframe 类型的数据直接转成 csv 文件进行保存。不需要保存列名称和序列号，因此可以使用下面代码。

```
#转换成 csv 文件，不需要保存列名和序号
flowers_data.to_csv("flowers.csv",header=False,index=False)
```

可以查看获得 flowers.csv 文件，类似如下

1185.jpg,2
1195.jpg,2
1223.jpg,1
1346.jpg,4
1480.jpg,5
1527.jpg,5