程设第二次作业

20377383 樊思涵

参照作业 pdf 建立包 GraphStat 的结构

本报告按照完成代码的时间逻辑排序

一、读取文件并完成序列化

为了避免每次都要重新读取两个文件并生成图这一繁琐的过程，我们先将两个文件读入并生成图，对其进行序列化：

Step1.读取 features 文件（node.py）

```
print("----------正在导入节点属性数据----------")
f=open(filename_features,encoding='UTF-8')
title = f.readline().strip().split(',') #读取第一行
txt_features=[]
line=[1]
while line:  # 直到读取完文件
    line = f.readline().strip()  # 读取一行文件，包括换行符
    txt_features.append(line)
f.close()  # 关闭文件
if txt_features[-1]=='':
    txt_features.pop()
list_features=[]
for i in txt_features:
    list_features.append(i.split(','))
#print(list_features)
print("----------数据导入完成----------")
#print(title)
```

Step2.将读取的信息转换为列表（node.py）

```
    dict_id={}      #用户属性字典
    for item in list_features:
        dict_index = {'views': "NULL", 'mature': "NULL", 'life_time': "NULL",
                'created_at': "NULL", 'updated_at': "NULL", 'dead_account': "NULL", 'language': "NULL", 'affiliate': "NULL"}
        for i in range(9):
            dict_index[title[i]] = item[i]
        dict_id[item[5]] = dict_index
    #print(len(dict_id))
    #print(dict_id)
    return dict_id
```

Step3.读取 edges 文件（graph.py）

```python
import networkx as nx
import pickle

def init_graph(filename_edges,dict_id):#（可以考虑用networkx中的 Graph 等。）
    """
    构建网络
    """
    print("----------正在导入边关系数据----------")
    f=open(filename_edges,encoding='UTF-8')
    title = f.readline().strip().split(',')     #读取第一行
    txt_edges=[]
    line=[1]
    while line:       #直到读取完文件
        line = f.readline().strip()  #读取一行文件，包括换行符
        txt_edges.append(line)
    f.close()  # 关闭文件
    if txt_edges[-1]=='':
        txt_edges.pop()
    list_edges=[]
    for i in txt_edges:
        list_edges.append(i.split(','))
    #print(list_edges)
    print("----------边关系数据导入完成----------")
```

Step4.使用 networkx 库生成图

```python
    G = nx.Graph()#生成一个空白图
    for id in dict_id:
        for key in dict_id[id]:
            G.add_node(id) #插入点
            G.nodes[id][key]=dict_id[id][key] #添加属性
    G.add_edges_from(list_edges) #连接节点
    return G
```

该图包含了所有节点的信息，只需要将该图序列化以后可直接利用。

Step5.将生成的图序列化

```python
def save_graph(G,filename_save):
    """
    序列化图信息
    图的信息暂时均以字符串形式存储
    """
    f=open(filename_save,"wb")
    pickle.dump(G,f)
    f.close()
```

二、检验序列化并完成 NetworkBuilder 的其他功能

Step1.

反序列化

```python

def load_graph(filename_save):
    """
    将网络加载至内存
    """
    f=open(filename_save,"rb")
    result =pickle.load(f)
    f.close()
    return result
```

Step2.

实现其他 NetworkBuilder 的功能

①（node.py）

```python
def get_features(G,id,features):
    """
    获取节点node的指定属性。
    属性包括views，mature，life_time，created_at，updated_at，dead_account，language和affiliate
    """
    lis_features=['views','mature','life_time','created_at','updated_at','dead_account','language','affiliate']
    if features in lis_features:
        return G.nodes[id][features]
    else:
        return -1

def print_node(G,id):
    """
    显示节点全部信息（利用 format）
    """
    print("id为{0}的全部信息为{1}".format(id,G.nodes[id]))
```

②一些简单的图的参数函数（过于简单的函数直接使用 networkx 自带的函数较为方便）（stat.py）

```python
#-*- coding: gbk -*-
import networkx as nx

def get_node_number(G):
    """
    计算节点数
    """
    number=nx.number_of_nodes(G)
    return number

def get_edge_number(G):
    """
    计算边数
    """
    number=nx.number_of_edges(G)
    return number

def cal_average_degree(G):
    """
    计算网络中的平均度
    """
    sum=0
    for i in G.nodes():
        sum+=G.degree(i)
    return sum/len(G.node)
```

```python
def cal_views_distribution(G):
    """
    计算 views 属性的分布
    """
    view_dic={}
    for id in G.nodes():#生成字典
        view_dic[G.nodes[id]["views"]]=view_dic.get(G.nodes[id]["views"],0)+1
    sum=0
    for key in view_dic.keys():
        sum+=view_dic[key]
    for key in view_dic.keys():
        view_dic[key]=view_dic[key]/sum
    return view_dic
```

度的分布与 views 的分布类似，可以用相同的字典计数法求得，但 nx 库存在已有函数，故直接调用

```python
def cal_degree_distribution(G):
    """
    计算网络的度分布
    """
    degree_dis=nx.degree_histogram(G)#统计从0到最大度的频次
    lis = [z / float(sum(degree_dis)) for z in degree_dis]#生成密度列表
    return lis

```

就此 NetworkBuilder 库的内容已经完成，下面尝试在主程序中调用。

三、在主程序中尝试调用已完成的 NetworkBuilder 包
进行序列化

```python
#-*- coding: gbk -*-
import GraphStat.NetworkBuilder.graph as gng
import GraphStat.NetworkBuilder.node as gnn
import GraphStat.NetworkBuilder.stat as gns

filename_edges=r'C:\Users\LF\Desktop\twitch_gamers\large_twitch_edges.csv'
filename_features=r'C:\Users\LF\Desktop\twitch_gamers\large_twitch_features.csv'
filename_save=r'C:\Users\LF\Desktop\twitch_gamers\save.txt'

dict_id=gnn.init_node(filename_features)
G=gng.init_graph(filename_edges,dict_id)
gng.save_graph(G,filename_save)

```

进行反序列化并输出信息

```python
G=gng.load_graph(filename_save)
gnn.print_node(G,'5') #随便输出一个id的信息
print("总结点数为%d."%gns.get_node_number(G))
print("总边数为%d."%gns.get_edge_number(G))
print("网络中的平均度为%d"%gns.cal_average_degree(G))

```

```
PS E:\code\py_code> python -u "e:\code\py_code\week4\main.py"
id为5的全部信息为{'views': '4987', 'mature': '1', 'life_time': '1288', 'created_at': '2015-04-03', 'updated_at': '2018-10-12', 'dead_account': '0', 'language': 'EN', 'affiliate': '1', 'numeric_id': '5'}
总结点数为168114.
总边数为6797557.
网络中的平均度为80
```

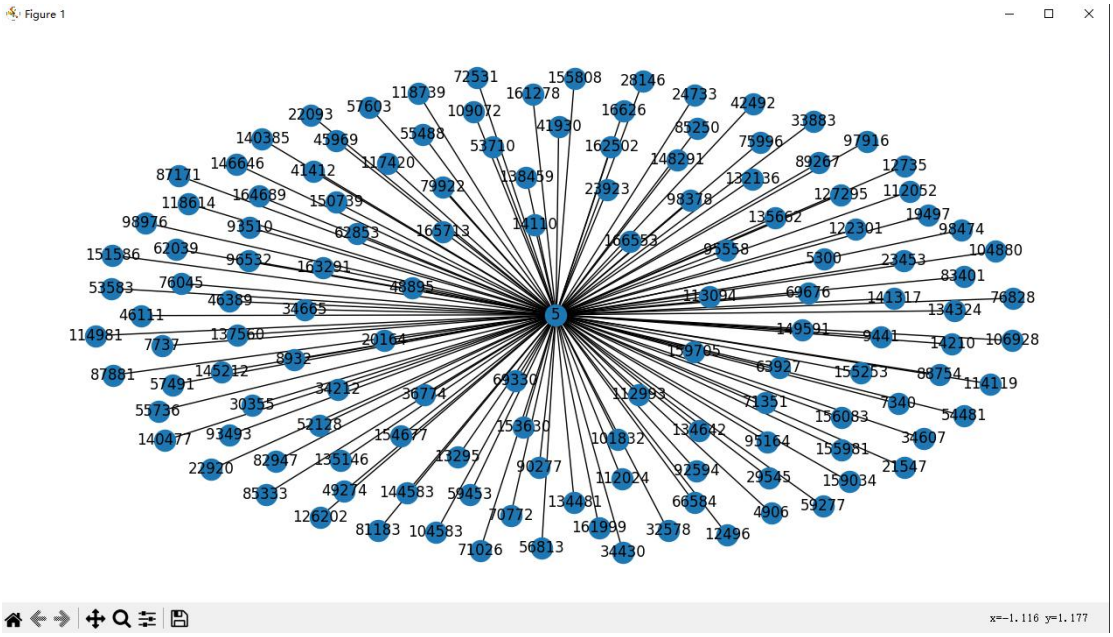
四、图的可视化

Step1.绘制节点的局部网络(plotgraph.py)

选择一个节点构建一个新的图

```python
def plot_ego(G,node):
    """
    绘制节点的局部网络
    """
    Q=nx.Graph()
    Q.add_node(node)
    lis=G.nodes()
    for i in lis:
        if G.has_edge(node,i):
            Q.add_node(i)
            Q.add_edge(i,node)
    nx.draw(Q,with_labels=True)
    plt.show()
```

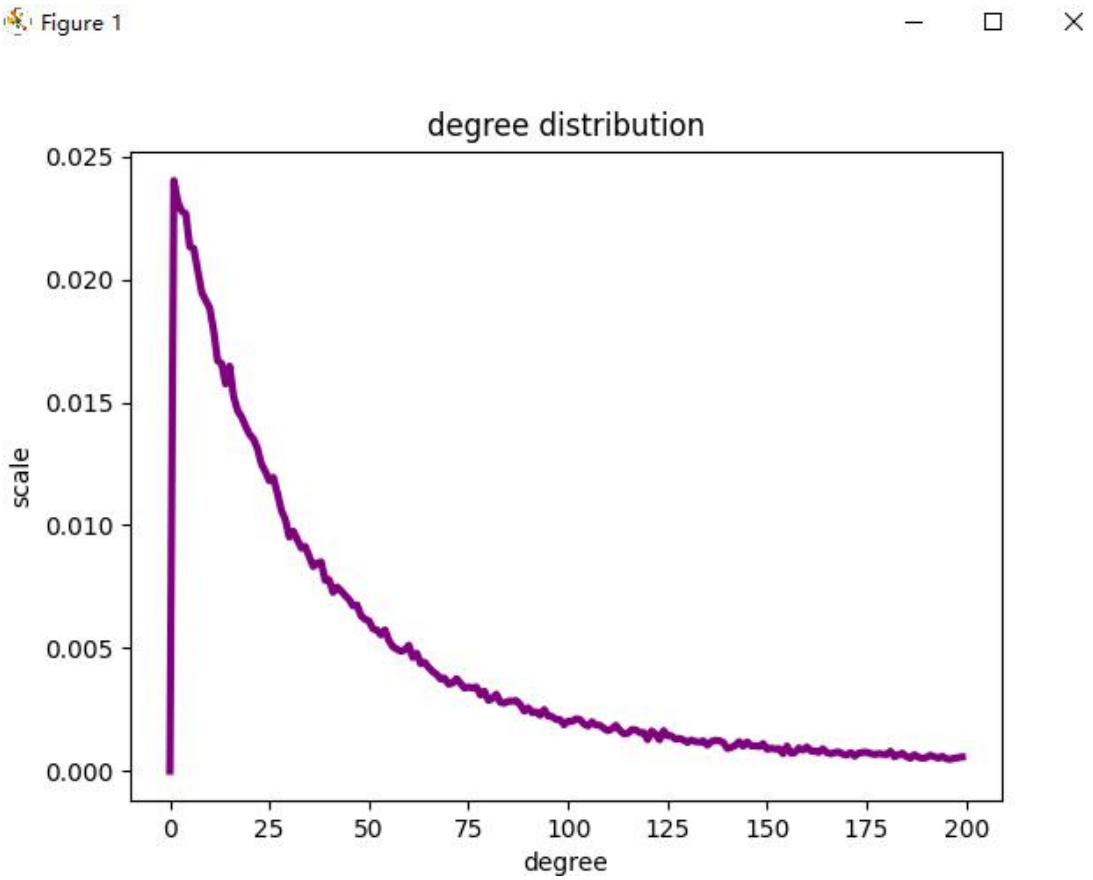
下面以 id 为 5 的节点为例输出结果



Step2.画出度的分布图

```python
def plot_degree_distribution(degree_dis): #（观察度分布的形态）
    """
    度的分布图
    由于节点分布过于分散，仅仅画出深度为0~200的分布
    """
    x=[i for i in range(200)]
    y=degree_dis[0:200]
    plt.plot(x,y,color="purple",linewidth=3.0)
    plt.title("degree distribution")
    plt.xlabel("degree")
    plt.ylabel("scale")
    plt.show()
```

由于节点的度分布过于分散，故只画出深度为 0~200 的分布
结果如下：



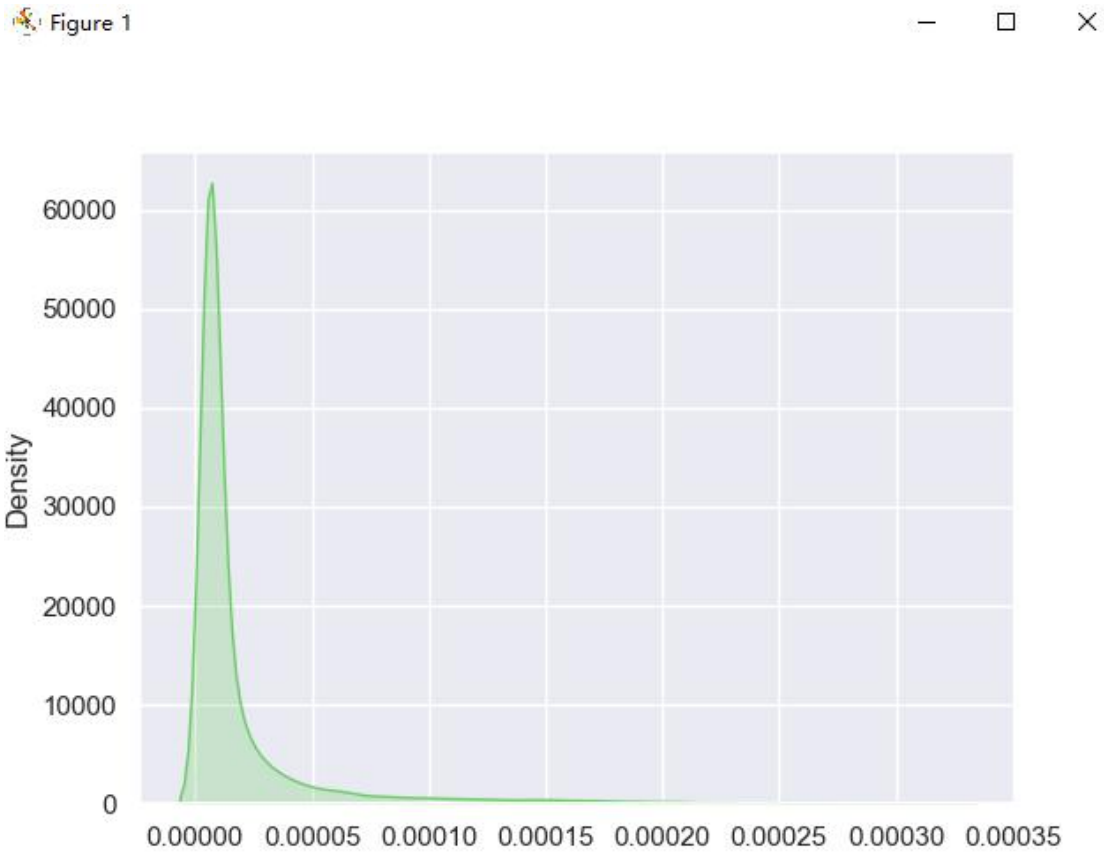
可以发现是一个近似反比例函数的图像（将点（0，0）排除在外）

Step3.观察属性的分布形态

```python
#-*- coding: gbk -*-
import seaborn as sns
import networkx as nx
import GraphStat.NetworkBuilder.stat as gns
import matplotlib.pyplot as plt

def plot_degree_distribution(G,features):
    """
    观察属性的分布形态
    """
    node=G.nodes()
    lis=['views', 'mature', 'life_time', 'created_at', 'updated_at', 'numeric_id', 'dead_account', 'language', 'affiliate']
    if features in lis:
        lis1=[]
        lis2=[0,50000,100000,150000]
        dic=gns.cal_distribution(G,features)
        for i in dic:
            lis1.append(dic[i])
        sns.set(palette='muted',color_codes=True)
        sns.distplot(lis1,lis2,hist=False,color='g',kde_kws={'shade':True})
        plt.show()
```

利用特征分布函数画出对应属性的分布图
下面以‘views’属性为例



下面是在主程序的代码

```python
#-*- coding: gbk -*-
import GraphStat.NetworkBuilder.graph as gng
import GraphStat.NetworkBuilder.node as gnn
import GraphStat.NetworkBuilder.stat as gns
import GraphStat.Visualization.plotgraph as gvpg
import GraphStat.Visualization.plotnodes as gvpn
import random

filename_edges=r'C:\Users\LF\Desktop\twitch_gamers\large_twitch_edges.csv'
filename_features=r'C:\Users\LF\Desktop\twitch_gamers\large_twitch_features.csv'
filename_save=r'C:\Users\LF\Desktop\twitch_gamers\save.txt'

#第一部分：输入数据与序列化
dict_id=gnn.init_node(filename_features)
G=gng.init_graph(filename_edges,dict_id)
gng.save_graph(G,filename_save)

#第二部分：反序列化与属性输出
G=gng.load_graph(filename_save)
gnn.print_node(G,'5') #随便输出一个id的信息
print("总结点数为%d."%gns.get_node_number(G))
print("总边数为%d."%gns.get_edge_number(G))
print("网络中的平均度为%d"%gns.cal_average_degree(G))

#第三部分：图的可视化
gvpg.plot_ego(G,str(random.randint(0,10)))
degree_dis = gns.cal_degree_distribution(G)
gvpg.plot_degree_distribution(degree_dis)
gvpn.plot_degree_distribution(G,'views')
```

序列化和反序列化 https://blog.csdn.net/reuxfhc/article/details/80043320

networkx 中文学习手册 https://blog.csdn.net/pylittlebrat/article/details/121906479