Graphs / Networks
Centrality measures, algorithms, interactive applications

Duen Horng (Polo) Chau
Georgia Tech

Partly based on materials by Professors Guy Lebanon, Jeffrey Heer, John Stasko, Christos Faloutsos, Le Song
Joint IC/CSE Seminar Announcement
Thursday (2/20), 10-11am, TSRB Banquet Hall

Design Techniques for Crowdsourcing Complex Tasks

Edith Law
Harvard University
(Graduated from CMU)

Edith is a faculty candidate; she’ll talk about some of her best work
0.5% bonus point for attending.
Talk may be recorded.
Recap...

• **Last time:** Basics, how to build graph, store graph, laws, etc.

• **Today:** Centrality measures, algorithms, interactive applications for visualization and recommendation
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?
More generally

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 = # incoming edges
- out degree = # outgoing edges

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. Create index for each column
2. Use `group by` statement to find node degrees

```
select count(*) from edges group by source_id;
```
High betweenness
= important “gatekeeper” or liaison

Betweenness of a node \( v \)

\[
\text{Betweenness of } v = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}
\]

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

Requires triangle counting

Real social networks have a lot of triangles

• Friends of friends are friends

But: triangles are expensive to compute

(3-way join; several approx. algos)

Can we do that quickly?
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 = \frac{1}{6} \text{Sum} \left( (\lambda_i)^3 \right)
(and, because of skewness, we only need the top few eigenvalues!)
Wikipedia graph 2006-Nov-04
≈ 3.1M nodes ≈ 37M edges

1000x+ speed-up, >90% accuracy

1000 1050 1100 1150 1200 1250 1300 1350

Accuracy(%)
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; spot most ‘popular’ node (→ steady state probability (ssp))

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

“state” = webpage
(Simplified) PageRank

Let $B$ be the transition matrix: transposed, column-normalized
(Simplified) PageRank

\[ B \ p = p \]
(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 randomly moving along the edges
• compute its steady-state probability (ssp)

Full version of algorithm:
  with occasional random jumps
Why? To make the matrix irreducible
Full Algorithm

- With probability $1-c$, fly-out to a random node
- Then, we have

$$p = c \mathbf{B} p + \frac{(1-c)}{n} \mathbf{1} \Rightarrow$$

$$p = \frac{(1-c)}{n} [\mathbf{I} - c \mathbf{B}]^{-1} \mathbf{1}$$
Full Algorithm

- With probability $1-c$, fly-out to a random node
- Then, we have
  
  $$p = cBp + \frac{1-c}{n} \mathbf{1} \Rightarrow p = \frac{1-c}{n} (I - cB)^{-1} \mathbf{1}$$

![Diagram of a network with nodes and edges]
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?
“Personalizing” PageRank

• With probability $1-c$, fly-out to a random node some preferred nodes
• Then, we have

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

$$p = \frac{(1-c)}{n} [I - c B]^{-1} 1$$
Why learn Personalized PageRank?

Can be used for recommendation, e.g.,

- If I like this webpage, what would I also be interested?
- If I like this product, what other products I also like? (in a user-product bipartite graph)
- Also helps with visualizing large graphs
  - Instead of visualizing every single nodes, visualize the most important ones

Again, very flexible. Can be run on any graph.
Building an interactive application

Will show you an example application (Apolo) that uses a "diffusion-based" algorithm to perform recommendation on a large graph

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

- Belief Propagation
  (powerful inference algorithm, for fraud detection, image segmentation, error-correcting codes, etc.)

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

- Guilt-by-association techniques
Building an interactive application

Why diffusion-based algorithms are widely used?

• **Intuitive to interpret**
  uses “network effect”, homophily, etc.

• **Easy to implement**
  Math is relatively simple

• **Fast**
  run time linear to #edges, or better

• **Probabilistic** meaning
Human-In-The-Loop Graph Mining

Apolo: Machine Learning + Visualization

CHI 2011
Finding More Relevant Nodes

Citation network

HCI Paper

Data Mining Paper
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

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
The garnet user interface development

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

Not Interested
Automatically generating user interface...
Decision-Theoretic User Interface...
Daniel S. Weld
Krzysztof Z. Gajos
Automatically generating interface...
Exploring the design space of user interface...
Predictability and accuracy in interaction...

Brad
Brad A. Myers
The garnet user interface development
Using HCI Techniques to Design a Multi...
Creating charts by demonstration.
The Amulet User Interface Development
Easily Adding Animations to Interfaces...
Simplifying video editing using metadata...
SILVER: simplifying video editing with...

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

Show: All

Recommendations:
Apolo 2010
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
"Model-based"  

"Prototyping"  

10 papers for each section
Apolo

Google Scholar

"Model-based"

"Prototyping"

10 papers for each section

Expert judges rated papers

1 + 0 = 1

1 + 1 = 2

0 + 0 = 0

::
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?

Important that you pick a **good** problem!

- Otherwise, you solve a “non-problem”, and nobody cares

Think about scalability early

- e.g., picking a scalable algorithm early on

When building interactive applications, use **iterative** design approach (as in Apolo)

- 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** (can **save you a lot of time**)


Practitioners’ guide to building (interactive) applications?

How to do iterative design?
What kinds of prototypes?
  • Paper prototype, lo-fi prototype, high-fi prototype
What kinds of evaluation?
  • 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