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)?
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:

\[
\text{edges(source_id, target_id)}
\]

1. If slow, first create index for each column
2. Use \texttt{group by} statement to find \texttt{out degrees}
   
   \[
   \text{select count(*) from edges group by source_id;}
   \]
High betweenness = “gatekeeper”

Betweenness of a node $v$

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

- Number of shortest paths between $s$ and $t$ that goes through $v$
- Number of shortest paths between $s$ and $t$

= 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
(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 Sum ( λ_i^3 )

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

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

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
- …
PageRank (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; most “popular” nodes are the ones with highest steady state probability (ssp)

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

“state” = webpage
**(Simplified) PageRank**

Let $B$ be the transition matrix: transposed, column-normalized

$$
\begin{array}{c}
1 \\
1 \\
1/2 \\
1/2 \\
1/2 \\
\end{array}
\begin{array}{c}
1 \\
1 \\
1/2 \\
1/2 \\
1/2 \\
\end{array}
\begin{array}{c}
p_1 \\
p_2 \\
p_3 \\
p_4 \\
p_5 \\
\end{array}
= 
\begin{array}{c}
p_1 \\
p_2 \\
p_3 \\
p_4 \\
p_5 \\
\end{array}
$$

How to compute SSP:
http://www.sosmath.com/matrix/markov/markov.html
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 person randomly moving along the edges/links
• A node’s PageRank score is the steady-state probability (ssp) of finding the person at that node

Full version of algorithm:

   With occasional random jumps to any nodes

Why? To make the matrix irreducible.

   Irreducible = from any state (node), there’s non-zero probability to reach any other state (node)
Full Algorithm

With probability $1-c$, fly-out to a random node.

Then, we have

$$p = c \mathbf{B} p + \frac{(1-c)}{n} \mathbf{1}$$
How to compute PageRank for huge matrix?

Use the power iteration method

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

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 run PageRank on any graphs

- All you need are the graph edges!

Should be in your algorithm “toolbox”

- Better than degree centrality
- Fast to compute for large graphs, runtime linear in the number of edges, $O(E)$

But can be “misled” (Google Bomb)

- How?
Personalized PageRank

Intuition: not all pages are equal, some more relevant to some people

Goal: rank pages in a way that those more relevant to you will be ranked higher

How? Make just one small change to PageRank
Personalized PageRank

With probability 1-c, fly-out to a random node some preferred nodes

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

\[ \begin{array}{cccc}
p_1' & 1 & 1 & 1/2 & 1/2 & 1/2 \\
p_2' & 1 & 1 & 1/2 & 1/2 & 1/2 \\
p_3' & 1 & 1 & 1/2 & 1/2 & 1/2 \\
p_4' & 1 & 1 & 1/2 & 1/2 & 1/2 \\
p_5' & 1 & 1 & 1/2 & 1/2 & 1/2 \\
\end{array} \]

\[ \begin{array}{cccc}
p_1 & 1 & 1 & 1 & 1 \\
p_2 & 0 & 0 & 0 & 0 \\
p_3 & 0 & 0 & 0 & 0 \\
p_4 & 0 & 0 & 0 & 0 \\
p_5 & 0 & 0 & 0 & 0 \\
\end{array} \]

\[ = 0.8 \]

Default value for c

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
Finding More Relevant Nodes

Citation network

HCI Paper

Data Mining Paper
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
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

End User Programming
End users creating effective software...
End user software engineering: chi...
Invited research overview: end-user...
Brad A. Myers
Margaret M. Burnett
Mary Beth Rosson
Andrew Jensen Ko
Alan F. Blackwell

Text Entry
In-stroke word completion.
Integrating isometric joysticks into...
Eyes on the road, hands on the wheel...
An alternative to push, press, and...
Maximizing the guessability of symb...
Few-key text entry revisited: mnemon...
Text entry from power wheelchairs: ...
Joystick text entry with date stamp, ...

Interface Generation
Huddle: automatically generating i...
UNIFORM: automatically generating...
Demonstrating the viability of auto...
Jeffrey Nichols
Brandon Rothrock
Duen Horng Chau

Not Interested
Automatically generating user inte...
Decision-Theoretic User
Daniel S. Weld
Krzysztof Z. Gajos
Automatically generating ...
Exploring the design space...
Predictability and accuracy...

Brad
Brad A. Myers
The garnet user interface developm...
Using HCI Techniques to Design a M...
Creating charts by demonstration.
The Amulet User Interface Developm...
Easily Adding Animations to Interfac...
Simplifying video editng using metad...
SILVER: simplifying video editing wit...
Apolo 2010
Apolo 2011

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

The cost structure of sensemaking


245 citations 8 versions
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

*Past, Present and Future of User Interface Software Tools*

Brad Myers, Scott E. Hudson, and Randy Pausch

Human Computer Interaction Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3891
Between subjects design
Participants: grad student or research staff
"Model-based"
"Prototyping"

10 papers for each section
Apolo

Google Scholar

“Model-based”

“Prototyping”

10 papers for each section

Expert judges rated papers

\[
\begin{align*}
1 + 0 &= 1 \\
1 + 1 &= 2 \\
0 + 0 &= 0 \\
&\vdots
\end{align*}
\]
Higher is better.

Apolo wins.

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

Judges’ Scores

<table>
<thead>
<tr>
<th>Model-based</th>
<th>*Prototyping</th>
<th>*Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apolo</td>
<td>Scholar</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>8</td>
<td>16</td>
</tr>
</tbody>
</table>

*Prototyping* scores are statistically significant, by two-tailed t test, $p < 0.05$
Practitioners’ guide to building (interactive) applications

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
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
If you want to know more about people…

http://amzn.com/0321767535