

基于新闻数据的社交网络图结构挖掘

摘要: 现实世界中包含着各种类型的复杂网络，如社交网络、技术网络、生物网络等。经过近几年的努力，复杂网络社区的研究取得了许多重要进展，发现了复杂网络的若干统计特征，其中包括小世界理论、无标度性质以及聚集性或网络传递性。而复杂网络不仅是一种数据的表现形式，它同样也是一种科学的研究的手段。目前，研究复杂网络方面收到了广泛的关注和研究。而复杂网络的另一个重要特征就是网络中所呈现出的社区结构。社区网络结构是由一组网络节点构成的集合，社区内部节点联系紧密，社区间节点联系稀疏。对社区的划分有利于进一步研究检测社区中节点的相似性和紧密性。通过对节点的度以及边权值的统计，可以明晰社区中节点的重要性排名和相关性强度。本文主要利用数据库语句和广度优先搜索、tarjan、PageRank、Louvain 等算法对图中节点、边、连通分量和其他性质进行研究。

关键词: 社交网络；去噪；PageRank 算法；社区发现；聚集系数

Social Network Graph Structure Mining Based on News Data

Abstract: In the real world, there are various types of complex networks, such as social networks, technology networks, biological networks. Through the efforts in recent years, the research on complex network communities has made many important progress, and found several statistical characteristics of complex networks, including small-world theory, scale-free property and aggregation or network transitivity. Complex network is not only a form of data expression, but also a means of scientific research. At present, the study of complex networks has received extensive attention and research. Another important feature of complex networks is the community structure presented in the network. Community network structure is composed of a group of network nodes. The nodes within the community are closely related, and the nodes between the communities are sparsely connected. The division of communities is conducive to further study and detect the similarity and compactness of nodes in communities. Through the statistics of node's degree and edge weight, the importance ranking and correlation intensity of nodes in the community can be clearly defined. This paper mainly uses database statements and breadth-first search, tarjan, PageRank, Louvain and other algorithms to study the nodes, edges and connected components in the graph.

Key words: community structure, denoising, PageRank algorithm, Louvain algorithm, convergence factor

1 背景陈述

人类历史上，但凡重要的技术革命都伴随着媒介革命，人类任何活动本质上都是信息活动，信息流的传递介质、管理方式的不同，将决定接受信息的不同，所有有关信息流媒介的变革一定是底层的变革——社区中的网络也是如此。一个复杂网络中的一个普遍特征就是社区结构，整个网络是由许多个社区组成。而利用社区中的节点相似性来研究其网络中的紧密联系是揭示社区结构的重要因素。因此，我们有必要对社区结构中节点的度、边权值等特性进行研究。

2 符号体系

表 1 符号体系

Table 1 Symbolic system

符号	含义
PR_i	第 i 个网页的 PageRank 值
O_j	网页 j 的出度
Q	模块度
A_{ij}	节点 i 和节点 j 之间边权重
	所有与节点 i 相连的边的权
k_i	重之和
c_i	节点 i 所属的社区
m	所有边的权重之和
Σ_{in}	社区 c 中边的权重总和
	社区 c 中节点相连的边的
Σ_{tot}	权重总和
	节点 v_2 的所有相邻节点个数
k	数
	节点 v_2 所有相邻节点之间互相连接的边的个数
n	

3 正文

3.1 预处理

预处理主要分为去重、去噪这两个部分，其中，去重部分主要针对代表人名的节点进行的。

3.1.1 去重

首先将每个新闻中人名这一行的名字提取出来作为节点。通过提取大量人名，我们发现不同新闻中出现同一人名的情况屡见不鲜。于是，在所有人物名提取完毕后，进行了去重处理。考虑到在大数据的背景下，处理的方法需要调整到适应人物过多的情况。因此利用了 python 中的一个数

据结构——集合 set。集合 (set) 是一个无序的不重复元素序列，即具有自动去重的性质。如果将每次读出的一个人名放入集合 (set) 当中，遍历一遍新闻之后，就能出现所有互不相同的点了。通过获取集合 (set) 的长度，即可获得不同点的个数。

3.1.2 去噪

在提取人名的过程中，发现人名的格式有很大的差别，在某种程度上看，有些提取出的人名并不符合现实。通过大量的观察与分析，我们主要采用三种方式提高人名的质量。

1. 根据提取小数据集的英文结果，观察到很多英文名称实际上是一句任意的英文长句，而不是一个名字。而在对比英文名的构造之后，可以确定基本上英文名字都是由不超过三个空格分隔开的。由此，通过计算所谓英文名称内部包含的空格数，删除掉空格数超过三的人名，便可以去除掉很多不必要的节点，提升了数据提取的质量。

2. 第二种方法是基于人名的格式来完成的。比如，在一个新闻中出现了‘(永春’的字样，而在另一个新闻中则是直接出现了人名‘永春’。在这种情况下，有足够的理由认为这两个名字其实是一个人。因此，可以去除掉名字中的‘(’，提升后期进行边统计时的准确度。

3. 第三种方法是去除掉名称中有‘@’符号的名字。因为很多新闻数据中，人名是以邮箱名呈现的，这也不是一个真正的人名。于是，去除掉这些邮箱名，有助于减少离群点的数量，增加结果的可信度。

3.1.3 数据库导入的两种方法

I. 使用 CSV 文件格式导入

需要在文件的开头一行，指明每一列含义，也必须用逗号分隔开。要导入的文件需放在 neo4j 数据库的安装文件夹下的 import 文件夹内，此导入方式适用的范围为图中点的个数在 1-10 万之间的网络图。在导入点之后给点加上索引可以大大加快边导入的速度。

Cypher 代码如下：

导入点的代码：

```
LOAD CSV WITH HEADERS
FROM "file:///person.csv" AS line
MERGE(p:person {id:line.ID,
name:line.name})
```

导入边的代码:

```
USING PERIODIC COMMIT 800
LOAD CSV WITH HEADERS
FROM "file:///rela.csv" AS line
MATCH (from:person {id:line.from_id}),
      (to:person {id:line.to_id})
MERGE (from)-[r:rel {property:line.property}]->(to)
```

II. import 命令行导入

这是 neo4j 数据库自己提供的一个导入工具，适用于很大范围内的数据的导入，而且速度非常快。使用的前提是目前数据库处于关闭状态，且数据库为空，需要一起导入所有的点和边。数据库，点文件，边文件都需要指明绝对路径，此处代码中为了简便略去只写了名字。

导入数据的代码:

```
neo4j-import --into graph.db
--nodes person.csv
--relationships edge.csv
--input-encoding UTF-8
```

文件的格式如下:

点的文件首行需要指明 ID，不同的属性可以在后面指明类型。

```
:ID(person),:LABEL,name,degree:int
```

```
1,person, Horace Ankrah,3
```

```
2,person, 洪波,9
```

边的文件首行需要指明起点和终点，以及这条边的类型。

```
:STARTID(person),:ENDID(person),
```

```
:TYPE,property:int
```

```
18913,77572,rel,1
```

```
77572,18913,rel,1
```

3.2 基础功能

3.2.1 图的验证

I. 功能介绍

图的验证主要提供一个网页界面，在输入一个人名后，输出和该人关系最强的十个邻居。根据这一要求，不仅要利用预处理过程得到的节点，而且需要统计出边的权值。为了输入关系最强的十个邻居，还需要对与某一节点相连的边的权值进行排序。

II. 算法介绍与改进

1. 边权值统计

(1) Pandas 包介绍

Pandas 中的 groupby() 和 sum() 函数可以实现二维表中数据的分组并求和的功能，因此可以把初步筛选到的“结点 1-结点 2-边权值”存到 pandas 中的 DataFrame 类型的 table 中，然后用 table.groupby(['person1','person2']).sum() 即可统计出所有边及对应的权值。

(2) 步骤

第一次统计边的权值时，使用了 python 的 Pandas 包，将所有的边都列出来，如果有两个人名再次出现在另一个新闻当中，也就是遇到了相同的边，则权值加一。但是，在实践中发现，由于边的个数总计超过了 1800 万，python 在导入的过程中速度非常慢，于是，进一步改进了算法。

改进后，决定利用 SQL server 数据库来统计边权值。大数据的方法是将所有的边都列出来，具体列出的是一个人名和另一个人名以及对应的一条边。而数据库中的 select 语句可以用于从表中选取数据，结果被存储在一个结果表中，称为结果集。接着，将列出的内容导入数据库中，利用数据库中的 select 语句就可以去重并直接统计出结果。统计相同元组的和，数据量比较大，而数据库在这一功能上性质较好，可以快速进行。

(3) select 语句去重并统计边权值

代码:

```
select distinct source ,target ,weight
from
node as node1 ,
(select distinct s as source ,t
as target ,SUM(v) as weight
from alledge
group by s,t having s<>t) as edge ,
node as node2
where node1.name = edge.source
and edge.target = node2.name
```

2. 输出关系最强的十个邻居

理论上，在节点与边权值导入数据库之后，可以直接利用数据库的语句统计出排名最高的十个邻居。实际上，则采用了优先队列计算排名，其过程主要分为两个步骤。

第一，要统计出每个节点的度，即一个节点的邻居个数。设一条边的两个节点为 u 和 v。初始设一个数组，数组的长度就是节点的个数，初

始值是 0，遍历每一条边时将数组的第 u 个值和第 v 个值加一，表示两个节点都有邻居，遍历结束就得到了节点的所有邻居。

第二，针对一个节点建立一个优先队列。考虑到有些节点可能有不足十个邻居，于是控制优先队列里的元素不超过 10。在统计过程中，先遍历所有邻居，把边权值压入队列，设定优先队列中的元素是由大到小排列的。如果队列已满，就对比优先队列中最小的数值和与新邻居的权值。如果与邻居的权值更大，则除去最小的权值，将该邻居加入优先队列；如果与邻居的权值更小，则不考虑这个邻居，继续遍历。最终，优先队列中的 10 个元素就是与该节点关系最强的 10 个邻居。

实际实现时使用的数据库代码为：

```
MATCH (n:person {name: 'person_name'})  
-[x:rel]-(t:person)  
RETURN DISTINCT t.name, x.property  
ORDER BY x.property DESC  
LIMIT 10
```

III. 页面的逻辑

前端是以 node.js 为框架，主要使用 Javascript 来进行前端和数据库的交互。前端和数据库的连接使用 node.js 的一个软件包 neo4j-driver。首先检测到用户点击查询的事件发生，之后在 html 中提取名字作为数据库查询的 query，然后在 Javascript 代码中嵌入查询语句即可得到数据库返回的查询结果，接着操纵 html 文件，最终将每个人关系最强的 10 个邻居以及它们之间的边权值返回到页面上。

IV. 结果展示

与“习近平”关系最强的十个人，如图 1 所示

与“奥巴马”关系最强的十个人，如图 2 所示

3.2.2 图的统计

I. 功能介绍

图的统计是计算出图的节点个数、边数、连通分量的个数以及最大连通分量的大小。利用预处理中去重过程可以得到互异的节点个数，并且在采用数据库计算边权值时，就可以直接得到边的个数。而连通分量的个数和最大连通分量的大小可以通过广度优先搜索和 tarjan 算法两种方法得到。

II. 算法介绍

Search Results	
Person	Weight
李克强	402
奥巴马	214
胡锦涛	192
张德江	192
杨洁篪	144
刘云山	127
刘晓明	127
刘延东	123
邓小平	120
马克思	110

图 1 习近平的邻居

Fig. 1 Neighbor of Xi Jinping

Search Results	
Person	Weight
习近平	214
胡锦涛	173
杨洁篪	84
克林顿	47
戴秉国	42
王岐山	41
崔天凯	38
拜登	30
克里	30
盖特纳	29

图 2 奥巴马的邻居

Fig. 2 Neighbor of Obama

1. 广度优先搜索

广度优先搜索 (BFS) 属于一种盲目搜寻法，目的是系统地展开并检查图中的所有节点，以找寻结果。已知图 $G=(V,E)$ 和一个源顶点 s ，广度优先搜索以一种系统的方式探寻 G 的边，从而“发现” s 所能到达的所有顶点，并计算 s 到所有这些顶点的距离 (最少边数)，该算法同时能生成一棵根为 s 且包括所有可达顶点的宽度优先树。对从 s 可达的任意顶点 v ，广度优先树中从 s 到 v 的路径对应于图 G 中从 s 到 v 的最短路径，即包含最小边数的路径。

在利用 (BFS) 计算连通分量个数时，首先要建立一个辅助队列，head, rear 是这个队列的操作

变量，当 `head=rear` 时，证明这个队列为空，入队移动 `rear`，出队移动 `head`。而这个队列主要是用来存放顶点数组中未被访问的节点。接着，将数组中所有顶点初始化标记为 0，也就是未被读到。然后开始读，读顶点先把它自身与它链表中的、没有被访问的点进行入队列处理，并与连通分量集合，再一个一个出队，则对链表中的点，和该点的链表上的点进行入队列与连通分量集合。这样才算遍历完一个顶点，再读下一个顶点。有时，有的顶点以及其后的所有点，都被读过，这是因为它是连通分量中的一员。当遍历完所有顶点，就可以得到所有的连通分量，这样也就知道了连通分量的个数和最大连通分量的大小。

2.Tarjan 算法

Tarjan 算法是一种由 Robert Tarjan 提出的求有向图强连通分量的时间复杂度为 $O(n)$ 的算法。首先我们要知道两个概念：时间戳 (DFN)，节点能追溯到的最早的栈中节点的时间戳 (LOW)。顾名思义，DFN 就是在搜索中某一节点被遍历到的次序号 (dfs num)，LOW 就是某一节点在栈中能追溯到的最早的父亲节点的搜索次序号。

Tarjan 算法是基于深度优先搜索的算法。在搜索过程中把没有 Tarjan 过的点入栈，并将该节点的 $DFN[i] = LOW[i] == ++dfsnum$ （也就是设成它自己），然后以这个节点为树根再进行搜索。当一颗子树搜索完毕时回溯，并在回溯时比较当前节点和目标节点的 LOW 值，将较小的 LOW 值赋给当前结点的 LOW，这样可以保证每个节点在以其为树根的子树的所有节点中 LOW 值是最小的。如果回溯时发现当前节 $DFN[i] == LOW[i]$ ，就将栈中当前结点以上的节点全部弹栈，这些点就组成了一个强连通分量。还要注意一点是，当目标节点进行过 Tarjan 但还在栈中，就拿当前节点 LOW 值与目标节点 DFN 值比较，把更小的赋给当前结点的 LOW。

因此，在搜索过程中可能会遇到以下三种节点：从没访问过的节点（固然不在栈中），访问过但不在栈中的节点，访问过但在栈中的节点。对于三种点进行分开讨论。

1) 对于从没访问过的节点：加入栈中让其 $DFN[i] = LOW[i] == ++dfsnum$ ，让 $vis[i]=1$ 表示该点入栈了。这类点的标志是 $!DFN[i]\&\&!vis[i]$ 。

2) 对于访问过但不在栈中的节点 ($!vis[i]$) 直接回溯即可，因为既然该节点访问过了又不在栈

中，就必定属于另一个强连通分量。这类点的标志是 $DFN[i]\&\&!vis[i]$ 。

3) 对于访问过且在栈中的节点，比较当前节点 LOW 值和目标节点 DFN 值，将较小的赋给当前结点 LOW 值然后回溯。这类点的标志是 $DFN[i]\&\&vis[i]$ 。

伪代码如下：

算法 1 求解有向图强连通分量

```

1: function TARJAN(u)
2:   visit[u]  $\leftarrow$  true
3:   for each(u, v) in QUERY do
4:     if visit[v] then
5:       ans(u, v)  $\leftarrow$  FIND(v)
6:     end if
7:   end for
8:   for each(u, v) in TREE do
9:     if !visit[v] then
10:      Tarjan(v)
11:      parent[v]  $\leftarrow$  u
12:    end if
13:   end for
14: end function

```

3. 数据库实现

Cypher 代码

```

CALL algo.scc('person', 'rel',
  {write: true, partitionProperty:
  'partition'})
YIELD loadMillis, computeMillis,
  writeMillis, setCount,
  maxSetSize, minSetSize;

```

III. 结果展示

基础功能部分的结果详见表 2，其中第一行代表小数据集，第二行代表大数据集；“连通分量”代表连通分量个数，“最大分量”代表最大连通分量节点数。

表 2 基础功能结果

Table 2 Fundamental Function

点	边	连通分量	最大分量
80188	1672855	1777	77337
361642	19811401	310	360792

3.3 高级功能

3.3.1 PageRank——影响力计算

I. 介绍

PageRank 是 Sergey Brin 和 Larry Page 在 1998 年的 WWW7 会议上提出来的，旨在解决链接分析中网页排名的问题。在此功能下，我们计算了图中每个人的影响大小，并根据排序结果得到影响力最大的前二十个人。

在衡量一个网页的排名时，我们首先要考虑几个因素：

- (1) 当一个网页被更多网页所链接时，其排名会越靠前；
- (2) 排名高的网页应具有更大的表决权，即当一个网页被排名高的网页所链接时，其重要性也会对应提高。

II. 原理

总体来看，PageRank 算法会预先给每个网页一个 PR 值，O 为网页的出度，即每一个网页被访问的概率为 $\frac{1}{O}$ 。我们假定所有网页的 PR 值的总和为 1，以便直观地反映概率。根据以上描述，我们可以建立一个网页的排名公式：一个网页的排名等于所有链接到该网页的网页的加权排名之和。

公式描述如下：

$$PR_i = \sum_{(j,i) \in E} \frac{PR_j}{O_j}$$

其中， PR_i 表示第 i 个网页的 PageRank 值，用来衡量每个网页的排名， O_j 为网页 j 的出度。

由此，网页之间的链接关系便形成了一个有向图 $G=(V,E)$ ，边 (j,i) 代表网页 j 链接到了网页 i 。我们用 n 维 PageRank 值向量表示，则有 $P = (PR_1, PR_2, \dots, PR_n)^T$ ，而 A 作为有向图 G 的转移矩阵表示为

$$A_{ij} = \begin{cases} \frac{1}{O_j}, & (i,j) \in E \\ 0, & \text{else} \end{cases} \quad (1)$$

因此，我们可以构建 n 个等式，表示成矩阵相乘的形式： $P = A^T P$ 。

III. 网页之间链接关系分类

1. 普通情况

针对图 3，根据以上公式，我们可以得出网页 A 的 PageRank 值： $PR(A) = \frac{PR(B)}{2} + \frac{PR(C)}{1}$ 。

2. 没有出链

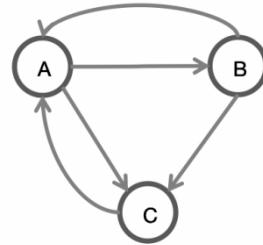


图 3 普通情况

Fig. 3 Normal

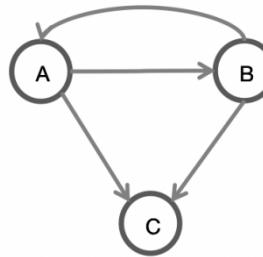


图 4 没有出链

Fig. 4 No Outbound Link

对于图 4，此时，网页 C 没有出链，即网页 C 对其他所有网页没有贡献 PR，为了满足 Markov 链的收敛性，我们设定网页 C 对所有网页（包括网页 C 本身）都有出链，则网页 A 的 PageRank 值 $PR(A) = \frac{PR(B)}{2} + \frac{PR(C)}{4}$ 。

3. 出链有循环圈

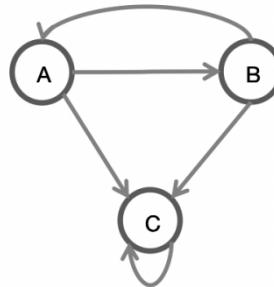


图 5 出链有循环圈

Fig. 5 Outbound Link With Loops

对于图 5，在这种情况下，我们发现在迭代的过程中，网页 C 的 PageRank 值只增不减，而且网页 C 只会为自己贡献 PageRank，这样显然是不公平、不合理的。为了解决这一问题，我们假设一个人不会一直只浏览一个网页 C，而是回会等概率的浏览其他网页。根据我们搜集到的资料，此时网页 A 的 PageRank 值： $PR(A) = \alpha(\frac{PR(B)}{2}) + \frac{(1-\alpha)}{4}$ 。

一般情况下，我们取 α 为 0.85，由此，我们将公式改进为

$$PR_i = \alpha \sum_{(j,i) \in E} \frac{PR_j}{O_j} + \frac{1-\alpha}{N}$$

其中 N 表示为网页总数。

IV. 优点

我们要想计算某个网页的 PageRank，首先需要知道 PageRank，这样就形成了一个循环圈，难以求解。但是 PageRank 算法利用幂迭代法破解了这个逻辑圈，经过 n 次迭代（ n 充分大），总可以找到一个值，使得我们可以默认每个网页的 PageRank 就在该值的小邻域内，即 $X_n \rightarrow X_0$ 。

V. 试验结果

1. Pgerank 数据库的代码

```
MATCH (node:person)
WITH collect(node) AS nodes
CALL apoc.algo.pageRank(nodes)
    YIELD node, score
RETURN node, score
ORDER BY score DESC
LIMIT 20
```

2. Pagerank 的结果

(1) 对于小数据集，PageRank 排名前 20 的人为：

习近平，胡锦涛，李克强，徐建平，程永华，邱小琪，伍江，杨洁篪，裴钢，吴志强，杨贤金，杨厚兰，罗林泉，孔子，邓小平，季志业，马锦明，诺贝尔，吴邦国。

(2) 对于大数据集，PageRank 排名前 20 的人为：

特朗普，孔子，安倍，毛泽东，乾隆，李克强，詹姆斯，孙子，范冰冰，蒋介石，王毅，邓小平，赵丽颖，马云，鲁迅，马克思，川普，刘强东，周迅，王菲。

3.3.2 社区挖掘

I. 背景

社区结构作为真实复杂网络所普遍具有的一个重要的拓扑特性，最近 10 年内得到了广泛而深入的研究。为了明确社区划分机制，我们首先分析了已知框架下，大量新闻信息所组成的社交网络、社区、社区发现以及相关定义。

社区网络是由节点和边组成的结构，节点代表新闻中的人名，而不同人在同一新闻中出现则两节点间就有一条边，如果对节点间关系的强度

进行区分的话，那么每条边的权重即为两个节点在同一新闻中出现的次数。社区发现则是社交网络分析的一个基本任务，社区，从直观上来看，是指网络中的一些密集群体，每个社区内部的节点间的联系相对紧密，各个社区之间的连接相对比较稀疏。通过大量的统计分析，我们已经得到了网络图，此时找出社区结构的过程便是社区发现。以社交网络为例，社区结构是客观存在的，但某个社区内的某个用户只和那些与其相连的用户产生互动，殊不知，在这个社区内，他和那些与其没有链接的用户其实也“很近”，如果做好友推荐，属于同一社区的用户之间应该优先进行推荐，对于一个大型网络调用社区发现算法，其实是对其按照某种标准进行了划分，在此基础上可对社区做进一步的挖掘。

II. GN 算法

1. 算法思想

在一个网络之中，边介数表示任意两个节点通过此边的最短路径的数目。通过网络只能够社区内部的边的最短路径相对较少（边介数小），而通过社区之间的边的最短路径的数目则会相对较多（边介数大）。GN 算法便是一种凝聚型的社区结构发现算法。该算法根据网络中社区内部高内聚、社区之间低内聚的特点，逐步去除社区之间的边，取得相对内聚的社区结构。如果一条边连接两个社区，那么两个社区节点之间的最短路径通过该边的次数就会最多，相应的边介数就会最大。如果删除该边，那么两个社区就会分割开。GN 算法就是基于以上思想反复计算当前网络的最短路径，计算每天边的边介数，删除边介数最大的边。最后在一定的临界条件下，算法停止，即可得到网络的社区结构。

2. 算法步骤

- (1) 计算每条边的边介数；
- (2) 删除边介数最大的边；
- (3) 重新计算网络中剩下的边的边介数；
- (4) 重复 (3) 和 (4) 步骤，直到网络中的任一顶点作为一个社区为止。

3. 算法缺点

- (1) 计算终止的临界值需要自行设定，即停止时的 ϵ 未知；
- (2) 在 m 条边和 n 个节点的情况下，时间复杂度为 $O(m^2 * n)$ ，比较耗时。
- (3) 最后共有多少个社区也是未知的。

III. Louvain 算法

1. 介绍

Louvain 算法是 Blondel 等人提出的用于大型网络中提取社区的方法。换句话说，Louvain 算法主要功能便是进行社区检测，其优化目标就是最大化整个社区网络的模块度。模块度也称模块化度量值，是目前常用的一种衡量网络社区结构强度的方法，最早由 Mark NewMan 提出。模块度定义为：

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

$$\delta(u, v) = \begin{cases} 1, & u == v \\ 0, & \text{else} \end{cases} \quad (2)$$

其中， A_{ij} 表示节点 i 和节点 j 之间边的权重，网络不是带权图时，所有边的权重可以看作是 1； $k_i = \sum_j A_{ij}$ 表示所有与节点 i 相连的边的权重之和（度数）； c_i 表示节点 i 所属的社区； $m = \frac{1}{2} \sum_{ij} A_{ij}$ 表示所有边的权重之和（边的数目）。根据公式我们可以看出，模块度值的大小主要取决于网络中节点的社区分配 c ，即网络的社区划分情况，可以用来定量的衡量网络社区划分质量，其值越接近 1，表示网络划分出的社区结构的强度越强，也就是划分质量越好。因此可以通过最大化模块度 Q 来获得最优的网络社区划分。

2. 算法原理

为了最大化公式中的 Q 值，Louvain 方法有两个迭代重复的阶段。

第一阶段，网络中的每个节点都分配给自己的社区。然后，计算每个节点 i 的模块度的变化，以便除去 i 所在社区并且将其移动到 i 的邻居社区。这个数值是通过两步来计算的：

- a. 删除 i 所在的原始社区；
- b. 插入 i 的邻居社区 j。

其中，步骤二的方程为

$$\Delta Q = \left[\frac{\sum in + 2k_{i,in}}{2m} - \left(\frac{\sum tot + k_i}{2m} \right)^2 \right] - \left[\frac{\sum in}{2m} - \left(\frac{\sum tot}{2m} \right)^2 - \left(\frac{k_i}{2m} \right)^2 \right] \quad (3)$$

$\sum in$ 是社区 c 中边的权重总和， $\sum tot$ 是社区 c 中节点相连的边的权重总和。对于每个节点 i，依次将该节点分配到每个邻居节点所在的社区，计算分配前后模块度 Q 的变化量，记录 Q 的变化量最大的邻居节点，将节点 i 分配到 Q 变化量最

大的邻居节点所在社区，否则节点 i 仍在原社区。重复上述步骤，直接 Q 的值不再变化，一旦达到局部模块度最大化，第一阶段就结束了。

第二阶段，对图进行压缩，将所有在同一个社区的节点压缩成一个新节点，社区内节点之间的边的权重转化为新节点的环的权重，社区间的边权重转化为新节点间的边权重。对每个节点重复此过程，直到整个图的模块度不再发生变化。

3. 优点

Louvain 算法在效率和效果上都比较不错。通常，我们认为该算法的时间复杂度为 $O(n \log n)$ 。并且，该算法可以发现层次性的社区结构，最大化整个社区网络的模块度。

IV. 实验结果

1. 社区发现的数据库代码

```
CALL algo.louvain.stream
    ('person', 'rel', {})
YIELD nodeId, community
RETURN algo.getNodeById(nodeId).name
    AS user, community
ORDER BY community;
```

2. 社区发现结果截图与分析

每个节点所属的社区详情请看附录 3。

(1) 统计信息

在我们的计算结果中，小数据集中共有 1887 个社区，大数据集有 376 个社区。

图 6 显示了小数据集中人数大于 1000 的社区人数分布情况。横坐标为社区编号，纵坐标为社区人数。可以看出社区 2 人数接近 35000，占据节点数的大多数，而剩余结点个数约等于或小于 5000。由此可初步分析得到该数据中大多数结点的连接都很紧密。

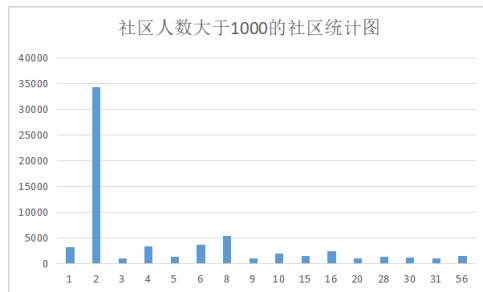


图 6 小数据集社区统计情况

Fig. 6 Community of Small Dataset

图 7 显示了大数据集中人数大于 100 的社区人数分布情况。横坐标为社区编号，纵坐标为社

区人数。可以看出社区 8 人数最多，占有 70000 余人，共有 9 个社区人数超过 10000. 与小数据集不同的是，该数据集中很多社区人数相当，可见有多个凝聚力比较强的社区，因此猜测效果图中社区能相对分散一些。

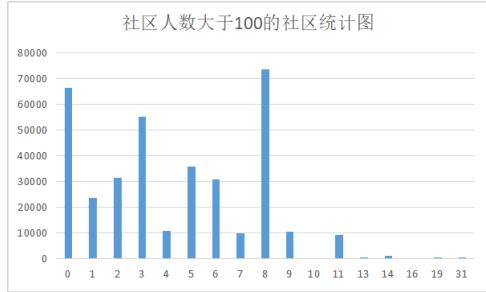


图 7 大数据集社区统计情况

Fig. 7 Community of Large Dataset

(2) 小数据集

图 8 展示了小数据集的所有点不同社区的划分情况。首先采用 Neo4j 对每个点进行了标记，再在 Gephi 中对不同的 Modularity Class 的值进行渲染，结点的大小按照度的大小来描绘，度越大的结点其面积越大；最后采用 Force Atlas2 布局算法得出最终的效果图。

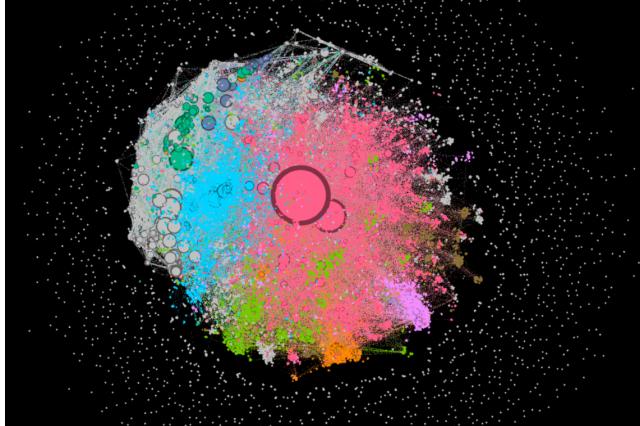


图 8 Louvain-small-Force Atlas2

Fig. 8 Louvain-small-Force Atlas2

对图 8 进行初步分析，可以得出以下结论：

- 红色社区占据了绝大多数结点（实际个数接近 35000）。且其中包括度最大的结点（实际为习近平）。
 - 所有的社区几乎为重叠状态，可以看出这些阶段虽然在不同的社区，但社区与社区之间联系都很紧密。
- 为了进一步分析，在图 8 中去掉了最大的社区和人数占比少于 2% 的社区，最后得出了社区

人数为 Top2-5 的效果图，如图 9。

虽然这几个社区里的结点之间也有联系，但和最大社区的连接紧密度相比还是较为低。

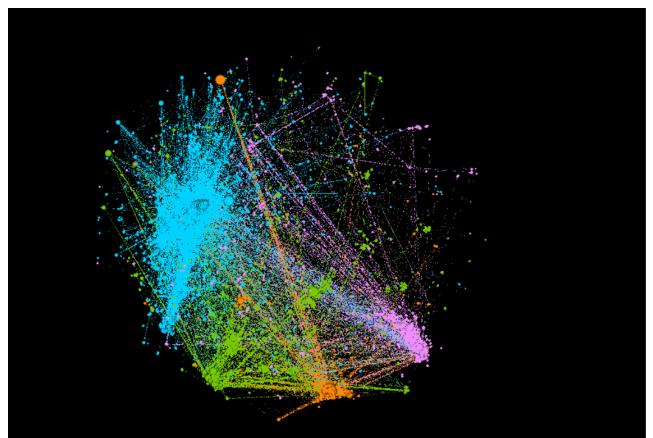


图 9 Louvain-several-Force Atlas2

Fig. 9 Louvain-several-Force Atlas2

图 10 为放大后社区边界节点的分布与连接情况。

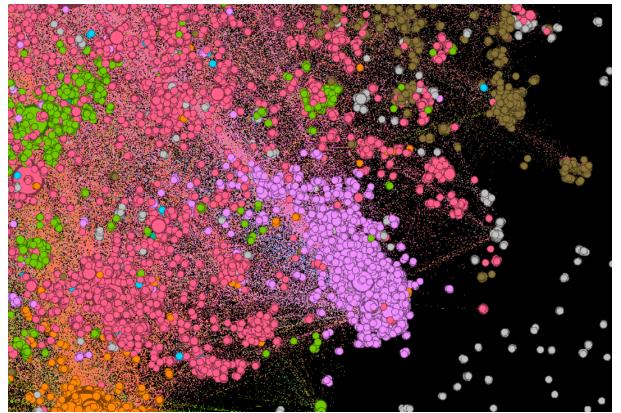


图 10 Louvain-part-Force Atlas2

Fig. 10 Louvain-part-Force Atlas2

(3) 大数据集

对于大数据集，由于硬件条件的限制，我们无法在 Gephi 上画出所有结点的社区情况。最终我们选取了社区人数在前十的社区，并对每个社区绘制了结点分布图，效果图如图 11 和 12 所示。

不同的社区结点分布状况完全不同。有的社区几乎所有的结点度都很大，有的社区一大部分结点很“聚集”而有一小部分很“离群”。

3.3.3 聚集系数

I. 概念介绍

聚集系数是表示一个图形中节点聚集程度的系数，证据显示，尤其是在特定的网络中，由于相对高密度连接点的关系，节点总是趋向于建立

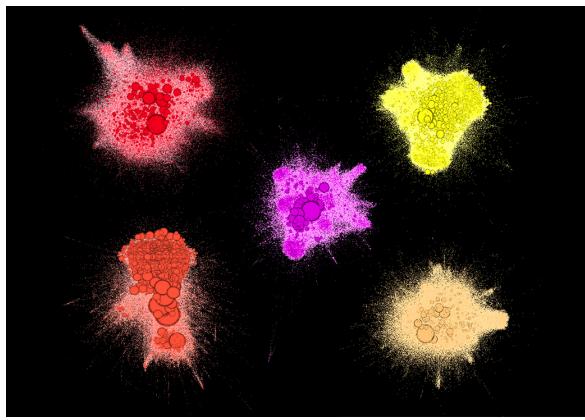


图 11 Community Detection1-Force Atlas2
Fig. 11 Community Detection1-Force Atlas2

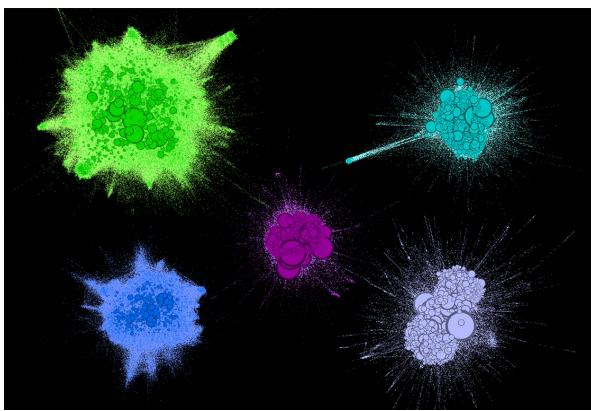


图 12 Community Detection2-Force Atlas2
Fig. 12 Community Detection2-Force Atlas2

一组严密的组织关系。在现实世界的网络，这种可能性往往比两个节点之间随机设立了一个连接的平均概率更大。

聚集系数主要分为三类：第一类，全局集聚。全局集聚系数是基于节点三元组的，一个三元组是由两条或三条无向边连接的三个节点。如果三元组是由两条边连接的三个节点，有时也称为2-星。全局集聚系数是所有三元组（包括开的和闭的）中封闭三元组的数目。第二类，局部集聚。图中一个节点的局部集聚系数表示了它的相邻节点形成一个完全图的紧密程度，Duncan J.Watts 和 Steven Strogatz 在 1998 年引入度量一个图是小世界网络的方法。第三类，平均集聚。整个网络的集聚系数由 Watts 和 Strogatz 定义为所有节点 n 的局部集聚系数的均值：如果一个图的平均集聚系数显著高于相同节点集生成的随机图，而且平均最短距离与相应随机生成的随机图相近，那么这个图被认为是小世界的。

II. 算法

1. 原理

节点 v 的聚集系数即为与 v 相邻的节点之间边的实际数与 v 相邻节点对的个数之比，表示为

$$CC_{v2} = \frac{n}{C_k^2} = \frac{2n}{k(k-1)}$$

其中 k 表示节点 v_2 的所有相邻的节点的个数，即 v_2 的邻居；n 表示节点 v_2 的所有相邻节点之间互相连接的边的个数。根据公式，我们可以看出难点在于统计一个节点的邻居节点之间有边的个数，于是，我们将问题转化为计算图中包含该节点的三角形个数。

2. 算法描述

- (1) 统计图中每个顶点的度数；
- (2) 对每个结点设一个集合 $S[i]$ ，然后将每个结点的邻居结点分别放入集合中；
- (3) 对图中每条边 (u,v) 进行遍历，每访问一条边 (u,v) ，求出 $S[v]$ 和 $S[u]$ 交集的大小，设为 D。设一个数组 A，初值为 0，第 i 个元素表示图中包含顶点 i 的三角形个数，对 $A[v]$ 和 $A[u]$ 累加 D；
- (4) 由于步骤 3 会将经过每个结点的三角形个数重复统计，得到原数量的两倍，因此最终结果需要将计算所得值个数除以 2 得到经过该结点的三角形个数。

3. 算法优点

结构简单，容易实现，能够有助于在复杂的网络中分析得到最终的结果。

4. 伪代码描述

聚集系数伪代码描述如算法 2 所示。

III. 实验结果

1. 聚集系数结果

对于小数据集，聚集系数为 1 的节点共有 51246 个，其中非零聚集系数最小的为 0.004259562，对应人名为“习近平”。

对于大数据集，聚集系数为 1 的节点共有 42755 个，其中非零聚集系数最小的为 0.018671969，对应人名为“宝贵”。

两个数据集中聚集系数为 1 且按照人名字字母序升序排列后前十的情况如下表 3.

2. Neo4j 的效果展示

聚集系数较小的人的社会关系网络，以“邓小平”为例，如图 13 所示。

聚集系数为 1 的人的社会关系网络，以“董志军”为例，如图 14 所示。

算法 2 CLUSTERING-COEFFICIENT

```

function CLUSTER((G, V))
2:   for  $i = 1 \rightarrow numNode$  do
3:      $degree[i] = degree$  of node $_i$ 
4:   end for
5:   for each  $(u, v)$  in edge do
6:     neighbor $[u].push(v)$ 
7:     neighbor $[v].push(u)$ 
8:   end for
9:   for each  $(u, v)$  in edge do
10:    triangle $[u], triangle[v] +=$ 
11:      size of (intersection between
12:        neighbor $[u]$  and neighbor $[v]$ )
13:   end for
14:   for each node  $- i$  do
15:     clustering $_coefficient[i] =$ 
16:       triangle $[i]/(degree[i] * (degree[i]))$ 
17:   end for
18: end function

```

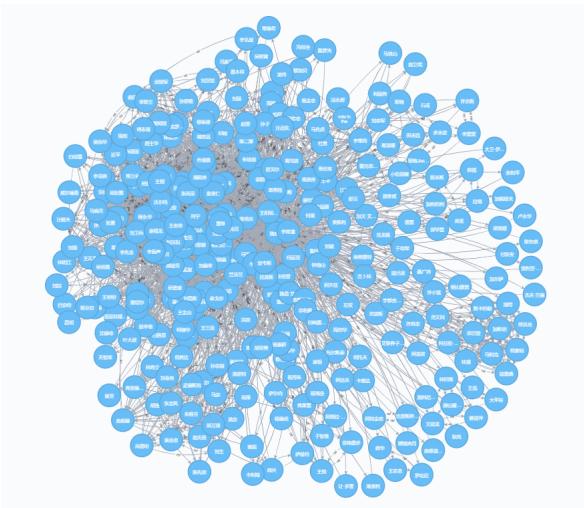


图 13 邓小平的社会关系网络

Fig. 13 Social Network of Deng Xiaoping

4 实验结果的进一步分析

4.1 度与聚集系数的关系

4.1.1 小数据集的结果分析

I. 相关系数

为了进一步探究度和聚集系数的关系，首先我们探索了二者的相关性，因此用 python 中 numpy 包的 corr 函数求了“所有节点的度”和“所有节点聚集系数”这两个序列的相关系数，计算结果为 -0.19060842400800726。

表 3 聚集系数结果

Table 3 Result of Clustering Coefficient

小数据集	大数据集
阿·克巴罗夫	阿艾克
阿·贝丘	阿邦正
阿·别嘉里耶娃	阿宝会
阿·德·阿尔达莫诺夫	阿宝叔
阿·卡斯纳	阿贝纳
阿·芒索尔	阿贝齐
阿巴·沫若	阿扁成
阿巴德	阿扁申
阿巴迪·泽姆	阿冰系
阿巴杜拉·戈麦达	阿波丸

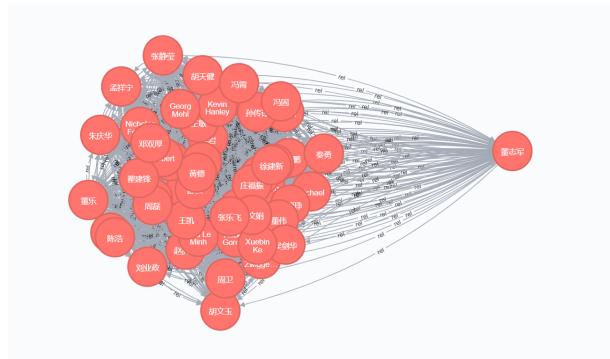


图 14 董志军的社会关系网络

Fig. 14 Social Network of Dong Zijun

这个结果表面节点的度和聚集系数之间呈负相关关系，但是线性相关性很弱。

II. 散点图表示

利用 python 中的 matplotlib 包，绘制了度和聚集系数的散点图。如图 15 所示。其中横坐标表示节点的度，纵坐标表示节点的聚集系数。

III. 现象与原因探究

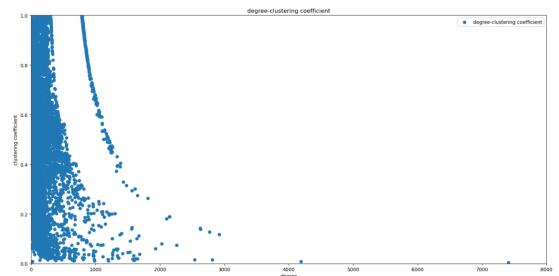


图 15 Small Degree clustering coefficient

Fig. 15 Small Degree clustering coefficient

1. 现象——中间有明显的“断层”

在 xlsx 中查看聚集系数大于 0.9 的结点的度

的情况，如图 16 所示。结果发现，在度为 319 之前的点的度分布很均匀，间隔很小，但是度突然从 319 变到 787，这是产生间隔的根本因素。



图 16 Node Clustering > 0.9

Fig. 16 Node Clustering > 0.9

经过分析，总结原因为以下两点：

(1) 小数据集共有 6 万余条新闻，这个数据量太小，在统计分析新闻任务关系时容易产生偶然误差；

(2) 新闻类型单一，该数据集全部为政治新闻，人物类型容易产生明显差异。一类人物是常见的政治人物如“习近平”、“李克强”等，他们出现的新闻较多，与他们相关的人也会非常多，其中不仅有政界其他领导人，还有每次报道的记者等，因此这类节点的度一般比较大。而另一类是一些不知名的人物，比如某个村的村长或某次会议的记者，他们可能只出现在一两条新闻里，且一般与这类人相关的人会比较少，因此他们的度也较小。因此人物类型明显的差异导致了节点度的分布不均匀，有一个明显的间隔。而在后文对大数据集的分析中可以看到，由于数据量更大，且新闻选区更随机更多样，因此度与聚集系数的关系图更均匀一些。

2. 现象二——靠近 Y 轴的节点数更多

在现实生活中，相较于政治界的大人物，新闻中出现不知名的人物的数量更多，而这些节点的度往往比较小，因此在上面度与聚集系数的散点图分布中靠近 Y 轴的节点更加密集。

3. 现象三——有很多聚集系数为 1 结点

在实验结果中，统计出聚集系数为 1 的节点共有 51246 个，总节点数为 80188，占比 64

(1) 数据量太小，很多人只出现在一个新闻里，而恰好这个新闻里的其他人也都是这种情况，因此这个新闻里的所有人的聚集系数都为 1。这

种新闻往往是发生概率比较小、报道次数很低的新闻；

(2) 很多新闻里的人数不超过 10，在这类新闻中的人的邻居更容易有关系，从而他们的聚集系数为 1 的概率更高。所有节点中有很多节点只有 2、3 个朋友。

4.1.2 大数据集的结果分析

I. 散点图表示

按照上述同样的方法，绘制了大数据集的度与聚集系数散点图，如图 17 所示。

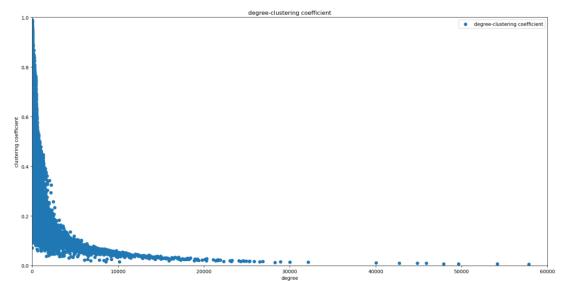


图 17 Big Degree clustering coefficient

Fig. 17 Big Degree clustering coefficient

II. 现象与原因探究

1. 现象一——度越大，聚集系数越小；度越小，聚集系数越大

在前文提到度与聚集系数满足如下关系： $cluster=2n/d*(d-1)$ 。因此我们画出了 $y=1/(x*(x-1))$ 的图像，如图 18 所示。可以看出散点图的函数趋势与该函数几乎是一致的。

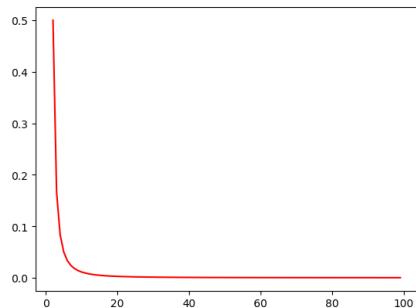


图 18 Function Graph

Fig. 18 Function Graph

对于形成这种趋势的原因，分析如下：当样本量足够多时，同一个 n （即邻居之间是朋友的边数）也就是分子相同时，不同的度的结点很多，

因此能够形成一条线的形状, 即一个 $y=n1/d*(d-1)$ 函数。然后多种 n 的情形下的“线条”的叠加, 就形成了上面散点图的情况。

2. 现象二——靠近 Y 轴的节点数更多

与小数据集类似, 度较小的结点占绝大多数, 因此靠近 Y 轴的部分更密集一些。

4.2 度与 pagerank 值的关系

由于大数据集和小数据集结果差不多, 因此以下使用小数据集的结果加以说明。

I. 相关系数

计算出所有节点度和 pagerank 值的相关系数为: 0.5550938262823185。

可以得出结论节点的度和 pagerank 值的正相关性很高。

II. 散点图

按照上述同样的方法, 绘制了小数据集的度与 pagerank 值的散点图, 如图 19 所示。

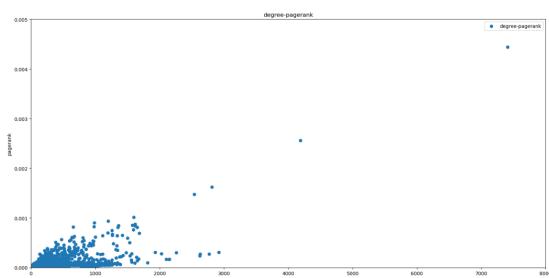


图 19 Small PageRank-Degreee
Fig. 19 Small PageRank-Degreee

III. 现象与原因探究

1. 现象一——度越大, pagerank 值越大。

2. 现象二——度越小, 节点分布越密集

这一点在上文已经分析到, 在新闻中出现的“小人物”的数量大于新闻中的“大人物”, 因此节点的分布更靠近 Y 轴。

3. 现象三——度较小时, 节点分布比较“凌乱”

度较小的时候, 其 pagerank 值没有很明显的规律, 这也反映出在新闻中出现次数少的那些“小人物”很难明确的确定出他们的排名关系。相反, 对于那些政治界的“大人物”, 他们的关联人物更多时, 相对的其 pagerank 排名也更高, 这与现实生活中的情况也相符。

4.3 聚集系数与 pagerank 值的关系

I. 相关系数

计算出所有节点的聚集系数与 pagerank 的相关系数, 结果为 -0.3642494799166566。可以得出结论聚集系数和 pagerank 值的有一定的负相关性。

II. 散点图

按照上述同样的方法, 绘制了小数据集的聚集系数与 pagerank 值的散点图, 如图 20 所示。

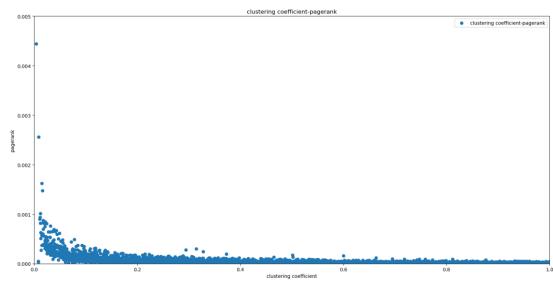


图 20 Small PageRank-clustering
Fig. 20 Small PageRank-clustering

III. 现象与原因探究

现象——聚集系数越大, pagerank 值越低

对于出现这种现象的原因, 我们总结为: “小人物”的朋友往往比较少, 他们的朋友之间常常因为工作等关系出现在一个新闻里, 因此小人物更容易形成较为聚集的“小团体”, 聚集系数往往较高; 但是“大人物”周围有关的人太多了, 而且可能是在不同场合下认识的, 比如: 一篇新闻里有习近平慰问某某村, 该村村长为 A, 另一篇新闻为习近平访问某国, 某国领导人为 B, 在现实生活中 A 和 B 很难有直接关系, 因此大人物的邻居之间有关系的概率较低, 所以聚集系数也较低。

4.4 PageRank 排名前 1000 人的度与聚集系数的关系

I. 散点图

为了进一步探究与验证, 我们选择绘出大数据集中 PageRank 排名前 1000 人的度与聚集系数的关系, 如图 21 所示。

II. 现象与原因探究

在图 21 所示中, 我们很明显地看出这一结果符合前面的分析, 具体表现如下:

- (1) 度: 离 Y 轴有一定距离, 这说明 pagerank

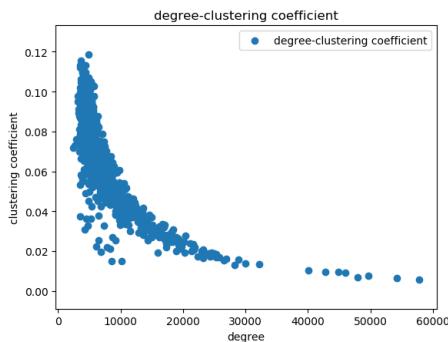


图 21 Big PageRank-Degree

Fig. 21 Big PageRank-Degree

前 1000 的节点的度都比较大，验证了我们前面分析的“度越大 PageRank 越大”的结论；

(2) 聚集系数：所有节点的聚集系数均在 0.12 以下，它们的聚集系数都很小，验证了我们前面分析的“PageRank 越大聚集系数越小”的结论；

(3) 度与聚集系数的关系图基本符合 $y=c/x^*(x-1)$ 的函数关系。

5 分工

分工情况如表 4

表 4 小组分工
Table 4 Doing Work

小组成员	工作内容
边关月	分词找人名、Tarjon 算法、图数据库计算连通分量、社区发现、论文撰写与排版、Web 前端、pagerank
范欣妍	点和边权值的统计、聚集系数计算、统计图表绘制、实验结果的统计与分析、论文撰写与排版、Gephi 画图
高艺境	网页制作、论文撰写与排版、结果整理

6 总结、展望与不足

6.1 总结与展望

在分析与利用大量新闻数据、建立社交网络图的过程中，主要实现了图的验证、图的统计两个基础功能和计算影响力、社区挖掘和计算节点的聚集系数三个高级功能。为了分析节点的个数，对数据进行了预处理。通过数据库，计算出最强前十个邻居。根据 Tarjan、Louvain 等算法的适用

性，得到了图的统计和社区的划分。总体来看，为了各项结果的准确性，进行了多种预处理，最终各项功能都可以顺利输出，完成度较高。

其中，这些功能不仅可以利用在新闻媒体领域，而且可以利用在打击违法犯罪上。设想如果社区网络图是一个犯罪嫌疑人关系图，我们便可以划分出不同的犯罪集团，并且得到与犯罪分子联系最前的前十个人。这不但有助于快速梳理犯罪关系，而且可以通过打击犯罪团伙对犯罪分子予以重创。

6.2 不足

在完成功能的过程中，我们也发现了许多不足之处。在处理数据时，由于数据量过大，有时没能选择适宜的解决方法，从而节约处理数据的时间。并且在预处理的过程中，仅考虑到三种除去无效节点的情况，而忽略了其他无效节点的处理，或许会对结果的准确度产生些许的影响。

7 参考文献

- [1] 复杂网络的分形研究方法综述 [J]. 王江涛, 杨建梅. 复杂系统与复杂性科学. 2013(04)
- [2] 郭玉泉. 复杂网络社区结构检测算法研究 [D]. 吉林大学, 2017
- [3] 复杂网络聚类方法 [J]. 杨博, 刘大有, 金弟, 马海宾. 软件学报. 2009(01)
- [4] 复杂网络社团结构发现方法研究 [D]. 何东晓. 吉林大学 2014
- [5] 基于模块度的复杂网络社团结构检测方法研究 [D]. 李君秋. 大连理工大学 2014
- [6] 姚莹. 基于遗传优化的复杂网络社区检测技术研究 [D]. 南京邮电大学, 2017
- [7] 复杂网络基础理论 [M]. 科学出版社, 郭世泽, 2012
- [8] 采用模糊层次聚类的社会网络重叠社区检测算法 [J]. 李刘强, 桂小林, 安健, 孙雨. 西安交通大学学报. 2015(02)
- [9] 宋道榜. 一种改进的 Greedy Louvain 方法及其应用 [D]. 兰州大学, 2018
- [10] Local modularity for community detection in complex networks [J]. Ju Xiang, Tao Hu, Yan Zhang, Ke Hu, Jian-Ming Li, Xiao-Ke Xu, Cui-Cui Liu, Shi Chen. Physica A: Statistical Mechanics and its Applications. 2016
- [11] Wavelets on graphs via spectral graph theory [J]. David K. Hammond, Pierre Vandergheynst, Rémi

Gribonval. Applied and Computational Harmonic Analysis . 2010

附录:

1. 所有结点的编号、名称、度的信息请看附件中“实验结果/大（小）数据集”中的“节点编号、结点名称和度.csv”
2. 所有节点的聚集系数请看附件中“实验结果/大（小）数据集”中的“聚集系数.xlsx”
3. 所有结点及其所在社区请看附件中“实验结果/大（小）数据集”中的“节点编号和所在社区.csv”
4. 每个社区人数统计请看附件中“实验结果/大（小）数据集”中的“社区及其人数.csv”
5. PageRank 的执行结果统计请看附件中“实验结果/大（小）数据集”中的“PageRank 前 20 名.csv”