# 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(\*[node\_list[num][i] for i in node\_list[num]]))

### 三、计算图的度[[1]](#endnote-1)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[[2]](#endnote-2)

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

    """

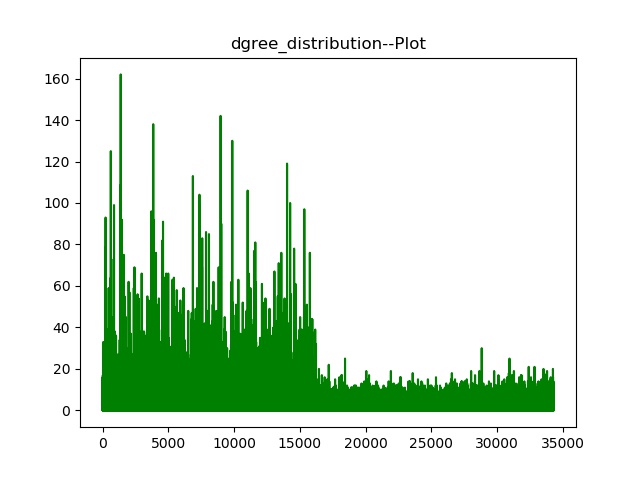
    plt.plot([i for i in range(len(dgree\_distribution\_list))],dgree\_distribution\_list,color='green')

    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(graph\_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\_dict))],[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\_dict))].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')

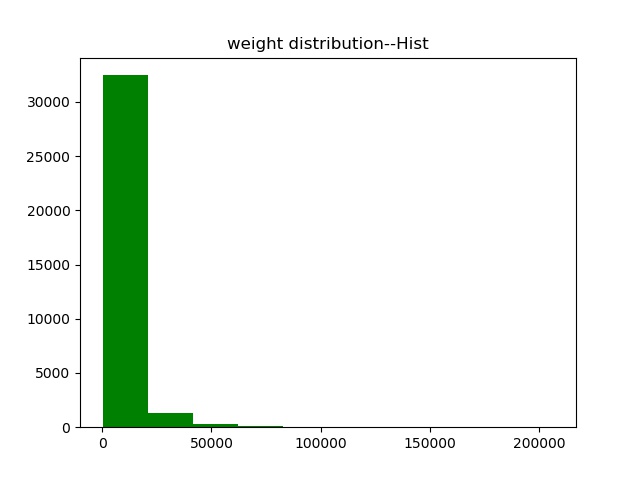
        plt.show()

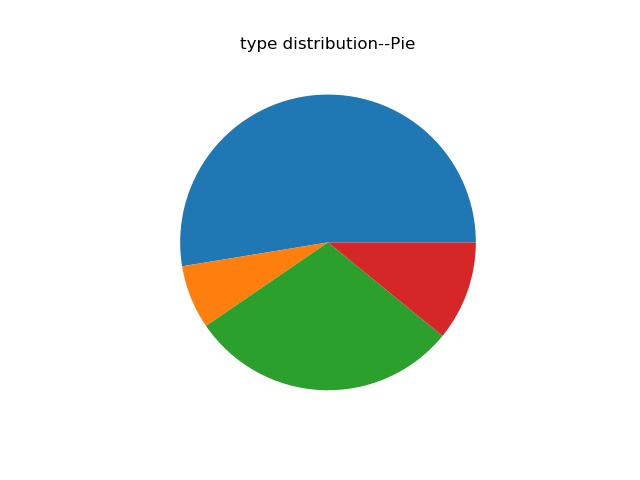
    else:

        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[[3]](#endnote-3)

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')

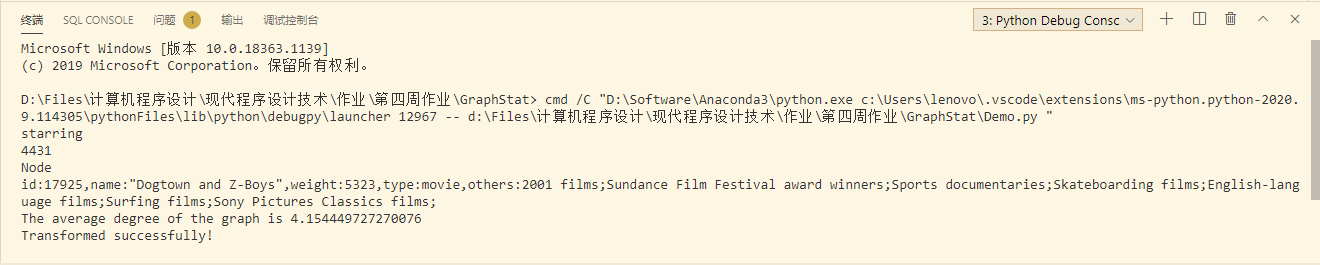
*#Visualization.plotnodes模块测试*

    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：



## 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！

1. <https://www.jianshu.com/p/4dd7cc2d95d5> 复杂网络和networkx [↑](#endnote-ref-1)
2. <http://c.biancheng.net/view/5736.html> Python pickle模块：实现Python对象的持久化存储 [↑](#endnote-ref-2)
3. <https://blog.csdn.net/nigelyq/article/details/78930330> ImportError: attempted relative import with no known parent package [↑](#endnote-ref-3)