基于 twitter 数据的网络关系提取及分析实验报告

一、实验目的

掌握从社交网络数据抽取用户关系并进行数据分析的方法。

二、实验环境

Windows 10 操作系统、python3.6、Gephi 0.9.2

三、实验内容

1.地震前后网络构建

根据附件 1 和附件 2 给出的推文数据，提取出用户发文时间、用户昵称、发文内容等关键字段，并根据这些信息构建地震前后的关系网络。

2.对构建好的网络进行分析

为了对比地震对用户关系带来的影响，分别对取得的英文使用者网络和日文使用者网络进行累积度分布变化统计、单独节点层面上的度变化统计并做对比分析。

3.网络可视化

使用复杂网络分析软件 Gephi 可以方便的对网络节点进行可视化。

四、实验步骤

1. 网络构建

(1) 分析 json 文件数据格式

分析 json 文件中所需字段所处位置结构。目标字段分别为”response”字段下的”content”,”trackback_date”,”trackback_author_nick”,”trackback_author_name”字段。

其中 nick 在 twitter 中唯一且只能由英文构成，name 可用多种语言。

(2) 提取关键字段（python3.6 代码示例）

```
1.    fp = open(file_name, encoding='utf-8')
2.    temp = json.loads(fp.read())
```

```
    page = temp['response']['list']
    key_record = defaultdict(int)
    for post in page:
        nick = post['trackback_author_nick']
        name = post['trackback_author_name']
        date = post['trackback_date']
        content = post['content']
    fp.close()
```

分别提取 json 文件中英文使用者和日文使用者的发帖信息。

(3) 数据清洗

将数据写入 Excel 文档或数据库中，并对数据进行去重处理。

按照地震时间（2011 年 3 月 11 日 13:46）将数据分段。时间戳可以转化为年月日格式。

为方便后期网络对比实验，只考虑地震前后共同用户变化。所以可以在这一步去除非公共用户信息，也可以在构建网络时去重非公共用户。

分别统计地震前后英文网络和日文网络的用户数量、用户发文数量、@关系数量。

(4) 网络构建

根据用户转发、评论关系构建网络。当用户 A 在其 content 字段中@用户 B，我们认为用户 A 与用户 B 之间存在联系。

分别对日语使用者地震前、日语使用者地震后、英文使用者地震前、英文使用者地震后各建网络。

NetworkX 是一个用 Python 语言开发的图论与复杂网络建模工具，内置了常用的图与复杂网络分析算法，可以方便的进行复杂网络数据分析、仿真建模等工作。networkx 支持创建简单无向图、有向图和多重图（multigraph）；内置许多标准的图论算法，节点可为任意数据；支持任意的边值维度，功能丰富，简单易用。

（网络构建部分代码 python3.6 版本示例：）

```
import networkx as nx
G = nx.Graph()
    # 创建一个权重表，字典形式存储
    w_nodes = defaultdict(defaultdict)
    # 然后按照时间一个一个的构建
    for row in range(1, len(data)):
        record = data[row]
        # check 网络中是不是已经有了这个节点
        c_nick = record['nick']
        if c_nick in G:
            pass
        # 如果没有 生成这个点
        else:
            G.add_node(c_nick)
        # check 这个帖子中 有没有 at 别人
        content = str(record['content'])
```

```
    if content:
        while find_string(content, '@'):
            # check 安特的这个人在不在网络
            ind = content.index('@')
            n_nick = ''
            while content[ind] != ' ' and content[ind] != ':' and ind < len(content) - 1:
                ind = ind + 1
                if content[ind] != ':':
                    n_nick += content[ind]
            if n_nick in G:
                pass
            # 如果不在生成这个点
            else:
                G.add_node(n_nick)
            # check 这两个人之间有没有边
            # #如果有
            if n_nick in neighbor_nodes(G,c_nick):
                # 权重+1
                G.add_edge(c_nick, n_nick)
                if c_nick in w_nodes:
                    if n_nick in w_nodes[c_nick]:
                        w_nodes[c_nick][n_nick] += 1;
            # 如果没有
            else:
                # 在这个节点和新的节点之间生成一条边
                G.add_edge(c_nick, n_nick)
                # 而且权重设为 1
                w_nodes[c_nick][n_nick] = 1;
            content = content[ind+1:]
```

(5) 去重非公共节点

```
def drop_diff_point(G1,G2):
    for node in G1.nodes():
        if not(node in G2):
            G1.remove_node(node)
    for node in G2.nodes():
        if not(node in G1):
            G2.remove_node(node)
    return G1,G2
```

(6) 将构建好的网络存入文件中

```
nx.write_gml(G,"Filename.gml")
```

2. 网络分析

(1) 使用 python 中 networkx 库对网络进行基本特性（平均度、最大联通片、群聚系数）分析

```
#输出网络平均度
def average_deg(G,name):
    d = nx.degree(G)
    print(name + "平均度")
    # print(d)
    print(np.array(list(d.values())).mean())
```

```
#输出最大联通片
def largest_com(G,name):
    largest_components=max(nx.connected_components(G),key=len)
    print(name + "的最大联通成分的大小为",end = '')
    print(len(largest_components))
```

```
#输出平均群聚系数
def average_clu(G,name):
    c = nx.average_clustering(G)
    print(name + "的平均群聚系数:",end = '')
    print(c)
```

(2) 个人层面度分析

```
def individualdegree(G1,G2,name):
    nodes1 = G1.nodes()
    nodes2 = G2.nodes()
    degree1 = []
    degree2 = []
    for node in nodes1:
        if node in nodes2:
            degree1.append(G1.degree(node))
            degree2.append(G2.degree(node))
    # plt.scatter(degree1,degree2)#在双对坐标轴上绘制度分布曲线
    # plt.subplot(121)
    plt.title(name)
    plt.xlabel('before')
    plt.ylabel('after')
    plt.loglog(degree1,degree2,'o')#在双对坐标轴上绘制度分布曲线
    plt.show()
```

(3) 累积度分析

```
#接下来做累积度分布
def cumlutive_degree_distribution(G):
    degree = []
    list = G.degree()
    for each_node in list:
        degree.append(list[each_node])
    xs = degree
    distKeys = range(min(xs), max(xs) + 1)
```

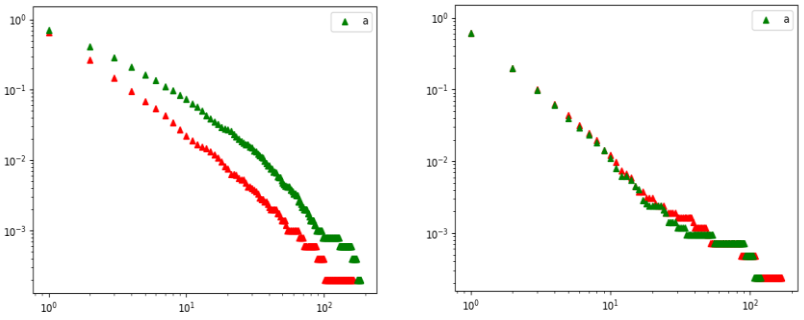
```
    pdf = dict([(k, 0) for k in distKeys])
    for x in xs:
        pdf[x] += 1
    pdf_temp=pdf
    scope = range(min(pdf),max(pdf)+1)
    for degree in scope:
        k=degree+1
        while k<=max(pdf):
            pdf[degree]+=pdf_temp[k]
            k+=1
    return pdf
#根据（图，名称，度列表，返回一个度分布图）
def draw_degree_chart(G,name,distribution):
    # degree=nx.degree_histogram(G)#返回图中所有节点的度分布序列
    degree = distribution
    # print(degree)
    y = np.array(list(degree.values()))
    # print(y)
    # y=[z/float(sum(degree))for z in degree]#将频次转化为频率，利用列表内涵
    y = y/y[0]#将频次转化为频率，利用列表内涵
    x=range(len(degree))#生成 X 轴序列，从 1 到最大度
    # y = degree
    if 'b' in name:
        color = 'r^'
        # 'r--':红色的需要;'bs':蓝色方块;'g^':绿色三角
        line = plt.loglog(x, y, color)  # 在双对坐标轴上绘制度分布曲线
        plt.legend(line, ("before"))
    else:
        color = 'g^'
        # 'r--':红色的需要;'bs':蓝色方块;'g^':绿色三角 o 圆
        line = plt.loglog(x, y, color)  # 在双对坐标轴上绘制度分布曲线
        plt.legend(line, ("a"))
    plt.title( name)
      plt.show()#显示图表
```

五、实验结果

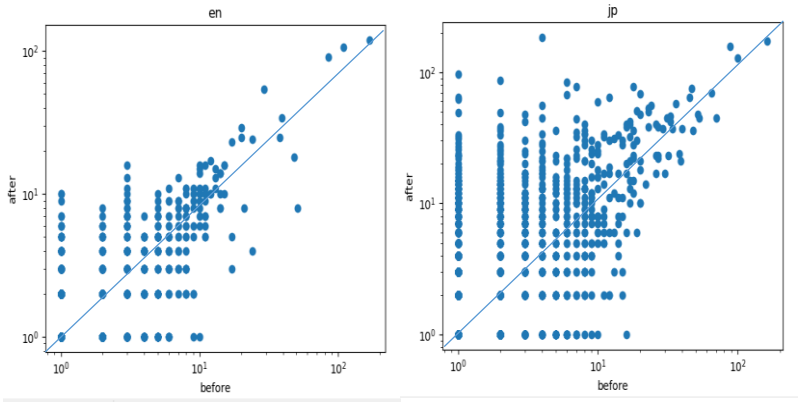
1. 网络基本特性信息统计对比

Tabel 1 Data description and basic network characteristics								
data	period	raw data			Extracted network			
		#users	#tweets	#links	#nodes	D	s	c
TP-JP	before	12863	40801	12956	5077	1.689	2591	0.000107
	after	32032	105668	52139	5077	3.214	3440	0.000208
TP-EN	before	17537	44993	15671	4200	1.274	1630	0
	after	17841	58429	15951	4200	1.235	1626	0.000847
D:average degree;s:size of giant connected component or giant strongly connected network;c:clustering coefficient.								

2. 累积度对比



3. 个人层面度对比



六、使用 Gephi 软件进行网络可视化和分析

1.打开之前保存的 gml 文件。

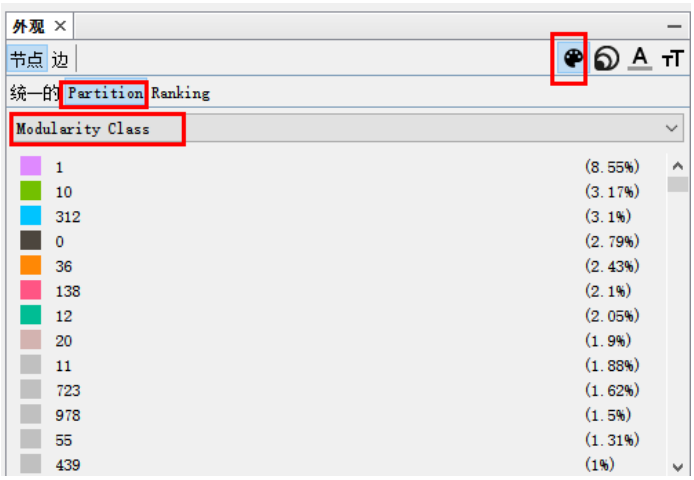
2.网络信息统计

在统计板块可以对网络进行基本信息统计。

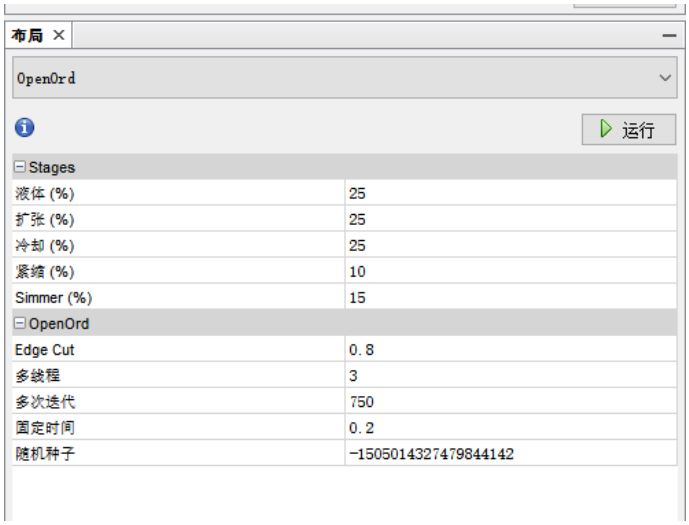
3.社区划分与渲染

另外，Gephi 内置快速模块化算法，可以根据网络结构对网络进行简单的社区划分。点击上图网络概述板块中的“模块化”，运行 community detection 算法。

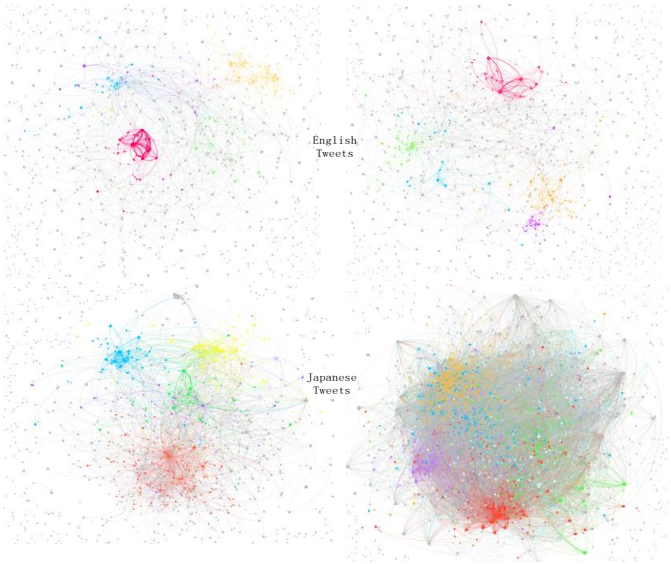
社区划分后，左上角外观部分可以对检测出的社区进行颜色渲染操作。



在左下角有布局选项，选择一个算法，点运行展开(lay out) 。



结果：



参考视频教程：https://www.udemy.com/gephi/

参考案例：https://blog.csdn.net/laoyang360/article/details/53616720