

COMPUTER SCIENCE 51

Spring 2009
cs51.seas.harvard.edu

Prof. Greg Morrisett
Prof. Ramin Zabih

computer
science 51

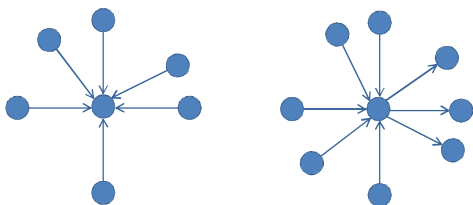


Importance in graphs

- What makes a node "important"?
 - More precisely: assign to each node a non-negative real number
 - Sum over all nodes is 1
 - Let's informally call a set of non-negative reals that sum to 1 a **probability distribution**
- Importance measures for web pages
 - PageRank (Page and Brin)
 - HITS (Kleinberg)



Some easy examples



Not a trivial problem

- Page with the most links to it?
 - Advantage: Easy to compute!
 - Disadvantage: Doesn't work very well
- Not all links are created equal
 - Having a link from the NYT or Harvard home page is much more important than a link from the CS51 home page
 - "A page is important if it has hyperlinks from important pages" (recursion!)
- Right math for an ill-defined problem

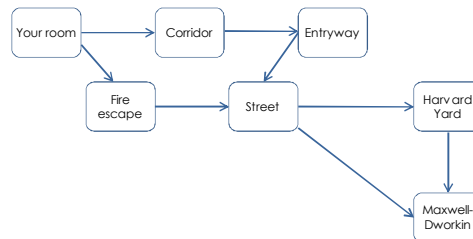


Walks in directed graphs

- Consider a "walk" on a graph
 - Start somewhere, take an outgoing edge
 - Do this repeatedly for a while
- Many intuitive examples of this
 - Quite important in advanced CS classes
 - Also: statistics, math, EE, etc.



Map example



- Note: only some edges shown in this example



Pirate-lite example

- "Generative model" for Pirate-speak
 - Reflects a **grammar** and **parser** also!

What is a walk on a graph?

- A walk from an initial node R on a graph is a function F such that
 - F goes from $t=\{0,1,2,\dots\}$ to nodes
 - $F(0) = R$
 - If $F(i) = v$ and $F(i+1) = v'$, then there is an edge from v to v'
- Note: HOP examples abound
 - Given a graph, return a walk
 - Given a graph and a function, check that function is a walk (to depth n)

What do walks represent?

- So far: permitted states of affairs
 - No path means not permitted
 - Ex: dorm room straight to Maxwell Dworkin
 - Ex: "Avast! Ye well-dressed CS51 faculty?!"
- It's natural for some edges to be more likely than others
 - Hopefully you rarely use the fire escape!
- Solution: probabilities

CS51 TF example

Rules for edge probabilities

- Intuitively, from a node you have to take one of the outgoing edges
 - Even if it goes back to you (self-loop)...
- Some edges are low probability
 - Example: terminal room to date = 10^{-6}
 - CS51 lecture to asleep = ???
- For each node, each outgoing edge will have a probability
 - Example: terminal room self-loop has probability .999999

Random walks on graphs

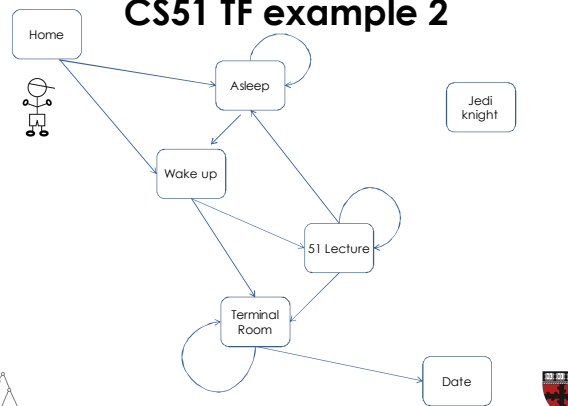
- We can now walk our graph by using a random number generator
 - Gives a number between 0 and 1.0
- Rule: to figure out where to go next, ask the random number generator
 - Example: $TF(7) = \text{terminal-room}$
 - If $\text{random} > 0.999999$, $TF(8) = \text{date}$
 - Far more likely that $TF(8) = \text{terminal-room}$
- "Markov chain" (see: Mark V. Shaney)

Importance?

- Where do random walks on this graph tend to end up?
 - This could be our importance measure!
- Let 100 TF's loose to do a random walk
 - Keep a count of each node visited
 - Stop each walk at depth 10 for simplicity (we'll relax this assumption shortly)
 - Divide the counts by 1000, and done
- Does this work?



CS51 TF example 2



Problem: rare states

- You need a huge amount of data!
 - Try running this on the web graph
 - Assume uniform outgoing probabilities
 - I.e., all hyperlinks are equally likely
- But this does have a nice intuition
 - Importance of a page is the frequency of visiting it on a random walk
 - We are close to Larry Page's idea
- Imitate infinite CS51 TF's?
 - Would this be ethical/affordable?

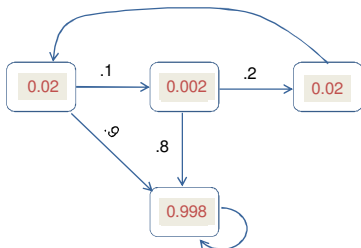


Solution: infinitely divisible TF's

- "Smear" the TF over the nodes
 - Somewhat like Schrodinger's cat
- Start with 100% of the TF at home
- At $t=1$, 95% asleep and 5% awake



Example



Application to web

- Nodes are web pages
 - Outgoing probabilities are uniform
 - i.e., 50-50 if page has 2 hyperlinks
- Start every page with same probability
- Models the behavior of a web surfer who picks random links
 - Importance of a page is how often such a surfer visits this page

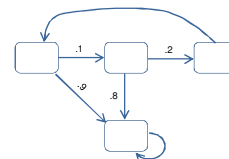


Update rule

- To compute the fractional TF for node v at time $t+1$:
 - Sum up all the ways to get here!
- Formally, look at each node w with an edge from w to v (**backlinks**)
 - At time t , the node w had fractional TF= x
 - It sends a portion of this to v
 - Multiply x by the probability of taking the edge from w to v

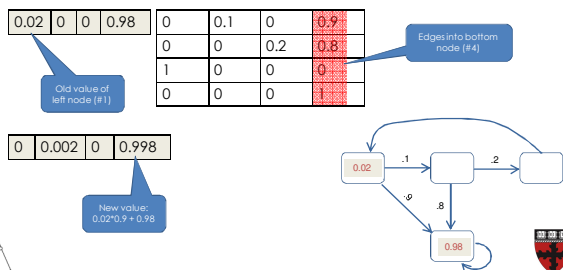
Implementation?

- List of current node probabilities
 - Ex: (list 0.02 0 0 0.98)
 - Invariant: sums to 1
- List of incoming edges per node
 - Ex: (list 0.9 0.8 0 1)
 - Invariant?
- MAP-REDUCE



Some suggestive notation

- Write the edge list vertically
 - One next to another, in an array



Questions for break

- Nice update rule
 - Probability distribution on nodes, at time t
 - As a function of distribution at time $t-1$
- Does it eventually stabilize?
 - Step(π^*) = π^*
- Does it matter where we start?
- Does it converge to something sane?
- How fast does it converge?

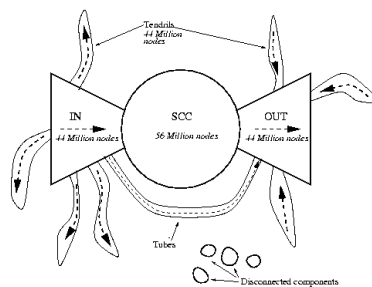
Convergence

- Does it always converge?
 - Is there an input where it loops?
 - Step(π) = π'
 - Step(π') = π



- Can we ensure unique convergence?

What does it converge to?



From: <http://www9.org/w9cdrom/160/160.html>
 Broder et al., 2000

Fixing both problems

- Add a small chance α of teleporting to a random web page
 - Think of this as evaporation/rainfall
 - Conceptually, graph is a clique!
- ♦ **Theorem:** unique convergence if graph is strongly connected, and has even a single self-loop
 - “Stationary distribution”



PageRank

- PageRank can be thought of as a model of user behavior. We assume there is a “random surfer” who is given a web page at random and keeps clicking on links, never hitting “back” but eventually gets bored and starts on another random page. The probability that the random surfer visits a page is its PageRank.
 - Brin and Page, “The Anatomy of a Large-Scale Hypertextual Web Search Engine”, <http://infolab.stanford.edu/~backrub/google.html>
- See also: Cleve Moler, “The World’s Largest Matrix Computation”, http://www.mathworks.com/company/newsletters/news_notes/clevescomer/oct02_cleve.html



Convergence speed?

- This is a surprisingly deep area
 - With many open questions
- There is (provably) no general way to rapidly find the stationary distribution
 - But some cases are known
- Big α leads to fast convergence
 - Obvious in the limit
- Google is widely believed to use .85
 - “Attention span” of about 6 links



Why no general solution?

- Suppose I give you a “black box” with a dial. Set dial, get a series of beeps.
- Question: can you find the dial setting that gets the most beeps?
 - Not without trying all of them (proof?)
- ♦ Fast way to compute an arbitrary stationary distribution would let you solve this problem!



Power method

- There are (many) clever ways to speed the computation
 - Algorithmic speedups are always much better than programming speedups
- One of the best ones involves exponentiation by repeated squaring
 - See: Greg's lecture Thursday



Extensions and alternatives

- Google versus Linkfarms
 - Simple solution: smart teleport
 - Personalization of search
- BadRank
- HITS (hubs and authorities)
 - Similar recursive algorithm
 - Hubs point to authorities

