Web 信息处理与应用实验二程序说明——邹易澄（PB14011009）

社区发现算法实现与比较

【实验内容】

1. 实现 spectral clustering 等几个社区发现算法。
2. 比较实验结果，并进行讨论，将结果可视化。

【实验环境】

编程语言：Matlab, C++

编程环境：Matlab R2015b, DevC++

运行环境：Win8.1, Matlab R2015b, JDK1.8

使用工具：Matlab BGL tools, Gephi0.9.1

【实验步骤及方法】

算法一：girvan-newman

此算法采用了较为朴素的思想，它是通过边来发现社区的，即去除社区之间连接的边，留下的就是社区。此算法的目的就是发现网络中中介中心性(betweenness)最大的边，并将其去除，直至留下 K 个社区。较先去除的边，中心性较低，而中介中心性则较大。

算法流程：

（1）计算网络中连通分量的个数 n；

（2）若 n 大于等于社区数目 k，停止；

（3）计算网络中所有边的 betweenness；

（4）去除 betweenneess 最高的边，转到（1）。

算法二：Average Link + Jaccard Similarity

Jaccard Distance 是用来衡量两个集合差异性的一种指标，它是 Jaccard 相似系数(Jaccard similarity coefficient)的补集，被定义为 1 减去 Jaccard 相似系数。其中 Jaccard 相似系数是用来衡量两个集合相似度的一种指标。

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

$$d_J(A,B) = 1 - J(A,B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|}$$

此算法首先应用 Jaccard 计算 A 的距离矩阵 D，接着应用 average 方法构造层次聚类树 Z，然后在 Z 上进行层次聚类。每一个步骤在 Matlab 中都有函数接口可以直接调用。

算法三：Ratio Cut

RatioCut 巧妙地把一个 NP 难度的问题转换成拉普拉斯矩阵特征值（向量）的问题，将离散的聚类问题松弛为连续的特征向量，最小的系列特征向量对应着图最优的系列划分方法。剩下的仅是将松弛化的问题离散化，即将特征向量再划分开，便可以得到相应的类别。

Ratio Cut 与拉普拉斯矩阵有着密切的关系。可以证明最小化 Ratio Cut 等价于最小化 f'Lf，其中 L=D-W 是一个拉普拉斯矩阵，D 是节点度的对角阵，W 是邻接矩阵。接下来，我们只要找到L的最小特征值λ及其对应的特征向量即可。

进一步，我们求拉普拉斯矩阵的前 K 个特征值，再对前 K 个特征值对应的特征向量进行 K-means 聚类。鉴于拉普拉斯矩阵的最小特征值为 0，我们可以从次小的特征值对应的特征向量来取。

算法流程：

（1）计算得到拉普拉斯矩阵 L = D - W；

（2）计算 L 的所有特征值；

（3）取最小的 k 个非零特征值对应的特征向量，组成矩阵 Vr；

（4）将取得的 k 个特征向量（即矩阵 Vr）进行 K-means 聚类。

算法四：Normalized Cut

Normalized cut 的思想与 Ratio Cut 大致相同。不过，它不是求 L 的特征值，而是求 D^(-1/2)LD^(-1/2)的特征值。它的结果比 Ratio Cut 更加均衡。

算法流程：

（1）计算得到拉普拉斯矩阵 L = D^(-1/2) * (D - W) * D^(-1/2)；

（2）计算 L 的所有特征值；

（3）取最小的 k 个非零特征值对应的特征向量，组成矩阵 Vn；

（4）将取得的 k 个特征向量（即矩阵 Vn）进行 K-means 聚类。

算法五：Modularity

该算法是对于某一种社区结构，考虑每个社区内连边数与期待值之差。实际连边越是高于随机期望，说明节点越有集中在某些社区内的趋势，即网络的模块化结构越明显。但是直接对社区及其内部的节点进行遍历的复杂度很大，一种处理的方式与 Ratio Cut 或 Normalized Cut 相似，也是通过特征向量来降低维度。

此算法用到的一个矩阵是 modularity matrix，即 B = A-ddT/2m。该矩阵的行列和都是

0，因为实际网络和随机洗牌后的网络度分布是不变的。

算法流程：

（1）计算得到矩阵 B = A-ddT/2m；

（2）计算 B 的所有特征值；

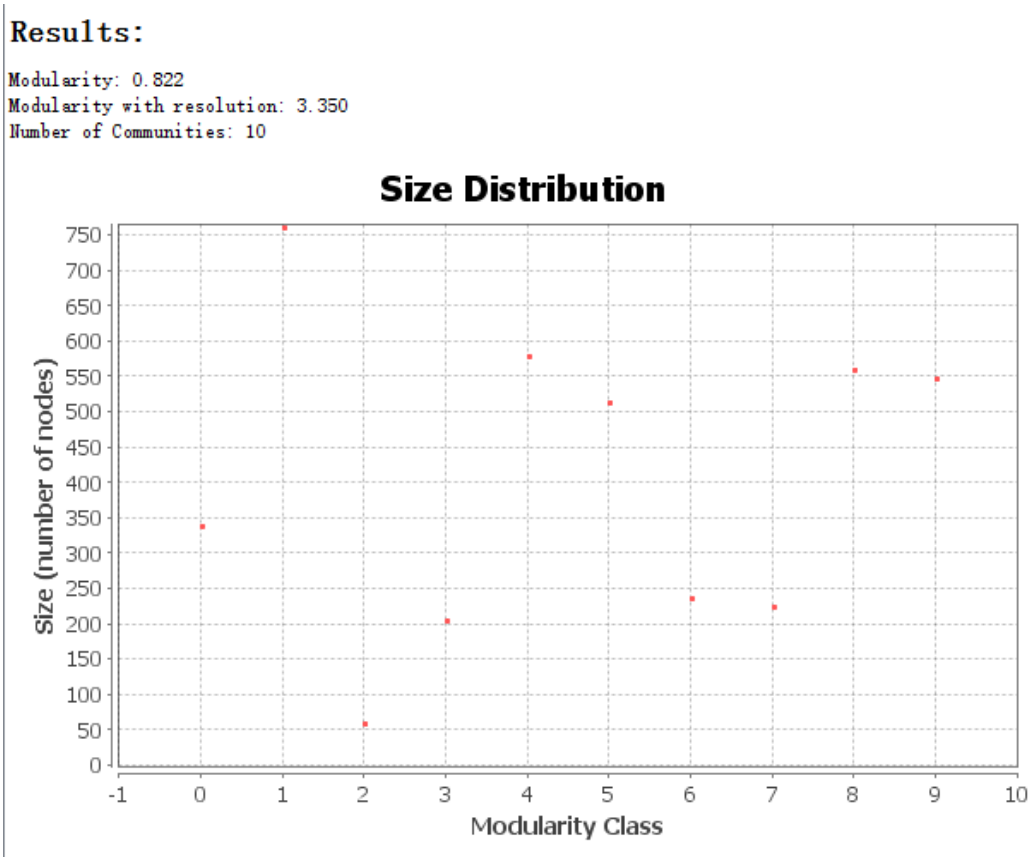
（3）取 B 最大的 k 个特征值对应的特征向量，组成矩阵 Vm；

（4）将取得的 k 个特征向量（即矩阵 Vm）进行 K-means 聚类。

第四组数据 K 的确定：

在本次实验中，第四组数据 k 的值取 10。

首先，我将未分类的数据输入 Gephi 处理。点击模块化选项，运行后结果如下显示：



结果显示预期的社区数目为 10，同时，原始数据有 10 个中心点，大致能以这 10 个中心点进行分类。

在 Stanford 的 Web Mining 主页上，我找到了原始数据，它是在 10 个 NetWork 中分别取得的数据。大致可以认为 10 个 NetWork 各自分别是一个社区。

Files

File	Description
facebook.tar.gz	Facebook data (10 networks, anonymized)
facebook_combined.txt.gz	Edges from all egonets combined
readme-Ego.txt	Description of files

【实验结果说明及演示】

Football 和 polbooks 五种算法的结果与 ground_truth 进行比较。结果如下：

名称	值
ACC_football_aj	0.1478
ACC_football_gn	0.1478
ACC_football_mo	0.1304
ACC_football_nc	0.2000
ACC_football_rc	0.2000
ACC_polbooks_aj	0.7714
ACC_polbooks_gn	0.7810
ACC_polbooks_mo	0.6476
ACC_polbooks_nc	0.7619
ACC_polbooks_rc	0.7714
NMI_football_aj	0.2633
NMI_football_gn	0.2637
NMI_football_mo	0.2515
NMI_football_nc	0.2515
NMI_football_rc	0.2633
NMI_polbooks_aj	0.4318
NMI_polbooks_gn	0.4388
NMI_polbooks_mo	0.2903
NMI_polbooks_nc	0.4257
NMI_polbooks_rc	0.4018

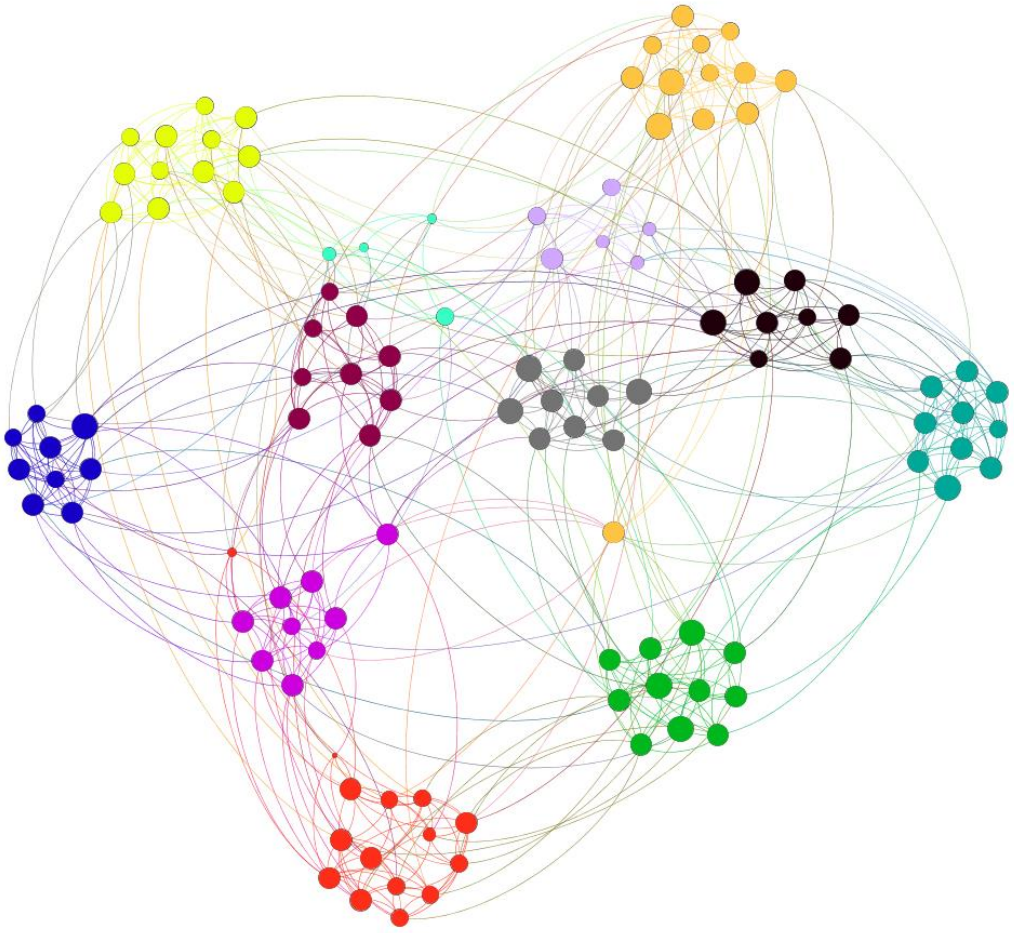
从比较数据可知，football 的社区发现结果不是很好，平均 ACC 正确率才 0.2 不到；相应的，NMI 值也不是很高。而对 polbooks 社区发现的结果相对较好，ACC 能达到 0.7 以上。可能的原因是，对 football 社区发现的计算模型不适合原数据，从下面可视化的结果可以发现，一般的社区发现算法都是将联系紧密的点归为一类，而原数据分类的标准与之不同。

另一方面，modularity 算法与其他算法得到的结果差别比较大。可能是因为此算法的思想和出发点与其他算法不同。其他算法的思想是将一个整体逐步分离，分出若干个社区。而 Modularity 则是将联系紧密的点合并为一个社区。

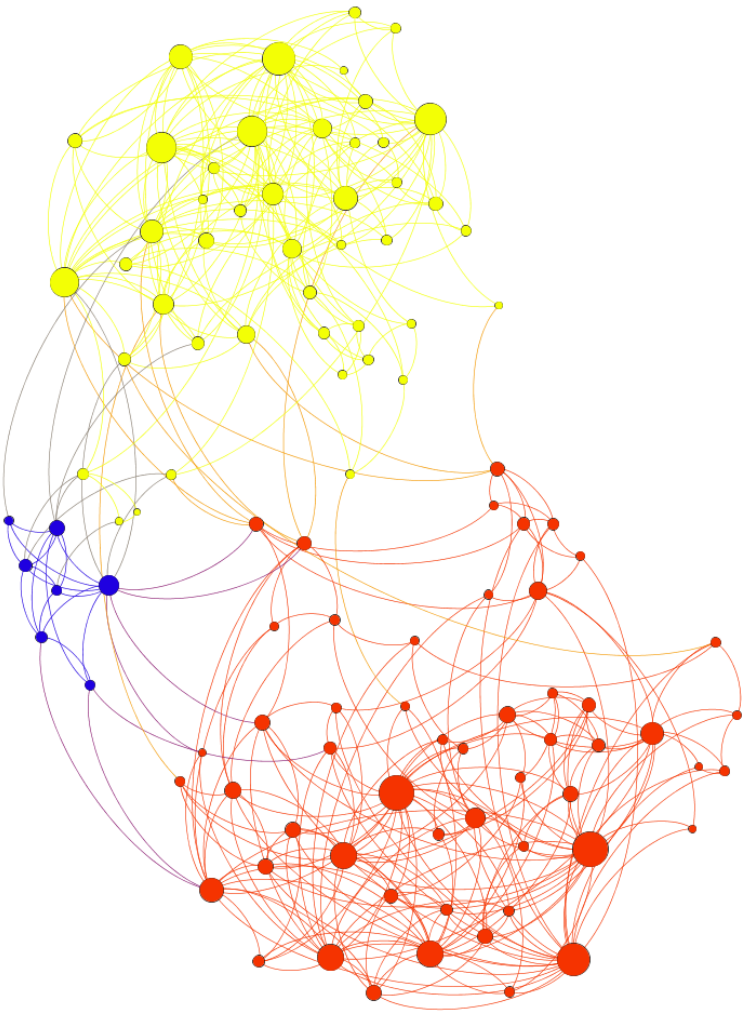
可视化结果：

可视化的结果都在 Gephi 中得到。下面贴上 4 组数据用 Girvan-Newman 得到的可视化结果。其余算法的图在 Lab2_picture 目录下。Mat 图数据通过 convert.cpp 转换为 csv 格式。

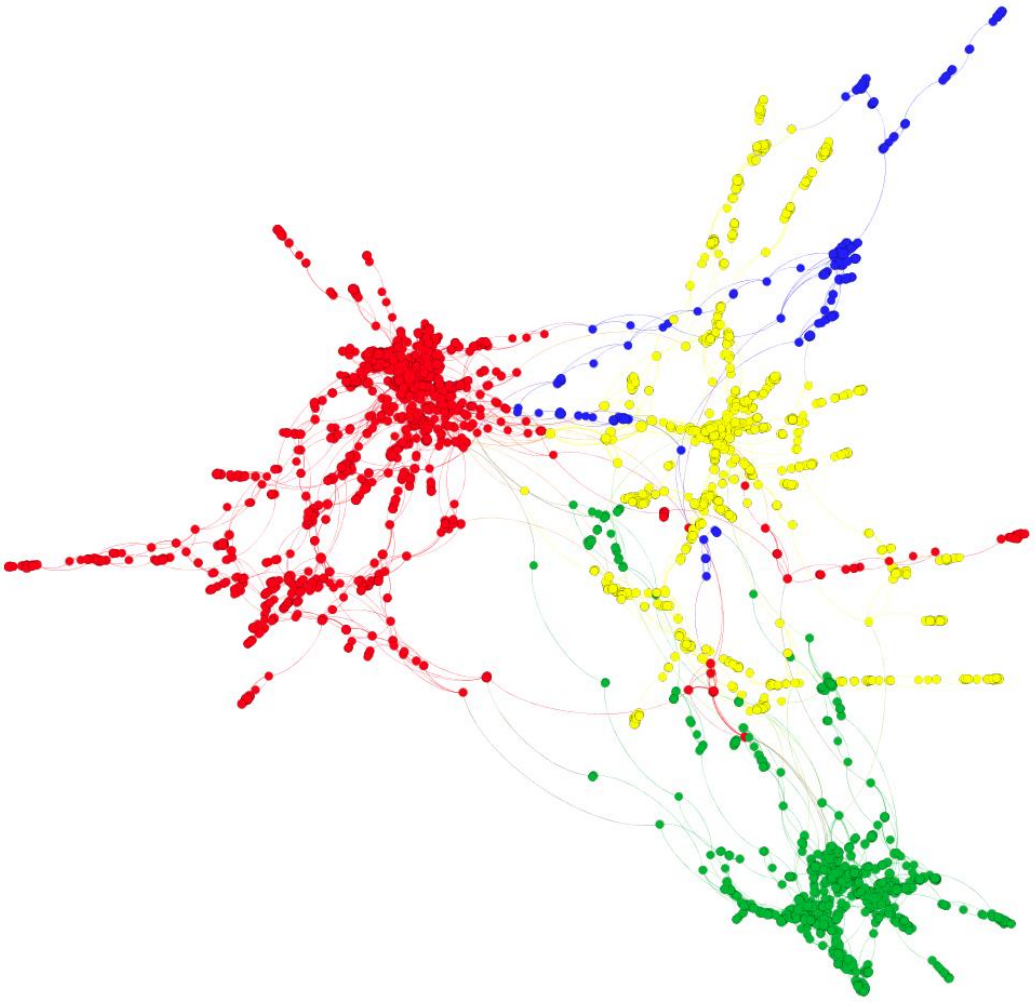
Football



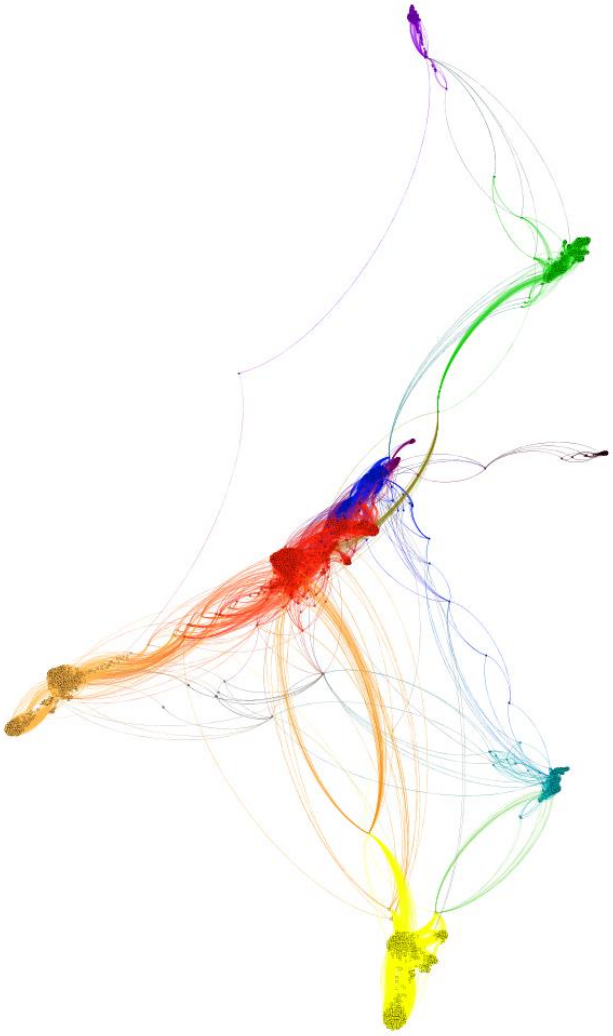
Polbooks



DBLP



Egonet



【实验总结】

通过本实验，我深刻理解了 5 种社区发现算法的原理，并用 Matlab 及其相应的工具完成了算法的实现。除此之外，我还了解了一些对社区发现及聚类的评价指标，并对此次实验的结果进行直观上的评价。

不过，实验中的算法还有改进之处。比如，第一个算法，每一次要计算所有边的 betweenness，并只去除其中最大的一条边，这样效率很差。对于顶点数和边数很多的图，可以一次性去除较多的边，减少 betweenness 的计算次数。对于后面三种用到 K-means 的算法，初始时中心点的确定会影响到最后聚类的效果。虽然理论上，当算法收敛之后，聚类的中心点是固定的，但是算法收敛的时间太长，一般 K-means 会指定一个最多的迭代次数。于是，一个好的中心点的确定会对结果产生较大影响。一个较好的方法是，先用其他较快的聚类算法估计中心点，之后按照估计的中心点开始 K-means 算法。