图的构建以及对节点属性的统计分析

任务描述

图是非常重要的一种数据结构。本次作业提供了 newmovies 数据集，希望基于该数据，在程序中读取并存储用户节点信息，建立无向图结构，并进一步实现相关统计和可视化功能。

任务目标

以若干指定的 Python 模块实现:

· 初始化加载图的数据，做好切分

· 对图的度进行统计计算

· 对图结构实现序列化存储/反序列化

· 对图的度的分布/属性的分布绘图

具体步骤

一、创建文件夹以及__init__.py，形成包结构

文件夹架构如下:

```
GraphStat/
    .dist
    __pycache__
    __init__.py
    Demo.py
    NetworkBuilder/
        __pycache__
        __init__.py
        graph.py
```

```
        node.py
        stat.py
    Visualization/
        __pycache__
        __init__.py
        plotgraph.py
        plotnodes.py
```

二、初始化图，预加载数据并实现输出功能

在 node.py 中，通过 init_node()函数实现读取数据并初始化，通过 print_node()函数实现输出，get_item()函数实现单个图节点的属性查询。

具体操作如下:

```
def init_node(node_file_location=r"D:/Files/计算机程序设计/现代程序设计技术/作业数据/newmovies.txt"):
    """
    输入节点文档所在位置，返回包含节点属性字典的列表，key 为节点的属性，值为对应的属性值
    """
    node=['id','name','weight','type','others']
    node_list=[]#存储节点信息字典的列表
    near_dict={}#存储节点邻接表的字典
    with open(node_file_location,'r',encoding='UTF-8') as f:
        lines=f.readlines()
        t_t=0#用于记录*出现的次数，读取到第二个*立刻断开，也能避开对第一个*后内容的读取
        for line in lines:
            if line[0]=='*':
                t_t+=1
                if t_t==1:#根据节点数
                    num=int(line.split(' ')[1].strip())
                    for i in range(num+1):
                        near_dict[i]=[]
                continue
            if t_t==1:
                temp=line.split('\t')
                s_dict={}
                for i in range(5):
                    s_dict[node[i]]=temp[i].strip()
```

```
                node_list.append(s_dict)
            if t_t==2:
                temp=line.split('\t')
                near_dict[int(temp[0])].append(temp[1])

    return node_list,near_dict

def _get_attr(node,key):
    """
    获取节点的属性，其中 node 为字典形式的节点信息
    """
    return node[key]

def get_item(node_list,key,num=0):
    """
    获取对应的节点属性，其中 num 是节点的序号
    """
    node=node_list[num]
    return _get_attr(node,key)

def print_node(node_list,num=0):
    """
    利用 format 函数，将节点属性输出至屏幕上，其中 num 是节点的序号
    """
    print("Node\nid:{},name:{},weight:{},type:{},others:{}".format(*[no
de_list[num][i] for i in node_list[num]]))
```

三、计算图的度并输出

见 stat.py，通过 cal_average_dgree()函数实现计算网的平均度，通过 cal_dgree_distribution()函数返回计算得到各个节点度的分布。

具体操作如下:

```
def cal_average_dgree(near_dict):
    """
    计算网络中的平均度
    """
    sum_dgree=0
    for i in near_dict:
        sum_dgree+=len(near_dict[i])

    return sum_dgree/len(near_dict)
```

```
def _get_attr_distribution(near_dict,num=0):
    """
    获取某个节点的度分布
    """
    return len(near_dict[num])

def cal_dgree_distribution(near_dict):
    """
    计算网络的度分布，返回一个列表，依节点次排布度
    """
    dgree_distribution_list=[_get_attr_distribution(near_dict,i) for i
in range(len(near_dict))]
    return dgree_distribution_list
```

四、序列化存取图的数据[ii]

见 graph.py，init_graph()函数负责合并已有数据为字典，save_graph()函数和 load_graph()函数分别负责序列化存储/反序列化读取图的数据。

具体操作如下:

```
import pickle as pe

def init_graph(node_list,near_dict):
    """
    返回一个字典，分别存储节点信息和边信息
    """
    graph_dict={}
    for i in range(len(node_list)):
        graph_dict[i]=(node_list[i],near_dict[i])
    return graph_dict

def save_graph(graph_dict,save_location=r"D:/Files/计算机程序设计/现代程
序设计技术/作业/第四周作业/graph_dict.txt"):
    """
    序列化图信息
    """
    with open(save_location,'wb') as w:
        pe.dump(graph_dict,w)
```

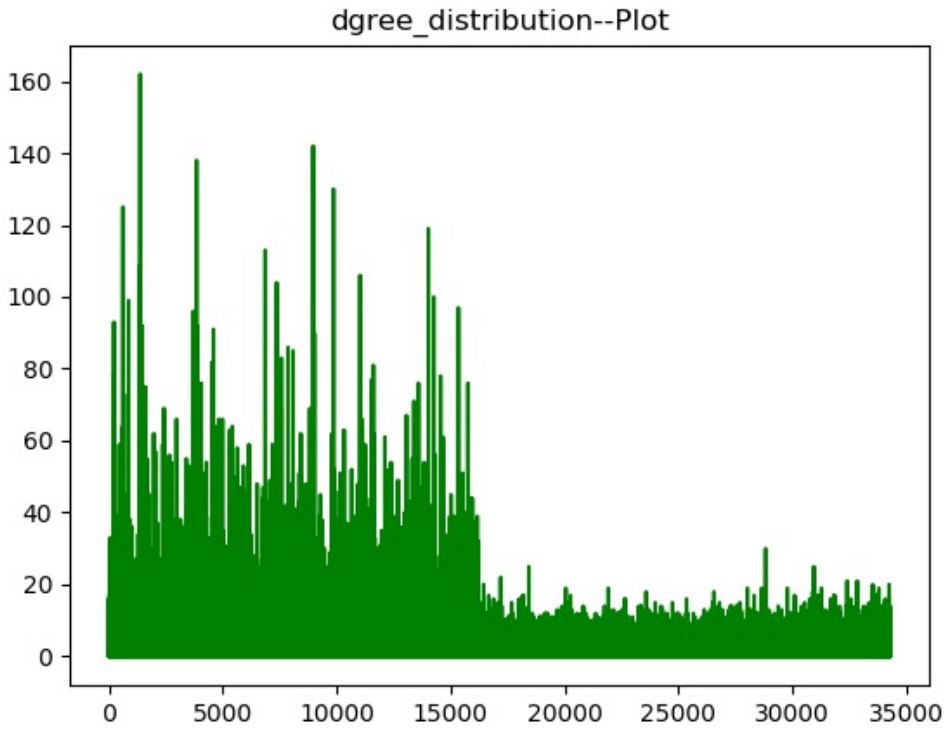
```
def load_graph(load_location=r"D:/Files/计算机程序设计/现代程序设计技术/作业/第四周作业/graph_dict.txt"):
    """
    反序列化图信息
    """
    with open(load_location,'rb') as r:
        graph_dict=pe.load(r)

    return graph_dict
```

五、绘制度的分布图

见 plotnodes.py。以函数 plotdgree_distribution()绘制了度随序号分布的线性图。具体操作如下:

```
import matplotlib.pyplot as plt

def plotdgree_distribution(dgree_distribution_list):
    """
    度的分布图
    """
    plt.plot([i for i in range(len(dgree_distribution_list))],dgree_distribution_list,color='green')
    plt.title('dgree_distribution--Plot')
    plt.savefig('D:/Files/计算机程序设计/现代程序设计技术/作业/第四周作业/dgree_distribution--Plot.jpg')
    plt.show()
```

图片如下:



六、依据节点属性绘制统计图

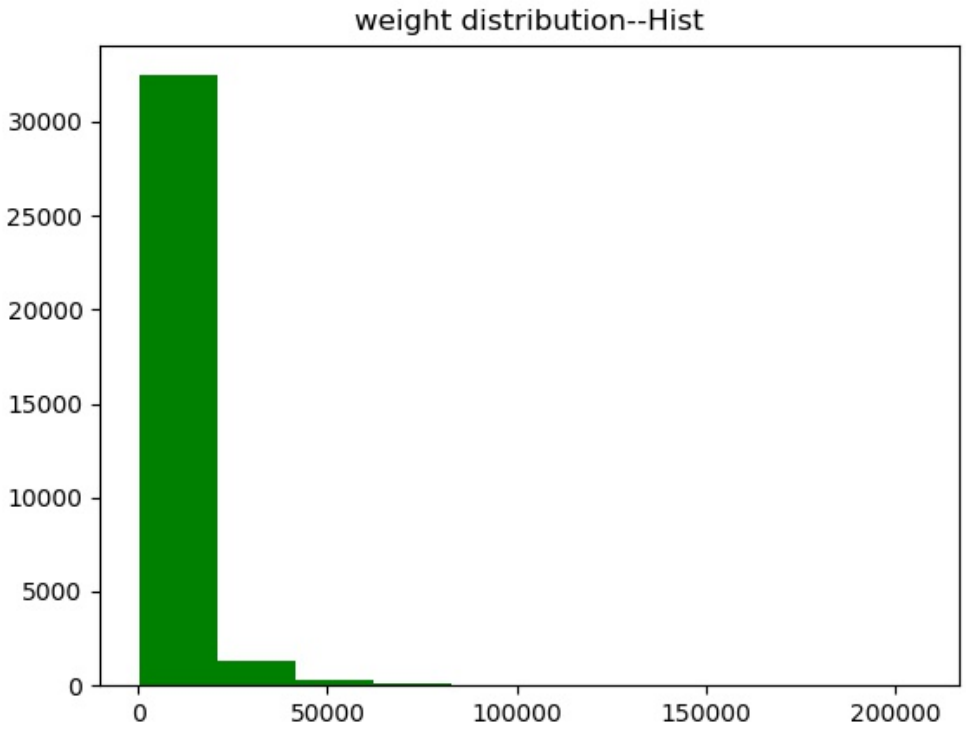
见 plotgraph.py，在函数 plot_nodes_attr 中，针对'weight'和'type'两个属性分别绘制了频率分布直方图、饼状图，具体操作如下：

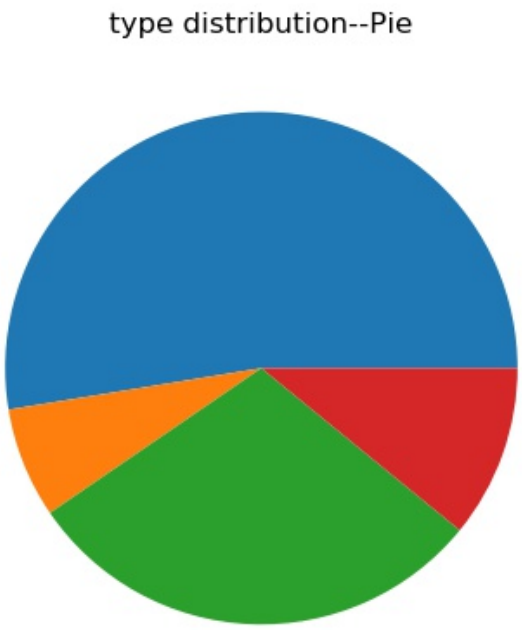
```
import matplotlib.pyplot as plt

def plot_nodes_attr(graph_dict,key):
    """
    绘制图中节点属性的统计结果，其他和名称不便于统计，此处仅统计权重和类型
    """
    if key=='weight':

        plt.hist([int(graph_dict[i][0]['weight']) for i in range(len(gr
aph_dict))],color='green')
        plt.title('weight distribution--Hist')
        plt.savefig('D:/Files/计算机程序设计/现代程序设计技术/作业/第四周作
业/weight distribution--Hist.jpg')
        plt.show()

        """
```

```
        plt.plot([graph_dict[i][0]['weight'] for i in range(len(graph_d
ict))],[i for i in range(len(graph_dict))],color='green')
        plt.title('weight distribution--Bar')
        plt.show()
        """
    elif key=='type':
        a=set([graph_dict[i][0]['type'] for i in range(len(graph_dict))
])
        plt.pie([[graph_dict[i][0]['type'] for i in range(len(graph_dic
t))].count(j) for j in a])
        plt.title('type distribution--Pie')
        plt.savefig('D:/Files/计算机程序设计/现代程序设计技术/作业/第四周作
业/type distribution--Pie.jpg')
        plt.show()
    else:
        print('Can not give a plot for this item!')
```

（注：由于运行内存不足，无法绘制成功柱状图，故留有注释在原位。）

绘制图如下：





七、创建 Demo 文件，依次调试模块 [iii]

以上所得运行结果均由此次运行 Demo 文件而来，具体操作如下：

```
import NetworkBuilder.node as n
import NetworkBuilder.stat as s
import NetworkBuilder.graph as g
import Visualization.plotgraph as plg
import Visualization.plotnodes as pln
import random

def main():
    #NetworkBuilder.node 模块测试
    node_list,near_dict=n.init_node()
    print(n.get_item(node_list,'type',random.randint(0,34282)))
    print(n.get_item(node_list,'weight',random.randint(0,34282)))
    n.print_node(node_list,random.randint(0,34282))
    #NetworkBuilder.Stat 模块测试
    print('The average degree of the graph is {}'.format(s.cal_average_
dgree(near_dict)))
    dgree_distribution_list=s.cal_dgree_distribution(near_dict)
    #NetworkBuilder.graph 模块测试
    graph_dict=g.init_graph(node_list,near_dict)
```

```
    g.save_graph(graph_dict)
    a=g.load_graph()
    if a==graph_dict:
        print('Transformed successfully!')
    #Visualization.plotgraph 模块测试
    plg.plot_nodes_attr(graph_dict,'weight')
    plg.plot_nodes_attr(graph_dict,'type')
    #Visualization.plotnodes 模块测试
    pln.plotdgree_distribution(dgree_distribution_list)

if __name__=='__main__':main()
```

八、检验测试结果

按照代码中的各个输出检测项目检测，均为正常：

```
终端  SQL CONSOLE  问题 1  输出  调试控制台                                    3: Python Debug Consc
Microsoft Windows [版本 10.0.18363.1139]
(c) 2019 Microsoft Corporation。保留所有权利。

D:\Files\计算机程序设计\现代程序设计技术\作业\第四周作业\GraphStat> cmd /C "D:\Software\Anaconda3\python.exe c:\Users\lenovo\.vscode\extensions\ms-python.python-2020.
9.114305\pythonFiles\lib\python\debugpy\launcher 12967 -- d:\Files\计算机程序设计\现代程序设计技术\作业\第四周作业\GraphStat\Demo.py "
starring
4431
Node
id:17925,name:"Dogtown and Z-Boys",weight:5323,type:movie,others:2001 films;Sundance Film Festival award winners;Sports documentaries;Skateboarding films;English-lang
uage films;Surfing films;Sony Pictures Classics films;
The average degree of the graph is 4.154449727270076
Transformed successfully!
```

不足与吐槽

·这次作业让我知道我们是有能够写出一个包的实力的。架构和结构并不复杂，只需要好好思考如何组织即可。而且，这样的模块化编程思维复用性高，能累积实现一些重复性的工作。

·关于可选项……时间和精力都跟不上……工业工程的孩子们太难了○|￣|_被物流管理折磨完之后可能只有时间完成基础任务了，实在不能再往下了……不过也有可能是我自身的问题。但是在看到networkx 库的时候，感觉这个库考虑了很多图论和网络规划的方法和数据结构，这对学过运筹学也力求不忘记的我很可能是一个有利的工具，之后我应该能捡起来学习这个库的。

·关于本次作业的绘图……我很想知道，如何能让我的电脑承担 3w+ 的数据在 matplotlib 上绘制柱状图？不能绘制的原因就在于无法加载出来，直接无响应……难道这样好的绘图工具就只能画画小图片了吗？

·能写出来并且运行效率不错的就是好代码了？这次作业不算很麻烦，我觉得自己写的还可以，不至于有比较大的问题~

附件列表

文件夹：GraphStat

dgree.distribution—Plot.jpg

type distribution—Pie.jpg

weight distribution—Hist.jpg

graph_dict.txt

参考网址

当然，一如既往地感谢帮我 debug 的网站！

---

[i] https://www.jianshu.com/p/4dd7cc2d95d5 复杂网络和 networkx

[ii] http://c.biancheng.net/view/5736.html Python pickle 模块：实现 Python 对象的持久化存储

[iii] https://blog.csdn.net/nigelyq/article/details/78930330 ImportError: attempted relative import with no known parent package