信 息 内 容 安 全 实 验 报 告

实验项目名称： 人民的名义关系分析

班级： SC011701

学号： 2017302207

姓名： 高丽

指导教师： 杨黎斌

实验时间： 2020.4.12

目录

1. 总体概述

实现了对“人民的名义”的人物关系分析和可视化。

用同样的方法实现了对“红楼梦”的人物关系分析和可视化。

2. 原理

2.1 PageRank 算法

PageRank 算法可以衡量一个节点的重要性程度，在实验中，也就是衡量一个人物的重要性，下面介绍 PageRank 算法：

PageRank 让链接来"投票"。

一个节点的“得票数”由所有链向它的节点的重要性来决定，到一个节点的超链接相当于对该节点投一票。一个节点的 PageRank 是由所有链向它的节点（“链入节点”）的重要性经过递归算法得到的。一个有较多链入的节点会有较高的等级，相反如果一个节点没有任何链入节点，那么它没有等级。

假设一个由 4 个节点组成的小团体：A，B，C 和 D。如果所有节点都链向 A，那么 A 的 PR（PageRank）值将是 B，C 及 D 的 Pagerank 总和。

$$PR(A) = PR(B) + PR(C) + PR(D)$$

继续假设 B 也有链接到 C，并且 D 也有链接到包括 A 的 3 个节点。一个节点不能投票 2 次。所以 B 给每个节点半票。以同样的逻辑，D 投出的票只有三分之一算到了 A 的 PageRank 上。

$$PR(A) = \frac{PR(B)}{2} + \frac{PR(C)}{1} + \frac{PR(D)}{3}$$

换句话说，根据链出总数平分一个节点的 PR 值。

$$PR(A)=\frac{PR(B)}{L(B)}+\frac{PR(C)}{L(C)}+\frac{PR(D)}{L(D)}$$

最后，所有这些被换算为一个百分比再乘上一个系数。由于“没有向外链接的节点”传递出去的 PageRank 会是 0，所以，通过数学系统给了每个节点一个最小值：

$$PR(A)=\left(\frac{PR(B)}{L(B)}+\frac{PR(C)}{L(C)}+\frac{PR(D)}{L(D)}+\cdots\cdots\right)d+\frac{1-d}{N}$$

所以一个节点的 PageRank 是由其他节点的 PageRank 计算得到。算法不断的重复计算每个节点的 PageRank。如果给每个节点一个随机 PageRank 值（非 0），那么经过不断的重复计算，这些节点的 PR 值会趋向于稳定，也就是收敛的状态。

2.2 社团结构检测——Louvain 算法

对“人民的名义”等社会关系网络进行社团结构探测算法可以发现其中的社团划分。

Modularity，中文称为模块度，是 Community Detection（社区发现/社团检测）中用来衡量社区划分质量的一种方法。要理解 Modularity，我们先来看社团和社团检测的概念。

社团检测，就是要在一个图（包含顶点和边）上发现社团结构，也就是要把图中的结点进行聚类，构成一个个的社团。

关于**社团**（community），目前还没有确切的定义，一般认为社团内部的点之间的连接相对稠密，而不同社团的点之间的连接相对稀疏。

Louvain 算法是一种基于图数据的社区发现算法，算法的优化目标为最大化整个数据的模块度，**模块度**的计算如下：

$$Q=\frac{1}{2m}*\sum_{ij}\left[A_{i,j}-\frac{k_i*k_j}{2m}\right]*\delta(C_i,C_j)$$

其中 m 为图中边的总数量，k_i表示所有指向节点 i 的连边权重之和，k_j同理。$A_{i,j}$ 表示节点 i，j 之间的连边权重。

Louvain 算法只是一种优化关系图模块度目标的一种实现。Louvain 算法的两步迭代设计：

1、最开始，每个原始节点都看成一个独立的社区，社区内的连边权重为 0。

算法扫描数据中的所有节点，针对每个节点遍历该节点的所有邻居节点，衡量把该节点加入其邻居节点所在的社区所带来的模块度的收益。并选择对应最大收益的邻居节点，加入其所在的社区。这一过程化重复进行指导每一个节点的社区归属都不在发生变化。

2、对步骤 1 中形成的社区进行折叠，把每个社区折叠成一个单点，分别计算这些新生成的“社区点”之间的连边权重，以及社区内的所有点之间的连边权重之和。用于下一轮的步骤 1。

3 实现和代码

3.1 总体思路

（1）首先,对于给定的剧情梗概，筛选出人物以及对应的特殊称呼，放入 people.txt 文件；

（2）读取剧情梗概文件，编码转换，分词、去特定的停用词，找出词性为人物姓名（jieba 中为“nr”）的词；

（3）我们假设两个人物在某一长度的文本中同时出现就认为这两个人物有关系，在这里我们选取这个长度为段落。即在同一段落中出现了两个不同的文本，即认为这两个人物有关系。对于一次关系，构建无向图中的一条边。读到相同的关系则+1；

（4）注意在处理过程中需要将一些特殊称呼变换为其本名，特殊称呼例如陈海和侯亮平之间的“猴子”，高育良和侯亮平、祁同伟之前的师生称呼……

（5）将处理完的网络的所有边输出到 csv 文件中；

（6）利用可视化工具 Gephi 打开，利用 PageRank 算法计算人物节点重要性，并

将重要性用节点大小的程度表示；

（7）利用其自带的 Louvain 模块检测算法找寻社团结构，并将同一个社团的节点用相同的颜色标注；

（8）导出图像。

3.2 实现代码

3.2.1 人民的名义

人民的名义的配置：

```
class NamePeople(object):

    def __init__(self):
        #转化编码
        self.peoplepath=path+"\\人民的名义材料\\people.txt"
        self.juqingpath=path+"\\人民的名义材料\\剧情梗概.txt"
        #停用词
        self.stopwords=['吕州','林城','银行卡','明白','白云','嗡嗡嘤嘤',
           '阴云密布','雷声','陈大','谢谢您','安置费','任重道远',
           '孤鹰岭','阿庆嫂','岳飞','师生','养老院','段子','老总']
        self.replacewords={'师母':'吴慧芬','陈老':'陈岩石','老赵':'赵德汉','达康':'李达康','高总':'高小琴',
                '猴子':'侯亮平','老郑':'郑西坡','小艾':'钟小艾','老师':'高育良','同伟':'祁同伟',
                '赵公子':'赵瑞龙','郑乾':'郑胜利','孙书记':'孙连城','赵总':'赵瑞龙','昌明':'季昌明',
                 '沙书记':'沙瑞金','郑董':'郑胜利','宝宝':'张宝宝','小高':'高小凤','老高':'高育良',
                 '伯仲':'杜伯仲','老杜':'杜伯仲','老肖':'肖钢玉','刘总':'刘新建',"美女老总":"高小琴"}
        self.edgepath=path+"\\边文件\\NamePeople_edge.csv"
        converformat(self.peoplepath,'gb2312')
        converformat(self.juqingpath,"gb2312")
```

3.2.2 红楼梦

```
class RedDream(object):

    def __init__(self):
        #转化编码
        self.peoplepath=path+"\\红楼梦材料\\people.txt"
        self.juqingpath=path+"\\红楼梦材料\\剧情梗概.txt"
        #这部分较少
        self.stopwords=['明白']
        self.replacewords={'宝玉':'贾宝玉','黛玉':'林黛玉','林妹妹':'林
黛玉',"宝钗":"薛宝钗"}
        self.edgepath=path+"\\边文件\\RedDream_edge.csv"
        converformat(self.peoplepath,'gb2312')
        converformat(self.juqingpath,"gb2312")
```

3.2.3 处理部分

```
G=NamePeople()#如果处理红楼梦则换成 G=RedDream()
stopwords=G.stopwords
replace_words=G.replacewords
juqingpath=G.juqingpath
peoplepath=G.peoplepath
path=getthisdirpath()
names={} #所有人物
relationships ={} #关系
lineNames =[] #每段人物
node=[] #存放处理后的人物/节点

def read_txt(path): #处理部分

    jieba.load_userdict(G.peoplepath) #加载所有人物
    #读取剧情
    with open (path,"r",encoding="utf-8") as f:
        lines=f.readlines()
    #对于每一段，分词、去停用词，得到词性，筛选人物名称
    for line in lines:
        poss=pseg.cut(line)
        lineNames.append([])
        for w in poss:
```

```
                if w.word in stopwords:
                    continue
                if w.flag != "nr" or len(w.word) <2 :
                    if w.word not in replace_words:
                        continue
                if w.word in replace_words: #将特殊称呼替换为正式名字
                    w.word=replace_words[w.word]
                lineNames[-1].append(w.word)  #为当前段增加一个人物
                if names.get(w.word) is None: #如果这个名字从来没出现过，添加关系
                    names[w.word] =0
                    relationships[w.word] ={}
                names[w.word] +=1 #该人物出现次数加 1
        #得到所有边
        for line in lineNames:
            for name1 in line:
                for name2 in line:
                    if name1 == name2:
                        continue
                    if relationships[name1].get(name2) is None: #之前没有这个关系
                        relationships[name1][name2] =1
                    else:
                        relationships[name1][name2] +=1 #有关系
    def write_csv():
        #写入边文件
        csv_edge_file = open(G.edgepath, "w", newline="")
        writer = csv.writer(csv_edge_file)
        #第一行
        writer.writerow(["source", "target", "weight","type"])
        for name,edges in relationships.items():
            for v,w in edges.items():
                if w>20:
                    node.append(name)
                    #无向图
                    writer.writerow((name,v,str(w),"undirected"))
        csv_edge_file.close()


    if __name__=='__main__':
        #读取内容
        read_txt(juqingpath)
        #输出边
        write_csv()
```

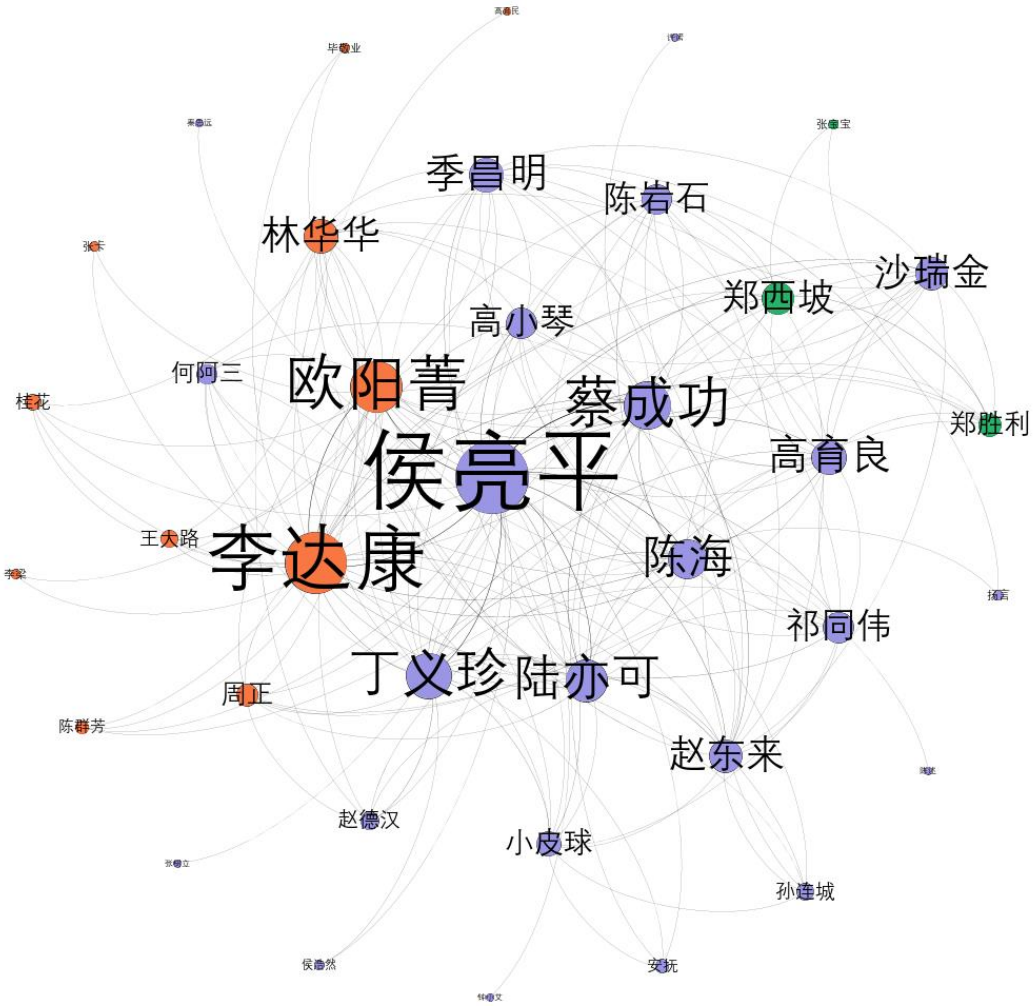
4 实验结果和分析

4.1 人民的名义

人民的名义人物关系：

其中，节点大小由 pagerank 算法计算的节点重要程度来决定；

颜色划分由社团结构划分来决定。



可见，其中侯亮平、李达康、蔡成功、欧阳菁、丁义珍、陆亦可等人物较为重要，李达康、欧阳菁等橙红色节点之间形成社团结构，侯亮平、蔡成功等

蓝色节点形成社团结构……

4.2 红楼梦

红楼梦关系网络用同样的方式画出：

其中，贾宝玉是明显的主角，其次是林黛玉、薛宝钗、贾母、贾政、王夫人等……

贾宝玉和林黛玉、薛宝钗这些人之间也形成了明显的社团结构。

仔细观察，可以发现宁国府和荣国府各自形成了比较紧密的社团结构。