http://www.di.unito.it/~tizzani

michele.tizzani@unito.it

Michele Tizzani - Università degli Studi di Torino (Italy)

Lecture 09.ns06

Hubs, closeness, betweenness, robustness
Computing betweenness values

Course: **Complex Networks Analysis and Visualization**
Sub-Module: **NetSci**

References

[ns1] Chapter 3

[ns2] Chapter 3 (3.6.B) —->

https://www.cs.cornell.edu/home/kleinber/networks-book/networks-book-ch03.pdf

Agenda

- Network measures (Chapter 3: [ns1])
  - Centralities
  - Centrality distributions
  - The friendship paradox
  - Ultra-small worlds
  - Robustness
  - Core decomposition
- Calculating betweenness values (Chapter 3.6.B: [ns2])

Network measures

[ns1] Chapter 3: Hubs

Real networks are heterogeneous

Some nodes (and links) are much more important (**central**) than others!

A First Course in Network Science by F. Menczer, S. Fortunato & C.A. Davis. Cambridge University Press, 2020
© 2020 F. Menczer, S. Fortunato & C.A. Davis. cambridgeuniversitypress.github.io/FirstCourseNetworkScience

Centrality measures

- **Centrality**: measure of importance of a node
- **Measures**:

1. Degree
2. Closeness
3. Betweenness

Degree

- **Degree of a node:** number of neighbors of the node

$$k_i = \text{number of neighbors of node } i$$

- High-degree nodes are called **hubs**

- **Average degree of the network:**

$$\langle k \rangle = \frac{\sum_i k_i}{N} = \frac{2L}{N}$$

```
G.degree(2) # returns the degree of node 2
G.degree()  # dict with the degree of all nodes of G
```

Closeness

Idea: a node is the more central the *closer* it is to the other nodes, on average

$$g_i = \frac{1}{\sum_{j \neq i} \ell_{ij}}$$

where ℓ_{ij} is the distance between nodes *i* and *j*

```
nx.closeness_centrality(G, node) # closeness centrality
                                 # of node
```

Betweenness

Idea: a node is the more central the *more often it is crossed by paths*

$$b_i = \sum_{h \neq j \neq i} \frac{\sigma_{hj}(i)}{\sigma_{hj}}$$

σ_{hj} = number of shortest paths from h to j

$\sigma_{hj}(i)$ = number of shortest paths from h to j running through i

Betweenness

Hubs usually have high betweenness, but there can be nodes with high betweenness **that are not hubs**

A First Course in Network Science by F. Menczer, S. Fortunato & C.A. Davis. Cambridge University Press, 2020
© 2020 F. Menczer, S. Fortunato & C.A. Davis. cambridgeuniversitypress.github.io/FirstCourseNetworkScience

Betweenness

- Betweenness can be easily extended to links
- **Link betweenness**: fraction of shortest paths among all possible node pairs that pass through the link

```
nx.betweenness_centrality(G)       # dict nodes ->
                                   # betweenness centrality
nx.edge_betweenness_centrality(G) # dict links ->
                                   # betweenness centrality
```

Centrality distributions

- On small networks it makes sense to ask which nodes or links are most important
- On large networks **it does not**
- **Solution:** statistical approach
- Instead of focusing on individual nodes and links, we consider **classes** of nodes and links with similar properties

© 2020 F. Menczer, S. Fortunato & C.A. Davis. cambridgeuniversitypress.github.io/FirstCourseNetworkScience

Centrality distributions

n_k = number of nodes with degree k

$f_k = \dfrac{n_k}{N}$ = frequency of degree k

A First Course in Network Science by F. Menczer, S. Fortunato & C.A. Davis. Cambridge University Press, 2020
© 2020 F. Menczer, S. Fortunato & C.A. Davis. cambridgeuniversitypress.github.io/FirstCourseNetworkScience

Centrality distributions

- For large N, the frequency f_k becomes the **probability** p_k of having degree k

- **Probability distribution:** plot of probability p_k versus k

A First Course in Network Science by F. Menczer, S. Fortunato & C.A. Davis. Cambridge University Press, 2020
© 2020 F. Menczer, S. Fortunato & C.A. Davis. cambridgeuniversitypress.github.io/FirstCourseNetworkScience

Cumulative distributions

- If the variable is *not integer* (e.g., betweenness), the range of the variable is divided into intervals (bins) and we count how many values fall in each interval

- **Cumulative distribution *P(x)*:** probability that the variable takes values *larger* than *x* as a function of *x*

- **How to compute it:** by summing the frequencies of the variable inside the intervals to the right of *x*

$$P(x) = \sum_{v \geq x} f_v$$

Logarithmic scale

- **Question:** how to plot a probability distribution if the variable spans a large range of values, from small to (very) large?

- **Answer:** use the **logarithmic** scale

- **How to do it:** report the logarithms of the values on the x- and y-axes

$$\log_{10} 10 = 1$$

$$\log_{10} 1{,}000 = \log_{10} 10^3 = 3$$

$$\log_{10} 1{,}000{,}000 = \log_{10} 10^6 = 6$$

Degree distributions

Heavy-tail distributions: the variable goes from small to large values

A First Course in Network Science by F. Menczer, S. Fortunato & C.A. Davis. Cambridge University Press, 2020
© 2020 F. Menczer, S. Fortunato & C.A. Davis. cambridgeuniversitypress.github.io/FirstCourseNetworkScience

Degree distributions

- The **heterogeneity parameter** says how broad the distribution is:

$$\kappa = \frac{\langle k^2 \rangle}{\langle k \rangle^2}$$

$$\langle k \rangle = \frac{\sum_i k_i}{N} = \frac{2L}{N}; \; \langle k^2 \rangle = \frac{\sum_i k_i^2}{N}$$

- If most degrees have the same value, say k_0:

$$\langle k \rangle \approx k_0, \; \langle k^2 \rangle \approx k_0^2 \implies \kappa \approx 1$$

- If the distribution is very heterogeneous: $\kappa \gg 1$

Betweenness distributions

Heavy-tail distribution: the variable goes from small to large values

A First Course in Network Science by F. Menczer, S. Fortunato & C.A. Davis. Cambridge University Press, 2020
© 2020 F. Menczer, S. Fortunato & C.A. Davis. cambridgeuniversitypress.github.io/FirstCourseNetworkScience

Degree centrality

Network	Nodes (N)	Links (L)	Average degree ($\langle k \rangle$)	Maximum degree (k_{max})	Heterogeneity parameter (κ)
Facebook Northwestern Univ.	10,567	488,337	92.4	2,105	1.8
IMDB movies and stars	563,443	921,160	3.3	800	5.4
IMDB co-stars	252,999	1,015,187	8.0	456	4.6
Twitter US politics	18,470	48,365	2.6	204	8.3
Enron Email	36,692	367,662	10.0	1,383	14.0
Wikipedia math	15,220	194,103	12.8	5,171	38.2
Internet routers	190,914	607,610	6.4	1,071	6.0
US air transportation	546	2,781	10.2	153	5.3
World air transportation	3,179	18,617	11.7	246	5.5
Yeast protein interactions	1,870	2,277	2.4	56	2.7
C. elegans brain	297	2,345	7.9	134	2.7
Everglades ecological food web	69	916	13.3	63	2.2

A First Course in Network Science by F. Menczer, S. Fortunato & C.A. Davis. Cambridge University Press, 2020
© 2020 F. Menczer, S. Fortunato & C.A. Davis. cambridgeuniversitypress.github.io/FirstCourseNetworkScience

Friendship paradox

- By choosing *nodes at random*, Tom has **the same chance** to be picked as everybody else
- By choosing *links at random*, Tom has a **higher chance** to be picked than everybody else

Friendship paradox

By following links, **the chance to hit a hub increases**

A First Course in Network Science by F. Menczer, S. Fortunato & C.A. Davis. Cambridge University Press, 2020
© 2020 F. Menczer, S. Fortunato & C.A. Davis. cambridgeuniversitypress.github.io/FirstCourseNetworkScience

Friendship paradox

- Average degree of a node = **2.29**
- Average degree of the neighbors of a node = **2.83 > 2.29**

- *Our friends have more friends than we do, on average* (**friendship paradox**)

A First Course in Network Science by F. Menczer, S. Fortunato & C.A. Davis. Cambridge University Press, 2020
© 2020 F. Menczer, S. Fortunato & C.A. Davis. cambridgeuniversitypress.github.io/FirstCourseNetworkScience

Friendship paradox

- **Question:** Where does the friendship paradox come from?

- **Answer:**
  1. By averaging the degree of the nodes, we pick them at random
  2. By averaging the degree of the neighbors, we choose them by following links: nodes with degree k will be counted k times, which inflates the average

- The *more hubs, the stronger* the effect

Ultra-small worlds

- In real networks, many shortest paths go through hubs
- **Example:** air transportation
- There may be no routes between airport A and B (if they are small), but it may be possible to go from A to B via a hub airport C
- The small-world property is typical of most networks of interest: if the network has hubs, paths are ultra-short (**ultra-small world**)

Ultra-small worlds

Shortest-path length distribution

A First Course in Network Science by F. Menczer, S. Fortunato & C.A. Davis. Cambridge University Press, 2020
© 2020 F. Menczer, S. Fortunato & C.A. Davis. cambridgeuniversitypress.github.io/FirstCourseNetworkScience

Robustness

- A system is **robust** if the failure of some of its components does not affect its function
- **Question:** how can we define the robustness of a network?
- **Answer:** we remove nodes and/or links and see what happens to its structure
- **Key point:** *connectedness*
- If the Internet were not connected, it would be impossible to transmit signals (e.g., emails) between routers in different components

Robustness

- **Robustness test:** checking how the connectedness of the network is affected as more and more nodes are removed

- **How to do it:** plot the relative size S of the largest connected component as a function of the fraction of removed nodes

- We suppose that the network is initially connected: there is only one component and $S = 1$

- As more and more nodes (and their links) are removed, the network is progressively broken up into components and S goes down

Robustness

A First Course in Network Science by F. Menczer, S. Fortunato & C.A. Davis. Cambridge University Press, 2020
© 2020 F. Menczer, S. Fortunato & C.A. Davis. cambridgeuniversitypress.github.io/FirstCourseNetworkScience

Robustness

- **Two strategies:**
  1. **Random failures:** nodes break down randomly, so they are all chosen with the **same probability**
  2. **Attacks:** hubs are deliberately targeted — the larger the **degree**, the higher the probability of removing the node

- In the first approach, we remove a fraction *f* of nodes, chosen at random

- In the second approach, we remove the fraction *f* of nodes with largest degree, from the one with largest degree downwards

Robustness

Conclusion: real networks are robust against random failures but fragile against targeted attacks!

© 2020 F. Menczer, S. Fortunato & C.A. Davis. cambridgeuniversitypress.github.io/FirstCourseNetworkScience

Core decomposition

- **Core:** dense part of the network, with high-degree nodes

- **Core decomposition:** procedure to identify denser and denser cores, by removing nodes of progressively higher degree. If we remove all nodes with degree k–1 or lower, the remaining portion of the network is called ***k*-core**

- ***k*-core decomposition procedure:** start with k=0
  1. Recursively remove all nodes with degree k, until none are left
  2. The set of removed nodes is the ***k*-th shell**, while the remaining ones form the **(*k*+1)-core**
  3. If there is no node left, terminate. Otherwise, increment k by one and repeat from step 1

Core decomposition

A First Course in Network Science by F. Menczer, S. Fortunato & C.A. Davis. Cambridge University Press, 2020
© 2020 F. Menczer, S. Fortunato & C.A. Davis. cambridgeuniversitypress.github.io/FirstCourseNetworkScience

Core decomposition

Core decomposition helps to visualize large networks, by pruning low-degree nodes and showing only the densest parts

```
nx.core_number(G)    # return dict with core number of each node
nx.k_shell(G,k)      # subnetwork induced by nodes in k-shell
nx.k_core(G,k)       # subnetwork induced by nodes in k-core
nx.k_core(G)         # innermost (max-degree) core subnetwork
```

Calculating betweenness

[ns2] Chapter 3.6: Advanced Material: Betweenness Measures and Graph Partitioning

Computing edges' betweenness

For each node $i \in N$

1. perform **BFS**
2. count **the number of shortest paths** from i
3. calculate **the amount of 'flow'** from i

Ulrik Brandes (2001) A faster algorithm for betweenness centrality, The Journal of Mathematical Sociology, 25:2, 163-177, DOI: 10.1080/0022250X.2001.9990249

Step 1: BFS from node *a*

Top-Down

Step 1: BFS from node *a*

Layer 1

Top-Down

Step 1: BFS from node *a*

Layer 1

Layer 2

Top-Down

Step 1: BFS from node *a*

Layer 1

Layer 2

Layer 3

Step 1: BFS from node *a*

Layer 1

Layer 2

Layer 3

Layer 4

Step 2: count the # shortest paths from *a*

Bottom-Up

Step 2: count the # shortest paths from *a*

$\sigma_{ak} = \sigma_{ai} + \sigma_{aj}$

Bottom-Up

Step 2: count the # shortest paths from a

Bottom-Up

Step 2: count the # shortest paths from a

Bottom-Up

Step 2: count the # shortest paths from a

Top-Down

Step 2: count the # shortest paths from a

Top-Down

Step 2: count the # shortest paths from a

Top-Down

Step 2: count the # shortest paths from a

Top-Down

Step 3: computing the "flow"

For each node *i* in the BFS structure, we add up all the flow arriving from edges directly below *i*, plus 1 for the flow destined for *i* itself.

We then divide this up over the edges leading upward from *i*, *in proportion to the number of shortest paths* coming *through each*.

Bottom-Up

Step 3: computing the "flow"

For each node i in the BFS structure, we add up all the flow arriving from edges directly below i, plus 1 for the flow destined for i itself.

We then divide this up over the edges leading upward from i, *in proportion to the number of shortest paths* coming *through each*.

Bottom-Up

Step 3: computing the "flow"

For each node i in the BFS structure, we add up all the flow arriving from edges directly below i, plus 1 for the flow destined for i itself.

We then divide this up over the edges leading upward from i, *in proportion to the number of shortest paths* coming *through each*.

Bottom-Up

Step 3: computing the "flow"

For each node i in the BFS structure, we add up all the flow arriving from edges directly below i, plus 1 for the flow destined for i itself.

We then divide this up over the edges leading upward from i, *in proportion to the number of shortest paths* coming *through each*.

1

Bottom-Up

Step 3: computing the "flow"

For each node *i* in the BFS structure, we add up all the flow arriving from edges directly below *i*, plus 1 for the flow destined for *i* itself.

We then divide this up over the edges leading upward from *i*, *in proportion to the number of shortest paths* coming *through each*.

Bottom-Up

Output: betweenness for each edge

- Repeat for all the nodes in G
- For each edge (i,j): b_{ij} = sum up all the 'flow' values / 2
- Complexity of Girvan-Newman:
  - $O(N^3)$ for dense graphs
  - $O(N^2)$ for sparse graphs
- We can do better… Leuven Algorithm: $O(N)$

Additional readings

- Linton C. Freeman, Centrality in social networks. A conceptual clarification, Social networks, vol. 1, no 3, 1979, p. 215–239
  https://www.bebr.ufl.edu/sites/default/files/Centrality%20in%20Social%20Networks.pdf
- Ulrik Brandes (2001) A faster algorithm for betweenness centrality, The Journal of Mathematical Sociology, 25:2, 163-177
  http://www.uvm.edu/pdodds/research/papers/others/2001/brandes2001a.pdf
- Newman, Mark EJ, and Michelle Girvan. "Finding and evaluating community structure in networks." Physical review E 69.2 (2004): 026113
  http://www.cse.cuhk.edu.hk/~cslui/CMSC5734/newman_community_struct_networks_phys_rev.pdf