Machine Learning for Graphs and Sequential Data

Graphs - Ranking

Lecturer: Prof. Dr. Stephan Günnemann

cs.cit.tum.de/daml

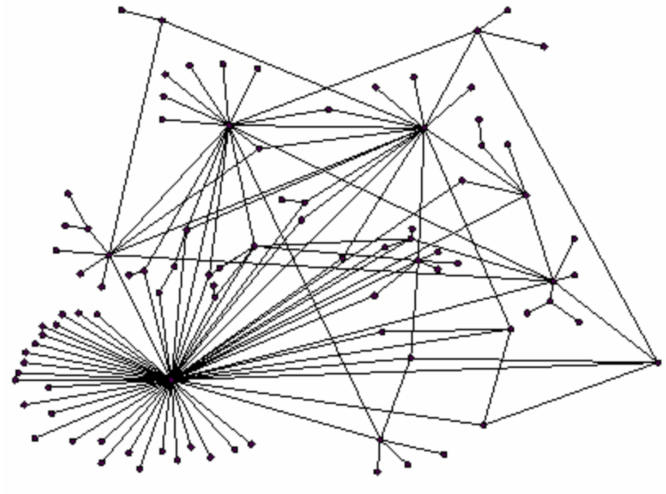
Summer Term 2024

Data Analytics and
Machine Learning

Roadmap

- **Chapter: Graphs**
  1. Graphs & Networks
  2. Generative Models
  3. **Ranking**
  4. Clustering
  5. Classification (Semi-Supervised Learning)
  6. Node/Graph Embeddings
  7. Graph Neural Networks (GNNs)

Motivation: Ranking of Nodes



- How to organize the Web?
- First try: Human curated Web directories
  - Yahoo, DMOZ, LookSmart
- Second try: Web Search
  - Information Retrieval investigates: Find relevant docs in a small and trusted set
    - Newspaper articles, Patents, etc.
  - But: **Web is huge**, full of untrusted documents, randomness, web spam, etc.

- Web pages are not equally "important"
  - www.some-personal-website.com vs. www.tum.de

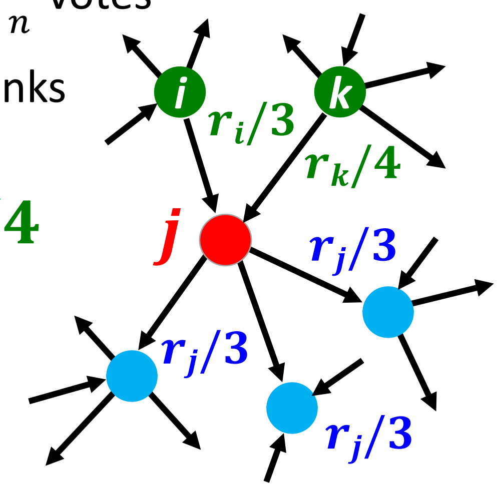
- There is large diversity in the web-graph node connectivity.
  Let's rank the pages by the link structure!

PageRank

- Core idea: **A page is important if many important pages point to it**
  - recursive formulation

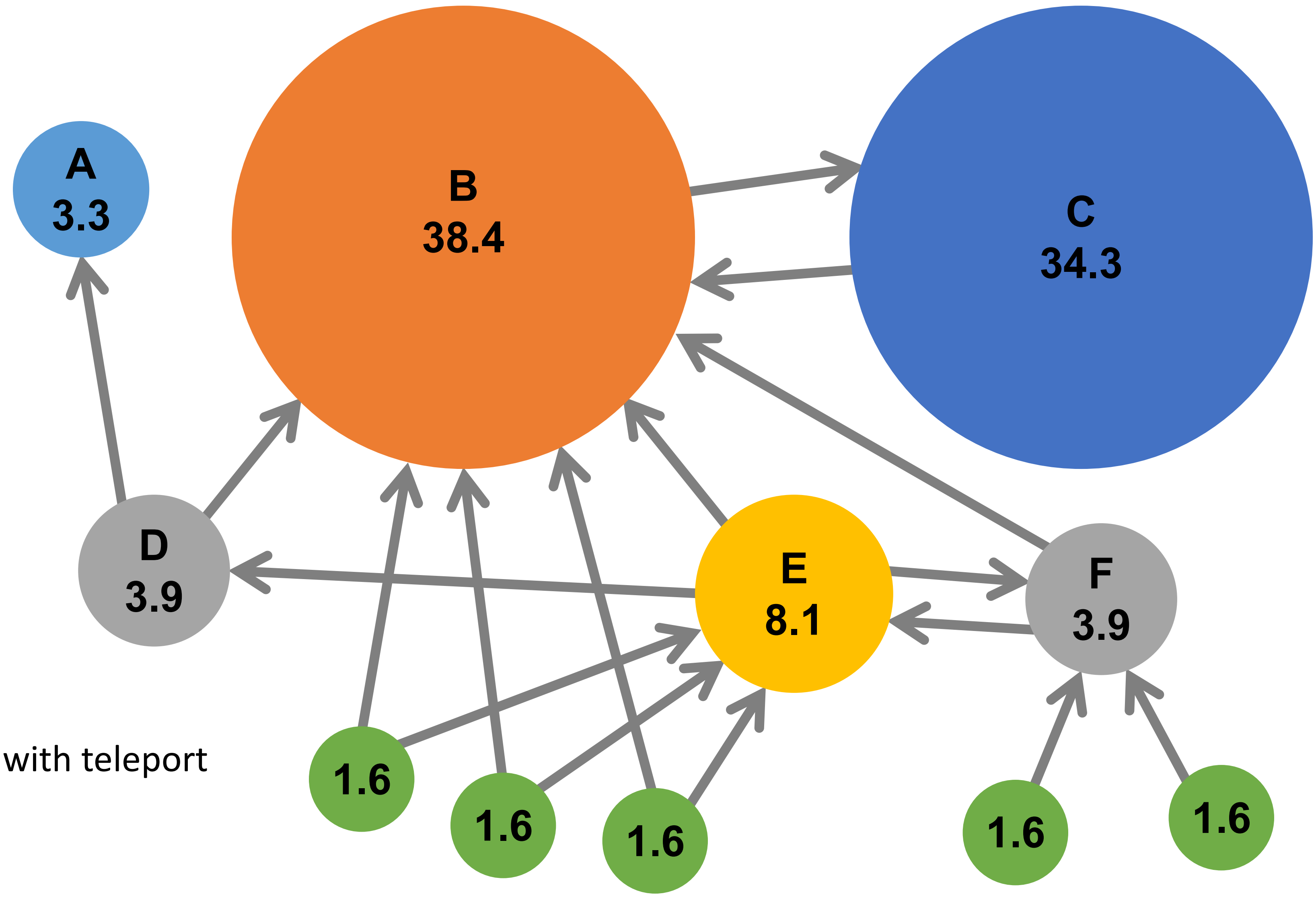
- "Voting" principle
  - each page votes for the importance of the pages it points to
  - a link's vote is proportional to the importance of its source page
  - If page j with importance r_j has n out-links, each link gets $\frac{r_j}{n}$ votes
  - Page j's own importance is the sum of the votes on its in-links

$$\boldsymbol{r_j = r_i/3 + r_k/4}$$



- **Rank of page j:** $\boldsymbol{r_j = \sum_{i \to j} \frac{r_i}{d_i}}$
  - d_i … out-degree of node i

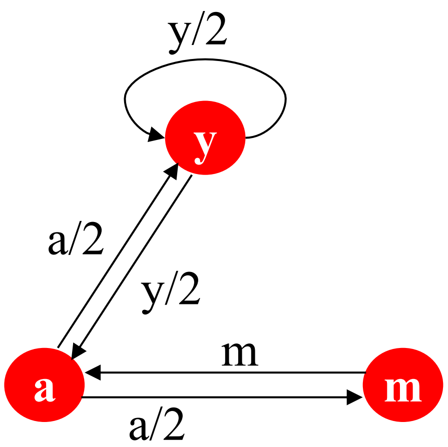
Example: PageRank Scores



with teleport

Computation via Solving Equations

- Rank of page j: $\boldsymbol{r_j} = \sum_{i \to j} \frac{\boldsymbol{r_i}}{\boldsymbol{d_i}}$
  - d_i … out-degree of node i



- Example:
  - 3 equations, 3 unknowns, no constants
    - No unique solution
    - All solutions equivalent modulo a scale factor
  - Additional constraint forces uniqueness: $\sum_i r_i = 1$
  - Solution: $\boldsymbol{r_y} = \frac{2}{5}, \ \boldsymbol{r_a} = \frac{2}{5}, \ \boldsymbol{r_m} = \frac{1}{5}$

Equations:

$$\mathbf{r_y} = \mathbf{r_y}/2 + \mathbf{r_a}/2$$
$$\mathbf{r_a} = \mathbf{r_y}/2 + \mathbf{r_m}$$
$$\mathbf{r_m} = \mathbf{r_a}/2$$

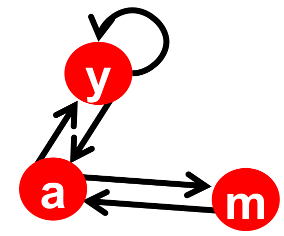
- Gaussian elimination method works for small examples but we need a better method for large web-size graphs

PageRank: Matrix Formulation

- Stochastic adjacency matrix $\boldsymbol{M}$
  - If $i \rightarrow j$, then $M_{ji} = \frac{1}{d_i}$ else $M_{ji} = 0$
  - $\boldsymbol{M}$ is a column stochastic matrix
    - Columns sum to 1
- Rank vector $\boldsymbol{r}$
  - r_i is the importance score of page i
  - $\sum_i r_i = 1$
  - $\forall i: r_i \geq 0$
- Equations $\boldsymbol{r_j} = \sum_{\boldsymbol{i \rightarrow j}} \frac{\boldsymbol{r_i}}{\boldsymbol{d_i}}$ can be written as:

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

- Analytical solution $\boldsymbol{r} = (\boldsymbol{Id} - \boldsymbol{M})^{-1}$ is intractable for large graphs



$$\boldsymbol{M} = \begin{array}{c|ccc} & \mathbf{y} & \mathbf{a} & \mathbf{m} \\ \hline \mathbf{y} & 1/2 & 1/2 & 0 \\ \mathbf{a} & 1/2 & 0 & 1 \\ \mathbf{m} & 0 & 1/2 & 0 \end{array}$$

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

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix}$$

Computation via Eigenvector

- Equations can be written as: $\boldsymbol{r} = \boldsymbol{M} \cdot \boldsymbol{r}$

- The rank vector $\boldsymbol{r}$ is an eigenvector of the stochastic matrix M
  - eigenvector with corresponding eigenvalue 1
  - Math background: largest eigenvalue of M is 1 since M is column stochastic (with non-negative entries)
    - We know r is unit length and each column of M sums to one, so $\boldsymbol{Mr} \leq \mathbf{1}$

- Finding $\boldsymbol{r}$ = finding eigenvector of $\boldsymbol{M}$ corresponding to the largest eigenvalue
  - you know how to do this efficiently (power iteration; see ML slides)

Notes on Computation

- Power iteration: iteratively compute $\boldsymbol{r} \leftarrow \frac{\boldsymbol{M} \cdot \boldsymbol{r}}{\|\boldsymbol{M} \cdot \boldsymbol{r}\|}$ until convergence
  - required for PageRank: $\sum_i r_i = 1$
- Let $\boldsymbol{y} = \boldsymbol{M} \cdot \boldsymbol{x}$ with $\sum_i x_i = 1$.
  Since $\boldsymbol{M}$ is column stochastic, it holds $\sum_i y_i = 1$

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

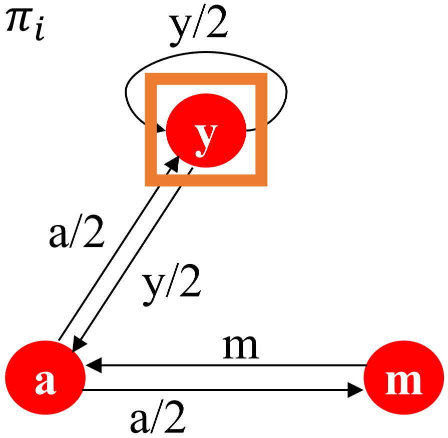
$$\boxed{\begin{matrix} r_y \\ r_a \\ r_m \end{matrix}} = \boxed{\begin{matrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{matrix}} \boxed{\begin{matrix} r_y \\ r_a \\ r_m \end{matrix}}$$

- No need for normalization!
- Start with random (normalized) vector $\boldsymbol{r}$, and iterate $\boldsymbol{r} \leftarrow \boldsymbol{M} \cdot \boldsymbol{r}$

- Important: Matrix $\boldsymbol{M}$ is sparse!
  - we only need to consider the (ingoing) neighbors of each node
- Iteratively compute $r_j \leftarrow \sum_{i \to j} \frac{r_i}{d_i}$ until convergence
  - first compute the updated value for each r_j, then assign them at once

Random Walk Interpretation

- Consider a random web surfer that moves between the web pages
  - At time t, the web surfer is in a random webpage i
  - At time $t+1$, the surfer follows an out-link from i uniformly at random
- The surfer's path (denoted by $X_1, X_2, X_3, \ldots$) forms a Markov chain
  - Web pages are the states of the Markov chain
  - The surfer starts from a random webpage: $\Pr(X_1 = i) = \pi_i$
  - Transition probabilities: $\Pr(X_{t+1} = j | X_t = i) = M_{ji}$
  - Note: the transition probability matrix of the Markov chain is $\boldsymbol{B} = \boldsymbol{M}^T$



Random Walk Interpretation

- Stationary distribution: the vector $\boldsymbol{\pi}^{\infty}$ is called stationary distribution if the following equality holds

$$\boldsymbol{\pi}^{\infty} = \boldsymbol{\pi}^{\infty}\boldsymbol{B}$$

- By definition, $\boldsymbol{\pi}^{\infty}$ (if exists) is equal to (transpose of) the rank vector $\boldsymbol{r}$.

- $\boldsymbol{\pi}^{\infty}$ can be computed by
  1. getting the eigenvector of $\boldsymbol{M}$ associated with the unit eigenvalue
  2. normalizing it to one.

Random Walk Interpretation

- Consider a random web surfer that moves between the web pages
  - The surfer's path (denoted by $X_1, X_2, X_3, \ldots$) forms a Markov chain

- Remember: $\Pr(X_t = i) \stackrel{\text{def}}{=} \pi_i(t)$
  - probability of reaching state i (here: page i) in step t

recap:
$\boldsymbol{\pi}(t) = \boldsymbol{\pi}\boldsymbol{B}^{(t-1)}$

- What happens if the surfer is doing infinitely many steps?
  - $\lim_{t\to\infty} \boldsymbol{\pi}(t)$ is called the limiting distribution (if it exists)

- Under some "technical conditions", a Markov chain has a limiting distribution which is equal to its unique stationary distribution
  - ➢ we have $\boldsymbol{r} = \lim_{t\to\infty} \boldsymbol{\pi}(t)$ // rank score of page i = r_i = $\lim_{t\to\infty} \Pr(X_t = i)$
  - limit of the sequence $\boldsymbol{\pi}\boldsymbol{B}$, $(\boldsymbol{\pi}\boldsymbol{B})\boldsymbol{B}$, $\big((\boldsymbol{\pi}\boldsymbol{B})\boldsymbol{B}\big)\boldsymbol{B}$, … equals to $\boldsymbol{r}$

Random Walk Interpretation

- Given the "technical conditions" we have $\boldsymbol{r} = \lim_{t\to\infty} \boldsymbol{\pi}(t)$
  - limit of the sequence $\boldsymbol{\pi B}$, $(\boldsymbol{\pi B})\boldsymbol{B}$, $\big((\boldsymbol{\pi B})\boldsymbol{B}\big)\boldsymbol{B}$, … equals to $\boldsymbol{r}$

➢ Probability of reaching a node does <u>not</u> depend on start point of surfer

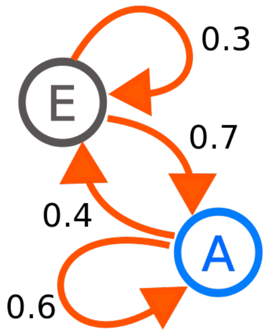
- Intuition: Assume that when $t \to \infty$, $\boldsymbol{B}^t$ converges to a matrix whose rows are the same. In this case: one row of $\lim_{t\to\infty} \boldsymbol{B}^t$ specifies the limiting distribution.

$$\lim_{t\to\infty} \boldsymbol{B}^{(t-1)} = \begin{bmatrix} a & b & c \\ a & b & c \\ a & b & c \end{bmatrix} \Rightarrow \lim_{t\to\infty} \boldsymbol{\pi}(t) = \lim_{t\to\infty} \boldsymbol{\pi B}^{(t-1)} = \underbrace{\left(\begin{bmatrix} \pi_1 & \pi_2 & \pi_3 \end{bmatrix} \begin{bmatrix} a & b & c \\ a & b & c \\ a & b & c \end{bmatrix} \right)}_{\boxed{= \begin{bmatrix} a & b & c \end{bmatrix}}}$$

Existence and Uniqueness

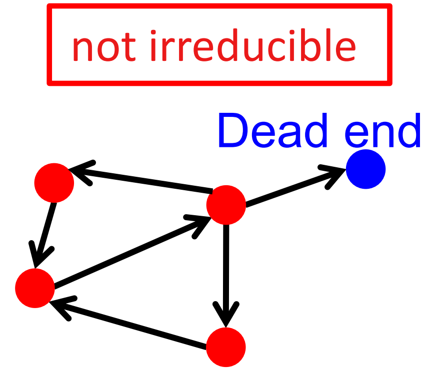
- What are the “technical conditions”?
  - Being **Irreducible** and **Aperiodic**

- **Irreducible**: it is possible to get to any state from any state
- **Aperiodic**: a state i is aperiodic if there exists n such that for all n' ≥ n: $\Pr(X_{n'} = i | X_1 = i) > 0$
  - A Markov chain is aperiodic if every state is aperiodic
  - An irreducible Markov chain only needs one aperiodic state to imply all states are aperiodic

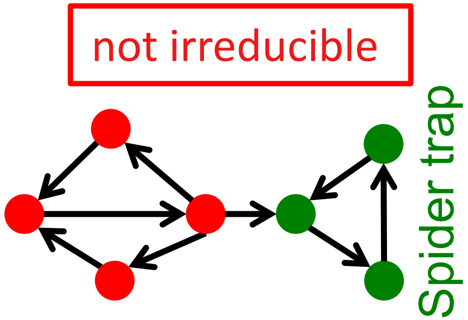


PageRank: Problems

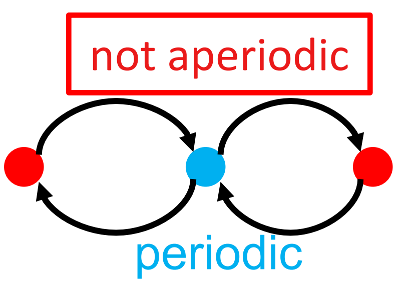
- Some pages are
  dead ends (have no out-links)
  - Random walk has "nowhere" to go to
  - Such pages cause importance to "leak out"



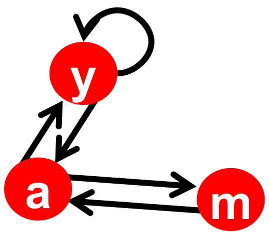
- Spider traps:
  (all out-links are within the group)
  - Random walk gets "stuck" in a trap
  - And eventually spider traps absorb all importance



- Periodic states:
  - If we start at the state, we will return to the state in fixed periods.



Solution: Random Teleports



- At each step, random surfer has **two options**:
  - With probability β, follow a link at random
  - With probability 1 – β, jump to some random page

- PageRank equation [Brin-Page, 98]

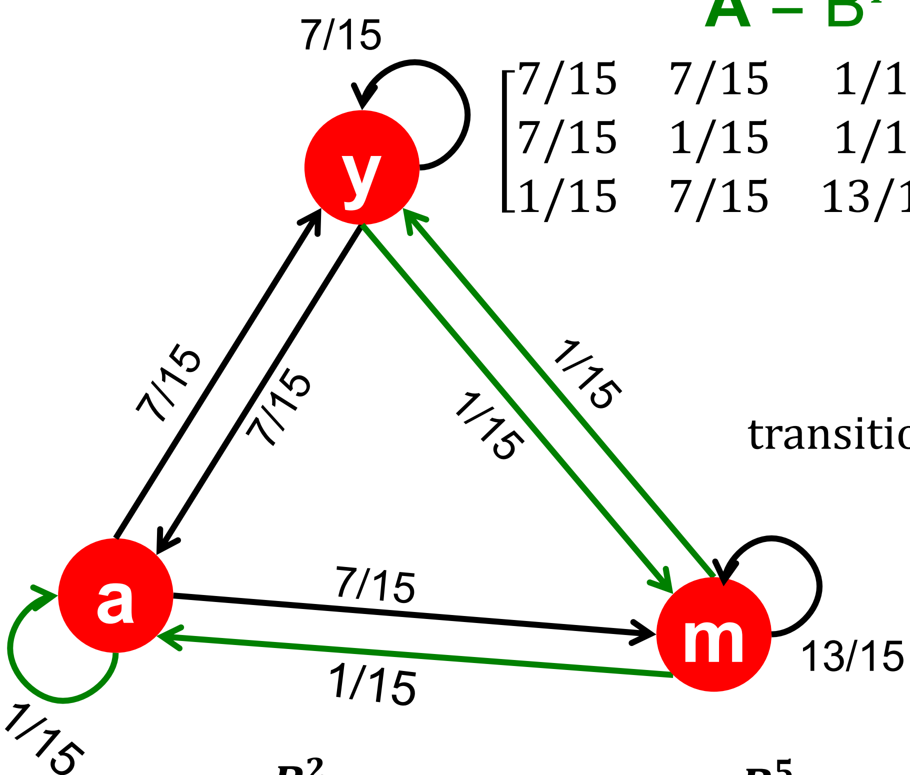
$$r_j = \sum_{i \to j} \beta \frac{r_i}{d_i} + (1-\beta)\frac{1}{N} \qquad // = \sum_{i \to j} \beta \frac{r_i}{d_i} + \sum_i (1-\beta)\frac{r_i}{N}$$

- In matrix notation: $A = \beta\, M + (1-\beta)\left[\frac{1}{N}\right]_{N \times N}$
  - final solution: $\boldsymbol{r} = \boldsymbol{A} \cdot \boldsymbol{r}$

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

This formulation assumes that $\boldsymbol{M}$ has no dead ends. We can either preprocess matrix $\boldsymbol{M}$ to remove all dead ends or explicitly follow random teleport links with probability 1.0 from dead-ends.

Illustration: Random Teleports (β = 0.8)

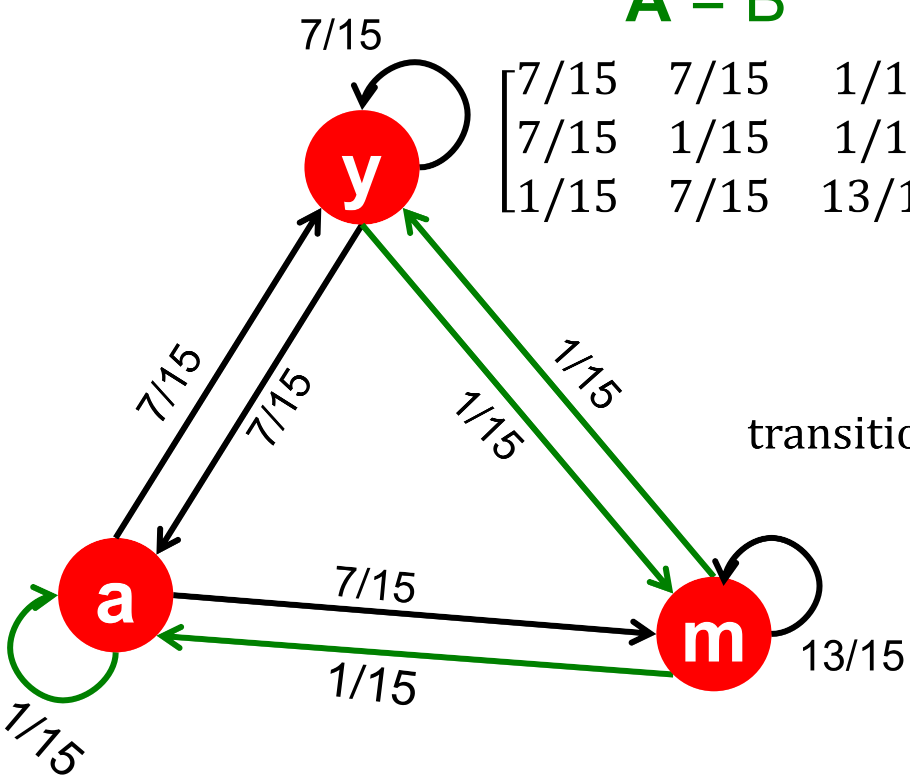


$$
\underbrace{\begin{bmatrix} 7/15 & 7/15 & 1/15 \\ 7/15 & 1/15 & 1/15 \\ 1/15 & 7/15 & 13/15 \end{bmatrix}}_{\mathbf{A} = \mathrm{B}^{T}} = 0.8 \underbrace{\begin{array}{c} \begin{array}{ccc} \mathrm{y} & \mathrm{a} & \mathrm{m} \end{array} \\ \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix} \end{array}}_{\mathbf{M}} + 0.2 \underbrace{\begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}}_{[1/N]_{NxN}}
$$

$$
\text{transition matrix} = \boldsymbol{B} = \boldsymbol{A}^{T} = \begin{array}{c} \mathrm{y} \\ \mathrm{a} \\ \mathrm{m} \end{array} \begin{bmatrix} 7/15 & 7/15 & 1/15 \\ 7/15 & 1/15 & 7/15 \\ 1/15 & 1/15 & 13/15 \end{bmatrix}
$$

$$
\begin{array}{c} \\ \mathrm{y} \\ \mathrm{a} \\ \mathrm{m} \end{array}
\begin{array}{c} \boldsymbol{B}^{2} \\ \begin{bmatrix} 0.440 & 0.253 & 0.306 \\ 0.280 & 0.253 & 0.466 \\ 0.120 & 0.093 & 0.786 \end{bmatrix} \end{array}
\ldots
\begin{array}{c} \boldsymbol{B}^{5} \\ \begin{bmatrix} 0.249 & 0.174 & 0.576 \\ 0.228 & 0.161 & 0.609 \\ 0.195 & 0.141 & 0.662 \end{bmatrix} \end{array}
\ldots
\begin{array}{c} \boldsymbol{B}^{19} \\ \begin{bmatrix} 0.212 & 0.151 & 0.636 \\ 0.212 & 0.151 & 0.636 \\ 0.212 & 0.151 & 0.636 \end{bmatrix} \end{array}
\begin{array}{c} \boldsymbol{B}^{20} \\ \begin{bmatrix} 0.212 & 0.151 & 0.636 \\ 0.212 & 0.151 & 0.636 \\ 0.212 & 0.151 & 0.636 \end{bmatrix} \end{array}
$$

Illustration: Random Teleports (β = 0.8)



$$\mathbf{A} = \mathrm{B}^T \qquad \mathbf{M} \qquad [\mathbf{1/N}]_{\mathbf{NxN}}$$

$$\begin{bmatrix} 7/15 & 7/15 & 1/15 \\ 7/15 & 1/15 & 1/15 \\ 1/15 & 7/15 & 13/15 \end{bmatrix} = 0.8 \begin{array}{c} \begin{array}{ccc} \mathrm{y} & \mathrm{a} & \mathrm{m} \end{array} \\ \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix} \end{array} + 0.2 \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}$$

$$\text{transition matrix} = \boldsymbol{B} = \boldsymbol{A}^{\boldsymbol{T}} = \begin{array}{c} \mathrm{y} \\ \mathrm{a} \\ \mathrm{m} \end{array} \begin{bmatrix} 7/15 & 7/15 & 1/15 \\ 7/15 & 1/15 & 7/15 \\ 1/15 & 1/15 & 13/15 \end{bmatrix}$$

$$\begin{array}{ccccccc} \boldsymbol{\pi} & & \boldsymbol{\pi B^2} & & \boldsymbol{\pi B^{15}} & & \boldsymbol{\pi B^{16}} \\ [1/3 \quad 1/3 \quad 1/3] & & [0.333 \quad 0.2 \quad 0.467] & \ldots & [0.212 \quad 0.151 \quad 0.636] & & [0.212 \quad 0.151 \quad 0.636] \end{array}$$

Notes on Computation

- Attention: **The matrix A is dense**!
  - N^2 non-zero entries
  - ➢ you should never compute $\boldsymbol{r}$ in such a way

- Consider the teleport by adding constant penalty to each term
  - ➢ iterate $r_j \leftarrow \sum_{i \to j} \beta \frac{r_i}{d_i} + (1-\beta)\frac{1}{N}$ until convergence
  - only neighbors need to be considered
- To maintain sparsity in matrix form multiply by $\beta \boldsymbol{M}$ then add a vector
  - $\boldsymbol{r} = \beta\, \boldsymbol{M}\, \boldsymbol{r} + (1-\beta)\left[\frac{1}{N}\right]_N$

- **Vertex-oriented computation**
  - each vertex performs local computations

Systems/Frameworks for Graph Processing

- Specialized systems for such kind of graph processing
  - *GraphLab (Dato, Turi)*
  - *Giraph* (open source counterpart to Google's Pregel)
  - *GraphX*: Library for graph processing on top of Spark

- **Crucial aspect: vertex-oriented programming**
  - each vertex performs local computations
  - GAS principle — **gather, apply, scatter**: each vertex (a) gathers information from adjacent vertices/edges (b) applies transformation, (c) scatters information to adjacent vertices
  - for PageRank only steps a + b required

- Similar concepts become also more frequent in Deep Learning Frameworks due to popularity of Graph Neural Networks

Some Problems with Page Rank

- Measures generic popularity of a page
  - Biased against topic-specific authorities
  - Solution: Topic-Sensitive PageRank
- Susceptible to Link spam
  - Artificial link topographies created in order to boost PageRank
  - Solution: TrustRank
- Uses a single measure of importance
  - Other models of importance
  - Solution: Hubs-and-Authorities (HITS, Hyperlink-Induced Topic Search)

Topic-Sensitive 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**

- Core idea: **Bias the random walk**
  - When walker teleports, pick a page from a set S
  - **Standard PageRank:** S = all pages
    - any page with equal probability
  - **Topic-Sensitive PageRank**: S = set of "relevant" pages
    - E.g., Open Directory (DMOZ) pages for a given topic/query
  - For each teleport set S, we get a different vector $\boldsymbol{r}_S$

Generalizing Topic-Sensitive PageRank

- As a matrix equation topic-sensitive PageRank takes the following form

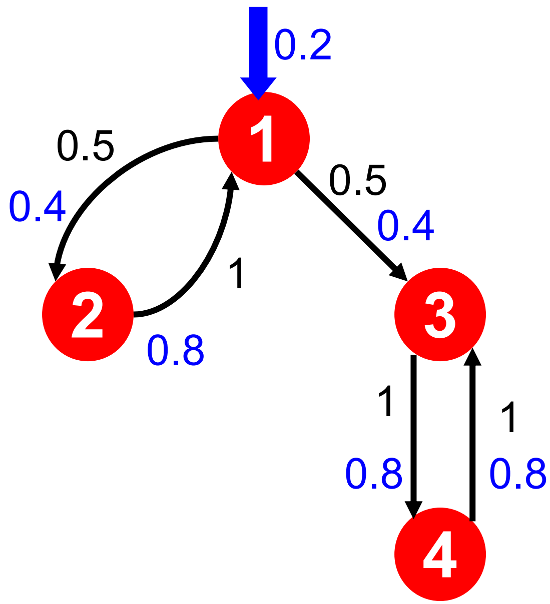
$$r = \beta M r + (1-\beta)\pi \qquad \text{where } \pi_i = \begin{cases} \frac{1}{|S|} & \text{if } i \in S \\ 0 & \text{otherwise} \end{cases}$$

- We can generalize this further to arbitrary teleport vectors π

$$r = \beta M r + (1-\beta)\pi \qquad \text{where } \sum_i \pi_i = 1$$

- The exact solution is $r = (1-\beta)(I-\beta M)^{-1}\pi$
  - Runtime scales worse than $O(N^2)$
  - Use the iterative approximate algorithm in practice
    - Multiply by $\beta \cdot \boldsymbol{M}$, then add restart vector $(1-\beta)\pi$, repeat, …
    - Maintains sparsity

Example: Topic-Sensitive PageRank



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

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

$\boldsymbol{S} = \{1\}, \quad \beta = 0.90$:
$\boldsymbol{r} = [0.17, 0.07, 0.40, 0.36]$

$\boldsymbol{S} = \{1\}, \quad \beta = 0.8$:
$\boldsymbol{r} = [0.29, 0.11, 0.32, 0.26]$

$\boldsymbol{S} = \{1\}, \quad \beta = 0.70$:
$\boldsymbol{r} = [0.39, 0.14, 0.27, 0.19]$

$\boldsymbol{S} = \{1,2,3,4\}, \quad \beta = 0.8$:
$\boldsymbol{r} = [0.13, 0.10, 0.39, 0.36]$

$\boldsymbol{S} = \{1,2,3\}, \quad \beta = 0.8$:
$\boldsymbol{r} = [0.17, 0.13, 0.38, 0.30]$

$\boldsymbol{S} = \{1,2\}, \quad \beta = 0.8$:
$\boldsymbol{r} = [0.26, 0.20, 0.29, 0.23]$

$\boldsymbol{S} = \{1\}, \quad \beta = 0.8$:
$\boldsymbol{r} = [0.29, 0.11, 0.32, 0.26]$

Discovering the Topic Set 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, …

PageRank: Variants (I)

- **"Normal" PageRank:**
  - Teleports uniformly at random to any node
  - All nodes have the same teleport probability of surfer landing there:
    $$\pi = (0.1 \quad 0.1 \quad 0.1 \quad 0.1 \quad 0.1 \quad 0.1 \quad 0.1 \quad 0.1 \quad 0.1 \quad 0.1)^T$$
- **Topic-Sensitive PageRank**:
  - Teleports to a topic specific set of pages
  - Nodes can have different probabilities of surfer landing there:
    $$\pi = (0.1 \quad 0 \quad 0 \quad 0.2 \quad 0 \quad 0.5 \quad 0 \quad 0 \quad 0 \quad 0.2)^T$$
- **Personalized PageRank (Random Walk with Restarts):**
  - Teleport is always to the same node:
    $$\pi = (0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0)^T$$

PageRank: Variants

- Spam is common in the web
  - Spammer's goal: Maximize the PageRank of target page t
  - Technique:
    - Get as many links from accessible pages as possible to target page t
    - Construct "link farm" to get PageRank multiplier effect

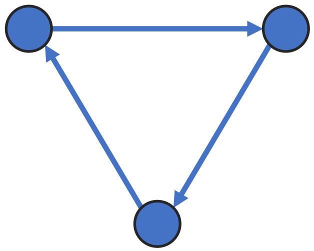
- Combating link spam via TrustRank
  - **Topic-sensitive PageRank with a teleport set of trusted pages**
  - Example: .edu domains, similar domains for non-US schools

Summary

- Core idea: Ranking of the nodes based on the link structure
- PageRank scores nodes depending on their incoming links
- With a teleport set we can rank nodes based on arbitrary factors, for example
  - Topic
  - Trust
  - Node identity
- Computing PageRank requires sparse matrix products for even moderately sized graphs

Questions

- Consider a directed cycle of length 3 as a Markov chain disregarding edge weights



- Is it irreducible? Is it aperiodic?
- How does the introduction of random teleports change the above 3-cycle?
- How can you make it aperiodic by inserting just a single edge?