Graphs / Networks
Centrality measures, algorithms, interactive applications

Duen Horng (Polo) Chau
Assistant Professor
Associate Director, MS Analytics
Georgia Tech

Partly based on materials by Professors Guy Lebanon, Jeffrey Heer, John Stasko, Christos Faloutsos, Parishit Ram (GT PhD alum; SkyTree), Alex Gray
Centrality = “Importance”
Why Node Centrality?

What can we do if we can rank all the nodes in a graph (e.g., Facebook, LinkedIn, Twitter)?

- Find **celebrities** or influential people in a social network (Twitter)
- Find “**gatekeepers**” who connect communities (headhunters love to find them on LinkedIn)
- What else?
Why Node Centrality?

Helps graph analysis, visualization, understanding, e.g.,

• Let us rank nodes, group or study them by centrality
• Only show subgraph formed by the top 100 nodes, out of the millions in the full graph
  • Similar to google search results (ranked, and they only show you 10 per page)
  • Most graph analysis packages already have centrality algorithms implemented. Use them!

Can also compute edge centrality. Here we focus on node centrality.
Degree Centrality (easiest)

Degree = number of neighbors

• For directed graphs
  • **In degree** = No. of incoming edges
  • **Out degree** = No. of outgoing edges

• For undirected graphs, **only degree is defined**.

• Algorithms?
  • Sequential scan through edge list
  • What about for a graph stored in SQLite?
Computing Degrees using SQL

Recall simplest way to store a graph in SQLite:

```
edges(source_id, target_id)
```

1. If slow, first create index for each column
2. Use `group by` statement to find in degrees

```
select count(*) from edges group by source_id;
```
Betweenness Centrality

High betweenness = “gatekeeper”

Betweenness of a node $v$

$$\sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

= how often a node serves as the “bridge” that connects two other nodes.

Betweenness is very well studied. [http://en.wikipedia.org/wiki/Centrality#Betweenness_centrality](http://en.wikipedia.org/wiki/Centrality#Betweenness_centrality)
(Local) Clustering Coefficient

A node’s clustering coefficient is a measure of how close the node’s neighbors are from forming a clique.

- 1 = neighbors form a clique
- 0 = No edges among neighbors

(Assuming undirected graph)

“Local” means it’s for a node; can also compute a graph’s “global” coefficient

Computing Clustering Coefficients...

Requires triangle counting

Real social networks have a lot of triangles
  • Friends of friends are friends

Triangles are expensive to compute
  (neighborhood intersections; several approx. algos)

Can we do that quickly?

Algorithm details:
Faster Clustering Coefficient Using Vertex Covers
http://www.cc.gatech.edu/~ogreen3/_docs/2013VertexCoverClusteringCoefficients.pdf
Super Fast Triangle Counting
[Tsourakakis ICDM 2008]

But: triangles are expensive to compute
(3-way join; several approx. algos)
Q: Can we do that quickly?
A: Yes!

#triangles = 1/6 \text{ Sum } ( \lambda_i^3 )

(and, because of skewness,
we only need the top few eigenvalues!)
Power Law in Eigenvalues of Adjacency Matrix

Eigenvalue

Rank of decreasing eigenvalue

Eigen exponent = slope = -0.48
Wikipedia graph 2006-Nov-04
≈ 3.1M nodes ≈ 37M edges

1000x+ speed-up, >90% accuracy

Accuracy (%) vs. Speedup

1000x, 97.4%
1277x, 94.7%
1329x, 92.8%

1000x+ speed-up, >90% accuracy
More Centrality Measures...

- Degree
- Betweenness
- Closeness, by computing
  - Shortest paths
- “Proximity” (usually via random walks) — used successfully in a lot of applications
- Eigenvector
- ...
Page Rank (Google)

PageRank: Problem

Given a directed graph, find its most interesting/central node

A node is important, if it is connected with important nodes (recursive, but OK!)
PageRank: Solution

Given a directed graph, find its most interesting/central node

Proposed solution:
use **random walk**; spot most “popular” node
(-> **steady state probability (ssp)**)

PageRank: Solution

A node has high **ssp**, if it is connected with **high ssp nodes** (recursive, but OK!)

“state” = webpage
Let $B$ be the transition matrix: transposed, column-normalized

(Simplified) PageRank
(Simplified) PageRank

B p = p

How to compute SSP:
http://www.sosmath.com/matrix/markov/markov.html
(Simplified) PageRank

- $B \ p = 1 \ * \ p$
- Thus, $p$ is the eigenvector that corresponds to the highest eigenvalue (=1, since the matrix is column-normalized)
- Why does such a $p$ exist?
  - $p$ exists if $B$ is nxn, nonnegative, irreducible [Perron–Frobenius theorem]
(Simplified) PageRank

- In short: imagine a particle/person randomly moving along the edges/links
- Compute its steady-state probability (ssp)

Full version of algorithm:
  With occasional random jumps to any nodes

Why? To make the matrix irreducible.
  Irreducible = from any state, there’s non-zero probability to reach any other state
With probability 1-c, fly-out to a random node

Then, we have

\[ p = c B p + \frac{(1-c)}{n} 1 \]

\[ p = \frac{(1-c)}{n} [I - c B]^{-1} 1 \]
How to compute PageRank for huge matrix?

Use the power iteration method

\[ \mathbf{p} = c \mathbf{B} \mathbf{p} + \frac{(1-c)}{n} \mathbf{1} \]

Can initialize this vector to any non-zero vector, e.g., all “1”s.
PageRank Explained with Javascript

Also great for checking the correctness of your PageRank Implementation.

http://www.cs.duke.edu/csed/principles/pagerank/
PageRank for graphs (generally)

You can compute PageRank for any graphs. Should be in your algorithm “toolbox”

- Better than simple centrality measure (e.g., degree)
- Fast to compute for large graphs (O(E))

But can be “misled” (Google Bomb)

- How?
Personalized PageRank

Make one small variation of PageRank

• Intuition: not all pages are equal, some more relevant to a person’s specific needs

• How?
With probability $1-c$, fly-out to a random node some preferred nodes

Can initialize this vector to any non-zero vector, e.g., all “1”s
Why learn Personalized PageRank?

For recommendation

• If I like webpage A, what else do I like?
• If I bought product A, what other products would I also buy?

Visualizing and interacting with large graphs

• Instead of visualizing every single nodes, visualize the most important ones

Very flexible — works on any graph
Related “guilt-by-association” / diffusion techniques

• **Personalized PageRank**
  (= Random Walk with Restart)

• “Spreading activation” or “degree of interest” in Human-Computer Interaction (HCI)

• Belief Propagation
  (powerful inference algorithm, for fraud detection, image segmentation, error-correcting codes, etc.)
Why are these algorithms popular?

- Intuitive to interpret
  uses “network effect”, homophily
- Easy to implement
  Math is relatively simple (mainly matrix-vector multiplication)
- Fast
  run time linear to #edges, or better
- Probabilistic meaning
Human-In-The-Loop Graph Mining

**Apolo:** Machine Learning + Visualization

*CHI 2011*

Apolo: Making Sense of Large Network Data by Combining Rich User Interaction and Machine Learning
Finding More Relevant Nodes

Apolo uses guilt-by-association (Belief Propagation, similar to personalized PageRank)
**Demo: Mapping the Sensemaking Literature**

**Nodes:** 80k papers from Google Scholar (node size: #citation)

**Edges:** 150k citations
The cost structure of sensemaking


245 citations 8 versions
Key Ideas (Recap)

Specify **exemplars**

Find **other** relevant nodes (BP)
Apolo’s Contributions

1. Human + Machine
   It was like having a partnership with the machine.

2. Personalized Landscape
Apolo 2009
Apolo 2011

22,000 lines of code. Java 1.6. Swing. Uses SQLite3 to store graph on disk.
User Study

Used citation network

Task: Find related papers for 2 sections in a survey paper on *user interface*

- Model-based generation of UI
- Rapid prototyping tools
Between subjects design
Participants: grad student or research staff
Apolo wins.

* Statistically significant, by two-tailed t test, $p < 0.05$
Apolo: Recap

A mixed-initiative approach for exploring and creating personalized landscape for large network data

Apolo = ML + Visualization + Interaction
Practitioners’ guide to building (interactive) applications

Think about scalability early

- Identify candidate scalable algorithms early on

Use iterative design approach, as in Apolo and industry

- Why? It’s hard to get it right the first time
- Create prototype, evaluate, modify prototype, evaluate, ...
- Quick evaluation helps you identify important fixes early — save you a lot of time overall
What kinds of prototypes?

• Paper prototype, lo-fi prototype, high-fi prototype

Important to involve REAL users as early as possible

• Recruit your friends to try your tools
• Lab study (controlled, as in Apolo)
• Longitudinal study (usage over months)
• Deploy it and see the world’s reaction!

• To learn more:
  • CS 6750 Human-Computer Interaction
  • CS 6455 User Interface Design and Evaluation
If you want to know more about people...  
http://amzn.com/0321767535