K 个影响力重要节点发现实验分析报告

指导教师：	曹玖新
姓名：	陈根文
学号：	**235183**
日期：	**2024 年 12 月 25 日**

摘要

近年来，随着信息网络的迅速发展，社交网络逐渐成为社会关注的核心。识别在社交网络中具有显著影响力的关键节点已经成为该领域的研究热点。在社交网络中，发现这些具有影响力的关键节点对于深入分析网络结构、研究网络演化以及分析用户之间关系的变化具有至关重要的意义。

本实验报告首先详细介绍了 SNAP 数据集，并通过节点可视化展示了其结构。随后，文章介绍了 Degree、PageRank、HITS、K-cores 等算法，这些算法用于发现影响力重要节点。文章提出了一种改进算法，灵感来自于卷积操作，并考虑了邻居节点的传播作用。该算法的创新之处在于综合考虑节点本身的属性和邻居节点的影响力，与传统的度数相关的算法不同，这种方法更为灵活。在实际应用中，文章指出了节点属性聚合时需注意邻居节点数量的差异，强调了对属性进行归一化的必要性。此外，文章强调了在聚合领域节点信息时不能忽略节点本身的属性，并提出通过调整系数大小来调整对二阶属性的利用程度。最后，文章介绍了评估方法，并进行了 Degree、PageRank、HITS、K-cores 算法在数据集上的结果分析，同时对比了几种方法的效果。整体而言，该研究提供了一个综合考虑节点属性和邻居节点影响力的新型算法，为社交网络中影响力节点的发现提供了新的思路和方法。

关键词：社会网络；重要节点；影响力；用户排名；复杂网络

第一章 数据集分析

本次实验使用是由斯坦福大学(SNAP)提供的 Higgs Twitter Dataset 数据集，斯坦福网络分析平台(SNAP)是一个通用的网络分析和图挖掘库。它是用 c++编写的，很容易扩展到具有数亿个节点和数十亿条边的大规模网络。它有效地操作大型图，计算结构属性，生成规则和随机图，并支持节点和边上的属性。

该数据集包含四个有向图，分别是社交图、转发图、回复图和提及图。

表 1 数据集描述

文件名称	描述
social_network.edgelist	好友/关注者有向图
retweet_network.edgelist	转发关系有向加权图
reply_network.edgelist	回复关系有向加权图
mention_network.edgelist	提及关系有向加权图

实验统计的网络信息主要包含：用户数目、社交联系、社交网络密度等信息，具体统计数据如表 2 所示。可以看出，这些网络虽然节点数目庞大，但其网络密度都很稀疏，同时平均最短路径也较短，符合六度分割理论。

表 2 数据集属性统计

数据集	Social Network	Mention Network	Reply Network	Retweet Network
用户数目	456626	116408	38918	256491
社交联系	14855842	150818	32523	328132
社交网络密度	5.521×10^{-5}	3.168×10^{-5}	5.591×10^{-5}	1.872×10^{-5}
社交网络核数	66	11	4	10
100 个节点的平均最短路径	2.906	0.271	0.130	0.093

图 1 展示了 Social Network 数据集中点的度数分布。可以看出，当对度数和点个数取对数后，图像的前段呈直线状。因此，可以认为该网络的度分布服从幂律分布，是一个经典的社交网络。

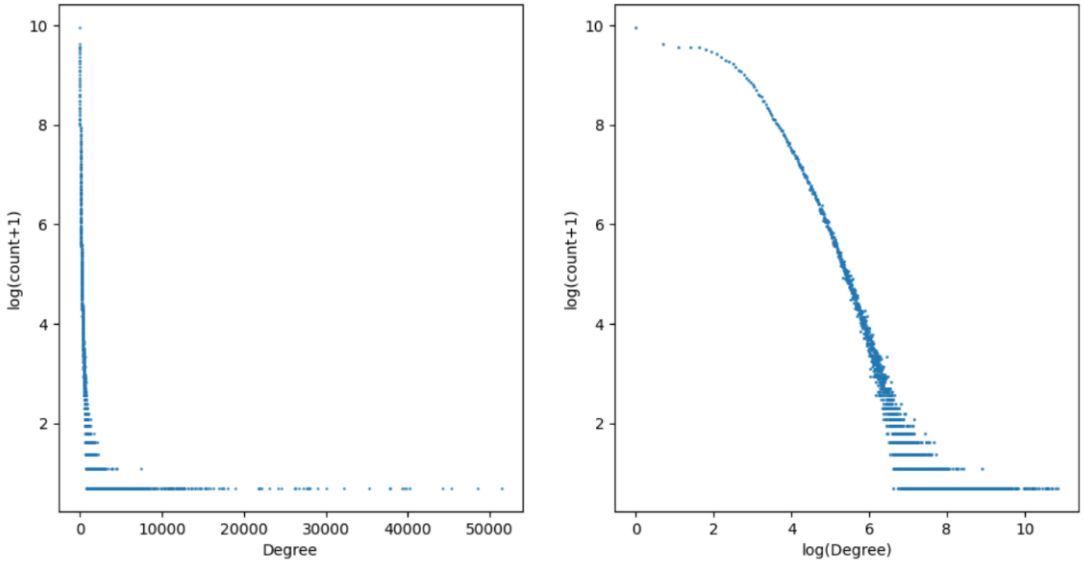


图 1 度分布散点图

除此之外，下表展示了数据集的图基本属性和结构特点。

表 3 社交图基本属性和结构特点分析

社交图 Social Network statistics	
总节点数	456626
总边数	14855842
最大弱联通的节点数	456290(0.999)
最大弱联通的边数	14855466(1.000)
最大强联通的节点数	360210(0.789)
最大强联通的边数	14102605(0.949)
平均聚类系数	0.1887
三角个数	83023401
三角形百分率(全局聚类系数)	0.002901
直径	9
90%有效直径	3.7

图 2 展示了随机抽取 3000 个用户节点的用户关注、提及、评论、转发网络结构图。从图中可以观察到，不同用户的影响力大小不同。

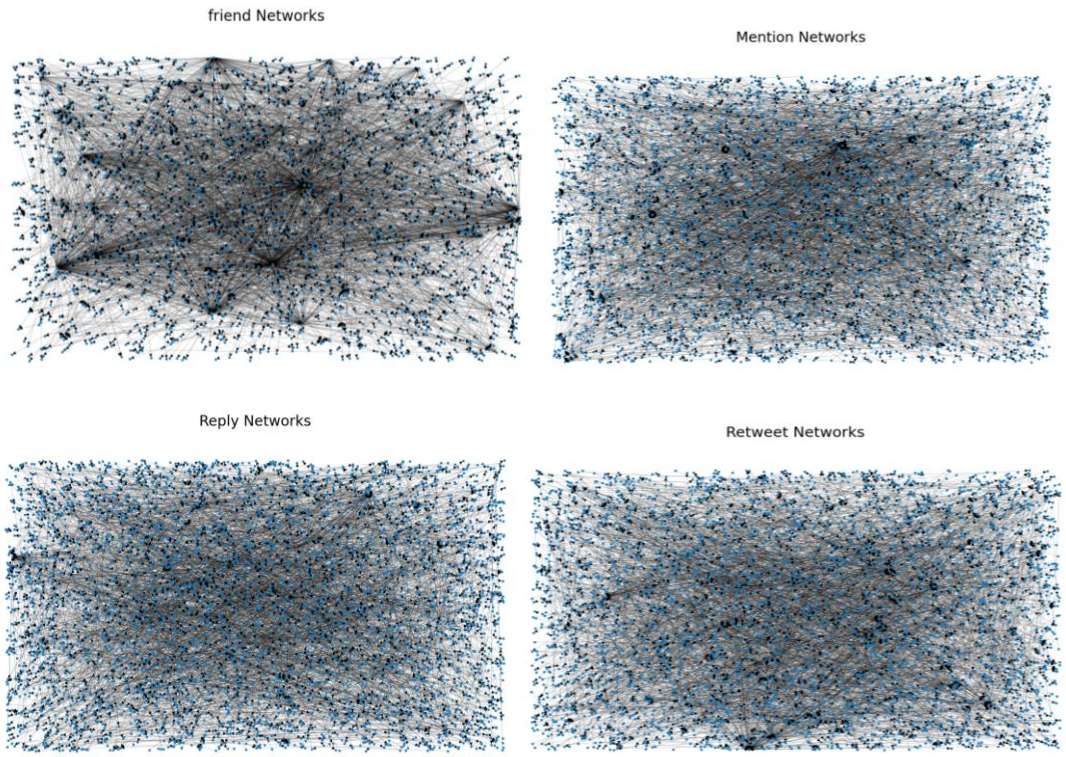


图 1 网络结构图

除此之外，我们页在这里选取了数据中的部分节点进行了可视化，可以较直观地观察到网络的局部特征，如图 3 所示，网络中存在一些更具“影响力” 的节点。

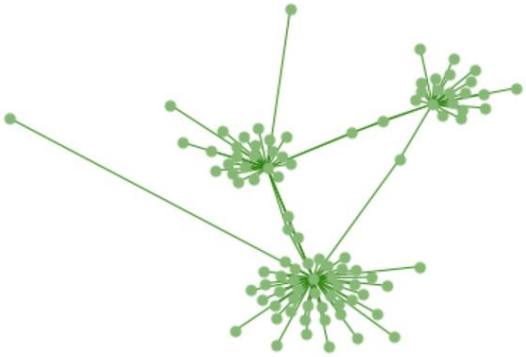


图 3 twitter 数据集社交图部分节点可视化

接下来对于网络中的节点联系性质进行分析，如图 4 所示节点中的有向边描述了关注或者交互的操作，对于数据集的说明中并没有详细解释交互的细节，但是我们从关注的角度去考虑，往往是被关注者对于关注者产生影响，在生活中人们关注明星，明星发布了信息进行带货，那么人们会跟风进行购买；而粉

丝本身并没有被明星关注，所以粉丝做出什么行为并不会反过来对于明星产生影响，所以在实验中，对于网络连接的指向进行了反转，即边是从被关注者指向关注者，与现实情况中的影响力方向一致。

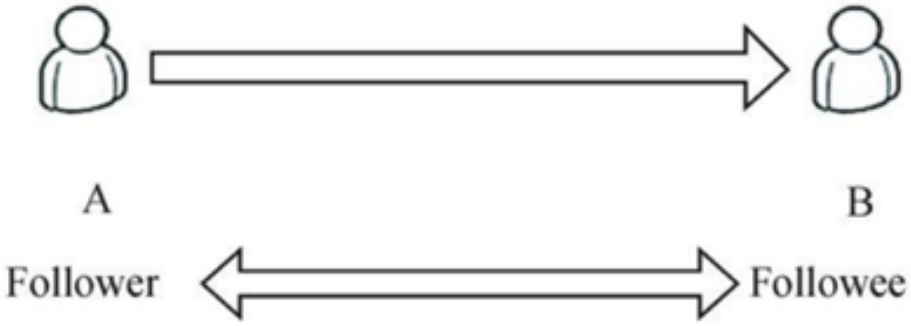


图 4 社交网络中用户关系描述图

第二章 算法介绍

1. Degree

在网络中，度数是最简单但往往非常有效的统计指标，对于网络图来说，与这个点相连的边的数量即为这个点的度数[错误!未找到引用源。]。如果是有向图，度数则可以细分为入度、出度、总度数三种，分别指指向该节点的边的数量，从这个节点出发的边的数量以及总的数量。

1.1 算法思想

Degree 算法的核心思想是基于节点的度来评估节点在网络中的重要性。节点的度是指与其直接相连的边的数量，用于衡量节点的连接程度。在一个图中，节点的度越高，代表它与其他节点有更多的直接连接，因此在网络中的地位也就越重要。这是基于节点度与节点重要性之间的正相关关系。

1.2 算法描述

（1）**节点度的定义：**对于一个图G, 节点v的度（degree）表示为$d(v)$。在有向图中，可以分为入度$d_{in}(v)$和出度$d_{out}(v)$。

$$d(v) = d_{in}(v) + d_{out}(v)$$

（2）**节点重要性的评估：**节点的度越高，其在网络中的重要性也就越高。这可以表示为：

$$Importance(v) \propto d(v)$$

其中∝表示正比关系

（3）**排序节点：**将所有节点按照度进行降序排列：

$$Sort(Nodes) = [v_1, v_2, v_3, \ldots., v_n]$$

其中n表示节点总数

（4）**选择关键节点：**从排序后的节点列表中选择前 K 个节点作为关键节点：

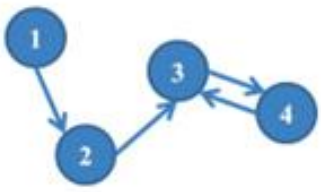
$$KeyNodes = [v_1, v_2, v_3, \ldots., v_k]$$

2. PageRank

2.1 算法思想

PageRank 的核心思想是基于有向图上的随机游走模型，这是一个一阶马尔可夫链。该模型描述了一个随机游走者如何沿着图的边随机移动，从一个节点访问到另一个节点[错误!未找到引用源。]。在满足某些条件的前提下，这个随机游走过程最终会收敛到一个平稳分布。在这个平稳分布中，每个节点被访问的概率即为其 PageRank 值，这个值可以被解释为节点的重要性。值得注意的是，PageRank 是递归定义的，这意味着一个页面的 PageRank 值部分地取决于链接到它的其他页面的 PageRank 值。因此，计算 PageRank 值通常需要迭代方法。

PR(1)	PR(2)	PR(3)	Sum
1/3	1/3	1/3	1
0	1/3	1/3	2/3
0	0	1/3	1/3
0	0	0	0



PR(1)	PR(2)	PR(3)	PR(4)	Sum
1/4	1/4	1/4	1/4	1
0	1/4	1/2	1/4	1
0	0	1/4	1/2	3/4
0	0	1/2	1/4	3/4

图 5 PageRank 算法

PageRank 一般定义的想法是在基本定义的基础上导入平滑项。给定一个含有n个结点的$v_i, i = 1,2, \ldots, n$的任意有向图，假设考虑一个在图上随机游走模型，即一阶马尔可夫链，其转移矩阵是 M ，从一个结点到其连出的所有结点的转移概率相等。 这个马尔可夫链未必具有平稳分布。 假设考虑另一个完全随机游走

的模型，其转移矩阵的元素全部为$1/n$， 也就是说从任意一个结点到任意一个结点的转移概率都是 $1/n$。 两个转移矩阵的线性组合又构成一个新的转移矩阵，在其上可以定义一个新的马尔可夫链。 容易证明这个马尔可夫链一定具有平稳分布，且平稳分布满足

$$R = \left(dM + \frac{1-d}{n}E\right)R = dMR + \frac{1-d}{n}ER$$

可以由上式写出每个结点的 PageRank，这是一般 PageRank 的定义。

$$PR(v_i) = d(\sum_{v_j \in M(v_i)} \frac{PR(v_j)}{L(v_j)}) + \frac{1-d}{n}), i = 1,2,\dots,n$$

2.2 算法描述

接下来介绍 PageRank 的迭代算法。

输入：含有n个结点的有向图，转移矩阵M，阻尼因子d，初始向量R_0；

输出：有向图的 PageRank 向量R。

（1）令$t = 0$;

（2）计算$R = dMR + \frac{1-d}{n}$ER;

（3）如果R_{t+1}与R_t充分接近，令$R_t = R_{t+1}$停止迭代。

（4） 否则$t = t + 1$,执行（2）

3. HITS

3.1 算法思想

HITS 算法和 PageRank 算法在原理上有些类似。都既考虑网页本身的链接数，也考虑所链接网页的权威性[错误!未找到引用源。]。其主要思想是：每个网页的重要性由两个指标刻画，权威值（Authority）与枢纽值（Hub）。一个权威值高的网页会被很多网页指向（如微博大 V）。一个枢纽值高的网页会指向很多网页（如大型目录网页），见下图。

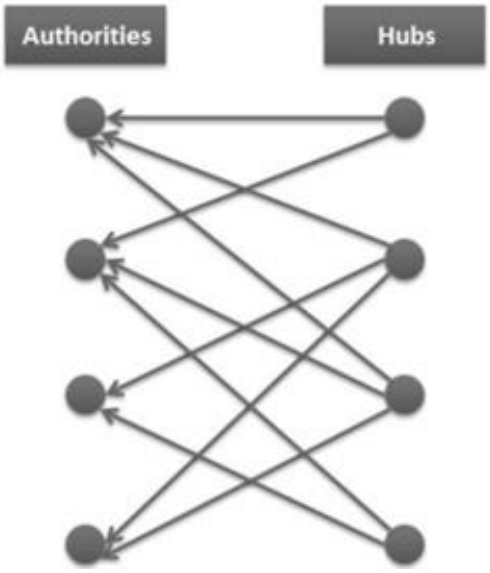


图 6 HITS 算法

HITS 采用互相增强原理，并基于两个假设：一个高质量的 authority 页面会被很多高质量的 hub 页面所指向；一个高质量的 hub 页面会指向很多高质量的 authority 页面。利用上述两个基本假设及相互增强关系等原则进行多轮迭代计算，每轮迭代计算更新每个页面的两个权值，直到权值稳定不再发生明显的变化为止。其中每个网页的 Authority 值和 Hub 值的计算公式如下：

$$auth(p) = \sum_{q \in p_{to}} hub(q)$$

$$hub(p) = \sum_{q \in p_{from}} auth(q)$$

3.2 算法描述

算法具体过程大致可分为以下五个步骤：

1) 将各节点的 auth 值和 hub 值均设为 1。
2) 利用公式计算并更新每个节点的 auth 值。
3) 利用更新的 auth 值，计算并更新节点的 hub 值。
4) 将 auth 值和 hub 值归一化。
5) 重复 2~4，直至收敛或达到停止迭代条件。

4. K-cores

4.1 算法思想

K-核算法的主要思想基于核心度，即位于网络核心部分的节点的影响力要高于边缘节点[错误!未找到引用源.]。算法分解流程如下：设网络的孤立节点的核心度为 0，首先把孤立节点从网络中去除，接着进行 K-核分解。

第一步，把所有核心度为 1 的节点从网络中移除，之后继续移除剩余度小于等于 1 的节点，直到网络中所有剩余节点的剩余度都大于 1 为止。在该步中所有移除的节点为 1-核节点，其核心度都等于 1。

第二步，首先移除剩余度等于 2 的节点，然后继续移除当前剩余度不大于 2 的节点，直到网络中所有剩余节点的剩余度都大于 2 为止。第二步中所有删除的节点为 2-核节点，其核心度都等于 2。

以此类推，直到网络中所有的节点都被移除。显然，较大核心度的节点意味着位于网络更中心的位置，有更大的影响力。

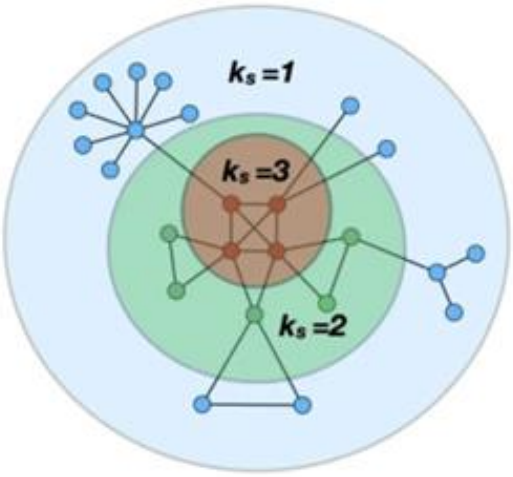


图 7 K-cores 算法

4.2 算法描述

（1）**初始化：**将网络中每个节点的度设为其初始度。

（2）**迭代删除：**

- 找到网络中度最小的节点（度最小的节点是指在当前迭代中，度数最小的节点）。
- 移除该节点以及与其相连的边。

- 更新网络中剩余节点的度。

（3）**迭代终止条件：**重复上述迭代删除步骤，直到网络中不存在度小于 K 的节点。

（4）**K-核提取：**从剩余网络中提取 K-核。即，得到一个子图，其中每个节点的度都至少为 K。

（5）**识别关键节点：**K-核中的节点即为关键节点，因为它们的度至少为 K，对网络的连接性和稳定性有着显著的贡献。

5. 改进算法

因为在很多情况下度数往往便捷且有效，所以受到卷积操作的启发，在本实验中提出了一种算法，该算法考虑了邻居节点的传播作用。在现实世界中，设想这样的情况，一些明星往往具有非常大体量的粉丝，如果这些粉丝中产生了一些自发的粉丝领袖，一旦明星发表了某些意见，那么粉丝领袖的跟随转发，会引发更多的转发和跟随[错误!未找到引用源。]。所以对于每个节点而言，其本身的节点属性固然重要，但如果其相连的节点“质量”不高或或是在整个网络中重要性较低，那么其影响力就相较于那些本身影响力大且其相邻的节点影响力也大的节点弱，并且此算法不一定要与度数绑定，如果有更好的节点影响力描述属性，也可以进行扩展。算法的公式展示为：

$$conv(p) = attr(p) + \sum_{q \in neighbor(p)} attr(q)/outdrgree(p)$$

在对于节点属性进行聚合的时候，需要注意每个节点连接的邻居节点数量是不同的，要对于属性进行归一化，不然邻居节点更多的节点会获得很大的影响力。需要注意的是，在聚合领域节点信息的时候，不能忽略了节点本身的属性，在公式中，对于聚合的属性设置了一个系数，通过调整系数大小，可以调整对于二阶属性的利用程度。

第三章 评估方法

独立级联的基本假设是节点 u 试图激活其邻接节点 v 的行为能否成功是一个概率为 $p(u,v)$的事件。且一个处于非活跃状态的节点被刚进入活跃状态的邻居节点激活的概率独立于之前曾尝试过激活该节点的邻居的活动。此外该模型

还做出了这样的假设：网络中任意的节点 u 只有一次机会尝试激活其邻居节点 v，无论能否成功，在以后的时刻中，u 本身虽然仍保持活跃状态，但它已经不再具备影响力，这一类节点称为无影响力的活跃节点。

算法如下：

1. 初始的活跃节点集合 A 。

2. 在 t 时刻，新近被激活的节点 u 对它的邻接节点 v 产生影响，成功的概率为 $p(u,v)$。若 v 有多个邻居节点都是新近被激活的节点，那么这些节点将以任意顺序尝试激活节点 v 。

3. 如果节点 v 被激活成功，那么在$t+1$时刻，节点 v 转为活跃状态，将对其邻接非活跃节点产生影响；否则，节点 v 在 $t+1$ 时刻状态不发生变化。

4. 该过程不断进行重复，直到网络中不存在有影响力的活跃节点时，传播过程结束。

线性阈值模型将图中的每个点描述为两种可能状态：不活跃（inactive）和活跃（active）。不活跃状态表示该个体还没有接受对应实体，而活跃状态表示该个体已经接受对应的实体。节点从不活跃状态变为活跃状态表示该节点接受了对应实体，也称之为被激活。

激活过程具体来讲：初始时同样只有种子节点被激活，随后每个时间步所有未被激活的节点都根据其已被激活的入邻居到它的线性加权和是否达到阈值来决定是否激活该节点。与独立级联不同的是，每个被激活都节点都有多次机会去激活自己尚未被激活的邻居节点。

那么算法步骤可以被分解为：

1. 初始化种子节点，然后激活种子节点作为初始激活集合。

2. 寻找尚未被激活且有激活节点作为邻居节点的节点，放入备选节点集合。

3. 依次对备选节点集合中的节点计算激活概率，然后尝试激活，被激活的节点将被放进激活集合。

4. 重复 23 步骤，直至没有可激活的节点。

本实验采用独立级联加线性阈值模型，阈值默认设置为 0.5。

第四章 实验分析

首先我们对 Degree、PageRank、HITS 和 K-cores 算法在给定数据集上运行。

1. Degree 算法

使用 Degree 算法得出的前一百个重要节点排序和各节点的度数中心性（由于数据太多，这里未完全展示）如图 8。

重要节点排序：

```
['1503', '206', '88', '138', '1062', '677', '352', '220', '317', '301', '383', '8', '1274',
'15', '6948', '7533', '3549', '1988', '6', '2055', '979', '205', '2417', '213', '335', '
960', '465', '3419', '396', '965', '4526', '327', '2670', '1276', '4259', '1', '1852', '
16452', '689', '323', '370', '43', '1267', '330', '2662', '407', '503', '2353', '331', '
408', '519', '42', '891', '93', '5226', '322', '16783', '2347', '7932', '349', '310', '
7551', '2138', '356', '4214', '59', '19153', '3158', '354', '2244', '78', '14648', '2179', '
3393', '857', '308', '816', '818', '463', '1247', '489', '223', '14590', '211', '10', '
6166', '216', '14549', '1881', '19', '1971', '750', '759', '1292', '81', '35481', '30230', '
6951', '14572', '4']
```

各节点的度数中心性：

```
[('1503', 0.1125343553243909), ('206', 0.106060771196824527), ('88', 0.09909444292362442), ('
138', 0.09677087325485902), ('1062', 0.0879189707090063), ('677', 0.0872137968792773), ('
352', 0.08657870243635368), ('220', 0.08592170818505337), ('317', 0.08289077470572133), ('
301', 0.08273747604708459), ('383', 0.07710484533260334), ('8', 0.07034218450588557), ('
1274', 0.0658154941144265), ('15', 0.0632137968792773), ('6948', 0.061501231864221184), ('
7533', 0.06101724609909663), ('3549', 0.060373391732822336), ('1988', 0.059707637558171364),
 ('6', 0.059401040240897894), ('2055', 0.058457158499863125), ('979', 0.0575373665480427),
('205', 0.057359978100191625), ('2417', 0.05356255132767588), ('213', 0.05280481795784287),
('335', 0.05062578702436354), ('960', 0.048350396934026825), ('465', 0.048052559540104026),
('3419', 0.04795182042157131), ('396', 0.04770435258691486), ('965', 0.041349028196003286),
('4526', 0.03930796605529702), ('327', 0.03814946619217082), ('2670', 0.03813194634546948),
('1276', 0.03746181220914317), ('4259', 0.03691212701888858), ('1', 0.03569011771147002), ('
1852', 0.03471119627703258), ('16452', 0.034286339994525045), ('689', 0.03361401587736107),
('323', 0.03341472762113332), ('370', 0.03313660005474952), ('43', 0.032424856282507526), ('
1267', 0.03168683274021352), ('330', 0.03150506433068711), ('2662', 0.0312006569942513), ('
407', 0.030105666575417463), ('503', 0.0300465370928004 4), ('2353', 0.029691760197098276),
```

图 8 Degree 算法前 100 重要节点排序和各节点的度数中心性

2. PageRank 算法

使用 PageRank 算法得出的各节点的 PageRank Score（由于数据太多，这里未完全展示）如图 9。

```
Node:1   ,  PageRank Score:0.01738357226025648
Node:88   ,  PageRank Score:0.002714551104752392
Node:1503   ,  PageRank Score:0.0025635933389870254
Node:138   ,  PageRank Score:0.0025262879766397134
Node:220   ,  PageRank Score:0.00246261093411527
Node:317   ,  PageRank Score:0.0023524366590045886
Node:206   ,  PageRank Score:0.0023372778153830757
Node:352   ,  PageRank Score:0.0023105904550850716
Node:677   ,  PageRank Score:0.002223956049592166
Node:301   ,  PageRank Score:0.001831418110137759
Node:1062   ,  PageRank Score:0.0018010666088643969
Node:6948   ,  PageRank Score:0.001647994894684677
Node:15   ,  PageRank Score:0.0016030488765817763
Node:383   ,  PageRank Score:0.0015316265662851539
Node:3549   ,  PageRank Score:0.0014873974561387742
Node:2055   ,  PageRank Score:0.0014762269491114665
Node:1274   ,  PageRank Score:0.0014659704129185956
Node:8   ,  PageRank Score:0.0014618359514216972
Node:979   ,  PageRank Score:0.0013713809367035448
Node:7533   ,  PageRank Score:0.0013473793207372567
Node:3419   ,  PageRank Score:0.001254927143180797
Node:205   ,  PageRank Score:0.001252972291846312
Node:6   ,  PageRank Score:0.0012233454060653218
```

图 9 PageRank 算法得出的各节点的 PageRank Score

3. HITS 算法

使用 HITS 算法得出的各节点的 Authority Score（由于数据太多，这里未完全展示）如图 10。

```
Node:1503   ,  Authority Score:0.0005568103177331815
Node:206   ,  Authority Score:0.0005311640437267166
Node:1062   ,  Authority Score:0.0004663517897160397
Node:138   ,  Authority Score:0.0004480514860843346 6
Node:301   ,  Authority Score:0.0004358765759135631
Node:383   ,  Authority Score:0.0004308204680619396
Node:88   ,  Authority Score:0.0003857071549339739
Node:1274   ,  Authority Score:0.0003747447195558527
Node:677   ,  Authority Score:0.0003678998163621042 6
Node:8   ,  Authority Score:0.0003574662125168383
Node:1988   ,  Authority Score:0.0003415835761604256 4
Node:205   ,  Authority Score:0.0003351820984072035 6
Node:352   ,  Authority Score:0.0003298206795199487 6
Node:2417   ,  Authority Score:0.0003263437490072752
Node:213   ,  Authority Score:0.0003242749710038884
Node:2055   ,  Authority Score:0.0003081789865543755
Node:220   ,  Authority Score:0.0003035231731133600 6
Node:335   ,  Authority Score:0.0003025632405754516
Node:15   ,  Authority Score:0.0002945229361291899
Node:317   ,  Authority Score:0.0002939375297153602
Node:396   ,  Authority Score:0.0002816718014711902
Node:6   ,  Authority Score:0.0002809091673766173
```

图 10 HITS 算法得出的各节点的 Authority Score

4. K-cores 算法

使用 K-cores 算法得出的前一百个重要节点排序和各节点的 K-cores Rank（由于数据太多，这里未完全展示）如图 11。

重要节点排序：

```
[('2', 125), ('4', 125), ('5', 125), ('6', 125), ('7', 125), ('8', 125), ('10',
125), ('15', 125), ('19', 125), ('20', 125), ('21', 125), ('26', 125), ('27', 125),
 ('290', 125), ('31', 125), ('32', 125), ('138', 125), ('34', 125), ('291', 125),
('39', 125), ('297', 125), ('299', 125), ('44', 125), ('301', 125), ('188', 125),
('253', 125), ('189', 125), ('211', 125), ('306', 125), ('308', 125), ('316', 125),
 ('220', 125), ('317', 125), ('322', 125), ('323', 125), ('325', 125), ('71', 125),
 ('327', 125), ('329', 125), ('330', 125), ('331', 125), ('1042', 125), ('359',
125), ('987', 125), ('994', 125), ('995', 125), ('998', 125), ('2662', 125), ('
721', 125), ('2055', 125), ('369', 125), ('1445', 125), ('206', 125), ('640', 125),
 ('373', 125), ('374', 125), ('376', 125), ('378', 125), ('920', 125), ('381',
125), ('1035', 125), ('80', 125), ('2670', 125), ('42', 125), ('393', 125), ('
1503', 125), ('81', 125), ('402', 125), ('901', 125), ('1357', 125), ('750', 125),
('1478', 125), ('759', 125), ('1101', 125), ('9646', 125), ('1104', 125), ('1115',
125), ('339', 125), ('891', 125), ('1959', 125), ('213', 125), ('459', 125), ('
5980', 125), ('57', 125), ('88', 125), ('3588', 125), ('2515', 125), ('1170', 125),
 ('63', 125), ('484', 125), ('161', 125), ('492', 125), ('677', 125), ('1982',
125), ('816', 125), ('1286', 125), ('1884', 125), ('527', 125), ('5231', 125), ('
355', 125)]
```

各节点的K-coresRank值：

```
[('2', 125), ('4', 125), ('5', 125), ('6', 125), ('7', 125), ('8', 125), ('10',
125), ('15', 125), ('19', 125), ('20', 125), ('21', 125), ('26', 125), ('27', 125),
 ('290', 125), ('31', 125), ('32', 125), ('138', 125), ('34', 125), ('291', 125),
('39', 125), ('297', 125), ('299', 125), ('44', 125), ('301', 125), ('188', 125),
('253', 125), ('189', 125), ('211', 125), ('306', 125), ('308', 125), ('316', 125),
 ('220', 125), ('317', 125), ('322', 125), ('323', 125), ('325', 125), ('71', 125),
 ('327', 125), ('329', 125), ('330', 125), ('331', 125), ('1042', 125), ('359',
125), ('987', 125), ('994', 125), ('995', 125), ('998', 125), ('2662', 125), ('
721', 125), ('2055', 125), ('369', 125), ('1445', 125), ('206', 125), ('640', 125),
 ('373', 125), ('374', 125), ('376', 125), ('378', 125), ('920', 125), ('381',
125), ('1035', 125), ('80', 125), ('2670', 125), ('42', 125), ('393', 125), ('
1503', 125), ('81', 125), ('402', 125), ('901', 125), ('1357', 125), ('750', 125),
```

图 11 K-cores 算法前 100 重要节点排序和各节点的 K-cores Rank

5.对比分析

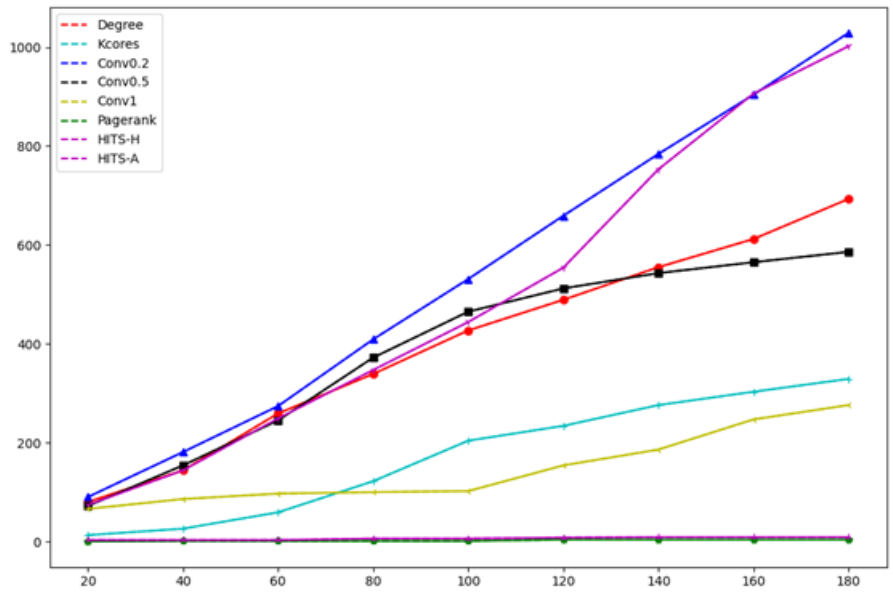


图 12 种子节点集较小时实验结果

上图所示是实验展示的是在评估中迭代轮数 k=2 的结果，横坐标表示选取影响力排名最高的种子节点集合的大小，从前 20 直到 180 个，纵坐标则表示影响的节点范围。从图中最好的算法进行分析，最好的是权重系数为 0.2 的聚合算法，这一算法聚合了领域的信息，又极大的保留了节点本身的信息，一直能够保持很好的影响力水平，其次是使用 HITS-H 的算法，这个算法比较注重于从节点出发的边的数量和质量。接下来是使用 0.5 的聚合算法和度数本身，这两个算法性能相近。再往下是使用 K 核算法和权重系数为 1 的聚合算法，因为本数据集较大，同样的核数会有很多的并列，所以用核数并不是很好描述节点的影响力程度，而权重系数为 1 的聚合算法，因为过于注重节点邻居的重要性而稀释了节点本身的影响力，导致算法效果不佳。最后是 HITS-A 和 PageRank，HITS-A 较为关注指向目标节点的连边的质量，那么在线性阈值和独立级联模型的评估方式下，很难取得较好的效果。关于 PageRank 算法，取得的效果十分低，认为是实验初期进行的有向边反转，使得算法并不适配，关于数据集建模与现实世界影响力方向以及算法的适配问题，还有待后续研究。

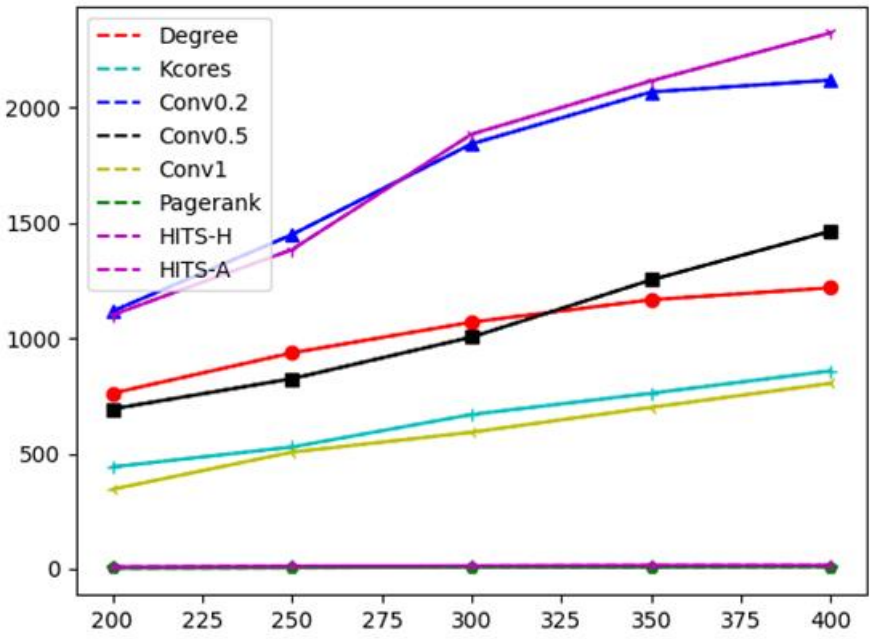


图 13 种子节点集较大时实验结果

在种子节点集较大的情况下，可以看到加权系数为 0.5 的聚合算法逐渐超过了度数本身，取得了较好的效果，但是 HITS 算法的效果也逐渐超过了系数为 0.2 的聚合算法，取得了最优，所以算法还需要进一步的改进。但是在这里 PageRank 算法取得的效果显然太低，所以可能是实验中设置出现了问题，最大的可能还是为了符合影响力传播的自然效果而直接反转有效图是不太正确的操作，还需要进一步的考虑。

总结

本实验报告全面介绍了 SNAP 数据集，并通过节点可视化展示了其网络结构。随后，探讨了 Degree、PageRank、HITS、K-cores 等经典算法在发现影响力关键节点方面的应用。本文也提出了一项改进算法，灵感源自度数在许多情境中的便捷和有效性，并在实验中结合卷积操作，考虑了邻居节点的传播作用。

我们以明星与粉丝关系为例，说明了在社交网络中一些节点可能拥有庞大的影响力，尤其是当这些节点的邻居中涌现自发的领袖时。本文强调了节点自身属性和邻居节点“质量”对影响力的共同作用，并强调该算法不仅仅限于度数，也可以根据不同的节点影响力描述属性进行扩展。

在节点属性聚合方面，本文强调了需要注意每个节点连接的邻居节点数量不同，必须对属性进行归一化，以避免邻居节点更多的节点获得过大影响力。此外，在聚合领域节点信息时，本文提到不能忽略节点本身的属性，通过设置属性聚合公式中的系数，可以调整对二阶属性的利用程度。

最后，本文介绍了评估方法，并对 Degree、PageRank、HITS、K-cores 算法在数据集上的结果进行了深入分析，同时对比了几种方法的效果。整体而言，该研究提供了一个综合考虑节点属性和邻居节点影响力的新型算法，为社交网络中影响力节点的发现提供了新的思路和方法。