A mathematical model of predicting error connections using link prediction algorithm and network topology

Abstract

With the development of modern technology, we have accumulated more and more networks data but the data is partially incomplete, inaccurate or sometimes distorted. In this case, it's really important to identify the error connections which based on the accurate judgement of the type of the networks.

In this paper, we build a mathematical model to find the error connections. Under the condition of unknown complex-system in the real world this model is based on the structural characteristics of the network. Different networks have different model.

The process of the construction of the model is divided into two parts. In the first part, we use the knowledge of graph theory to transform the network into collections of abstract edges and points. We use MATLAB and CodeBlocks software tools to collect the topology information of the network.

The information include the number of nodes(N), the number of edges(M), Global efficiency(GE)、the number of connected blocks and the number of nodes which in the biggest connected blocks, Clustering coefficient(C1), Clustering coefficient based on sociological definition(C2), assortativity coefficient(R).Then we use the excel tool to storage the node information. After data analysis we can conclude the network structure.

Based on the result of the first part, we chose the appropriate link prediction algorithm for different types if networks. We chose LP and GNC+PA for our error connections prediction and get a good prediction effect.

Contents

1 Introduction

1.1 Background

Networks is a powerful tool to describe the structure of the real system. From the world's largest networks Internet and World Wide Web to the Metabolic network and Neural networks of animals, from Social networks to Information networks, Complex Networks filled in Natural and Human society. In Graph Theory, system elements and individual are represented by Node and Edge between two nodes indicates linkages between system elements.

With the development of modern technology, we have accumulated more and more data of networks, but the data of networks often contains Noise. Therefore, networks often contain false edges and missing edges, and part of the data is incomplete, inaccurate or sometimes distorted in varying level.

We may need to do some experiments which are expensive and cumbersome to determine these error edges. But if we can predict some edges which may wrong, experiments can be targeted, thereby avoiding the blind meaningless work.

Through the analysis of network data, we can determine the topology of network. By using some of the norm in Link Prediction, we can predict which edges were more likely to be the wrong edge, thus improve work efficiency.

1.2 Restatement of the problem

(1) Develop a mathematical model to understand the structure and organization mechanics of the network. The structural characteristics of the different types of networks and the organization principle are not always the same.

(2) Propose an effective method to identify the error connections. Show the completeness of how the structural characteristics are discovered; explain the validity and the accuracy of the mathematical model as well as the accuracy of the algorithm.

1.3 Our work

We reviewed articles ,books , learn and understand the basic type and topology of the network. After analyzing the problem in graph theory, we calculated the parameters of these graph by using C++ and MATLAB. The information include: the maximum number of nodes of the max connected component, the number of connected component, the global efficiency (GE),clustering coefficient(C) and distribution coefficient(R) in undirected graph

and strongly connected component the global efficiency (GE), clustering coefficient(C) and distribution coefficient(R) in undirected graph and strongly connected component, the number of nodes in the maximum strongly connected component, the number of strongly connected component, the global efficiency(GE), the average of degree of nodes, clustering coefficient(C).

The title only gives us data of abstract nodes and edges but link prediction can analyze network through the topology without the specific meaning of node and connection. So we use link prediction modeling.

After analyzing the topological properties of these networks, we refer to some known network and we construct hybrid similarity index model with the PA (preferential attachment) indicators and GCN(Generalized Common Neighbors measure) indictors. We use this model to measure score of each connection. The connection which has lower score was more likely to be the error connection.

2 Problem analysis

2.1 How to develop a mathematical model to understand the structure and organization mechanics of the network

With different structure characteristic of different types of networks, we establish the different mathematical model to understand the organizational structure and mechanics of network. Because we only know the general type of the network, directed or undirected of the graph and a large number of nodes' abstract information. And no one tell us the specific meaning of each nodes. We can't see the features of the network structure directly and can't tell it corresponds to which kind of complex system in real life. So we chose data processing of node information first. Then make all kinds of topological characteristics of all figure, in order to through these features to analyze the nature of the networks.

2.2 Propose an effective method to identify the error connections.

Because there are 10% error connections which have been added randomly in the system. Whatever the network is, the error connections doesn't affect the characteristics of the entire network. Therefore, our judgment on the wrong side can be based on the existing network nodes' information and the network structure. All we need is to find an algorithm, to compute the probability of each

edge arrears. At last we sort the edge by the division value and find the error connections.

3 Assumption

- The 10% error connections doesn't change the original topological.
- The analysis of the network is a complicated system in the real world which can be applied noised to.
- Don't consider the effect of the change of time.

4 Symbol Description

Symbol	Meaning
M	Number of existed edges
N	Number of nodes
Z	Average degree
k_i^{in}	Out-degree of node i
k_i^{out}	In-degree of node i
D	Distance between nodes
L	Average distance of path
GE	Global efficiency
C_i	Clustering coefficient of nodes whose degree is i
C	Clustering coefficient
$\Gamma_{in\ (x)}$	Set of the point which on the edge of x as start point
r	Assortativity coefficient
E_I	Edges exist in the node i adjacent nodes
k_i	The degree of node i

A	adjacency matrix
S	Third-order neighbor contribution
ε	The coefficient of LP
$\Gamma_{out\ (x)}$	Set of the point which on the edge of x as start point

The description of the relevant symbols:

distence[1]: The length of the shortest path between two nodes.

degree[1]: The degree k_i of node i in an undirected network is defined as the number of edge linked to node i.

average degree[1]: The average degree of a network is the average value of all the nodes' degree, denoted by Z.

out − degree[1]: The out-degree k_i^{out} of node i is the number of edge from node i to other nodes.

in−degree[1]: The in-degree k_i^{in} of node i is the number of edges from other nodes to node i.

Assortativity coefficient[1]: Assortativity coefficient is used to indicate that whether the nodes with similar degree is inclined to linked to each other. If the nodes with large degree is linked with other large-degree node generally, we called the network's degree positive correlated, or the network is assortative. On the contrary, if the nodes with large degree is linked with small-degree node generally, we called the network's degree is negative correlated, or the network is disassortative.

Clustering coefficient[1]: Assume that the degree of node i in a network is k_i, i.e. node i has k_i edge to link to other nodes (which is called neighbor or adjacent point). The clustering coefficient is the rate of the real number of edges of node i and the maximum possible number of edges of node i.

5. Develop a mathematical model

5.1 Analyze the network topology

5.1.1 Determine the type of the graph

First of all, because we do not need to consider the meaning of the nodes, we can consider the complex network as a graph in order to analyze it. We divide the graphs in to 4 types as shown in figure 5-1-1according to the glossary of graph.

The analysis of the 6 network structure we propose is mainly based on unweighted undirected graph and unweighted directed graph.

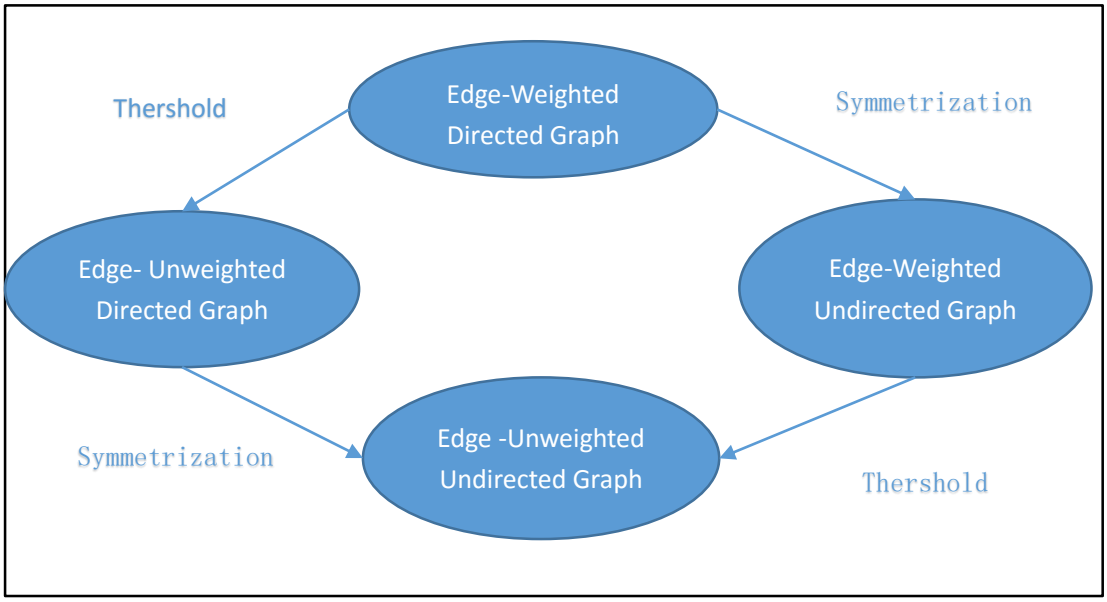


Figure 1 The type of figure[1]

5.1.2 Mathematical model for the three kinds of undirected graph

We count the degree of nodes, density, Global efficiency, and assortativity coefficien to determine an unknown network

Statistical results are as follows:

Table 1: Topological characteristics of three undirected network

Attribute	Info-network undirected	Bio-network undirected	Social network undirected
N	4554	2186	50398
M	5778	10491	44268
NC	4473/35	2180/4	2319/6555
GE	0.163761	0.227305	0.000397259
C1	0.0119282	0.263871	0.00749277
C2	0.0307122	0.470929	0.00334268
R	-0.1286	0.4862	-0.1747

M/N: the number of nodes N and connections M

NC: the number of connected blocks and the number of nodes in the biggest connected blocks

GE: Global efficiency[1]。

$$\mathbf{GE} = \frac{\mathbf{1}}{\frac{\mathbf{1}}{\mathbf{2}\boldsymbol{N}(\boldsymbol{N}-\mathbf{1})}\sum_{\boldsymbol{i}\geq\boldsymbol{j}}\frac{\mathbf{1}}{\boldsymbol{d_{ij}}}}$$

In order to avoid the average path length being infinity. We choose global efficiency to measure the calculation effectiveness in entire network. We use BFS (breadth-first search) algorithm implement the formula in C++.

C： Clustering coefficient

$$\boldsymbol{C_i = \frac{E_i}{(k_i(k_i-1))/2} = \frac{2E_I}{k_i(k_i-1)}}$$

$$\boldsymbol{C_1 = \frac{1}{N}\sum_{I=1}^{N} c_I}$$

We use this parameter to measure whether there exist a relationship between one node's neighbors. And measure the tightness of network by this parameter.

The definition of clustering coefficient in sociological, describe the clustering characteristics of network based on the relative number of triangles. We can analyze network and research the transversality of network.

$$C_2 = \frac{number\ of\ closed\ triplets * 3}{number\ of\ connected\ triples\ of\ vertices/3} = \frac{\sum_{i=1}^{N}\sum_{,j\neq i,k\neq i,j\neq k} a_{ij}a_{ik}a_{jk}}{\sum_{i=1}^{N}\sum_{j\neq i,k\neq i,j\neq k} a_{ij}a_{ik}}$$

R[1]: Assorativity coefficient

$$< \mathrm{jk} > - < \mathrm{j} >< \mathrm{k} >= \sum_{j,k} jk(e_{jk} - q_j q_k)$$

$$\sigma_q^2 = \sum_k k^2 q_k^2 - [\sum_k k q_k]^2$$

$$\mathrm{r} = \frac{1}{\sigma_q^2}\sum_{j,k} jk(e_{jk} - q_j q_k)$$

Because of the distribution of degree can't uniquely identify a network, we decided to introduce assorativity coefficient to portray relation between similar nodes in the network.

5.1.3 Mathematical model for the three kinds of directed graph

For the directed graph of bio-network, info-network and social network. We chose the following statistics as the reference for analyzing the structure of network topology:

Table 2: Three directed network topology characteristics

Attribute	Info-network directed	Bio-network directed	Social network directed
N	2591	293	25440

Z	3.50946	7.72355	59.2134
G	285/2301	242/51	24231/1210
C1	0.00255819	0.120004	0.0879446
C2	0.0989905	0.191294	0.134941
GE	0.024727	0.258622	0.30739

N: The number of nodes.

Z: The average out/in degrees of networks

Being used to judge the figure is dense or not.

G: number of strongly connected component /the number of nodes in the biggest strongly connected component

C: Clustering coefficient

C2: Sociological definition of clustering coefficient.

GE: Global efficiency[1].

5.2 Model overview

Due to a given node simulating a reality complicated system and the topology of the network is able to fully reflect the real linking properties of the complex system. We take the method of link prediction. We handle the topological structure of network directly, and do not think about the really node or the attribute. Because there are many ways to deal with link prediction, we finally choose LP, GCN+PA respectively to predict the spurious interactions of the directed graph and undirected graph, according to the characteristics of the topological structure we have calculated. When we get the result, we find that there are a lot of node which have little relation with the graph. Therefore, a noise suppression is necessary. Then we try deleting the node with one degree, and get rid of the set of continuous or heavy edge.

5.2.1 The prediction for error connections of undirected graph

LP[2](local path)

LP[2](local path)

Considering LP is based on the similarity index of the path, we can apply it in directed graph and undirected graph at the same time. However, we only use it to predict the contribution of undirected graph that considering the third-order neighbor on the basic of common neighbor index:

$$S = A^2 + \alpha A^2$$

In different network, the contribution of the next nearest neighbors are generally different. Different network will have different ε value. However, for

every real network, the optimal ε value is an index number which is calculated complexly and got from the whole network. What's more, the contribution to the second order neighbor prediction accuracy is small relative to the common neighbors.

Comprehensively considered, we define $\alpha = 0.001$, which is the adjacency matrix of the network. In the answer matrix we calculate by MATLAB, Si,j represent the relevancy of the two nodes. Then what we do is sort the value of edge and regard the top 10% sides as the spurious interactions.

Prediction results:

Table 4 the result of prediction in three undirected graph

type	Info-network undirected	Bio-network undirected	Social network undirected
accuracy rating	0.37455	0.34068	0.53748

As we can see, the LP is better in Social network.

5.2.2 The prediction for error connections of directed graph

PA+GCN

PA index is designed which was based on the laws in network evolution process: "the rich get richer", and it points out the fact that the nodes which have many out-degrees have more connections to nodes which have many in-degrees [4]. We can use PA index to judge error connections in network. GCN index combines CN\CC\RC with reasonable weight, as shown in figure 2. GCN is very gifted in locally, but it doesn't fit the network which have low general efficiency. Also PA is gifted in overall situation so we can combine these two index to improve accuracy. And we can find GCN+PA have the highest accuracy in almost all real network.

PA(Preferential Attachment Index)[3]

$$S_{xy} = k_{out}(x) \times k_{in}(y)$$

GCN(Generalized Common Neighbors measure)

$$S_{xy} = S_{xy}^{asy} + S_{xy}^{sy} = |\Gamma_{out}(x) \cap \Gamma_{in}(x)| + \frac{1}{2}[|\Gamma_{out}(x) \cap \Gamma_{out}(x)| + \Gamma_{in}(x) \cap \Gamma_{in}(x)]$$

Measures	GPN	HCN	FW	CNN	SSN	PBN	ESN	NDW	ACN	EEN
CN	0.505	0.836	0.722	0.795	0.669	0.924	0.828	0.838	0.892	0.953
Salton	0.505	0.836	0.697	0.788	0.669	0.911	0.828	0.837	0.892	0.949
Jaccard	0.505	0.836	0.696	0.786	0.669	0.911	0.828	0.837	0.892	0.948
SØrensen	0.505	0.836	0.696	0.786	0.669	0.911	0.828	0.837	0.892	0.948
HP	0.505	0.836	0.702	0.790	0.669	0.905	0.827	0.838	0.892	0.948
HD	0.505	0.836	0.696	0.786	0.669	0.911	0.828	0.837	0.892	0.948
LHN	0.505	0.836	0.676	0.781	0.669	0.891	0.827	0.837	0.892	0.945
AA	0.505	0.836	0.726	0.796	0.669	0.925	0.828	0.838	0.892	0.954
RA	0.505	0.836	0.730	0.797	0.669	0.924	0.828	0.838	0.892	0.954
ERA	0.505	0.836	0.708	0.796	0.669	0.917	0.828	0.838	0.892	0.952
PA	0.792	0.878	0.837	0.795	0.921	0.949	0.902	0.869	0.730	0.876
LP	0.508	0.863	0.726	0.847	0.849	0.955	0.905	0.872	0.925	0.955
CC	0.504	0.763	0.566	0.753	0.640	0.846	0.808	0.836	0.850	0.953
BC	0.504	0.816	0.526	0.713	0.645	0.855	0.833	0.843	0.898	0.953
GCN	0.514	0.959	0.739	0.878	0.740	0.947	0.880	0.885	0.942	0.959
GCN+PA	0.795	0.974	0.743	0.886	0.925	0.958	0.917	0.897	0.958	0.940

Figure 2 the prediction accuracy by 16 kind similarity index in 10 real directed networks [2]

When we calculate the score. If the network is small, we try two kinds of way: PA and GCN, such as bio-network and info-network. And we only use PA in social network because the network is too large.

The result of prediction:

Table 4 the result of prediction in three directed graph

Type	**Info-network directed**	**Bio-network directed**	**Social network directed**
Accuracy	0.38568	0.54206	0.75857

6 Analysis of the model

Our model can be divided into two parts. The first part is the analysis of network, and in the second part, we choose different index by parameters of network, and then judge error connections by link prediction.

About selecting the topological properties: although the topological nature of our choice are typical, there are also existing some network topologies nature relatively similar with other. Although there are some differences in the details, the actual network is unknown, we can't distinguish between the two networks. So the characterize of the network is slightly less.

We use LP index to predict error connection in undirected graph, but because of the limitation of this index and the value of ε is constant, we did not get good results in predicting in info-network and bio-network but our prediction got 50% accuracy in social network. And other index based on similarity of nodes don't

have good results.

We mainly use PA and GCN index to prediction error connection in directed graph but we have some adjustment with different network. We use PA to get corking result and lower computational complexity in social network. And we use GCN+PA to get more accurate results in the other network.

Overall, with the situation that we don't know the type of network, we will have better predict accuracy if the general efficiency or Average degree is high.

7 Reference

[1]Xianfan Wang,Xiang Li,Guanrong Chen.Network Science: An Introduction.Higher Education Press 2012.4

[2]Yangfu Zhang.Link prediction in directed and weighted networds. Retrieved from:

http://xueshu.baidu.com/s?wd=paperuri%3A%28760b0a3c756f62225bbb23cd71c9e23e%29&filter=sc_long_sign&tn=SE_xueshusource_2kduw22v&sc_vurl=http%3A%2F%2Fcdmd.cnki.com.cn%2Farticle%2Fcdmd-10530-1011243243.htm&ie=utf-8

[3] Carlson J, Doyle J. Highly optimized tolerance: Robustness and power laws in complex systems[J].
Phys.Rev.Lett.,2000, 84(11):2529-2532

[4] BarabsiA-L, AlbertR.Emergenceofscalinginrandomnetworks［J］. Science，1999，286:509 — 512

8 The appendix

```
1. get_assotative_coefficient.m
2. function r=get_assortative_coefficient(Nodes)
3. %get assortative coefficient,see Ref[Newman,Mixing patterns in
   networks]
4. %Input:Nodes--N*N adjacent matrix,Nodes(i,j)=1;i is outdegree,j is
   indegree
5. %Output:r--assortative coefficient
6. %
7. %Modified by Rock on 06.02.28
8. %Modified by Rock on 06.12.10 for r=0
9. %%Modified 07.03.20 for edge_num=0
10. %%Modified 07.09.13 for Out/Indegree
11.
12. TEST=0;
```

```
13.
14. if TEST==1
15.         Nodes=[0,1,1;0,0,1;1,1,0];
16.         Nodes=[0,1,0,1;1,0,1,0;0,1,0,1;1,0,1,0];
17.         Nodes=sparse(Nodes);
18. end
19.
20. N=length(Nodes);
21. edgeNum=nnz(Nodes);%include outEdge and
    inEdge,ave_degree=edgeNum/N
22.
23. Outdegree=zeros(N,1);
24. Indegree=zeros(N,1);
25.
26. %for i=1:N
27. %         Outdegree(i)=nnz(Nodes(i,:));
28. %         Indegree(i)=nnz(Nodes(:,i));
29. %end
30.
31. Outdegree=sum(Nodes(1:end,:));
32. Indegree=sum(Nodes(:,1:end));
33.
34. %sum1 is sum(i*j)
35. %sum2 is sum(i+j)
36. %sum3 is sum(i^2+j^2)
37. sum1=0;
38. sum2=0;
39. sum3=0;
40.
41. [Row,Col,Weight]=find(Nodes);
42. Len=length(Row);
43.
44. for temp=1:Len
45.         i=Row(temp);
46.         j=Col(temp);
47.         sum1=sum1+(Indegree(j)-1)*(Outdegree(i)-1);
48.         sum2=sum2+(Indegree(j)-1)+(Outdegree(i)-1);
49.         sum3=sum3+(Indegree(j)-1)^2+(Outdegree(i)-1)^2;
50. end
51. if edgeNum==0%Modified 07.03.20
52.         r=0;
53. else
54.         if ((sum1/edgeNum)-
    (sum2/(2*edgeNum))^2)*((sum3/(2*edgeNum))-
```

```
    (sum2/(2*edgeNum))^2)==0 %%Modified by Rock on 06.12.10 for r=0
55.             r=0;
56.     else
57.             r=((sum1/edgeNum)-
    (sum2/(2*edgeNum))^2)/((sum3/(2*edgeNum))-
    (sum2/(2*edgeNum))^2);
58.     end
59. end
60.
61. return
2 CN.m

 sim = train * train;
3.katz.m
 sim = inv( sparse(eye(size(train,1))) - lambda * train);
 sim = sim - sparse(eye(size(train,1)));
3 LocalPath.m

 sim = train*train;
 sim = sim + lambda * (train*train*train);
```

4. PA.m

```
 deg_row = sum(train,2);
 sim = deg_row * deg_row';
clear deg_row deg_col;
```

5 Jaccard.m

```
 sim = train * train;
 deg_row = repmat(sum(train,1), [size(train,1),1]);
 deg_row = deg_row .* spones(sim);

 deg_row = triu(deg_row) + triu(deg_row');
 sim = sim./(deg_row.*spones(sim)-sim); clear deg_row;
 sim(isnan(sim)) = 0; sim(isinf(sim)) = 0;
```

6 getParamentOfGraph.cpp

```
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <iostream>
#include <vector>
#include <queue>
#include <stack>
using namespace std;
const int MAXM = 1E7;
```

```
const int MAXN = 60000;
struct Side{
	int u,v;
}side[MAXM];
vector<int>vv[MAXN];
int siz,totEdge;//the number of nodes,the number of edges
int hh[MAXM];
bool vis[MAXN];
void ppGraph(){//print graph
	for(int i = 1;i <= siz;i ++){
		cout<<i<<':';
		for(int j = 0;j < vv[i].size();j ++)
			cout<<vv[i][j]<<' ';
		cout<<endl;
	}
	cout<<endl;
}

void ppDiscretization(){//print the result of discretization
	for(int i = 0;i < totEdge;i ++){
		cout<<side[i].u<<' '<<side[i].v<<endl;
	}
}

void buildGraph(bool flag){//if flag is 0,graph is undirected
	for(int i = 0;i < totEdge;i ++){
		vv[side[i].u].push_back(side[i].v);
		if(flag == 0)vv[side[i].v].push_back(side[i].u);//wu xiang tu
	}
}

void getNC(){//undirected graph connected componet
	memset(vis,0,sizeof(vis));
	int totT = 0,maxNum = 0;
	queue<int>q;
	for(int i = 1;i <= siz;i ++){
		if(vis[i] == 0){
			vis[i] = 1;
			q.push(i);
			totT ++;
			int Tnum = 1;
			while(!q.empty()){
				int now = q.front();q.pop();
				int nn = vv[now].size();
```

```
                    for(int j = 0;j < nn;j ++){
                        if(vis[vv[now][j]])continue;
                        Tnum ++;
                        q.push(vv[now][j]);
                        vis[vv[now][j]] = 1;
                    }
                }
                maxNum = max(maxNum,Tnum);
            }
        }
        cout<<"the number of node of the max Connected
component :"<<maxNum<<"the number of Connected
component :"<<totT<<endl;
}

void getE(){//global efficiency
        double totDis = 0.0;
        for(int i = 1;i <= siz;i ++){
            memset(vis,0,sizeof(vis));
            queue<pair<int,int> >q;
            q.push(make_pair(i,0));
            vis[i] = 1;
            while(!q.empty()){
                pair<int,int> tmp = q.front();q.pop();
                if(tmp.second)totDis += 1.0/tmp.second;
                int now = tmp.first,nn = vv[now].size();
                for(int j = 0;j < nn;j ++){
                    int xx = vv[now][j];
                    if(vis[xx])continue;
                    q.push(make_pair(xx,tmp.second+1));
                    vis[xx] = 1;
                }
            }
        }
        cout<<"the average length of Shortest path of all nodes "<<totDis/siz/(siz-
1)<<endl;
}

bool have(int b,int a){//judge whether have edge b->a or not
        int nn = vv[b].size();
        for(int i = 0;i < nn;i ++){
            if(vv[b][i] == a)return 1;
        }
        return false;
```

```
}

void getC(){//two kind of clustering coefficient
	//ppGraph();
	double C1 = 0,C2 = 0;
	long long totLian = 0;
	long long totTri = 0;
	for(int i = 1;i <= siz;i ++){
		int nn = vv[i].size();
		if(nn <= 1)continue;
		int tmpTri = 0;
		for(int one = 0;one < nn;one ++){
			for(int two = one + 1;two < nn;two ++){
				tmpTri += have(vv[i][two],vv[i][one]);
			}
		}
		double tmpC1 = (double)tmpTri*2/nn/(nn-1);
		C1 += tmpC1;

		totTri += tmpTri;
		totLian += nn*(nn-1)/2;
	}
	C1 /= siz;
	C2 = (double)totTri/totLian;
	cout<<"average Clustering coefficient of all nodes :"<<C1<<' '<<C2<<endl;
}

void getZ(){//average degree of nodes
	cout<<"average degree of nodes"<<(double)totEdge/siz<<endl;
}

//strongly connected component in directed graph
int t,dfn[MAXN],low[MAXN],sum;
int gNum[MAXN];
stack<int>s;
void dfs(int u){
	dfn[u] = low[u] = ++t;
	s.push(u);
	int nn = vv[u].size();
	for(int i = 0;i < nn;i ++){
		int v = vv[u][i];
		if(!dfn[v])dfs(v);
		if(dfn[v] != -1)low[u] = min(low[u],low[v]);
	}
```

```
	if(low[u] == dfn[u]){
		int v;
		do {
			v = s.top();s.pop();
			dfn[v] = -1;
			low[v] = sum;
			gNum[sum] ++;
		}while(v != u);
		sum ++;
	}
}
void getG(){
	t = sum = 0;
	for(int i = 1;i <= siz;i ++){
		if(!dfn[i])dfs(i);
	}
	int maxNum = 0;
	for(int i = 0;i < sum;i ++){
		maxNum = max(maxNum,gNum[i]);
	}
	cout<<"the number of nodes of the max Strongly connected
component :"<<maxNum<<"the number of Strongly connected
component"<<sum<<endl;
}
//strongly connected component in directed graph


void getR1(){//reciprocity
	int totm_d = 0;
	for(int i = 1;i <= siz;i ++){
		int nn = vv[i].size();
		for(int j = 0;j < nn;j ++){
			totm_d += have(vv[i][j],i);
		}
	}
	cout<<"the reciprocity is : "<<(double)totm_d/totEdge<<endl;
}

void Undir(){//get undirected graph parameters
	buildGraph(0);
	cout<<"Total Nodes is :"<<siz<<endl;
	getNC();
	getE();
	getC();
```

```
}

void Dir(){//get directed graph parameters
	buildGraph(1);
	cout<<"Total Nodes is :"<<siz<<endl;
	getZ();
	getG();
	getE();
	getR1();

	for(int i = 0;i <= siz;i ++)vv[i].clear();
	buildGraph(0);
	getC();
}

int main(){
	freopen("in.txt","r",stdin);
	//build graph and discretization
	int a,b;
	char s[100];cin>>s;
	totEdge = 0;
	while(~scanf("%d,%d",&a,&b)){
		side[totEdge] = (Side){a,b};
		hh[totEdge*2] = a;
		hh[totEdge*2+1] = b;
		totEdge ++;
	}
	sort(hh,hh+totEdge*2);
	siz = unique(hh,hh+totEdge*2) - hh;
	for(int i = 0;i < totEdge;i ++){
		side[i].u = lower_bound(hh,hh+siz,side[i].u) - hh + 1;
		side[i].v = lower_bound(hh,hh+siz,side[i].v) - hh + 1;
		//cout<<side[i].u<<' '<<side[i].v<<endl;
	}
	//ppDiscretization();
	//build graph and discretization

	//Undir();//Get Parameters of Undirected Graph
	Dir();//Get Parameters of Directed Graph
	return 0;
}
```

6 GCN_and_PA.cpp

```
#include <cstdio>
#include <cstring>
```

```
#include <algorithm>
#include <iostream>
#include <vector>
#include <queue>
#include <stack>
using namespace std;
const int MAXM = 1E7;
const int MAXN = 60000;
struct Side{
	int u,v;
}side[MAXM];
vector<int>vv[MAXN];
int siz,totEdge;//the number of nodes,the number of edges
int hh[MAXM];
bool vis[MAXN];
void ppGraph(){//print graph
	for(int i = 1;i <= siz;i ++){
		cout<<i<<':';
		for(int j = 0;j < vv[i].size();j ++)
			cout<<vv[i][j]<<' ';
		cout<<endl;
	}
	cout<<endl;
}

void ppDiscretization(){//print the result of discretization
	for(int i = 0;i < totEdge;i ++){
		cout<<side[i].u<<' '<<side[i].v<<endl;
	}
}

void buildGraph(bool flag){//if flag is 0,graph is undirected
	for(int i = 0;i < totEdge;i ++){
		vv[side[i].u].push_back(side[i].v);
		if(flag == 0)vv[side[i].v].push_back(side[i].u);//wu xiang tu
	}
}

void getNC(){//undirected graph connected componet
	memset(vis,0,sizeof(vis));
	int totT = 0,maxNum = 0;
	queue<int>q;
	for(int i = 1;i <= siz;i ++){
		if(vis[i] == 0){
```

```
                vis[i] = 1;
                q.push(i);
                totT ++;
                int Tnum = 1;
                while(!q.empty()){
                        int now = q.front();q.pop();
                        int nn = vv[now].size();
                        for(int j = 0;j < nn;j ++){
                                if(vis[vv[now][j]])continue;
                                Tnum ++;
                                q.push(vv[now][j]);
                                vis[vv[now][j]] = 1;
                        }
                }
                maxNum = max(maxNum,Tnum);
        }
    }
    cout<<"the number of node of the max Connected component :"<<maxNum<<"the number of Connected component :"<<totT<<endl;
}

void getE(){//global efficiency
    double totDis = 0.0;
    for(int i = 1;i <= siz;i ++){
        memset(vis,0,sizeof(vis));
        queue<pair<int,int> >q;
        q.push(make_pair(i,0));
        vis[i] = 1;
        while(!q.empty()){
            pair<int,int> tmp = q.front();q.pop();
            if(tmp.second)totDis += 1.0/tmp.second;
            int now = tmp.first,nn = vv[now].size();
            for(int j = 0;j < nn;j ++){
                int xx = vv[now][j];
                if(vis[xx])continue;
                q.push(make_pair(xx,tmp.second+1));
                vis[xx] = 1;
            }
        }
    }
    cout<<"the average length of Shortest path of all nodes "<<totDis/siz/(siz-1)<<endl;
}
```

```
bool have(int b,int a){//judge whether have edge b->a or not
	int nn = vv[b].size();
	for(int i = 0;i < nn;i ++){
		if(vv[b][i] == a)return 1;
	}
	return false;
}

void getC(){//two kind of clustering coefficient
	//ppGraph();
	double C1 = 0,C2 = 0;
	long long totLian = 0;
	long long totTri = 0;
	for(int i = 1;i <= siz;i ++){
		int nn = vv[i].size();
		if(nn <= 1)continue;
		int tmpTri = 0;
		for(int one = 0;one < nn;one ++){
			for(int two = one + 1;two < nn;two ++){
				tmpTri += have(vv[i][two],vv[i][one]);
			}
		}
		double tmpC1 = (double)tmpTri*2/nn/(nn-1);
		C1 += tmpC1;

		totTri += tmpTri;
		totLian += nn*(nn-1)/2;
	}
	C1 /= siz;
	C2 = (double)totTri/totLian;
	cout<<"average Clustering coefficient of all nodes :"<<C1<<' '<<C2<<endl;
}

void getZ(){//average degree of nodes
	cout<<"average degree of nodes"<<(double)totEdge/siz<<endl;
}

//strongly connected component in directed graph
int t,dfn[MAXN],low[MAXN],sum;
int gNum[MAXN];
stack<int>s;
void dfs(int u){
	dfn[u] = low[u] = ++t;
	s.push(u);
```

```
	int nn = vv[u].size();
	for(int i = 0;i < nn;i ++){
		int v = vv[u][i];
		if(!dfn[v])dfs(v);
		if(dfn[v] != -1)low[u] = min(low[u],low[v]);
	}
	if(low[u] == dfn[u]){
		int v;
		do {
			v = s.top();s.pop();
			dfn[v] = -1;
			low[v] = sum;
			gNum[sum] ++;
		}while(v != u);
		sum ++;
	}
}
void getG(){
	t = sum = 0;
	for(int i = 1;i <= siz;i ++){
		if(!dfn[i])dfs(i);
	}
	int maxNum = 0;
	for(int i = 0;i < sum;i ++){
		maxNum = max(maxNum,gNum[i]);
	}
	cout<<"the number of nodes of the max Strongly connected component :"<<maxNum<<"the number of Strongly connected component"<<sum<<endl;
}
//strongly connected component in directed graph


void getR1(){//reciprocity
	int totm_d = 0;
	for(int i = 1;i <= siz;i ++){
		int nn = vv[i].size();
		for(int j = 0;j < nn;j ++){
			totm_d += have(vv[i][j],i);
		}
	}
	cout<<"the reciprocity is : "<<(double)totm_d/totEdge<<endl;
}
```

```
void Undir(){//get undirected graph parameters
	buildGraph(0);
	cout<<"Total Nodes is :"<<siz<<endl;
	getNC();
	getE();
	getC();
}

void Dir(){//get directed graph parameters
	buildGraph(1);
	cout<<"Total Nodes is :"<<siz<<endl;
	getZ();
	getG();
	getE();
	getR1();

	for(int i = 0;i <= siz;i ++)vv[i].clear();
	buildGraph(0);
	getC();
}

int main(){
	freopen("in.txt","r",stdin);
	//build graph and discretization
	int a,b;
	char s[100];cin>>s;
	totEdge = 0;
	while(~scanf("%d,%d",&a,&b)){
		side[totEdge] = (Side){a,b};
		hh[totEdge*2] = a;
		hh[totEdge*2+1] = b;
		totEdge ++;
	}
	sort(hh,hh+totEdge*2);
	siz = unique(hh,hh+totEdge*2) - hh;
	for(int i = 0;i < totEdge;i ++){
		side[i].u = lower_bound(hh,hh+siz,side[i].u) - hh + 1;
		side[i].v = lower_bound(hh,hh+siz,side[i].v) - hh + 1;
		//cout<<side[i].u<<' '<<side[i].v<<endl;
	}
	//ppDiscretization();
	//build graph and discretization

	//Undir();//Get Parameters of Undirected Graph
```

```
    Dir();//Get Parameters of Directed Graph
    return 0;
}
```

NodeXLGraph2

2015年12月11日 11:12

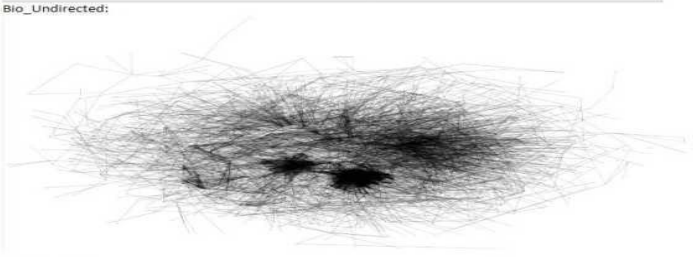
NodeXL Basic is a free and open-source network analysis and visualization software package for Microsoft Excel

来自 <https://en.wikipedia.org/wiki/NodeXL>

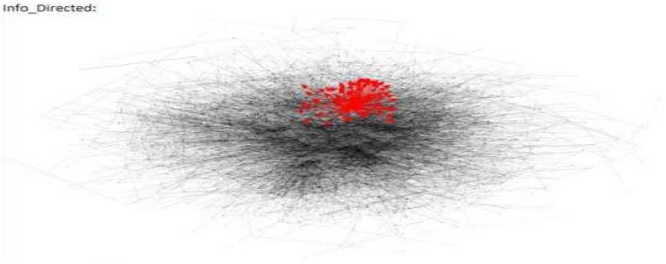
We use this plug-in analysis the 6 network given in attachment

Bio_Directed:

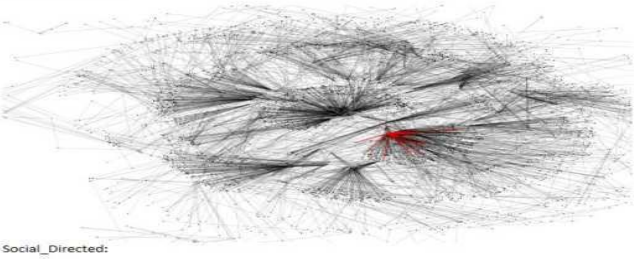
Bio_Undirected:



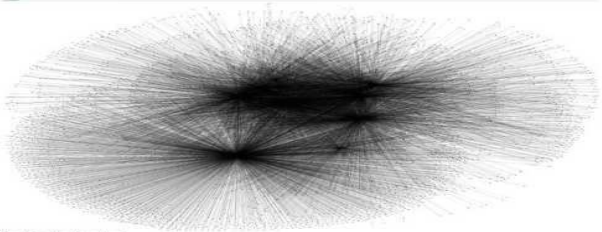
Info_Directed:



Info_Undirected:



Social_Directed:



Social_Undirected: