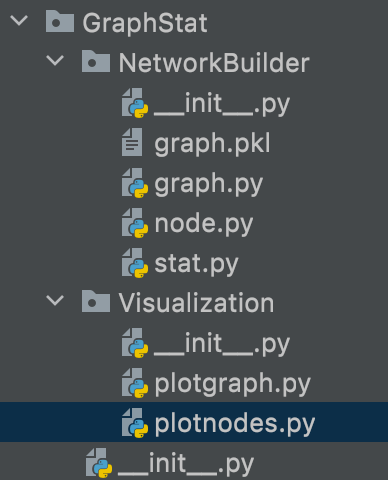
现代程序设计第四次作业

代码实现



Graph.py

import pickle  
  
'''def init\_node(path):  
 #邻接表的字典存储  
 with open(path,'r',encoding='utf-8') as f:  
 file = f.read()  
 info\_list = file.split('\n')  
 #print(len(node\_list))  
 count = 0  
 for i in info\_list:  
 count += 1  
 if i == '\*Edges':  
 break  
 #print(count)  
 #edges = info\_list[count:]  
 node\_list = node[:count-1]  
 #for i in range(len(edges)):  
 #edges[i] = edges[i].split('\t')  
 return node\_list'''  
  
def init\_graph(path):  
 #邻接表的字典存储  
 with open(path,'r',encoding='utf-8') as f:  
 file = f.read()  
 info\_list = file.split('\n')  
 count = 0  
 for i in info\_list:  
 count += 1  
 if i == '\*Edges':  
 break  
 print(count)  
 edges = info\_list[count:]  
 #node\_list = node[:count-1]  
 for i in range(len(edges)):  
 edges[i] = edges[i].split('\t')  
  
 #邻接表初始化  
 connect\_dict = {}  
 for i in range(count-2):  
 connect\_dict[i] = {}  
 #print(connect\_dict)  
  
 #构件图结构  
 for i in range(len(edges)):  
 n1 = int(edges[i][0])  
 n2 = int(edges[i][1])  
 if n1 not in connect\_dict[n2]:  
 connect\_dict[n2][n1] = 1  
 if n2 not in connect\_dict[n1]:  
 connect\_dict[n1][n2] = 1  
 return edges,connect\_dict  
  
def save\_graph(connect\_dict):  
 print("正在保存")  
 with open('graph.pkl','wb') as f:  
 pickle.dump(connect\_dict, f)  
  
def load\_graph(path):  
 print('正在加载')  
 with open(path,'rb')as f:  
 connect\_dict = pickle.load(f)  
 return connect\_dict  
  
'''def print\_node(node\_list):  
 print(node\_list[0])  
 for i in range(1,len(node\_list)):  
 #for i in range(1,10):  
 node\_content = node\_list[i].split('\t')  
 #print(len(node\_content))  
 print('id:{}\tname:{}\tweight:{}\ttype:{}others:{}'.format(node\_content[0],node\_content[1],node\_content[2],node\_content[3],node\_content[4]))  
'''  
  
'''edges, connect\_dict = init\_graph(r'/Volumes/HIKVISION/何熙1908/大三上课程/现代程序设计/第三次作业/newmovies.txt')  
print(connect\_dict[10462])'''  
'''save\_graph(connect\_dict)  
connect\_dict = load\_graph('graph.pkl')  
print(connect\_dict[0])'''  
#print(edges[0:10])

node.py

def init\_node(path):  
 with open(path,'r',encoding='utf-8') as f:  
 file = f.read()  
 node\_list = file.split('\n')  
 #print(len(node\_list))  
 count = 0  
 for i in node\_list:  
 count+=1  
 if i == '\*Edges':  
 break  
 #print(count)  
 node\_list = node\_list[:count-1]  
 return node\_list  
  
def get\_node(path):  
 return  
  
def print\_node(node\_list):  
 print(node\_list[0])  
 for i in range(1,len(node\_list)):  
 #for i in range(1,10):  
 node\_content = node\_list[i].split('\t')  
 #print(len(node\_content))  
 print('id:{}\tname:{}\tweight:{}\ttype:{}others:{}'.format(node\_content[0],node\_content[1],node\_content[2],node\_content[3],node\_content[4]))

stat.py

def cal\_average\_degree(graph):  
 total = 0  
 for key in graph:  
 total += len(graph['key'])  
 ans = total/len(graph)  
 return ans  
  
def cal\_degree\_distribution(graph):  
 degree\_dict = {}  
 for key in graph:  
 degree\_dict[key] = len(graph[key])  
 return degree\_dict

plotgraph.py

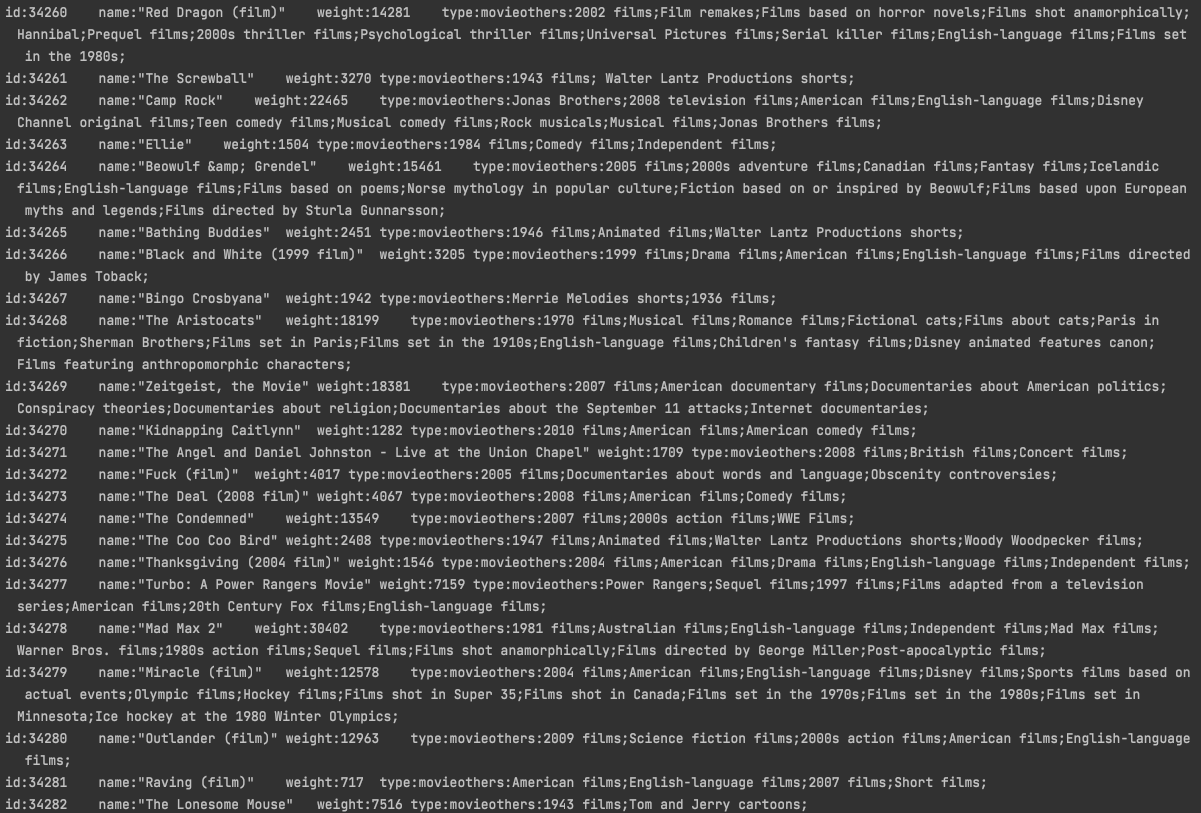
import matplotlib.pyplot as plt  
import networkx as nx  
  
def plotdgree\_distribution(graph):  
 G = nx.Graph()  
  
 #count = 0  
  
 nodes = []  
 for key in graph:  
 nodes.append(key)  
 #count+=1  
 #if count>1000:  
 #break  
 G.add\_nodes\_from(nodes)  
  
 count = 0  
 edges = []  
 for key in graph:  
 #count+=1  
 #if count>1000:  
 #break  
 for node in graph[key]:  
 edges.append((key,node))  
 #print(edges)  
 G.add\_edges\_from(edges)  
  
 nx.draw\_networkx(G)  
 plt.show()

plotnodes.py

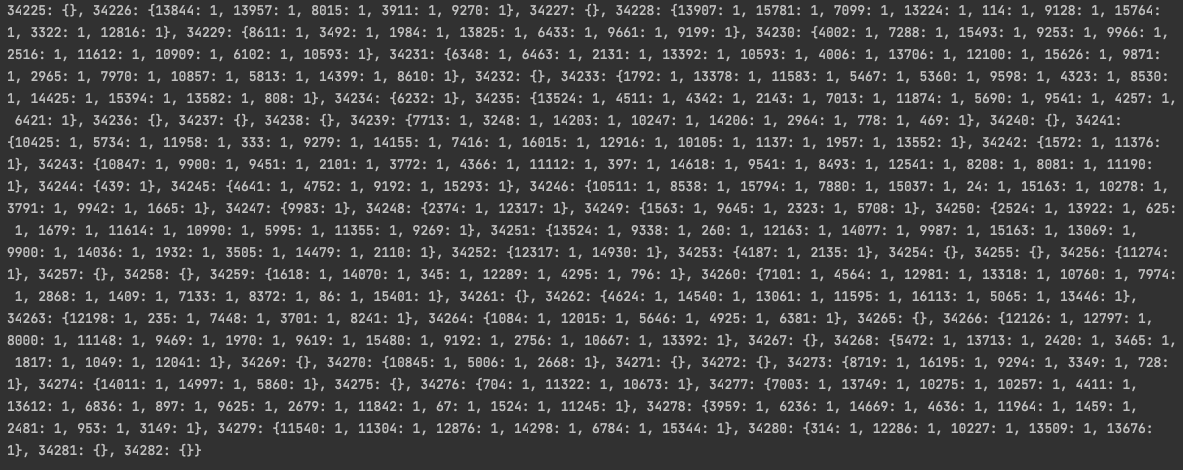
import networkx as nx  
import matplotlib.pyplot as plt  
import GraphStat.NetworkBuilder.graph as gh  
  
def plot\_nodes\_attr(graph, feature):  
 G = nx.Graph()  
  
 # count = 0  
  
 nodes = []  
 for key in graph:  
 nodes.append(key)  
 # count+=1  
 # if count>1000:  
 # break  
 G.add\_nodes\_from(nodes)  
  
 edges = []  
 for key in graph:  
 # count+=1  
 # if count>1000:  
 # break  
 for node in graph[key]:  
 edges.append((key, node))  
 # print(edges)  
 G.add\_edges\_from(edges)  
  
 count = 0  
 degree = []  
 for key in feature:  
 # count+=1  
 # if count>1000:  
 # break  
 degree.append(feature[key])  
 print(degree)  
 nx.draw(G, nodelist = nodes, node\_size = [(i \* 100) for i in degree])  
 plt.show()

结果展示

节点属性



节点图结构字典



绘图结果

