CS4225/CS5425 Big Data Systems for Data Science

Graphs and PageRank

Bryan Hooi
School of Computing
National University of Singapore
bhooi@comp.nus.edu.sg

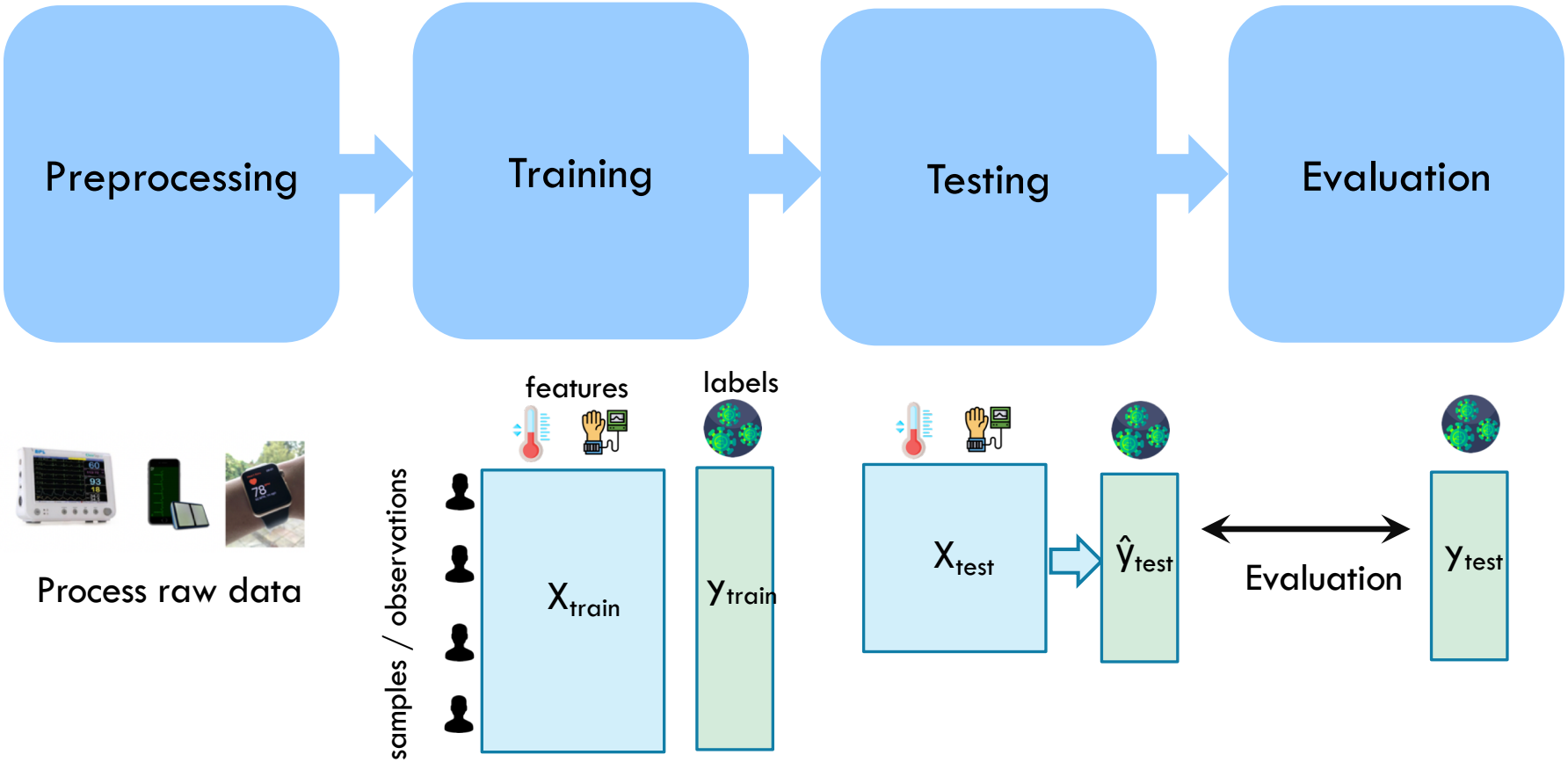
School of Computing

Announcements

- Tutorial on graph processing is today
- HW2 is due 31 Oct 11.59pm

Week	Date	Topics	Tutorial	Due Dates
1	12 Aug	Overview and Introduction		
2	19 Aug	MapReduce - Introduction		
3	26 Aug	MapReduce and Relational Databases		
4	2 Sep	MapReduce and Data Mining	Tutorial: Hadoop	
5	9 Sep	NoSQL Overview 1		Assignment 1 released
6	16 Sep	NoSQL Overview 2		
Recess				
7	30 Sep	Apache Spark 1	Tutorial: NoSQL & Spark	Assignment 2 released
8	7 Oct	Apache Spark 2		Assignment 1 due (13 Oct)
9	14 Oct	Large Graph Processing 1	Tutorial: Graph Processing	
10	21 Oct	Large Graph Processing 2		
11	28 Oct	Stream Processing	Tutorial: Stream Processing	Assignment 2 due (31 Oct)
12	4 Nov	Deepavali – No Class		
13	11 Nov	**Test**		

Recap: Typical Machine Learning Pipeline



Recap: Logistic Regression Prediction

Features

Number of '$'	Number of '!'
5	2

$x = \begin{pmatrix} 5 \\ 2 \end{pmatrix}$

(Predicted probability of the email being spam)

? $\hat{y}$

Parameters:

Weights $w = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$

1	2

Bias $b = -5$

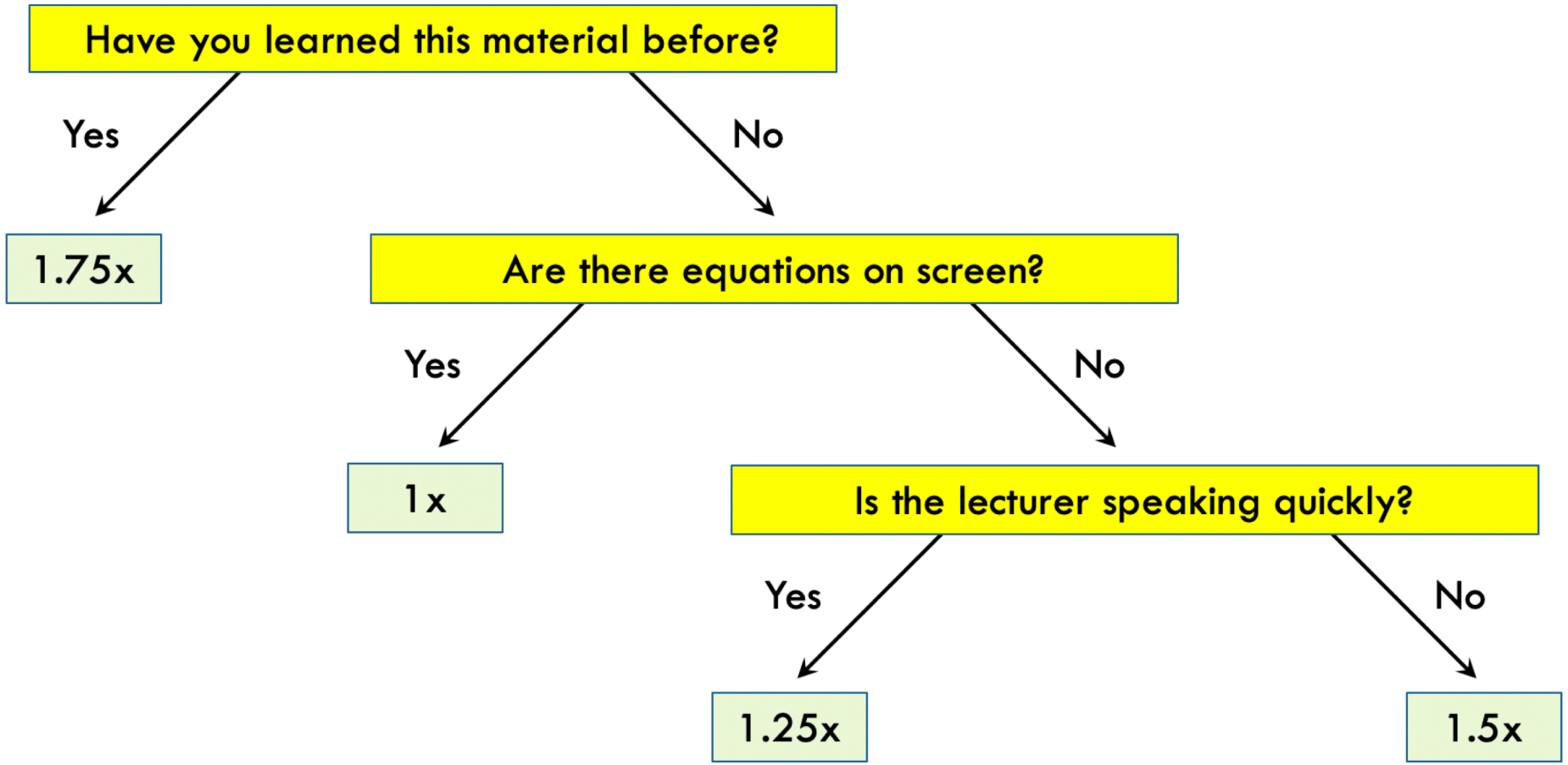
-5

Prediction:

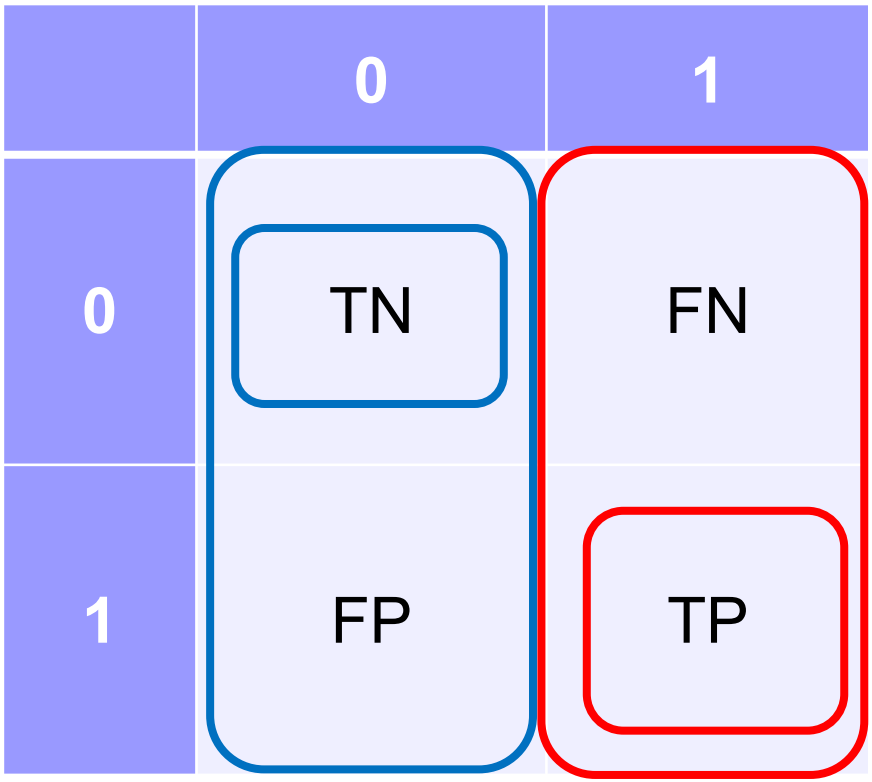
Sigmoid function, Dot product

$$\hat{y} = \sigma(x \cdot w + b) = \sigma\left(\begin{pmatrix} 5 \\ 2 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 2 \end{pmatrix} - 5\right) = \sigma(9 - 5) = \frac{1}{1 + e^{-4}} = 0.982$$

Recap: Decision Trees



Recap: Sensitivity / Specificity

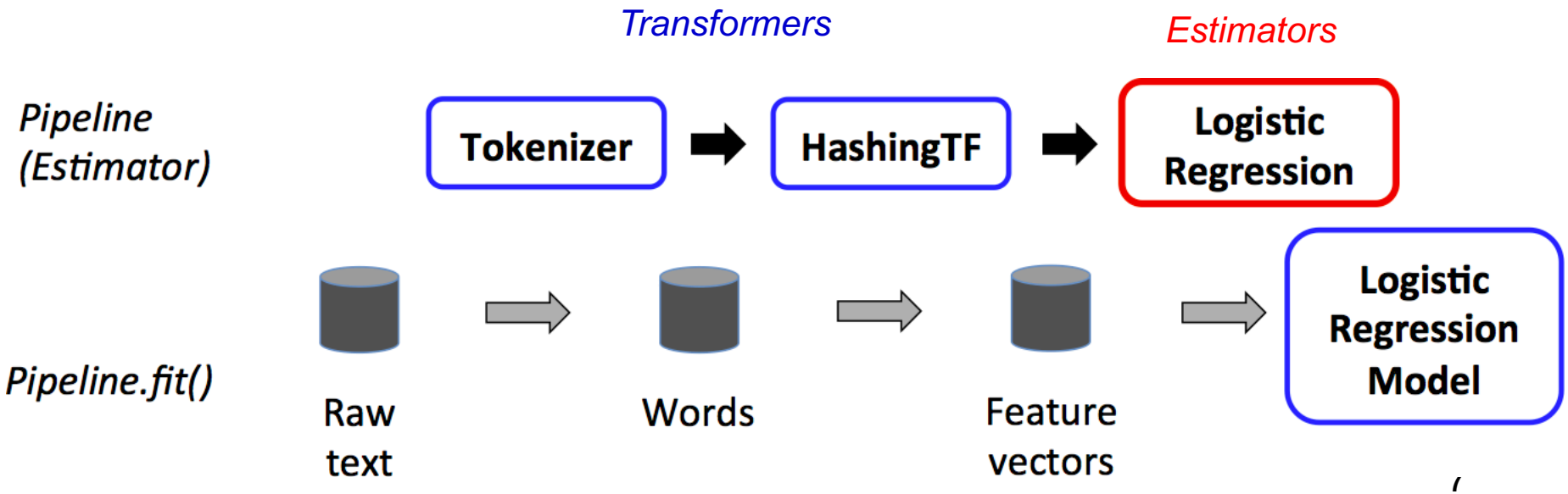


Sensitivity: fraction of positive cases that are detected
Specificity: fraction of actual negatives that are correctly identified

Q: Assume specificity = 97%; if I tested negative, does it mean my probability of being negative is 97%?
A: no (that is the wrong direction).

Recap: Spark Pipelines

- A pipeline chains together multiple Transformers and Estimators to form an ML workflow.
- It is an Estimator. When Pipeline.fit() is called:
  - Starting from the beginning of the pipeline:
  - For *Transformers*, it calls transform()
  - For *Estimators*, it calls fit() to fit the data, then transform() on the fitted model



Today's Plan

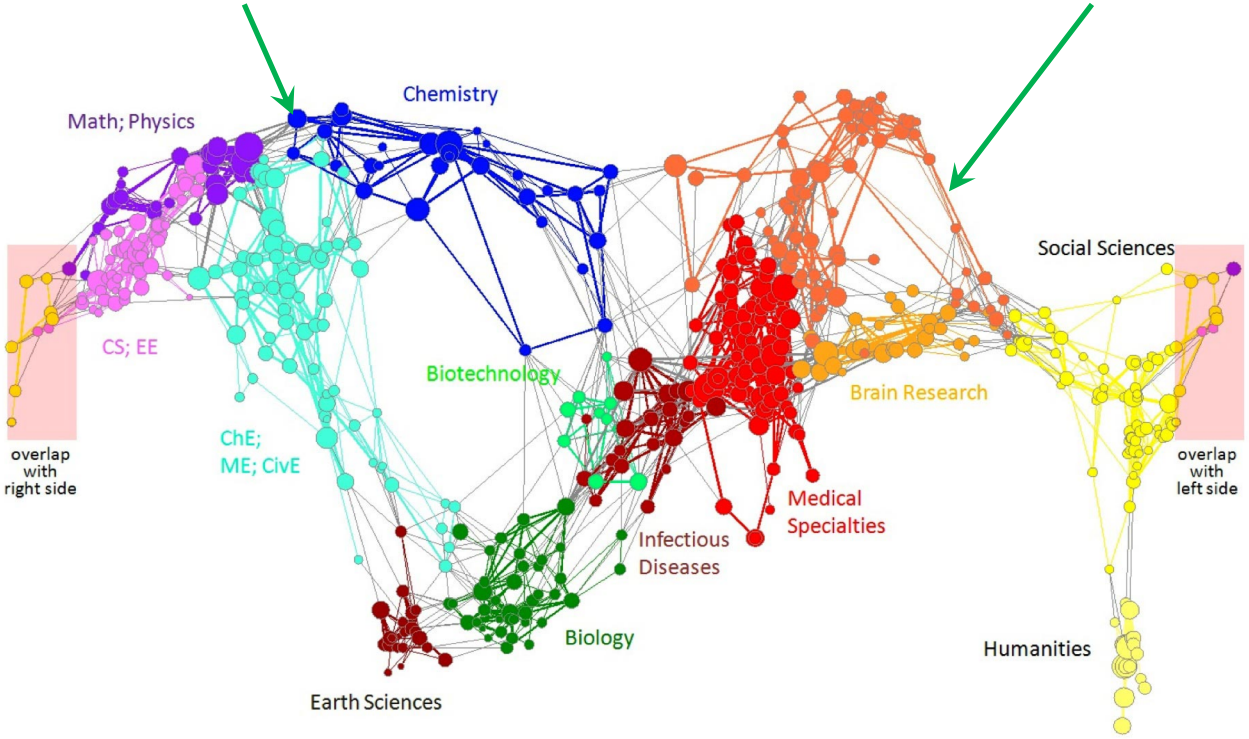
- **Graphs: Introduction**
- Simplified PageRank
  - Flow Formulation
  - Random Walk Formulation
- PageRank with Teleports
- Topic Sensitive PageRank

Graph Data: Information Networks

Graphs are everywhere!

Nodes represent objects (in this case, journals)

Edges represent relationships (in this case, citations)
- Can be *undirected* (e.g. friendship) or *directed* (e.g. citations, webpage hyperlinks)

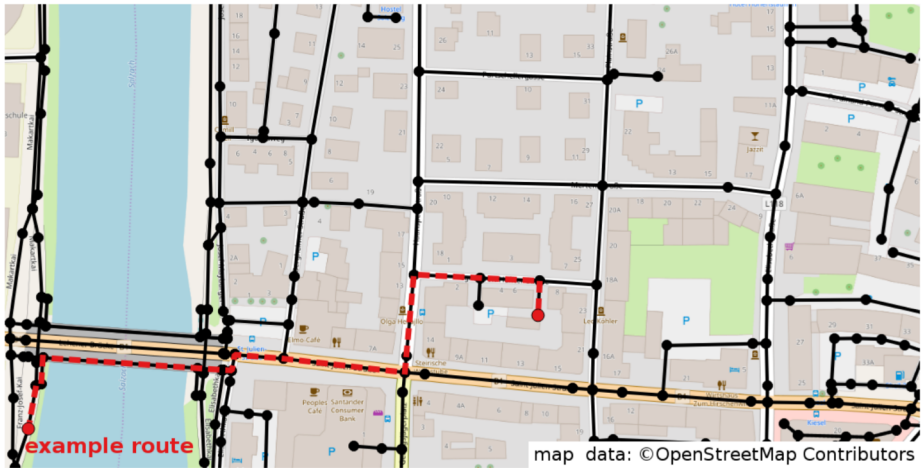


Citation networks and Maps of science
[Börner et al., 2012]

Graph Data: Social Networks

Facebook social graph
[Backstrom-Boldi-Rosa-Ugander-Vigna, 2011]

Graph Data: Traffic Networks



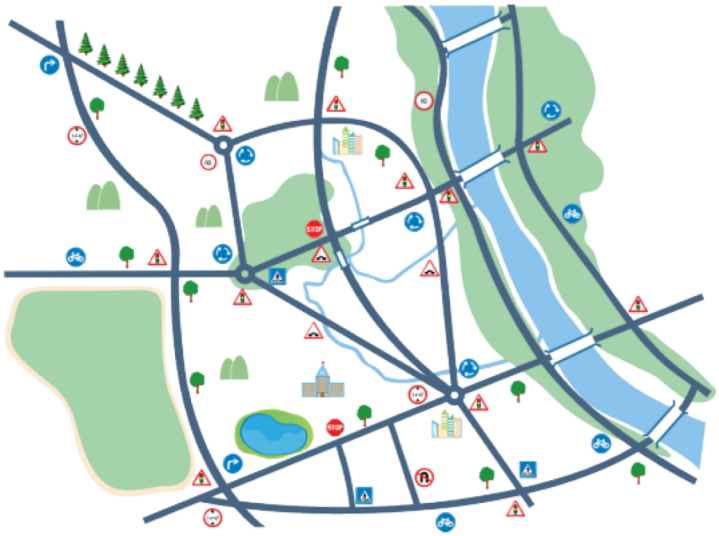


Figure 1: How can we predict how risky each road intersection is, based on nearby road features?

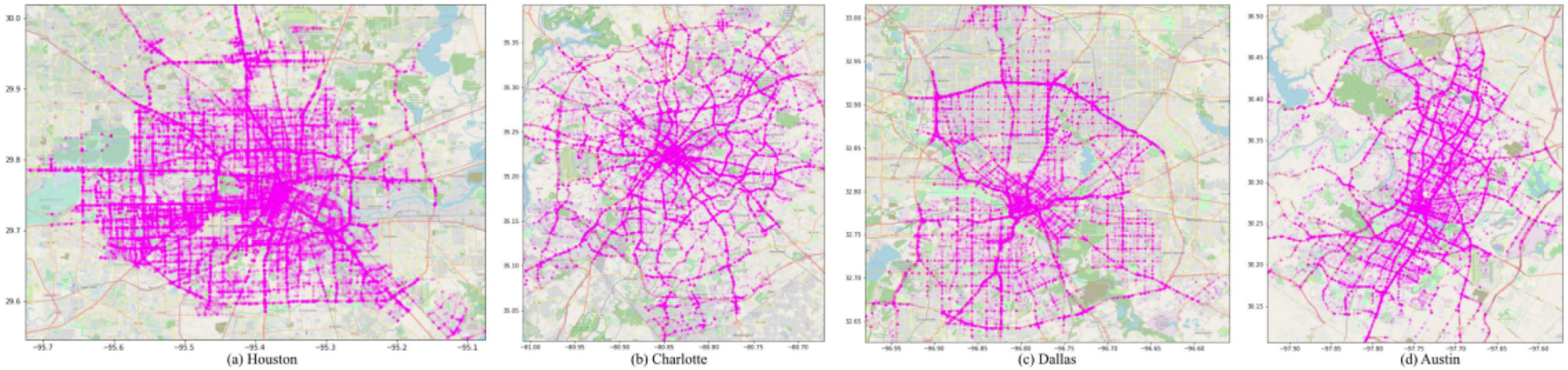
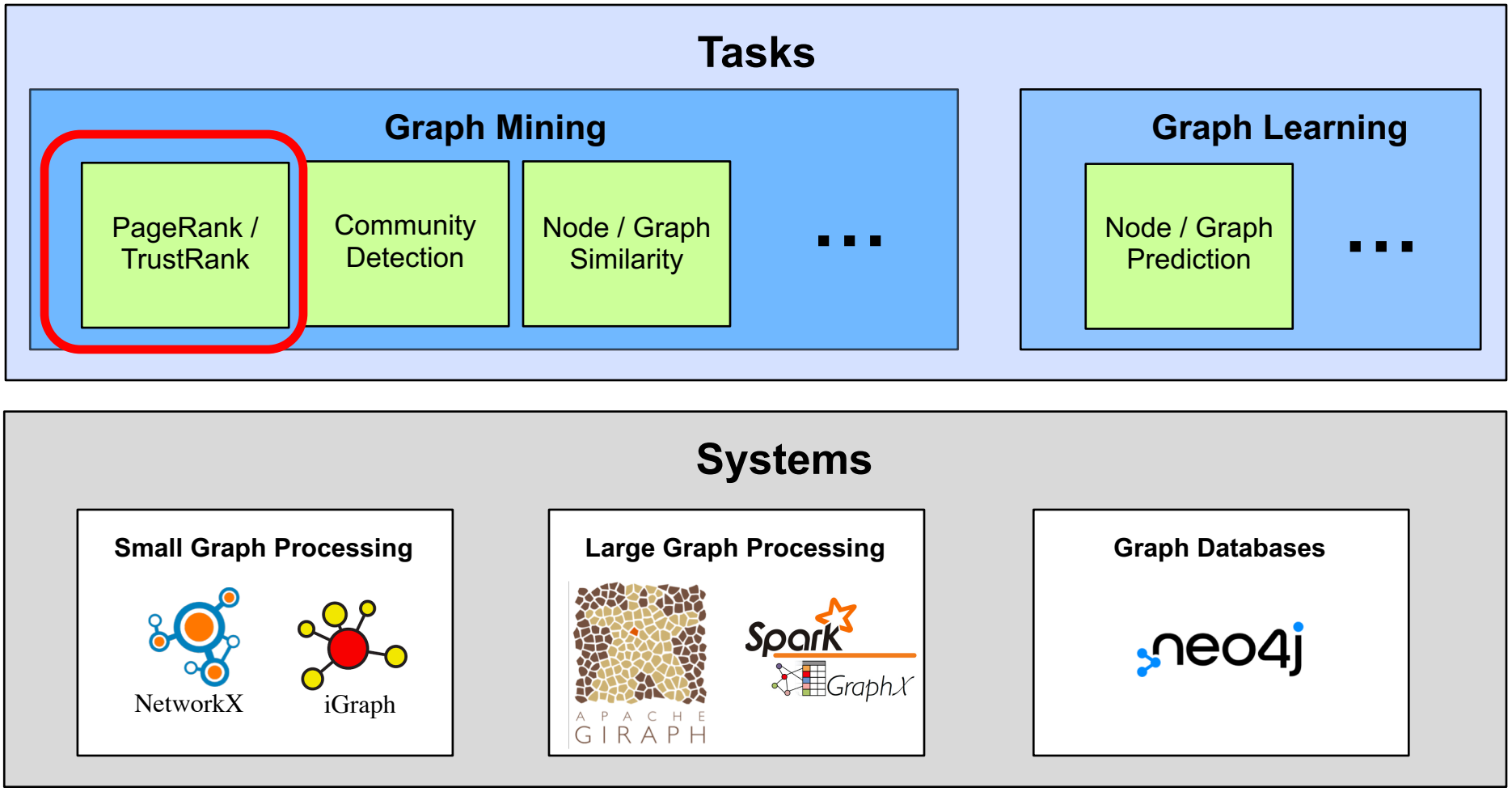


Figure 3: Traffic accident locations of Houston, Charlotte, Dallas, and Austin.

Overview: Graph Processing

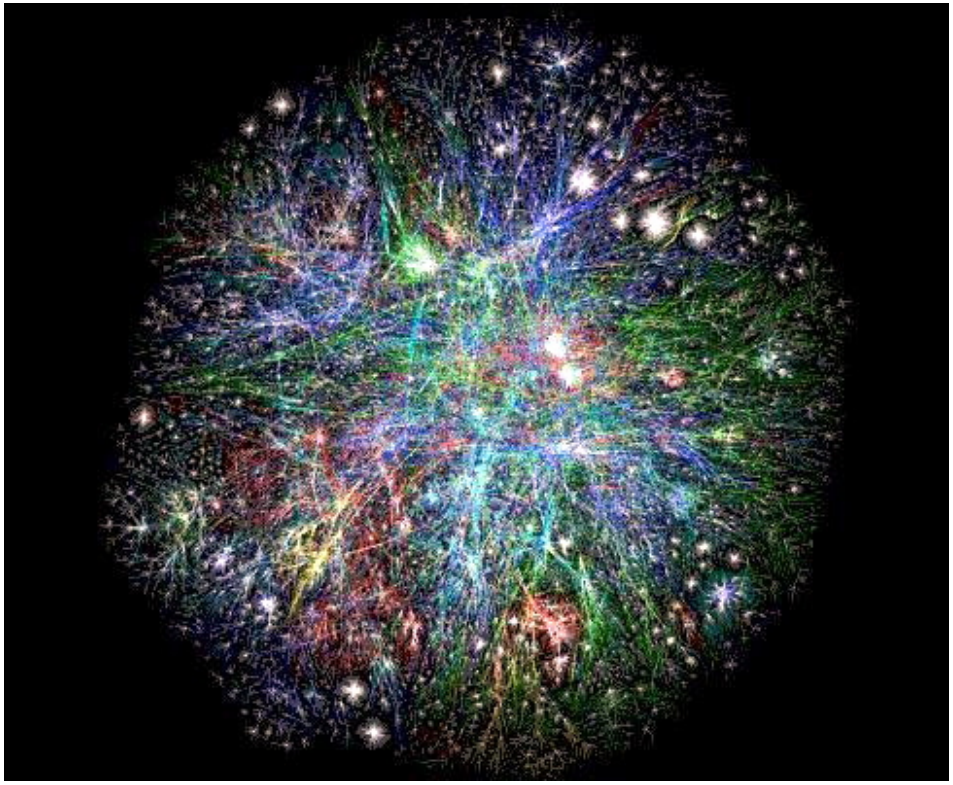


Today's Plan

- Graphs: Introduction
- **Simplified PageRank**
  - **Flow Formulation**
  - Random Walk Formulation
- PageRank with Teleports
- Topic Sensitive PageRank

Web as a Graph

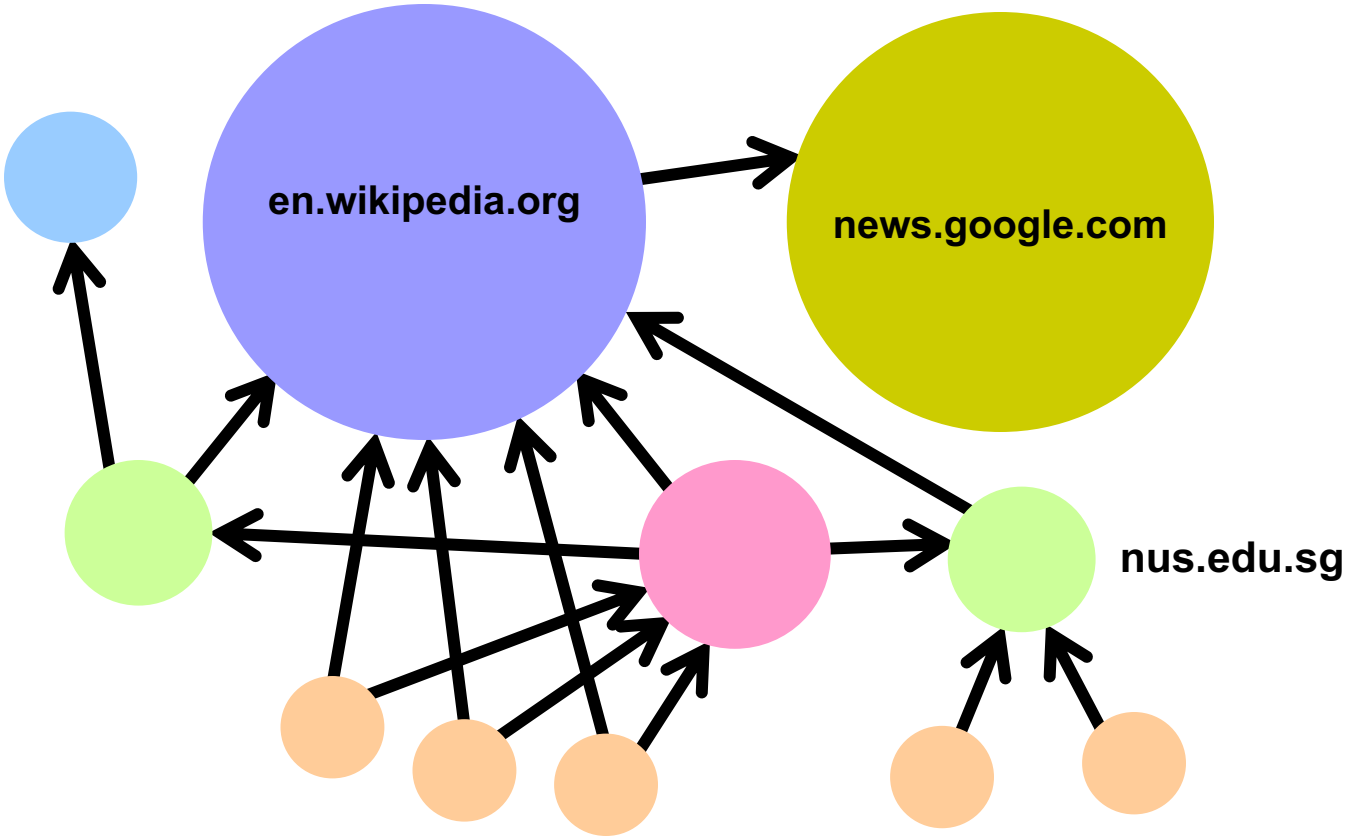
- **Web as a directed graph:**
  - **Nodes: Webpages**
  - **Edges: Hyperlinks**



Graph of the World Wide Web (http://www.bordalierinstitute.com/ (The Bordalier Institute))

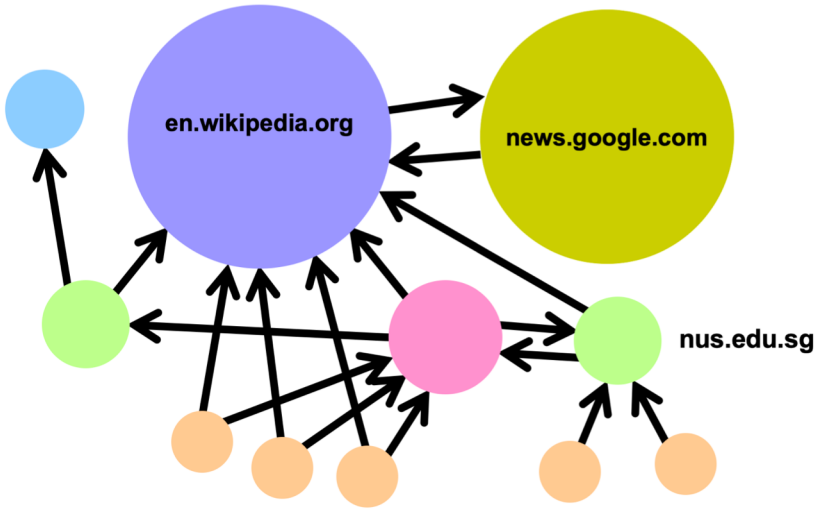
PageRank: Ranking Pages on the Web

- **All web pages are not equally "important":** www.joe-schmoe.com vs. en.wikipedia.org
  - Measuring the **importance** of pages is necessary for many web-related tasks (e.g. search, recommendation)
  - PageRank-like methods are also used in many other applications (bioinformatics, etc.)



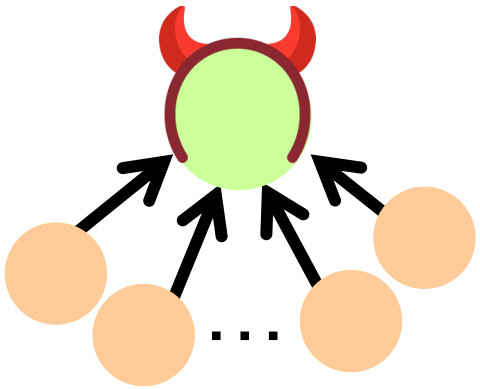
Links as Votes

- **Idea: Links as votes**
  - Page is more important if it has **more in-links**
    - Assume that incoming links are harder to manipulate. For example, anyone can create an out-link from their page to en.wikipedia.edu, but it is hard to get en.wikipedia.edu to create a link to their page
- **Think of in-links as votes**



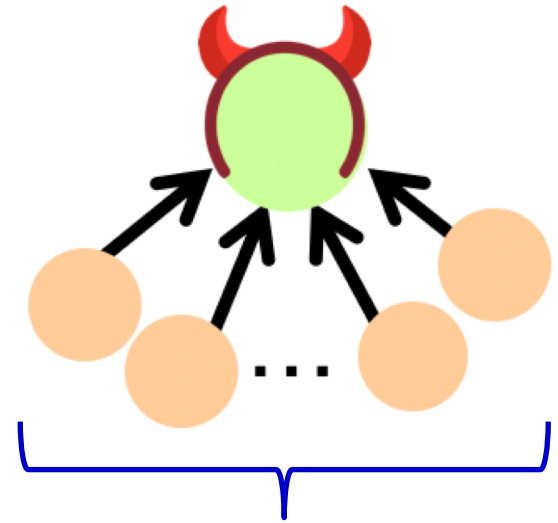
Links as Votes

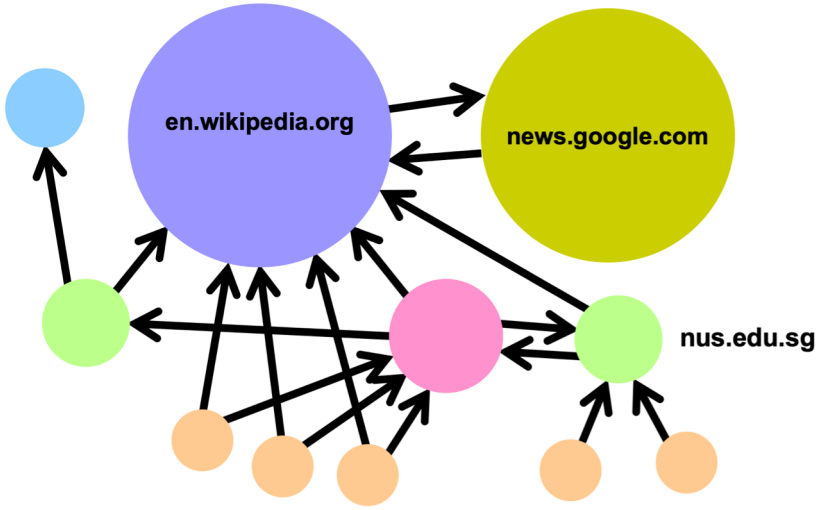
- **Think of in-links as votes**
- Naïve solution: What if we rank each page based on its *number* of in-links?
- **Problem:** malicious user can create a huge number of 'dummy' web pages, to link to their one page, to drive up its rank!



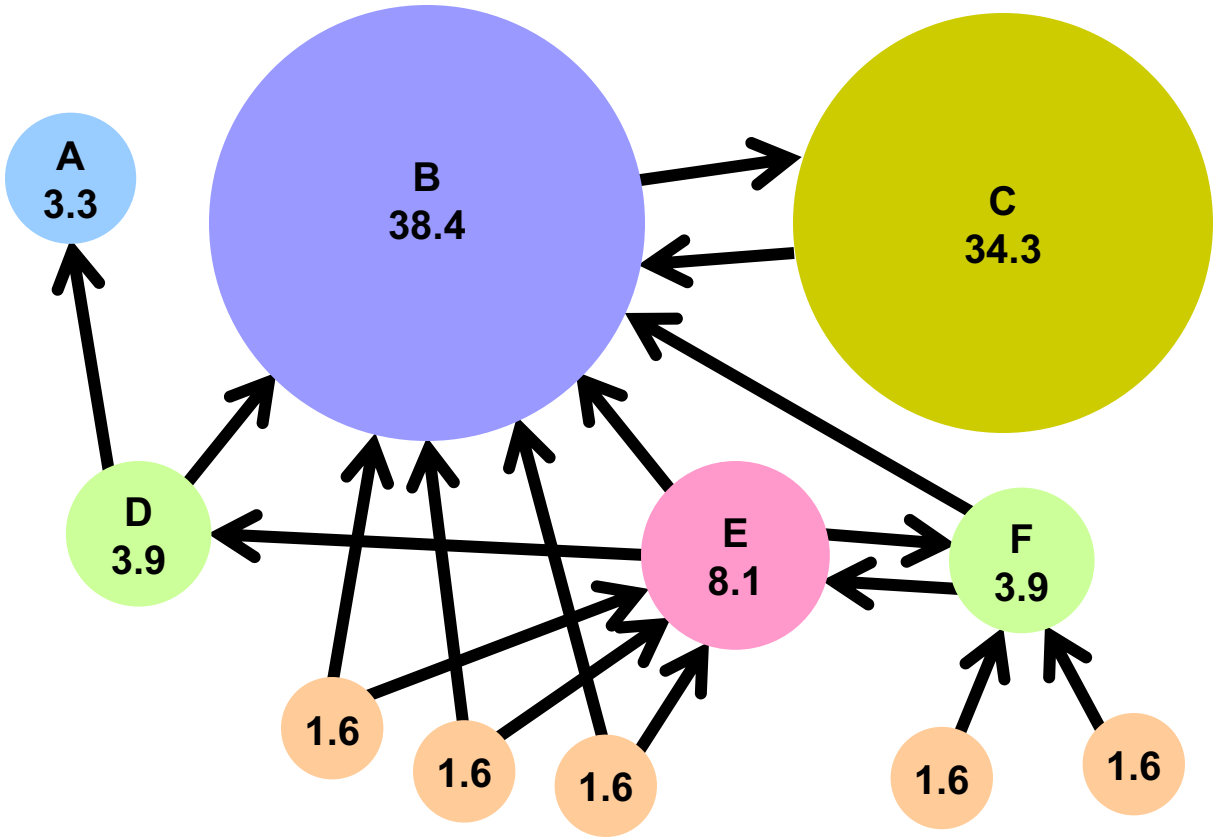
Links as Votes

- **Solution:** make the number of 'votes' that a page has proportional to its own importance. Then, as long as the 'dummy' pages themselves have low importance, they will contribute little votes as well.
  - Links from important pages count more – recursive definition!
  - This is the main idea of PageRank, which recursively defines the importance of a page based on the importance of the pages linking to it



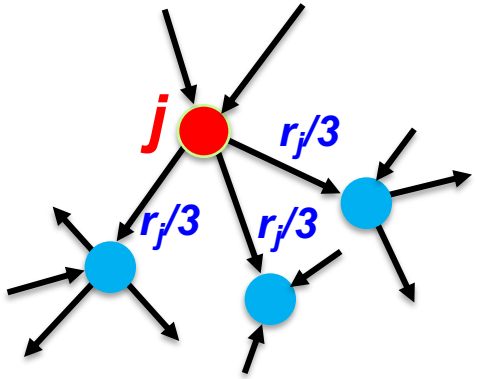


Example: PageRank Scores



‘Voting’ Formulation

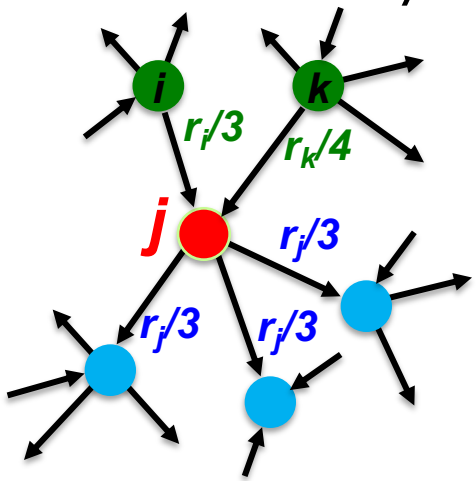
- Each link’s vote is proportional to the **importance** of its source page
- For each page ***j***, define its “importance” (or rank) as r_j
- If page ***j*** with importance r_j has ***n*** out-links, each link gets r_j / n votes



‘Voting’ Formulation

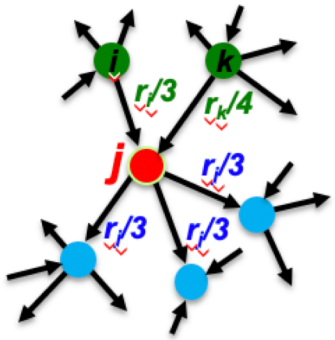
- Each link’s vote is proportional to the **importance** of its source page
- For each page ***j***, define its “importance” (or rank) as r_j
- If page ***j*** with importance r_j has ***n*** out-links, each link gets r_j / n votes
- Page ***j***’s own importance is the sum of the votes on its in-links
  - Analogy: each page receives a certain amount of candies from its incoming neighbors. It distributes these candies evenly to its outgoing neighbors.

$$r_j = r_i/3 + r_k/4$$



PageRank: The "Flow" Model

- A "vote" from an important page is worth more
- A page is important if it is pointed to by other important pages
- Define a "rank" or importance r_j for page j



$r_j = r_i/3 + r_k/4$

$$r_j = \sum_{i \to j} \frac{r_i}{d_i}$$

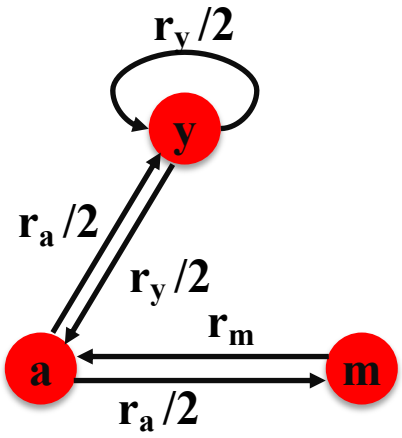
d_i = number of out-links (or "out-degree") of node i

Importance of j

Sum of importances of pages linking to j, each divided by number of out-links

"Simplified PageRank"

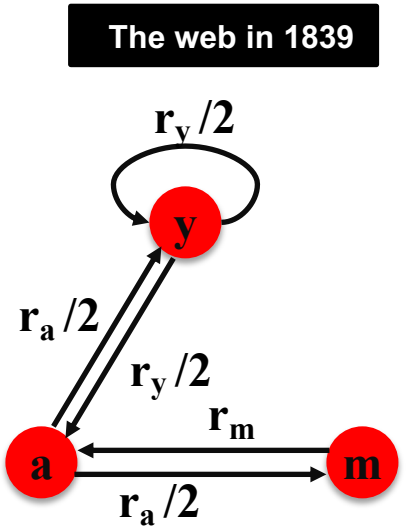
The web in 1839



"Flow" equations:

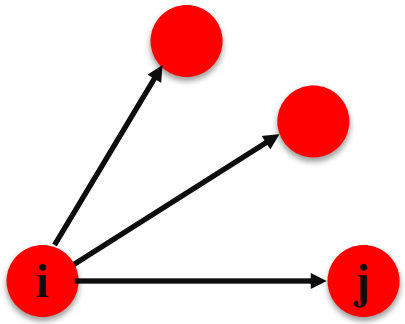
$$r_y = r_y/2 + r_a/2$$
$$r_a = r_y/2 + r_m$$
$$r_m = r_a/2$$

Solving the Flow Equations

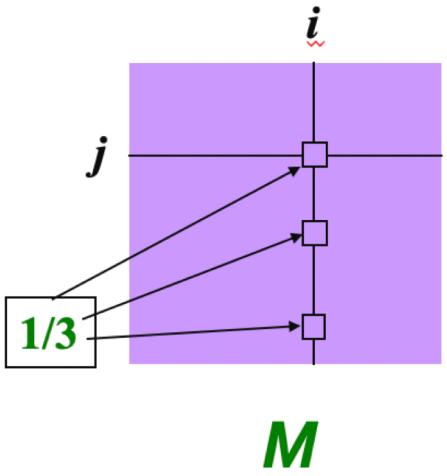


- **3 equations, 3 unknowns, no constants**
  - No unique solution
  - All solutions are rescalings of each other
- **Additional constraint forces uniqueness:**
  - $r_y + r_a + r_m = 1$
  - **Solution:** $r_y = \frac{2}{5}$, $r_a = \frac{2}{5}$, $r_m = \frac{1}{5}$
- Solving the equations through substitution or 'Gaussian elimination' works for small examples, but we need a better method for large web-size graphs
- **We need a new formulation!**

PageRank: Matrix Formulation



- **Stochastic adjacency matrix M**
  - Let page i has d_i out-links
  - If $i \to j$, then $M_{ji} = \frac{1}{d_i}$ else $M_{ji} = 0$
    - M is a **column stochastic matrix**
      - Columns sum to 1
- **Rank vector r:** vector with an entry per page
  - r_i is the importance score of page i
  - $\sum_i r_i = 1$
- **The flow equations can be written**



$$\boldsymbol{r} = \boldsymbol{M} \cdot \boldsymbol{r}$$

This is equivalent to our earlier formulation:

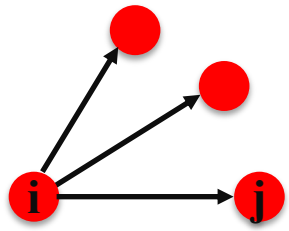
For all nodes $\boldsymbol{j}$: $r_j = \sum_{i \to j} \frac{r_i}{\mathrm{d_i}}$

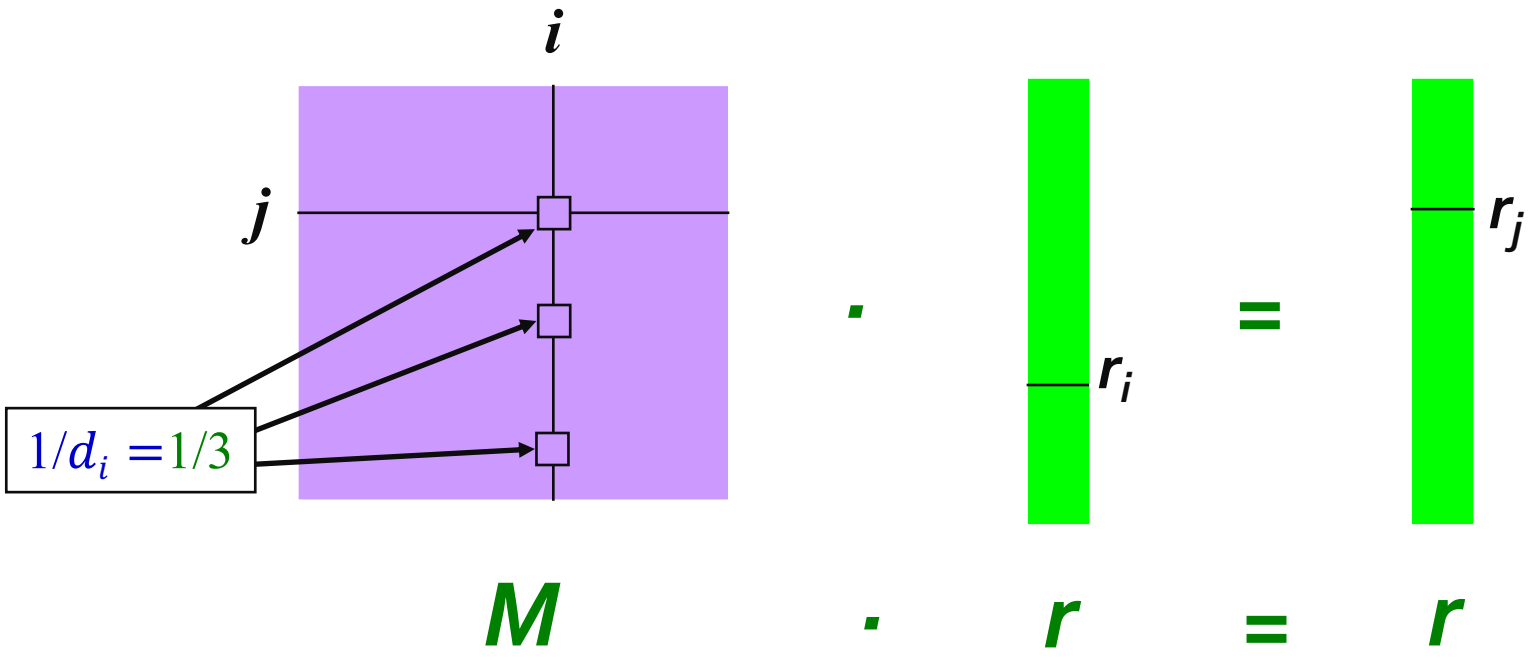
Why are the 2 formulations equivalent?

- **Remember the flow equation:** $r_j = \sum_{i \to j} \frac{r_i}{\mathrm{d_i}}$
- **Flow equation in the matrix form**

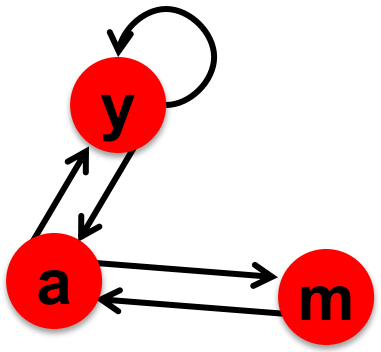
$$M \cdot r = r$$

- **Suppose page *i* links to 3 pages, including *j***





Example: Flow Equations & M



	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

$$r = M \cdot r$$

$$r_y = r_y/2 + r_a/2$$
$$r_a = r_y/2 + r_m$$
$$r_m = r_a/2$$

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{bmatrix} ½ & ½ & 0 \\ ½ & 0 & 1 \\ 0 & ½ & 0 \end{bmatrix} \begin{bmatrix} r'_y \\ r'_a \\ r'_m \end{bmatrix}$$

Solving the Flow Equations

- **Given the flow equation:**

$$r = M \cdot r$$

- **We can efficiently solve for *r*!**
  The method is called Power iteration

Power Iteration Method

- **Given a web graph with *n* nodes, where the nodes are pages and edges are hyperlinks**

Power iteration: a simple iterative scheme

- Suppose there are *N* web pages
- Initialize: $\mathbf{r}^{(0)} = [1/N,\ldots,1/N]^T$
- Iterate: $\mathbf{r}^{(t+1)} = \mathbf{M} \bullet \mathbf{r}^{(t)}$
- Stop when $|\mathbf{r}^{(t+1)} - \mathbf{r}^{(t)}|_1 < \varepsilon$

$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{d_i}$$

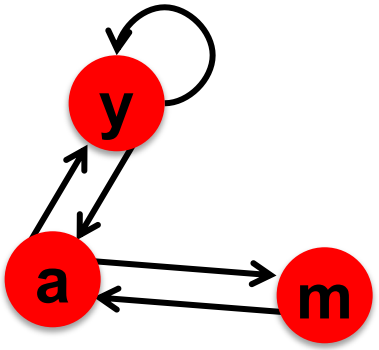
d_i …. out-degree of node i

$|x|_1 = \sum_{1 \le i \le N} |x_i|$ is the L_1 norm
Can use any other vector norm, e.g., Euclidean

Power Iteration: Example

Power Iteration:

- Suppose there are *N* web pages
- Initialize: $\mathbf{r}^{(0)} = [1/N,\ldots,1/N]^T$
- Iterate: $\mathbf{r}^{(t+1)} = \mathbf{M} \bullet \mathbf{r}^{(t)}$
- Stop when $|\mathbf{r}^{(t+1)} - \mathbf{r}^{(t)}|_1 < \varepsilon$



$r_y = r_y/2 + r_a/2$
$r_a = r_y/2 + r_m$
$r_m = r_a/2$

- **Example:**

$$\begin{pmatrix} r_y \\ r_a \\ r_m \end{pmatrix} = \begin{matrix} 1/3 & 1/3 & 5/12 & 9/24 & & 2/5 \\ 1/3 & 3/6 & 1/3 & 11/24 & \ldots & 2/5 \\ 1/3 & 1/6 & 3/12 & 1/6 & & 1/5 \end{matrix}$$

Iteration 0 Iteration 1 ⋯

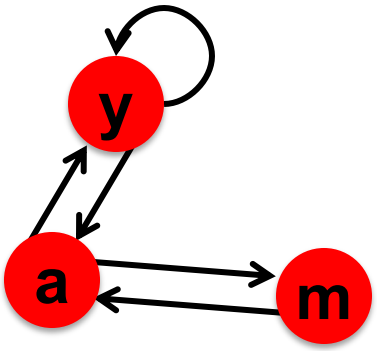
Today's Plan

- Graphs: Introduction
- **Simplified PageRank**
  - Flow Formulation
  - **Random Walk Formulation**
- PageRank with Teleports
- Topic Sensitive PageRank

Random Walk Interpretation

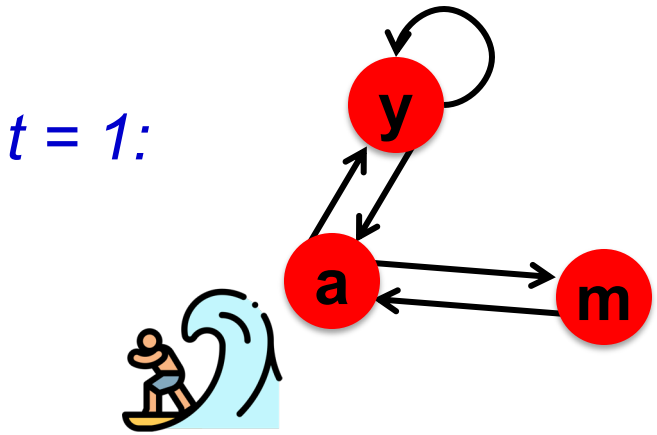
- Imagine a random web surfer:
  - At time t = 0, surfer starts on a random page
  - At any time t, surfer is on some page i
  - At time $t + 1$, the surfer follows an out-link from i uniformly at random
  - Process repeats indefinitely

t = 0:



Random Walk Interpretation

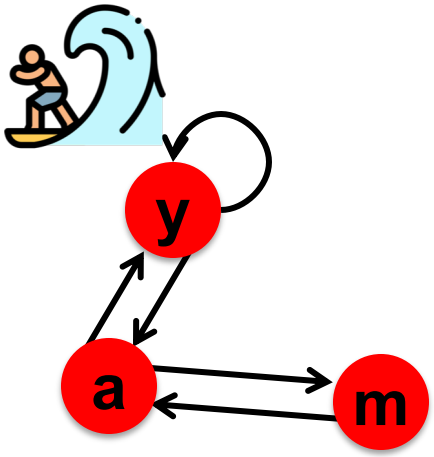
- Imagine a random web surfer:
  - At time t = 0, surfer starts on a random page
  - At any time t, surfer is on some page i
  - At time $t + 1$, the surfer follows an out-link from i uniformly at random
  - Process repeats indefinitely



Random Walk Interpretation

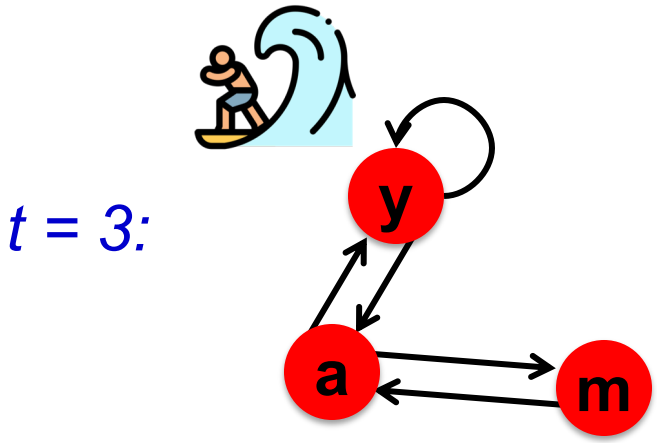
- Imagine a random web surfer:
  - At time t = 0, surfer starts on a random page
  - At any time t, surfer is on some page i
  - At time $t + 1$, the surfer follows an out-link from i uniformly at random
  - Process repeats indefinitely

t = 2:



Random Walk Interpretation

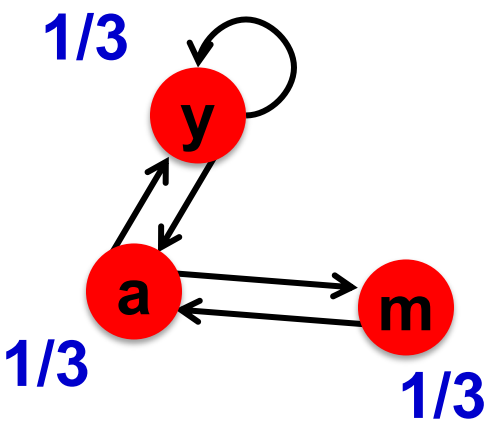
- Imagine a random web surfer:
  - At time t = 0, surfer starts on a random page
  - At any time t, surfer is on some page i
  - At time $t + 1$, the surfer follows an out-link from i uniformly at random
  - Process repeats indefinitely



Random Walk Interpretation

- Imagine a random web surfer:
  - At time t = 0, surfer starts on a random page
  - At any time t, surfer is on some page i
  - At time $t + 1$, the surfer follows an out-link from i uniformly at random
  - Process repeats indefinitely
- Let:
  - $p(t)$ … vector whose ith coordinate is the prob. that the surfer is at page i at time t
  - So, $p(t)$ is a probability distribution over pages

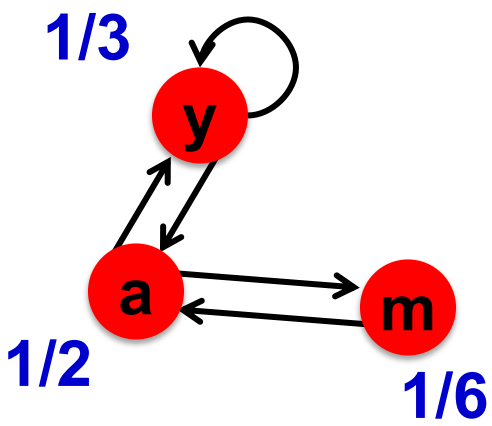
p(t) where t = 0:



Random Walk Interpretation

- Imagine a random web surfer:
  - At time t = 0, surfer starts on a random page
  - At any time t, surfer is on some page i
  - At time $t + 1$, the surfer follows an out-link from i uniformly at random
  - Process repeats indefinitely
- Let:
  - $p(t)$ … vector whose ith coordinate is the prob. that the surfer is at page i at time t
  - So, $p(t)$ is a probability distribution over pages

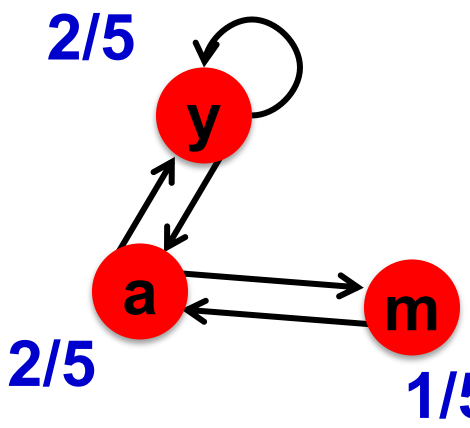
p(t) where t = 1:



Random Walk Interpretation

- Imagine a random web surfer:
  - At time t = 0, surfer starts on a random page
  - At any time t, surfer is on some page i
  - At time $t + 1$, the surfer follows an out-link from i uniformly at random
  - Process repeats indefinitely
- Let:
  - $p(t)$ … vector whose ith coordinate is the prob. that the surfer is at page i at time t
  - So, $p(t)$ is a probability distribution over pages
- **Stationary Distribution**: as $t \to \infty$, the probability distribution approaches a 'steady state' representing the long term probability that the random walker is at each node, which are the PageRank scores

p(t) where $t \to \infty$:



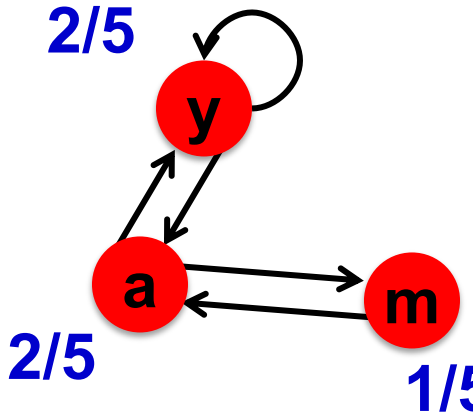
Equivalence between Random Walk and Flow Formulations

- Where is the surfer at time t+1?
  - Follows a link uniformly at random

$$\boldsymbol{p}(\boldsymbol{t}+\mathbf{1}) = \boldsymbol{M} \cdot \boldsymbol{p}(\boldsymbol{t})$$

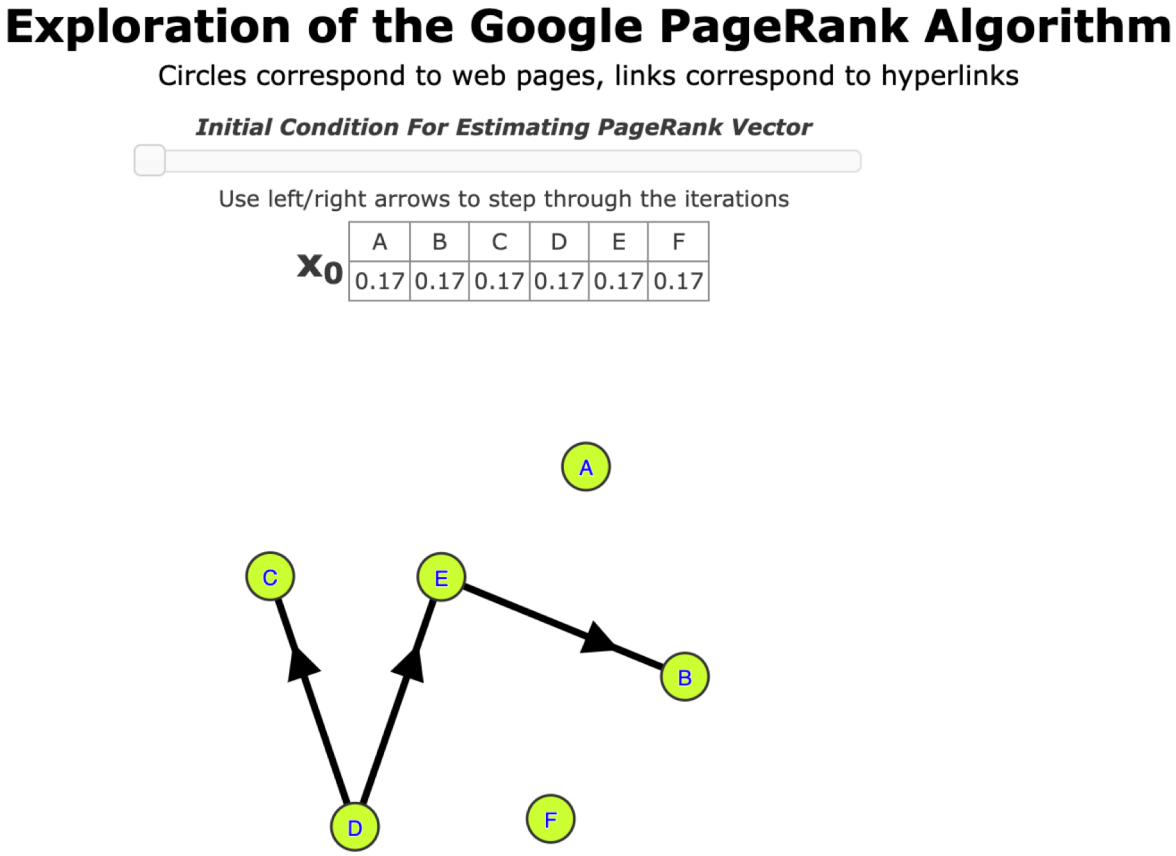
- Suppose the random walk reaches a stationary state $\boldsymbol{p}_s$: then $\boldsymbol{p}_s = \boldsymbol{M} \cdot \boldsymbol{p}_s$
- In the previous 'flow' formulation of PageRank, the rank vector $\boldsymbol{r}$ was also defined by $\boldsymbol{r} = \boldsymbol{M} \cdot \boldsymbol{r}$
  - Thus, the two (recursive) definitions are the same!

***p(t)* where $t \rightarrow \infty$:**



PageRank: Animation

- https://www.learnforeverlearn.com/pagerank/



Today's Plan

- Graphs: Introduction
- Simplified PageRank
  - Flow Formulation
  - Random Walk Formulation
- **PageRank with Teleports**
- Topic Sensitive PageRank

PageRank: Three Questions

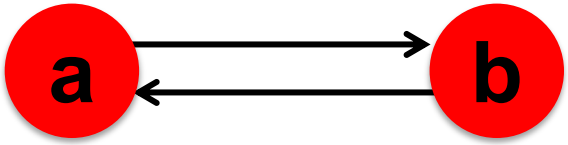
$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{\mathrm{d_i}}$$

or equivalently

$$r = Mr$$

- **Does this converge?**
- **Does it converge to what we want?**
- **Are results reasonable?**

Does this converge?



$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{\mathrm{d_i}}$$

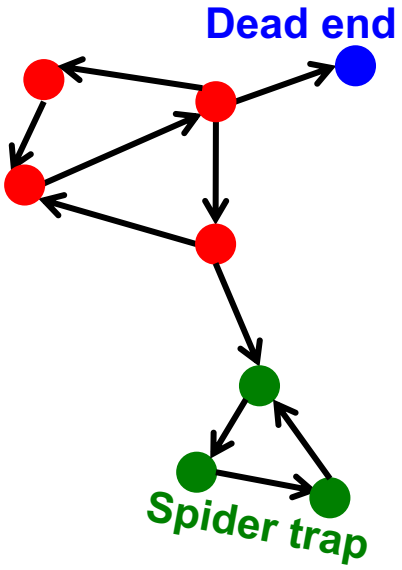
- **Example:**

r_a	=	1	0	1	0
r_b		0	1	0	1

Iteration 0, 1, 2, …

Does it converge to what we want?

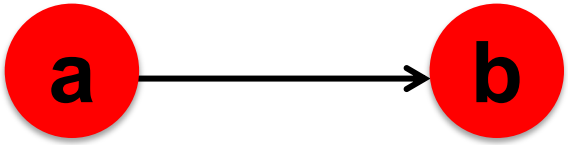
A: not always. 2 problems:



- **(1)** Some pages are **dead ends** (have no out-links)
  - Random walk has "nowhere" to go to
  - Such pages cause importance to "leak out"

- **(2) Spider traps:** (all out-links are within the group)
  - Random walk gets "stuck" in a trap
  - And eventually spider traps absorb all importance

Problem 1: Dead Ends



$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{\mathrm{d_i}}$$

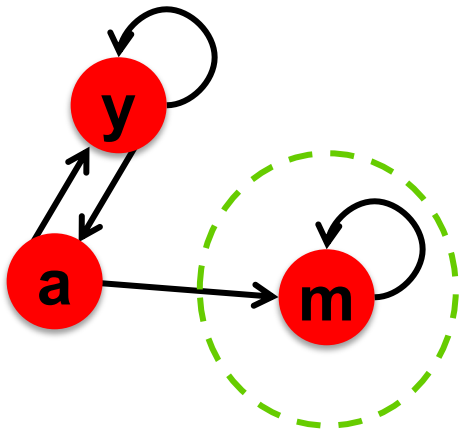
- **Example:**

$$\begin{matrix} r_a \\ r_b \end{matrix} = \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{matrix}$$

Iteration 0, 1, 2, …

Problem 2: Spider Traps

- **Power Iteration:**
  - Set $r_j = 1/N$
  - $r_j = \sum_{i \to j} \frac{r_i}{d_i}$
    - And iterate
- **Example:**



$$r_y = r_y/2 + r_a/2$$
$$r_a = r_y/2$$
$$r_m = r_a/2 + r_m$$

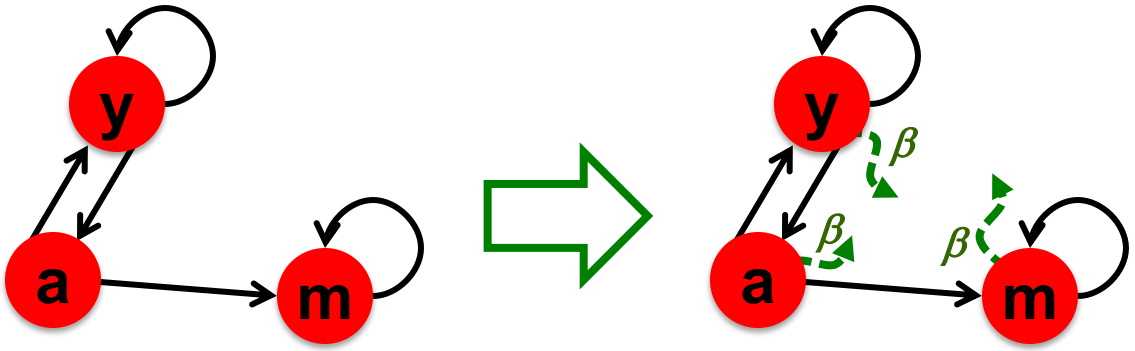
$$\begin{pmatrix} r_y \\ r_a \\ r_m \end{pmatrix} = \begin{matrix} 1/3 & 2/6 & 3/12 & 5/24 & & 0 \\ 1/3 & 1/6 & 2/12 & 3/24 & \ldots & 0 \\ 1/3 & 3/6 & 7/12 & 16/24 & & 1 \end{matrix}$$

Iteration 0, 1, 2, …

All the PageRank score gets "trapped" in node m.

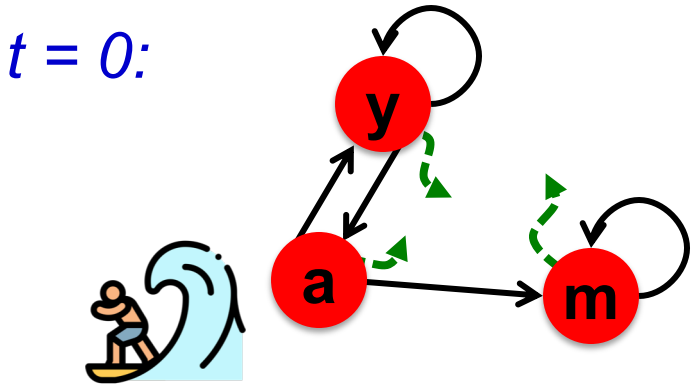
Solution: Teleports!

- **The Google solution for spider traps: At each time step, the random surfer has two options**
  - With prob. β, follow a link at random
  - With prob. **1-**β, jump to some random page
  - Common values for β are in the range 0.8 to 0.9



Solution: Teleports!

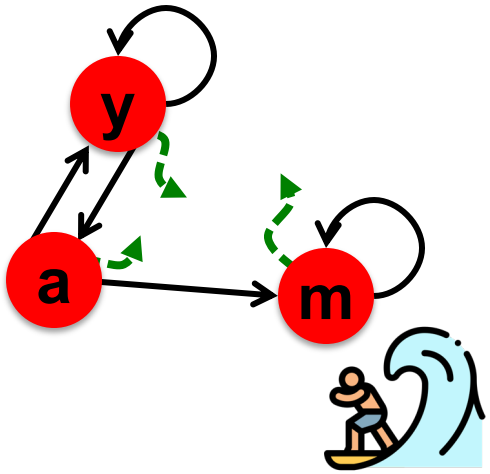
- **The Google solution for spider traps: At each time step, the random surfer has two options**
  - With prob. β, follow a link at random
  - With prob. **1-**β, jump to some random page
  - Common values for β are in the range 0.8 to 0.9



Solution: Teleports!

- **The Google solution for spider traps: At each time step, the random surfer has two options**
  - With prob. β, follow a link at random
  - With prob. **1-**β, jump to some random page
  - Common values for β are in the range 0.8 to 0.9

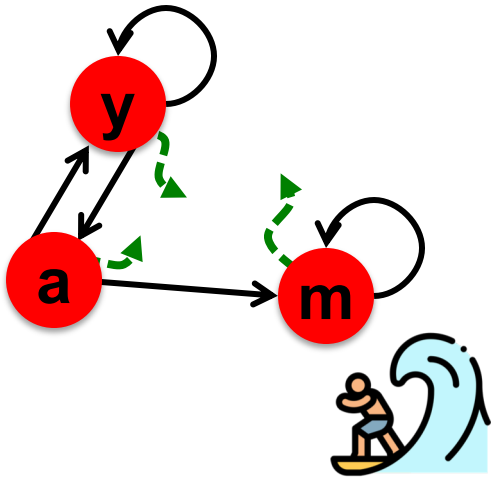
t = 1:



Solution: Teleports!

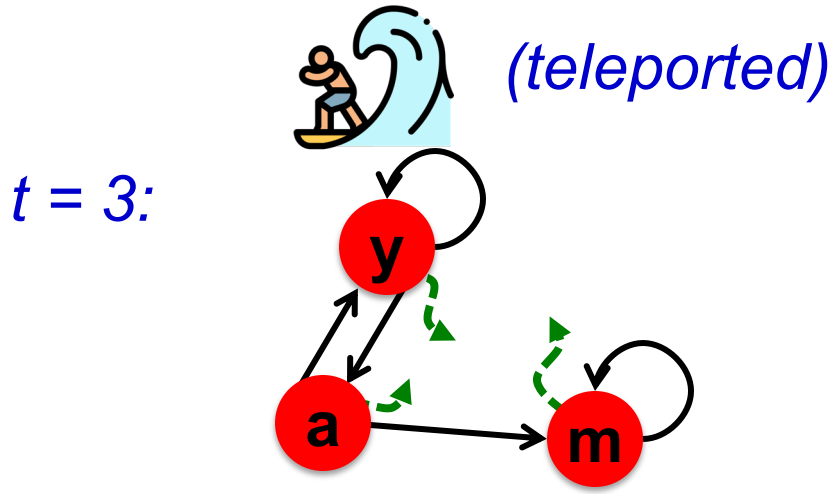
- **The Google solution for spider traps: At each time step, the random surfer has two options**
  - With prob. β, follow a link at random
  - With prob. **1-**β, jump to some random page
  - Common values for β are in the range 0.8 to 0.9

t = 2:



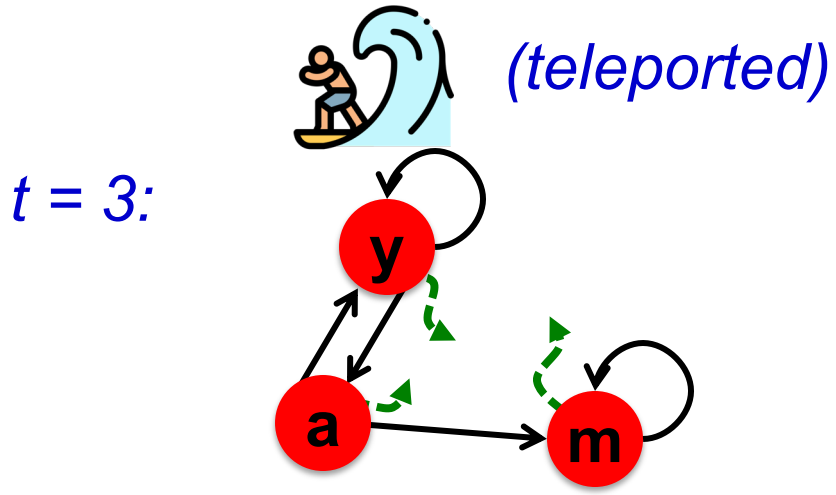
Solution: Teleports!

- **The Google solution for spider traps: At each time step, the random surfer has two options**
  - With prob. β, follow a link at random
  - With prob. **1-**β, jump to some random page
  - Common values for β are in the range 0.8 to 0.9



Solution: Teleports!

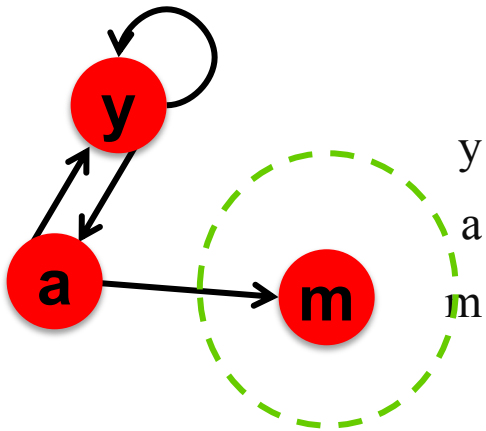
- **The Google solution for spider traps: At each time step, the random surfer has two options**
  - With prob. β, follow a link at random
  - With prob. **1-**β, jump to some random page
  - Common values for β are in the range 0.8 to 0.9



- **Conclusion:** surfer will quickly teleport out of any spider trap

Problem: Dead Ends

- **Power Iteration:**
  - Set $r_j = 1/N$
  - $r_j = \sum_{i \to j} \frac{r_i}{d_i}$
    - And iterate



	y	a	m
y	½	½	0
a	½	0	0
m	0	½	0

$r_y = r_y/2 + r_a/2$
$r_a = r_y/2$
$r_m = r_a/2$

- **Example:**

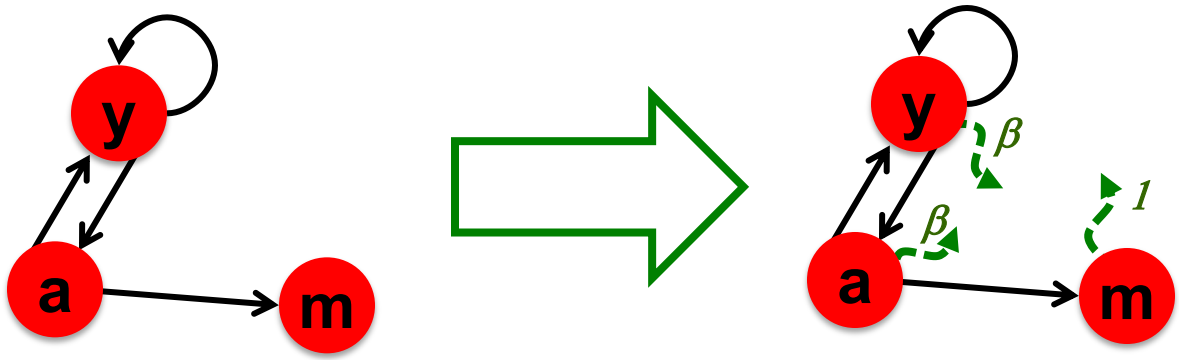
$$\begin{pmatrix} r_y \\ r_a \\ r_m \end{pmatrix} = \begin{matrix} 1/3 & 2/6 & 3/12 & 5/24 & & 0 \\ 1/3 & 1/6 & 2/12 & 3/24 & \ldots & 0 \\ 1/3 & 1/6 & 1/12 & 2/24 & & 0 \end{matrix}$$

Iteration 0, 1, 2, …

Here the PageRank "leaks" out since the matrix is not stochastic.

Solution: If at a Dead End, Always Teleport

- **Teleports:** Follow random teleport links with probability 1.0 from dead-ends
  - Adjust matrix accordingly



Why Teleports Solve the Problem?

Why are dead-ends and spider traps a problem and why do teleports solve the problem?

- **Spider-traps** cause random walker to get stuck in them, absorbing all importance
  - **Solution:** Never get stuck in a spider trap by teleporting out of it in a finite number of steps
- **Dead-ends** cause importance to "leak" out of the system
  - The matrix is not column stochastic
  - **Solution:** Make matrix column stochastic by always teleporting when at a dead end

Random Teleport: Equations

- At each step, random surfer has two options:
  - With probability β, follow a link at random
  - With probability $1\text{-}\beta$, jump to some random page

- **PageRank equation** [Brin-Page, 98]

$$r_j = \sum_{i \to j} \beta \underbrace{\frac{r_i}{d_i}}_{\text{Same as simplified PageRank}} + \underbrace{(1-\beta)\frac{1}{N}}_{\text{Teleport term}}$$

d_i … out-degree of node i

(This formulation assumes that there are no dead ends. If there are, we can make a small modification to the equation, to always teleport from dead ends)

The Google Matrix

- **PageRank equation** [Brin-Page, '98]

$$r_j = \sum_{i \to j} \beta \frac{r_i}{d_i} + (1 - \beta) \frac{1}{N}$$

- **The Google Matrix *A*:**

$$A = \beta\, M + (1 - \beta) \left[\frac{1}{N}\right]_{N \times N}$$

$[1/N]_{NxN}$…N by N matrix where all entries are 1/N

- **We have a recursive problem: $r = A \cdot r$**
  And the Power method still works!
- **What is β?**
  - In practice $\beta = 0.8, 0.9$ (roughly, 5-10 steps on avg before a teleport)

Some Problems with Page Rank

- **Measures generic popularity of a page**
  - Biased against topic-specific authorities
  - **Solution:** Topic-Specific PageRank
- **Uses a single measure of importance**
  - Other models of importance
  - **Solution:** Hubs-and-Authorities
- **Susceptible to Link spam**
  - Artificial link topographies created in order to boost page rank
  - **Solution:** TrustRank

Today's Plan

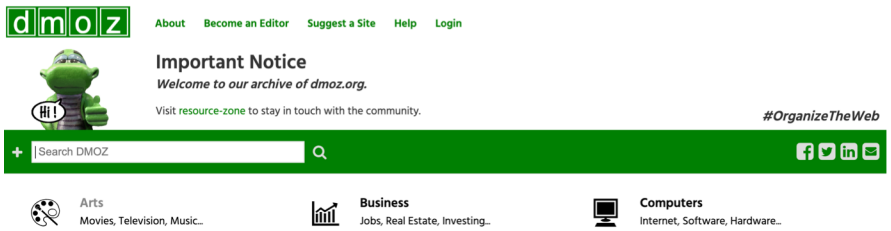
- Graphs: Introduction
- Simplified PageRank
  - Flow Formulation
  - Random Walk Formulation
- PageRank with Teleports
- **Topic Sensitive PageRank**

Topic-Specific PageRank

- **Instead of generic popularity, can we measure popularity within a topic?**
- **Goal:** Evaluate Web pages not just according to their popularity, but by how close they are to a particular topic, e.g. "sports" or "history"
- **Allows search queries to be answered based on interests of the user**
  - **Example:** Query "Trojan" wants different pages depending on whether you are interested in sports, history and computer security

Topic-Specific PageRank

- Random walker has a small probability of teleporting at any step
- **Teleport can go to:**
  - **Standard PageRank: Any page with equal probability**
    - To avoid dead-end and spider-trap problems
  - **Topic Specific PageRank: A topic-specific set of "relevant" pages (teleport set)**
- **Idea: Bias the random walk**
  - When random walker teleports, it picks a page from a set ***S***
  - ***S*** contains only pages that are relevant to the topic
    - E.g., Open Directory (DMOZ) pages for a given topic/query
  - For each teleport set ***S***, we get a different vector r_S



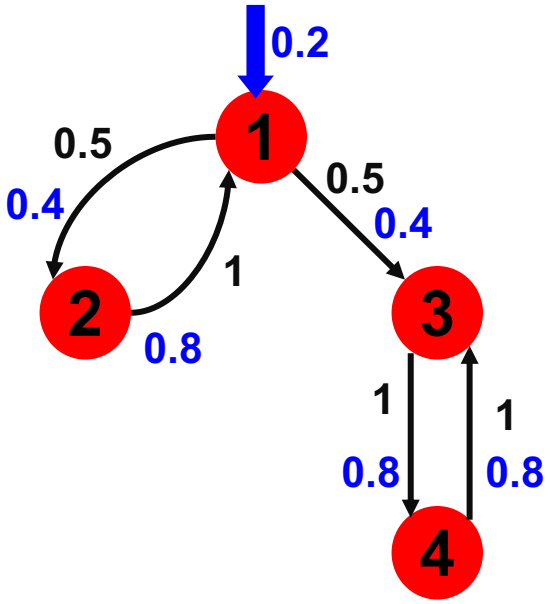
Matrix Formulation

- **To make this work all we need is to update the teleportation part of the PageRank formulation:**

$$A_{ij} = \begin{cases} \beta M_{ij} + (1-\beta)/|S| & \text{if } i \in S \\ \beta M_{ij} + 0 & \text{otherwise} \end{cases}$$

  - ***A*** is stochastic!
- We weighted all pages in the teleport set ***S*** equally
  - **Could also assign different weights to pages!**
- **Compute as for regular PageRank:**
  - Multiply by ***M***, then add a vector
  - Maintains sparseness

Example: Topic-Specific PageRank



$S = \{1\}$ $\beta = 0.8$

Node	Iteration				
	0	1	2	…	stable
1	0.25	0.4	0.28		0.294
2	0.25	0.1	0.16		0.118
3	0.25	0.3	0.32		0.327
4	0.25	0.2	0.24		0.261

Probabilities (no teleport)

Probabilities (with teleport)

S={1}, β=0.90:
r=[0.17, 0.07, 0.40, 0.36]
S={1} , β=0.8:
r=[0.29, 0.11, 0.32, 0.26]
S={1}, β=0.70:
r=[0.39, 0.14, 0.27, 0.19]

S={1,2,3,4}, β=0.8:
r=[0.13, 0.10, 0.39, 0.36]
S={1,2,3} , β=0.8:
r=[0.17, 0.13, 0.38, 0.30]
S={1,2} , β=0.8:
r=[0.26, 0.20, 0.29, 0.23]
S={1} , β=0.8:
r=[0.29, 0.11, 0.32, 0.26]

Discovering the Topic Vector S

- **Create different PageRanks for different topics**
  - The 16 DMOZ top-level categories:
    - arts, business, sports,…
- **Which topic ranking to use?**
  - User can pick from a menu
  - Classify query into a topic
  - Can use the **context** of the query
    - E.g., query is launched from a web page talking about a known topic
    - History of queries e.g., "basketball" followed by "Jordan"
  - User context, e.g., user's bookmarks, …

Acknowledgements

- mmds.org – Mining of Massive Datasets - Jure Leskovec, Anand Rajaraman, Jeff Ullman
- https://jakobmiksch.eu/post/openstreetmap_routing/
- https://www.edrawmax.com/online-map-maker.html
- https://learnforeverlearn.com/pagerank/