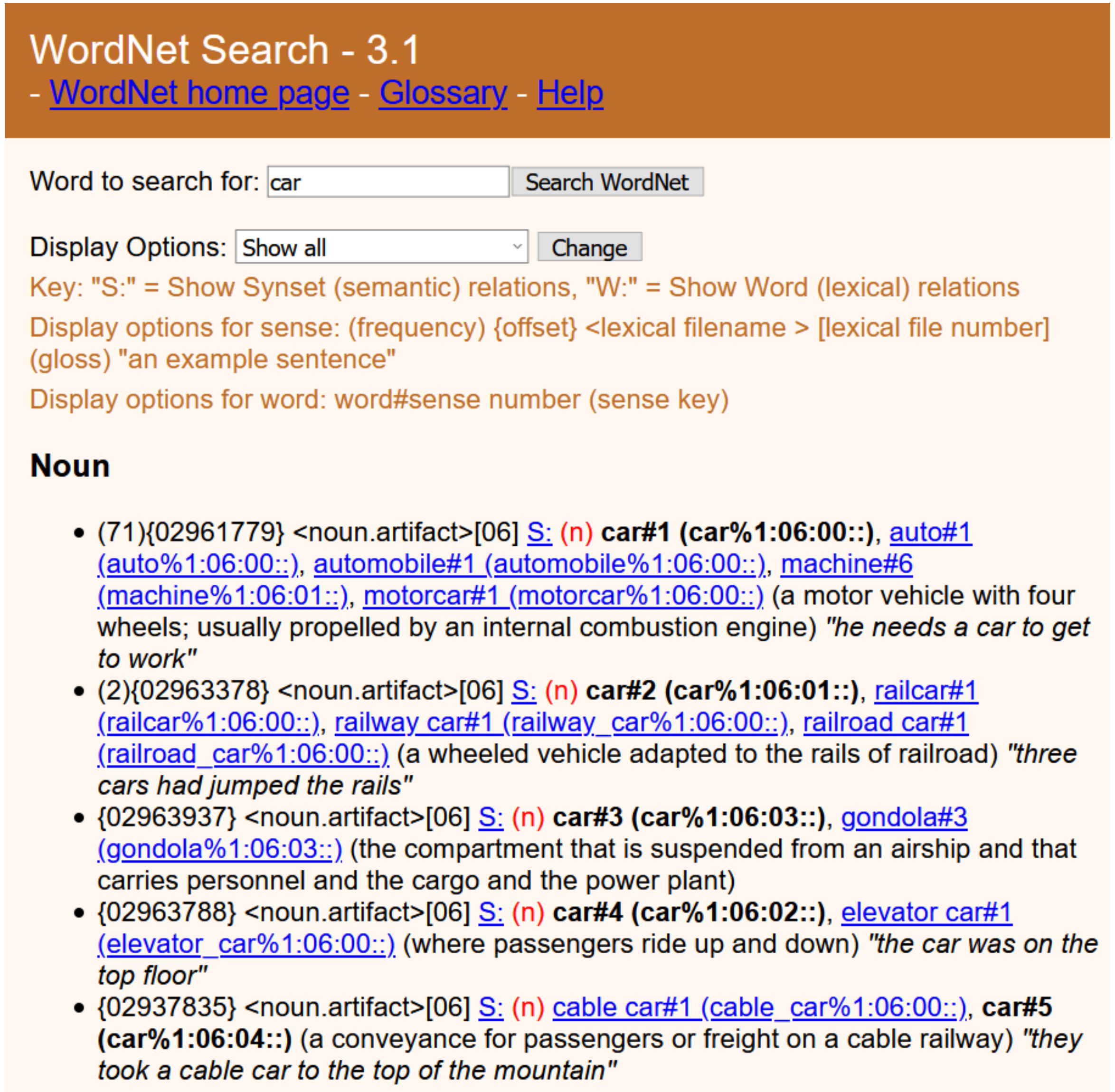
Technische Hochschule Ingolstadt

Word Net

Princeton University, 1985

semantic and lexical relations between words

Alternatives:
GermaNet
BabelNet
OpenThesaurus



WordNet Search - 3.1
- WordNet home page - Glossary - Help

Word to search for: car [Search WordNet]

Display Options: Show all [Change]

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations

Display options for sense: (frequency) {offset} <lexical filename > [lexical file number] (gloss) "an example sentence"

Display options for word: word#sense number (sense key)

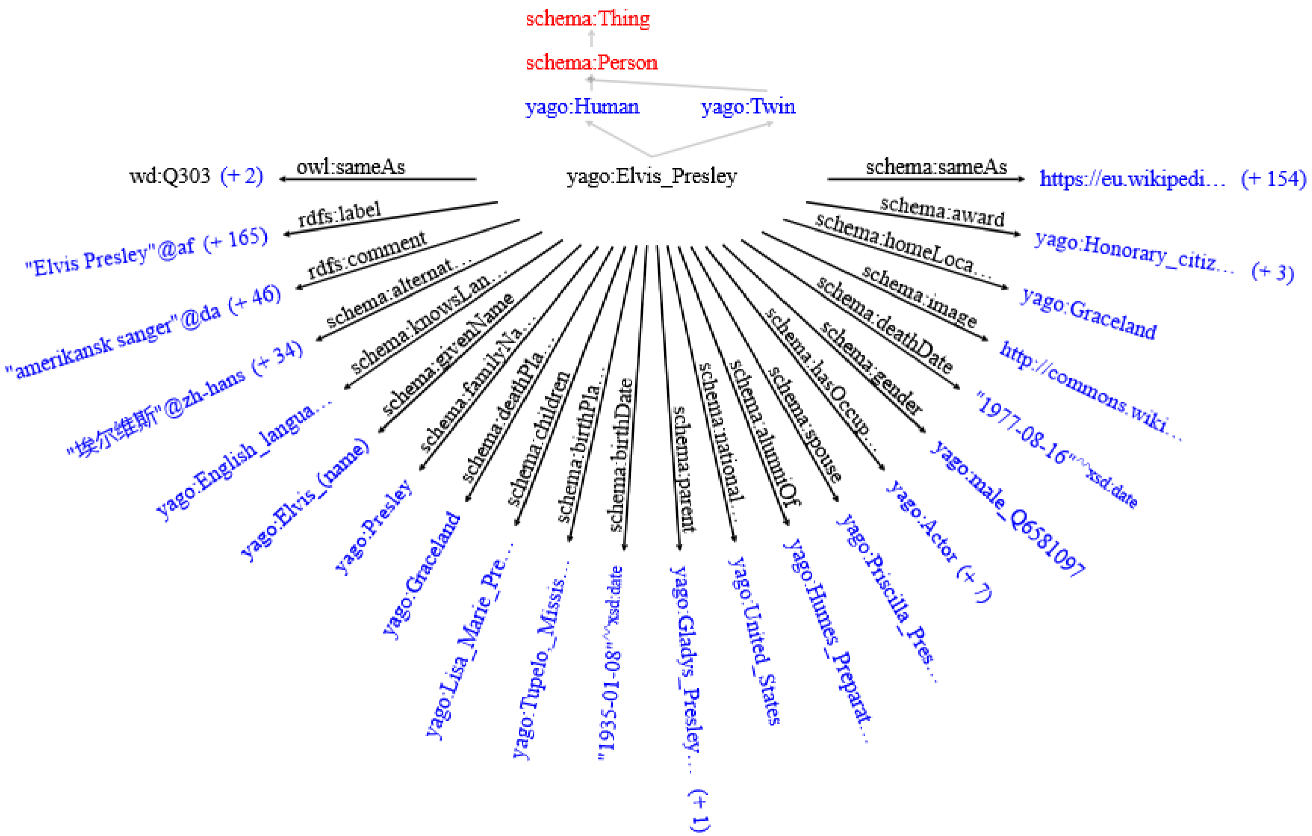
Noun

- (71){02961779} <noun.artifact>[06] S: (n) **car#1 (car%1:06:00::)**, auto#1 (auto%1:06:00::), automobile#1 (automobile%1:06:00::), machine#6 (machine%1:06:01::), motorcar#1 (motorcar%1:06:00::) (a motor vehicle with four wheels; usually propelled by an internal combustion engine) *"he needs a car to get to work"*
- (2){02963378} <noun.artifact>[06] S: (n) **car#2 (car%1:06:01::)**, railcar#1 (railcar%1:06:00::), railway car#1 (railway_car%1:06:00::), railroad car#1 (railroad_car%1:06:00::) (a wheeled vehicle adapted to the rails of railroad) *"three cars had jumped the rails"*
- {02963937} <noun.artifact>[06] S: (n) **car#3 (car%1:06:03::)**, gondola#3 (gondola%1:06:03::) (the compartment that is suspended from an airship and that carries personnel and the cargo and the power plant)
- {02963788} <noun.artifact>[06] S: (n) **car#4 (car%1:06:02::)**, elevator car#1 (elevator_car%1:06:00::) (where passengers ride up and down) *"the car was on the top floor"*
- {02937835} <noun.artifact>[06] S: (n) cable car#1 (cable_car%1:06:00::), **car#5 (car%1:06:04::)** (a conveyance for passengers or freight on a cable railway) *"they took a cable car to the top of the mountain"*

http://wordnetweb.princeton.edu/perl/webwn

Yago

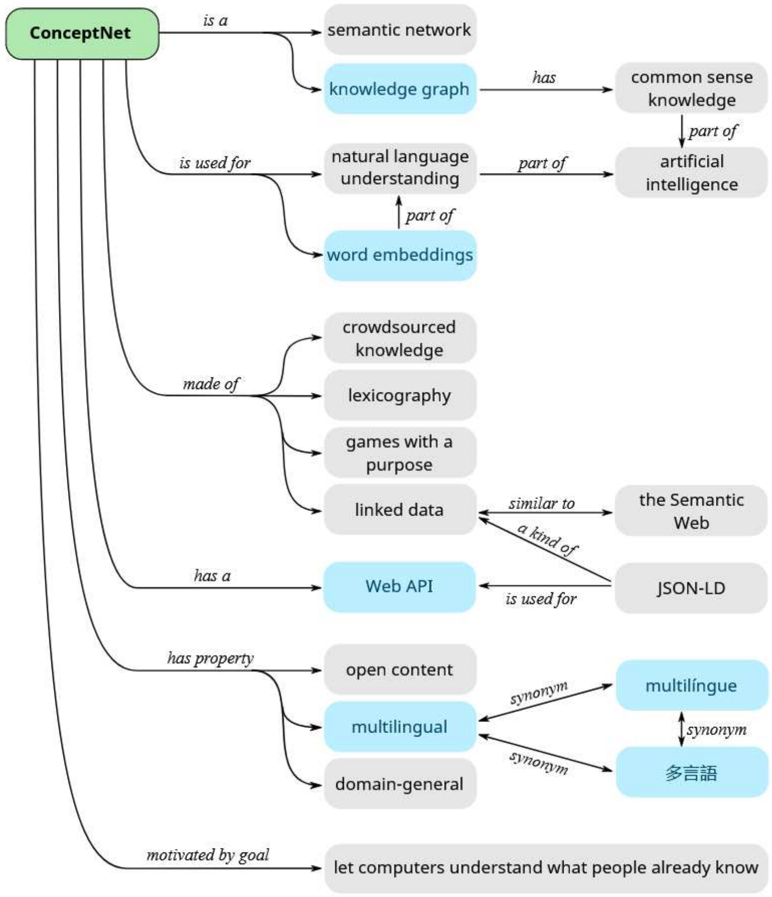
- https://yago-knowledge.org/



Technische Hochschule Ingolstadt

Concept Net

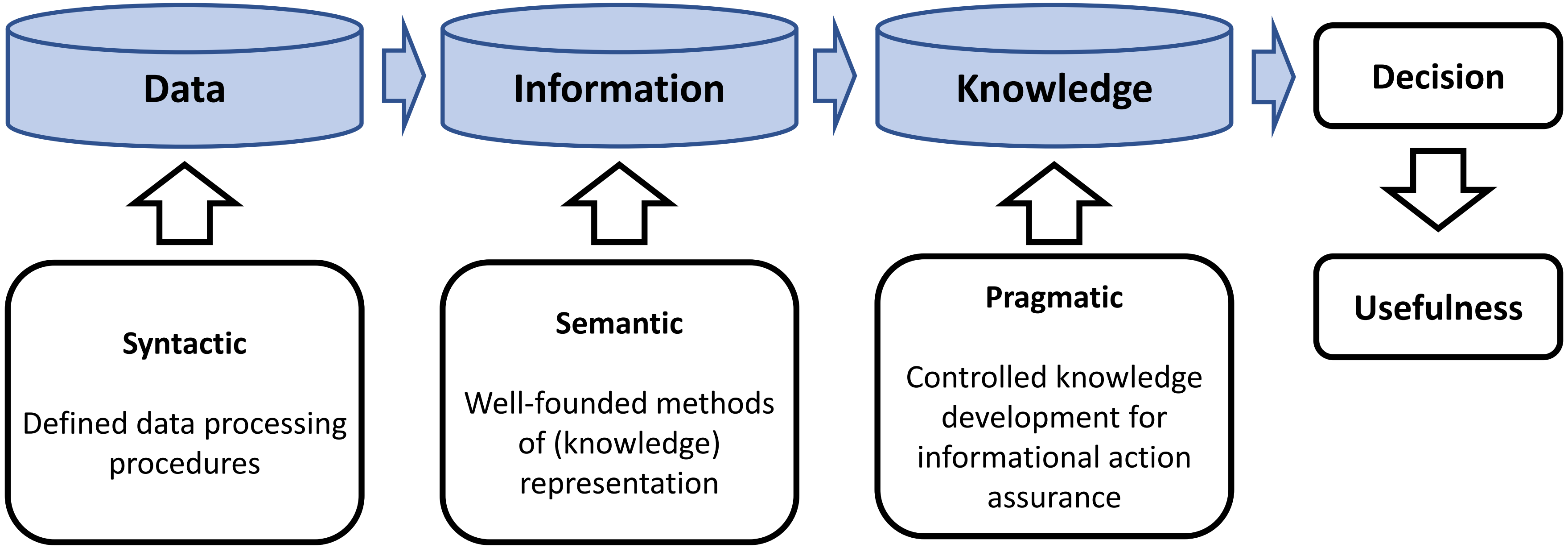
https://conceptnet.io/



Wikipedia

- **DBPedia** => extract structured content
- **Freebase** (part of google knowledge graph), today wikidata => extracted from wikipedia (A,B,C) Relationen..
- **Yago** (Saarbrücken): Ontology
- **ConceptNet**

Data, Information, Knowledge



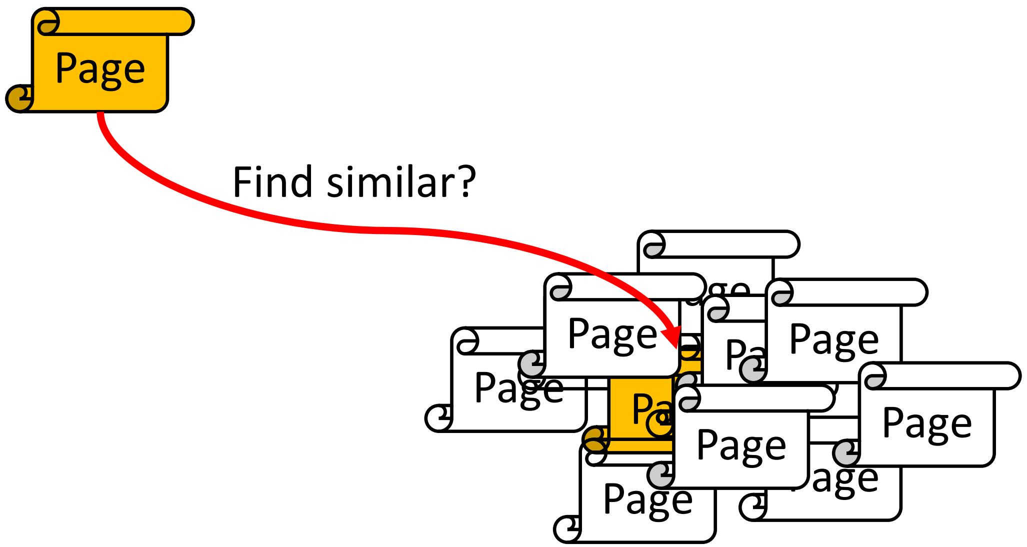
Syntax/Semantics/Pragmatics

Search on different abstraction levels:

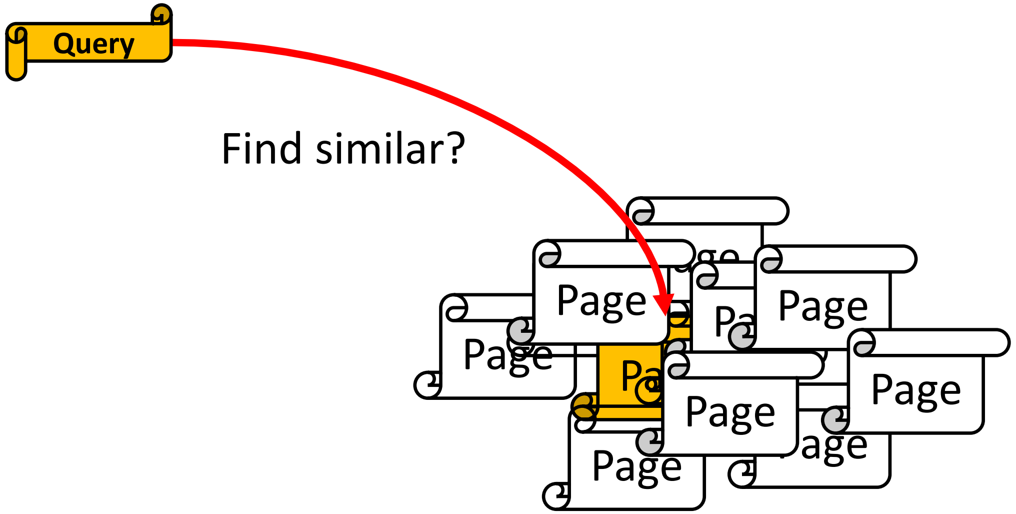
- **Syntax:** Document as sequence of symbols (e.g. string search in texts, color/texture/contour in images)
- **Semantics:** Meaning of a document (e.g. text semantics, objects occurring in an image).
- **Pragmatik:** Use of a document (purpose), e.g.: Does the document solve my problem? What is the message of the text / image?

IR deals with the semantics and pragmatics of documents

Search



Search



Problems related to Search

Problems related to Search

I. **Vagueness**
   1. User cannot specify his information request precisely
   2. vague query conditions
   3. iterative question formulation

Problems related to Search

I. Vagueness

1. User cannot specify his information request precisely
2. vague query conditions
3. iterative question formulation

II. Uncertainty

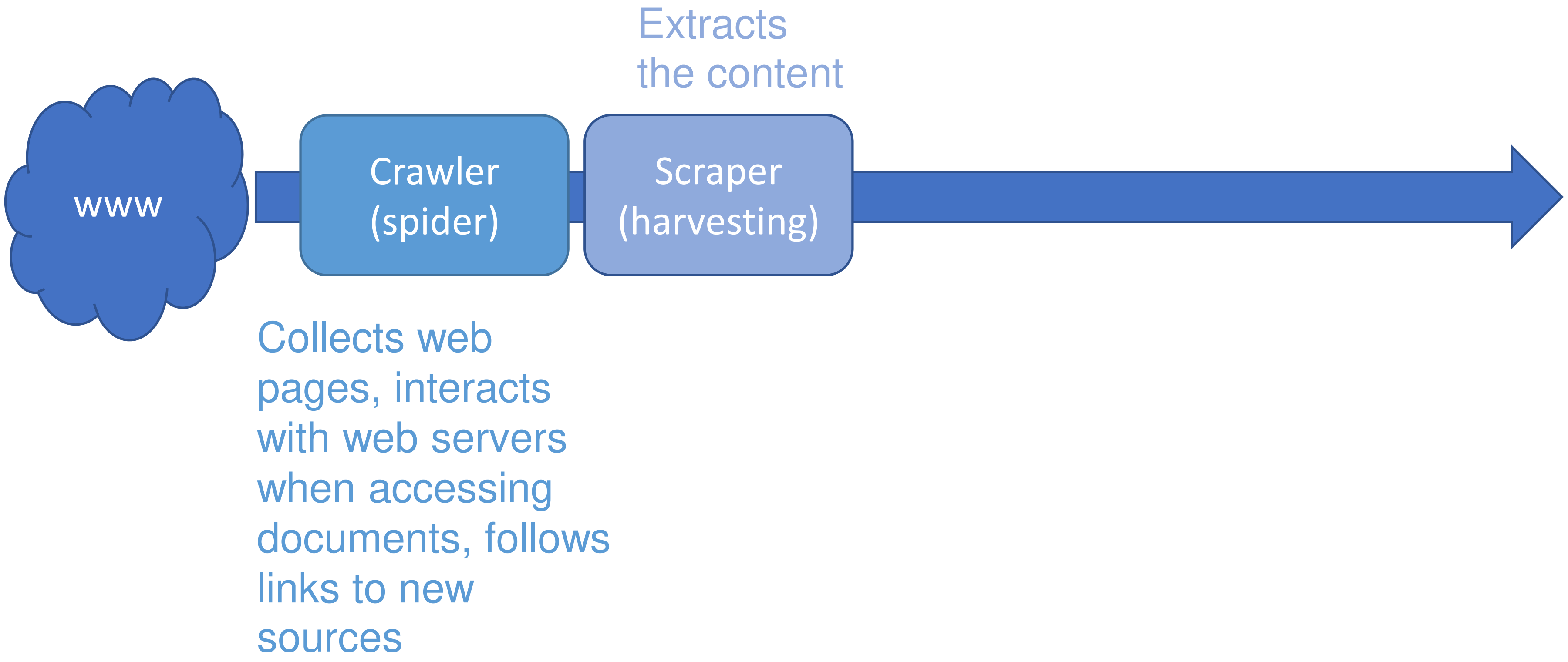
1. system has uncertain (insufficient) knowledge about the content of managed objects
2. uncertain representation (incorrect answers)
3. incomplete representation (missing answers)

Search Engines

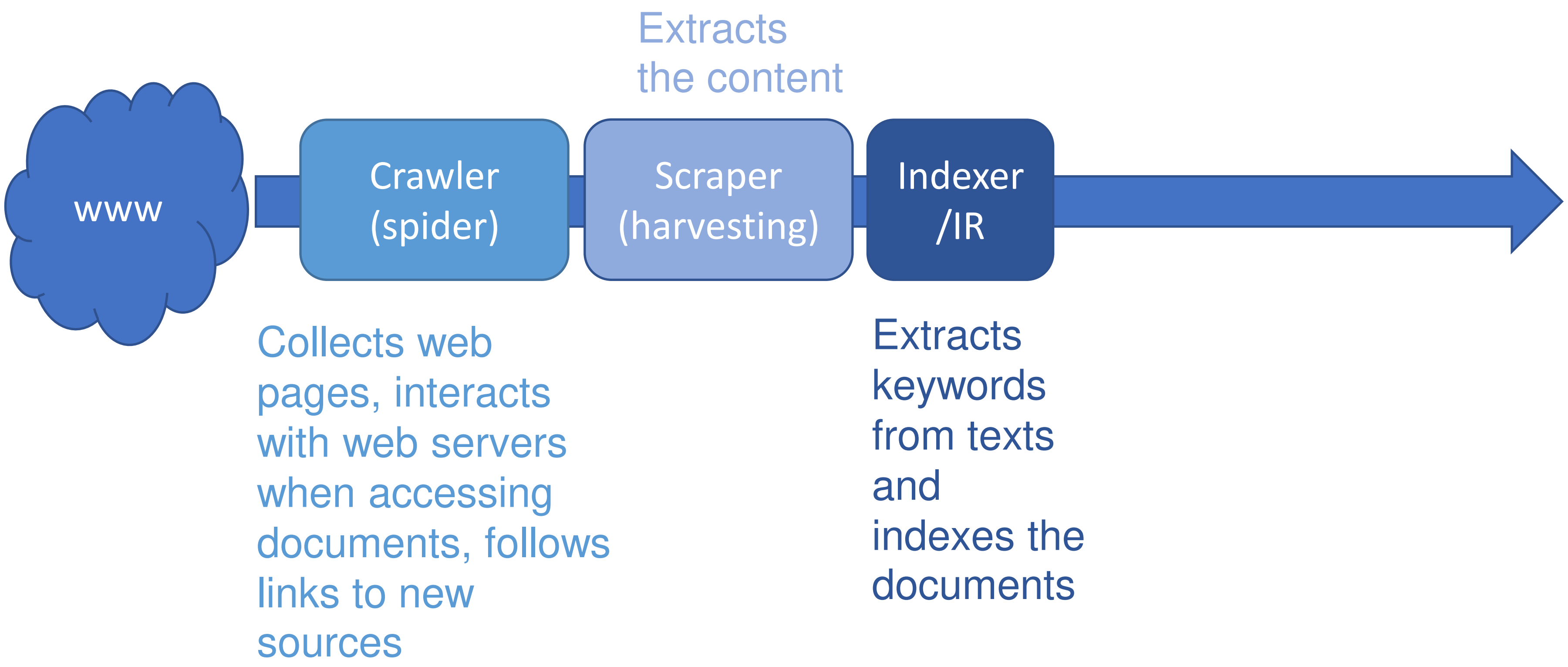
Search Engines

Collects web pages, interacts with web servers when accessing documents, follows links to new sources

Search Engines



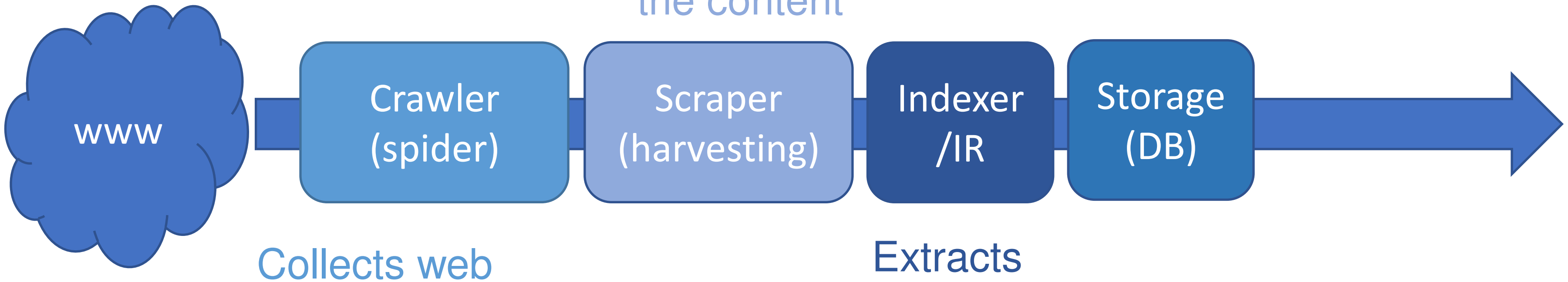
Search Engines



Search Engines

Efficient storage of extracted and processed data (e.g. in a database with inverted lists)

Extracts the content



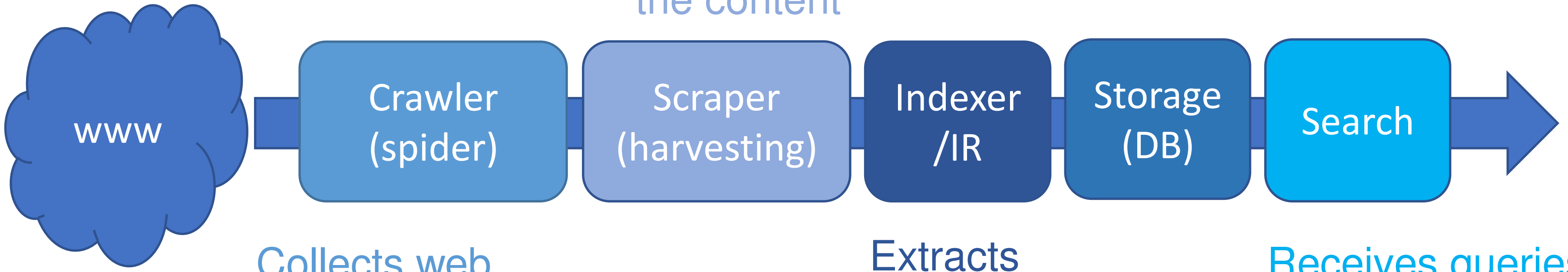
Collects web pages, interacts with web servers when accessing documents, follows links to new sources

Extracts keywords from texts and indexes the documents

Search Engines

Efficient storage of extracted and processed data (e.g. in a database with inverted lists)

Extracts the content



Collects web pages, interacts with web servers when accessing documents, follows links to new sources

Extracts keywords from texts and indexes the documents

Receives queries, searches the database for the appropriate answers.

Scrapy

Fun Task:

Extract live information from websites, e.g. https://www.ingolstadt.de/Rathaus

Scrapy

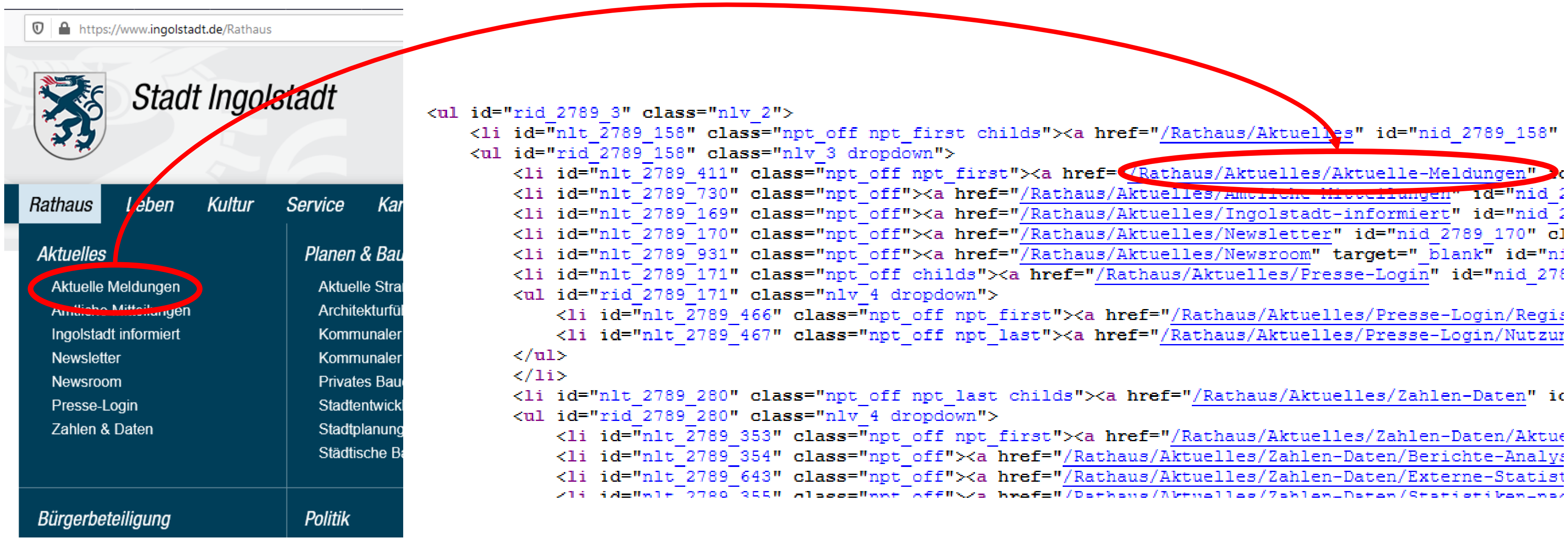
Fun Task:

Extract live information from websites, e.g.
https://www.ingolstadt.de/Rathaus

Scrapy

Fun Task:

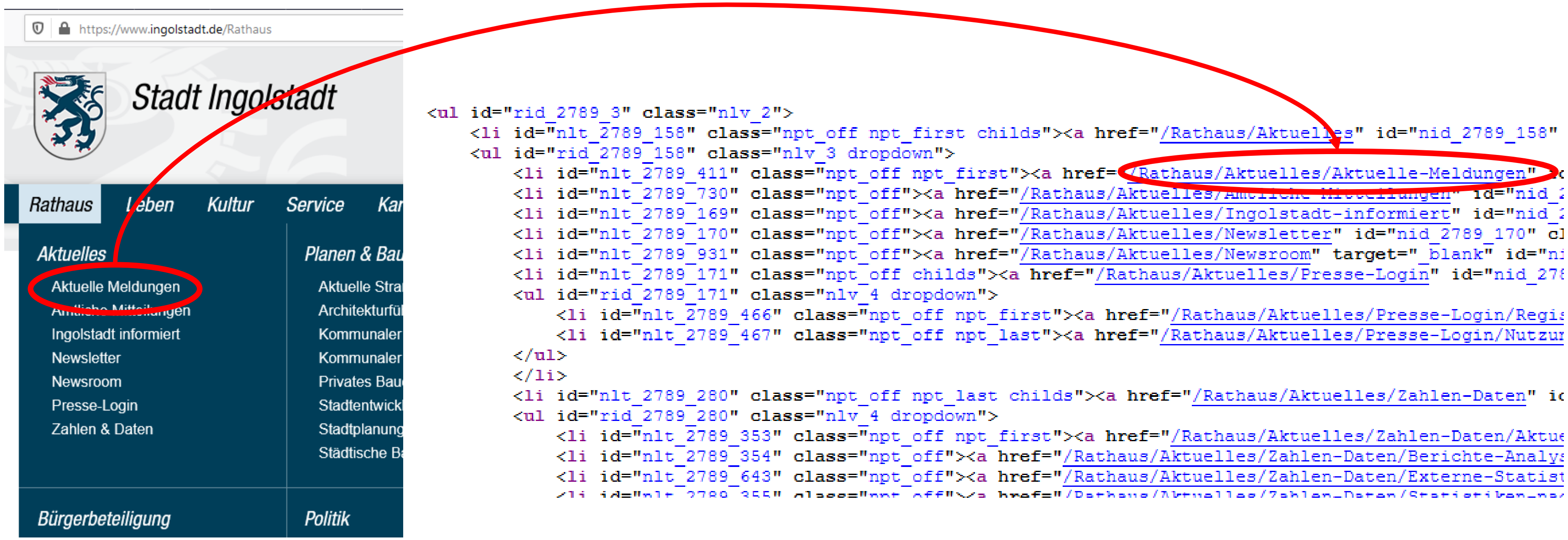
Extract live information from websites, e.g.
https://www.ingolstadt.de/Rathaus



Scrapy

Fun Task:

Extract live information from websites, e.g.
https://www.ingolstadt.de/Rathaus



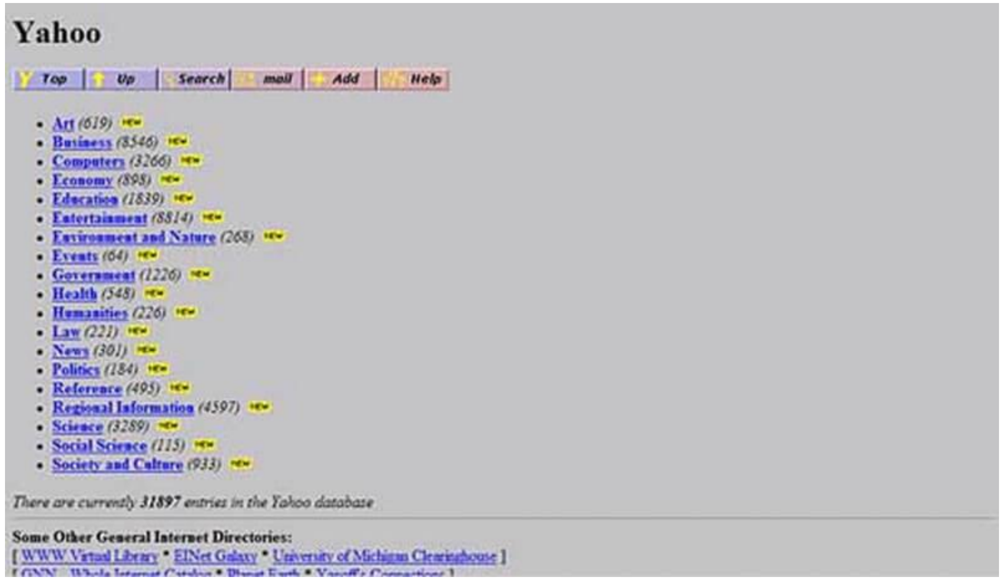
How? https://docs.scrapy.org/en/latest/intro/tutorial.html

How to organize the Web?

https://searchengineland.com/yahoo-directory-close-204370

How to organize the Web?

- 1st try: Human curated **Web directories**



https://searchengineland.com/yahoo-directory-close-204370

How to organize the Web?

- 1st try: Human curated **Web directories**
- 2nd try: **Web Search**

https://aeroadmin.com/articles/en/2020/how-search-engines-work/

How to organize the Web?

- 1st try: Human curated **Web directories**
- 2nd try: **Web Search**
  - Information Retrieval

    Find relevant docs in a small and trusted set

    - Newspaper articles, Patents, etc.

https://aeroadmin.com/articles/en/2020/how-search-engines-work/

How to organize the Web?

- 1st try: Human curated **Web directories**
- 2nd try: **Web Search**
  - Information Retrieval

    Find relevant docs in a small and trusted set
    - Newspaper articles, Patents, etc.

Problem: Web is huge, full of untrusted documents, random things, web spam, etc.

https://aeroadmin.com/articles/en/2020/how-search-engines-work/

Web Search: Two Challenges

1. **Web contains many sources of information**

   Who to "trust"?

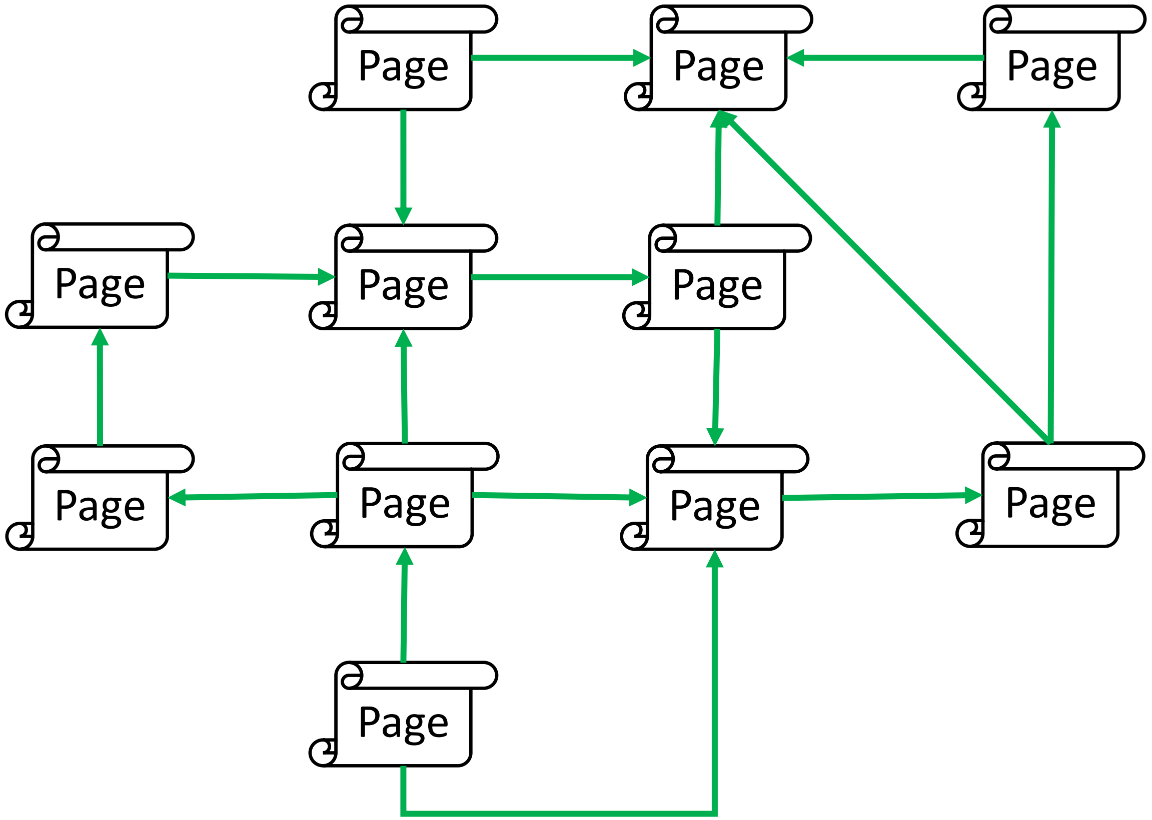
   Trick: Trustworthy pages may point to each other!

2. **What is the *best* answer to query "newspaper"?**

   No single right answer

   Trick: Pages that actually know about newspapers might all be pointing to many newspapers

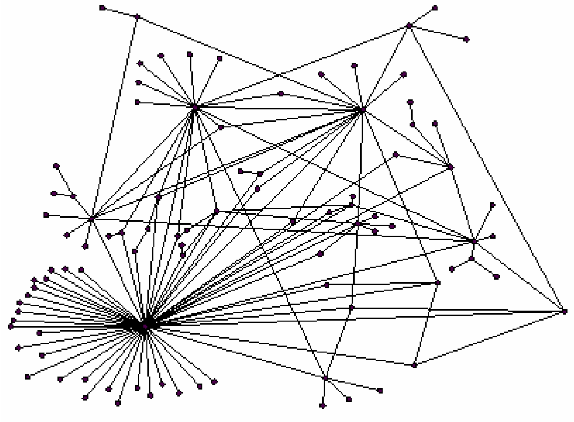
The Internet as Directed Graph



Ranking Nodes of the Graph

Ranking Nodes of the Graph

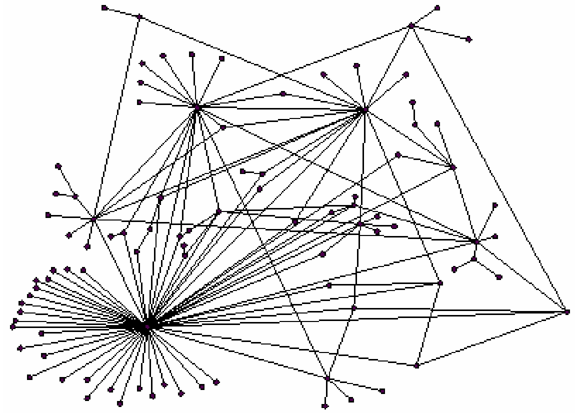
- **Not all web pages are equally "*important*"**
- **There is a large diversity in the web-graph node connectivity.**



Ranking Nodes of the Graph

- **Not all web pages are equally "*important*"**
- **There is a large diversity in the web-graph node connectivity.**

Methods for computing *importance* of nodes in a graph:



- PageRank
- Topic-Specific (Personalized) PageRank

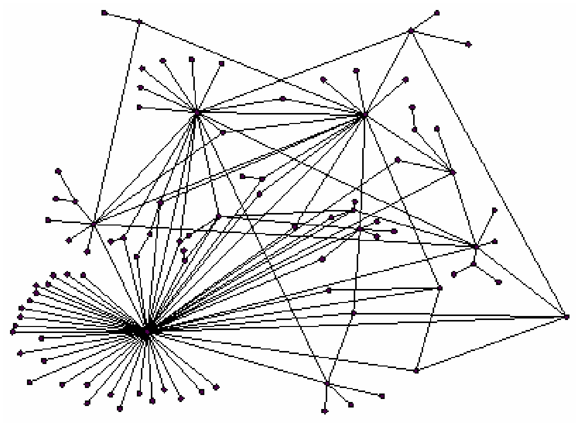
Ranking Nodes of the Graph

- **Not all web pages are equally "*important*"**
- **There is a large diversity in the web-graph node connectivity.**

Let's rank the pages by the link structure!

Methods for computing *importance* of nodes in a graph:

- PageRank
- Topic-Specific (Personalized) PageRank



Idea: Links $\approx$ Votes

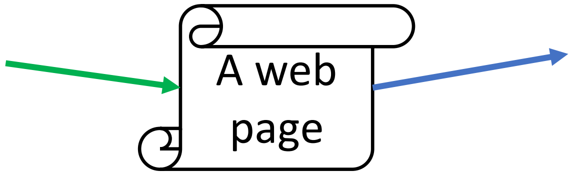
Idea: Links ≈ Votes

- **Page is more important if it has more links**

Idea: Links ≈ Votes

- **Page is more important if it has more links**

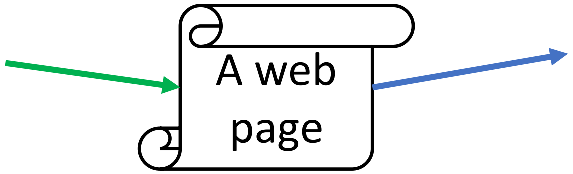
  Question: What about in-coming and Out-going links?



Idea: Links ≈ Votes

- **Page is more important if it has more links**

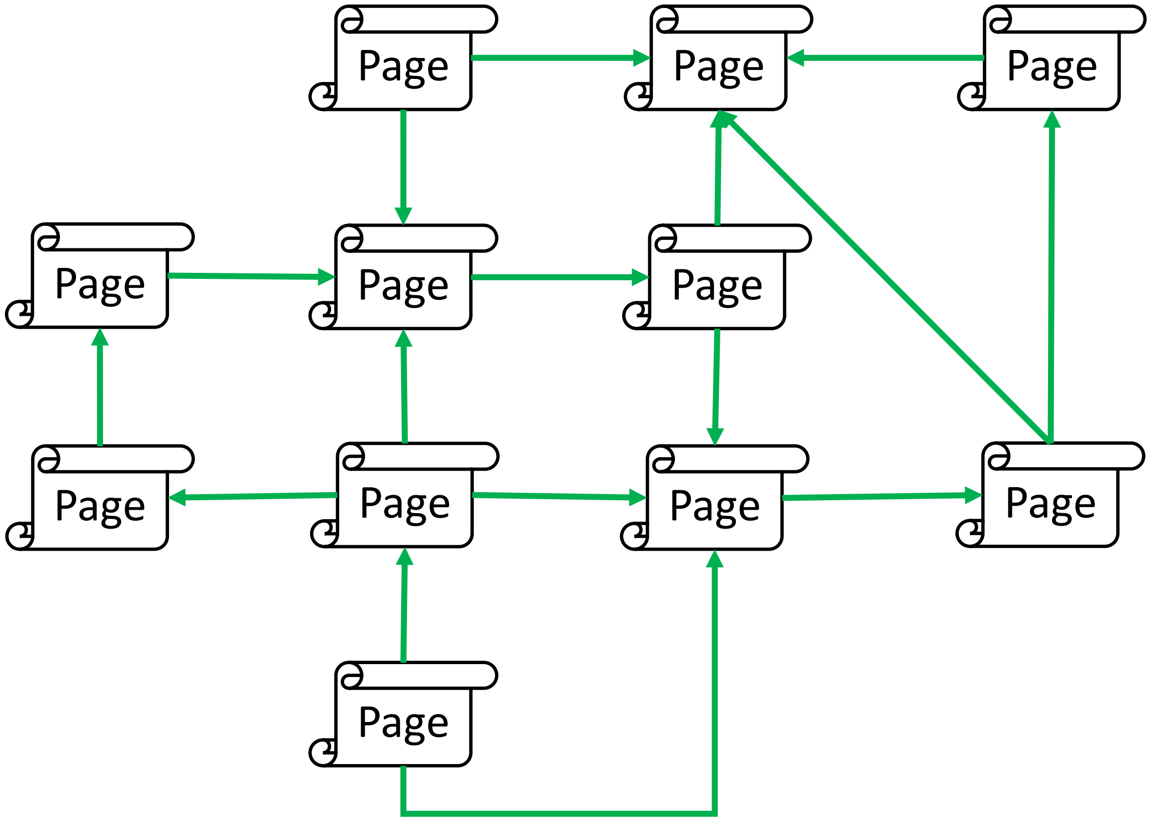
  Question: What about in-coming and Out-going links?



- **Are all in-links equal?**
  - Idea: Links from important pages count more
  - Recursion…

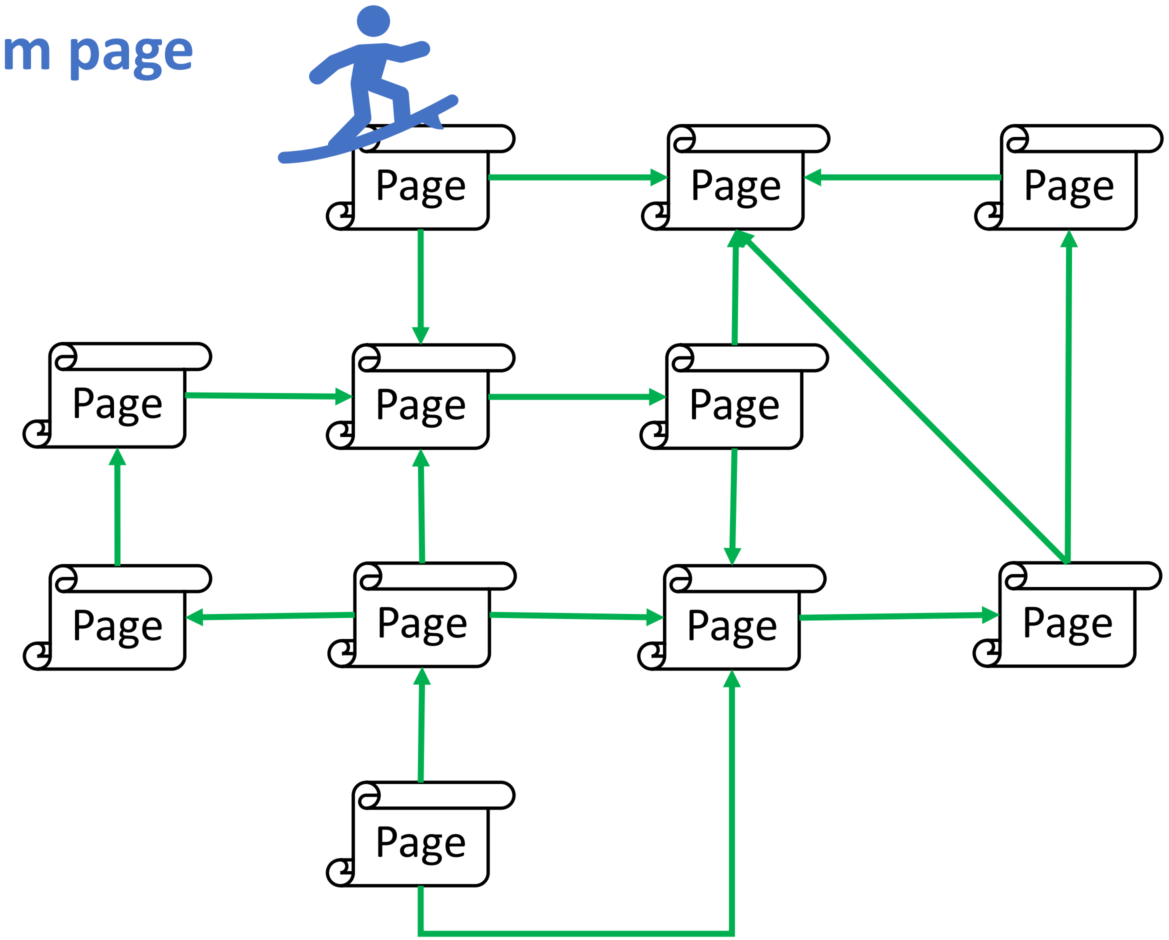
How to move around the graph?

This is Jeremy



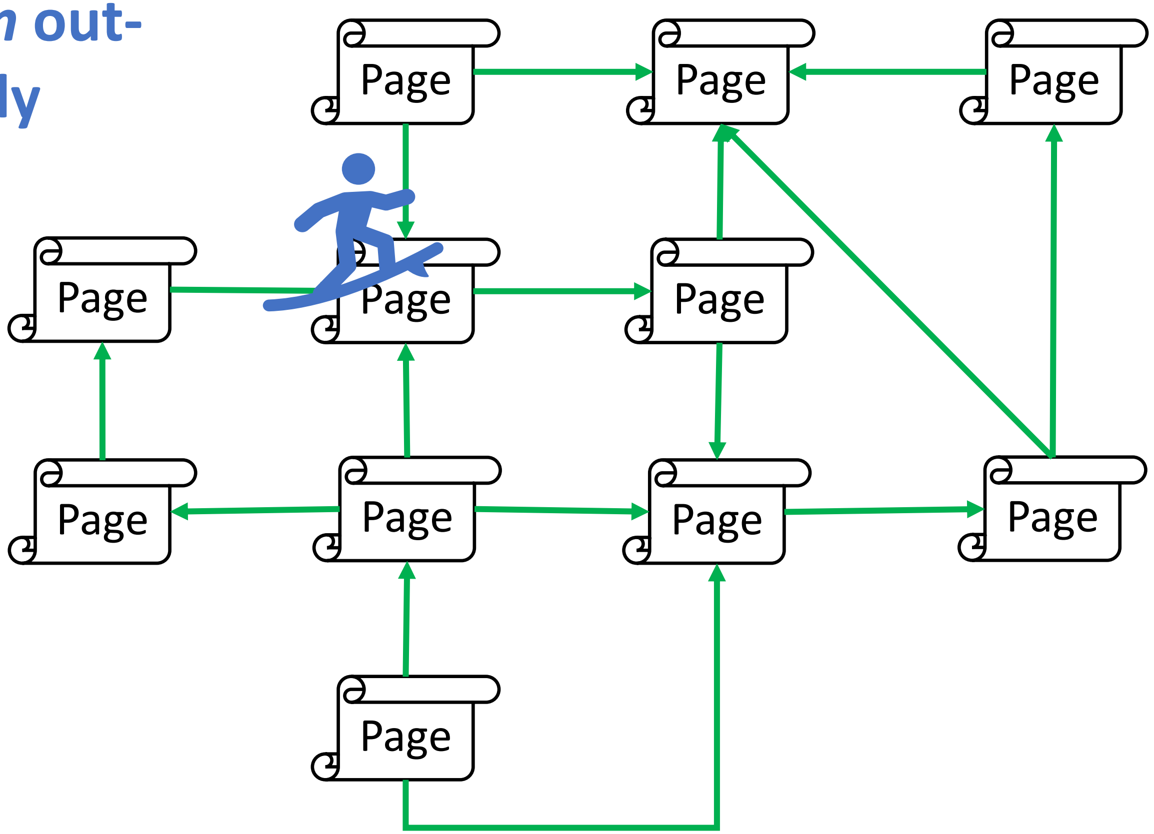
How to move around the graph?

1. **Start at random page**



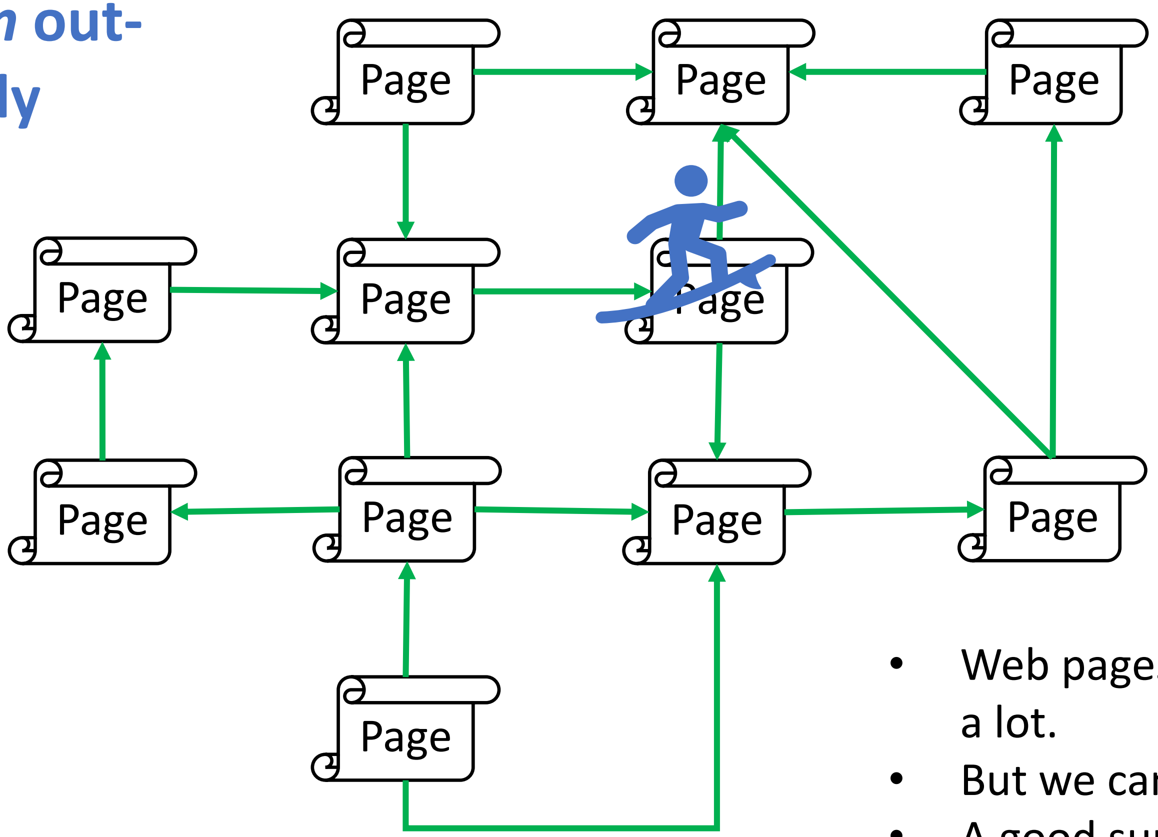
Random Surfer Model

1. **Start at random page**
2. **Follow *random* out-links repeatedly**



Random Surfer Model

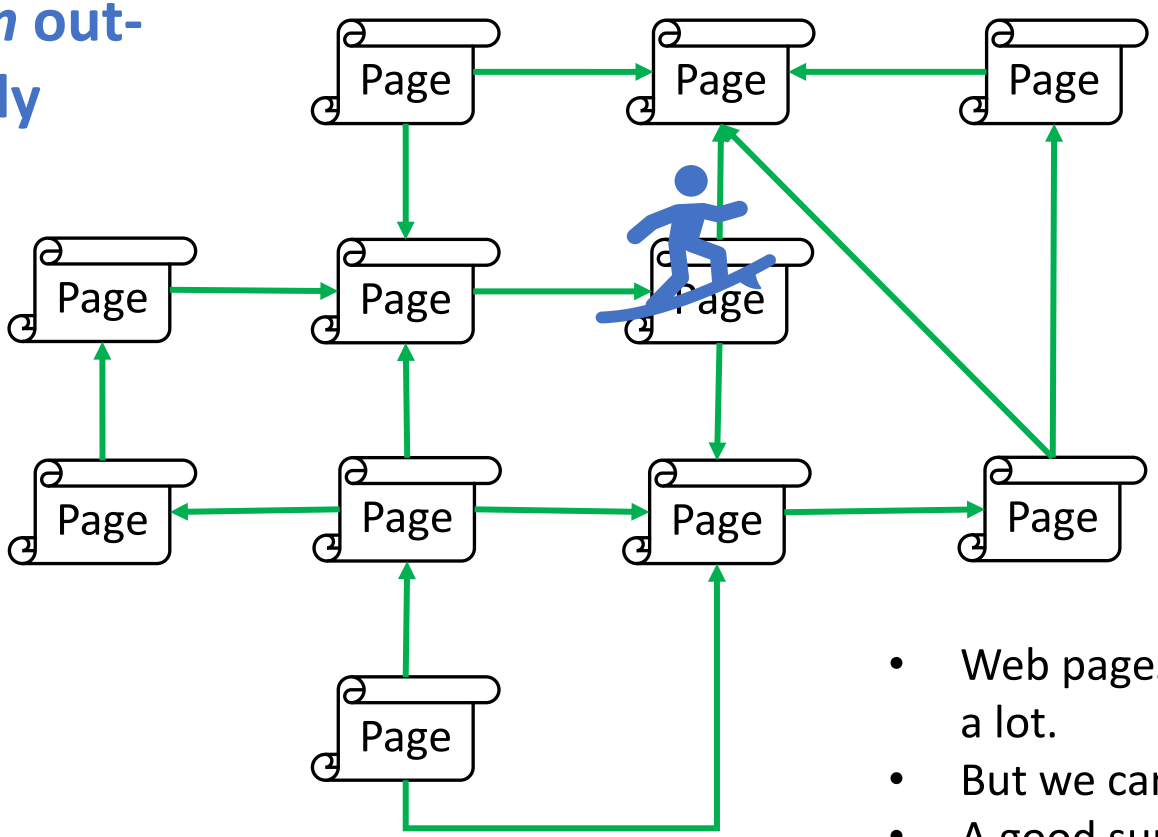
1. **Start at random page**
2. **Follow *random* out-links repeatedly**



- Web pages are **important** if people visit them a lot.
- But we can't watch everybody using the Web
- A good surrogate for visiting pages is to assume **people follow links randomly**

Random Surfer Model and Page Rank

1. **Start at random page**
2. **Follow *random* out-links repeatedly**

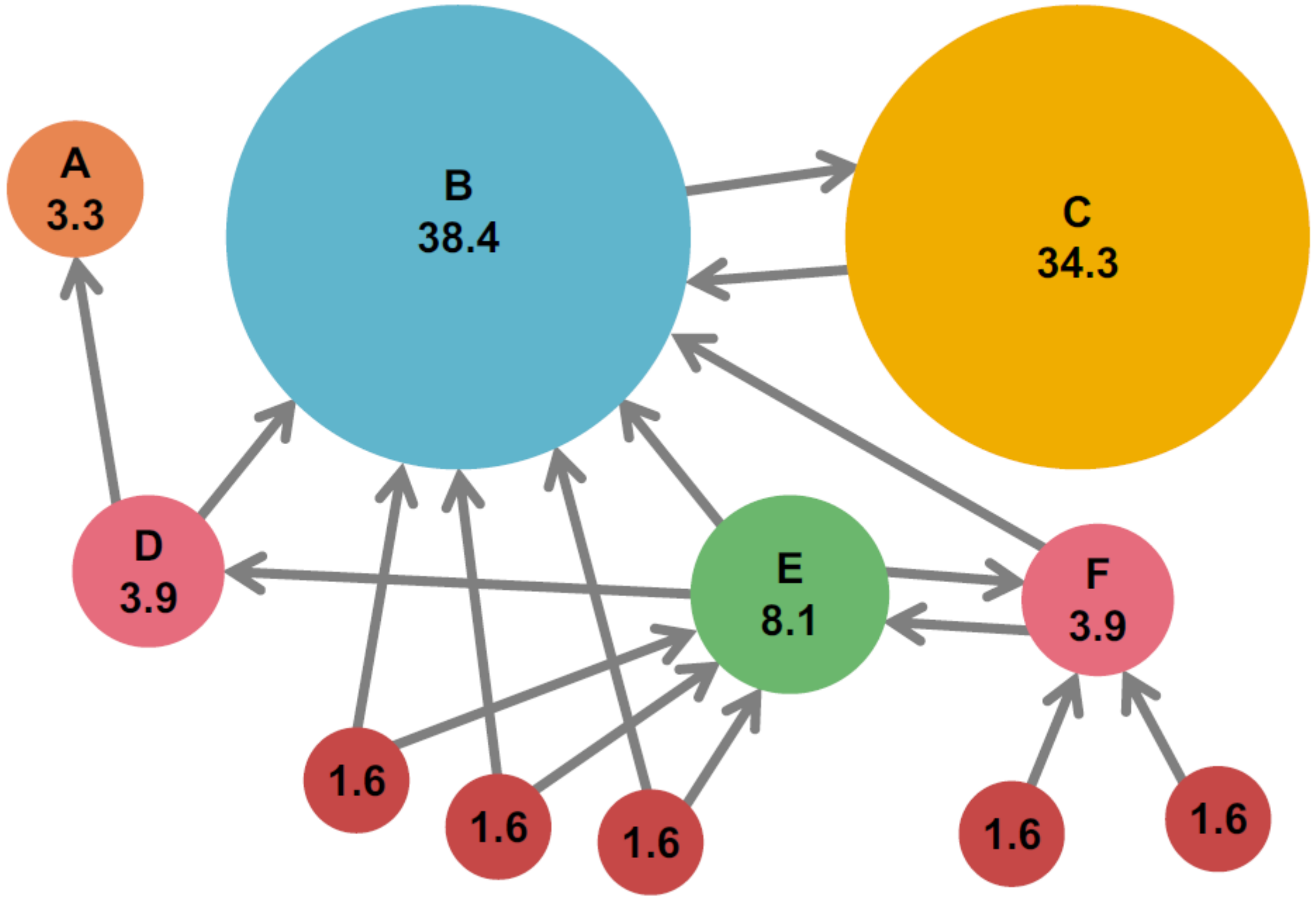


=

limiting probability of being at a page at any point in time.

- Web pages are **important** if people visit them a lot.
- But we can't watch everybody using the Web
- A good surrogate for visiting pages is to assume **people follow links randomly**

Example: Importance Scores



Intuition behind Importance Recursion

Solve the recursive equation:

“importance of a page j

=

its share of the importance of
each of its predecessor pages”

Intuition behind Importance Recursion

Page Rank r_j

Solve the recursive equation:

"importance of a page j

=

its share of the importance of each of its predecessor pages"

Intuition behind Importance Recursion

Page Rank r_j

Solve the recursive equation:

"importance of a page j
=
its share of the importance of
each of its predecessor pages"

Idea: Each link's vote is proportional to the importance of its source page

Intuition behind Importance Recursion

Page Rank r_j

Solve the recursive equation:

"importance of a page j
=
its share of the importance of
each of its predecessor pages"

Idea: Each link's vote is proportional to the importance of its source page

Intuition behind Importance Recursion

Page Rank r_j

Solve the recursive equation:

"importance of a page j
=
its share of the importance of each of its predecessor pages"

Idea: Each link's vote is proportional to the importance of its source page

- page j has importance r_j and n out-links

Intuition behind Importance Recursion

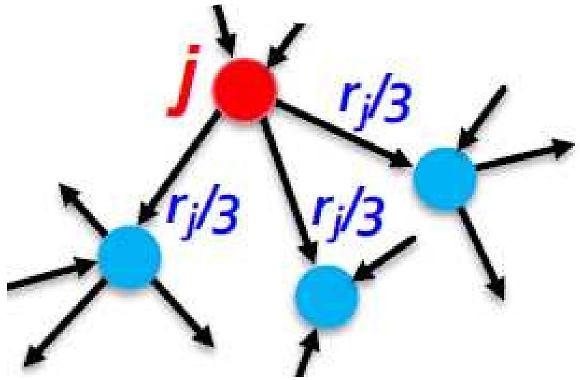
Page Rank r_j

Solve the recursive equation:

"importance of a page j
=
its share of the importance of each of its predecessor pages"

Idea: Each link's vote is proportional to the importance of its source page

- page j has importance r_j and n out-links
  $\Rightarrow$ each link gets $\frac{r_j}{n}$ votes



Intuition behind Importance Recursion

Page Rank r_j

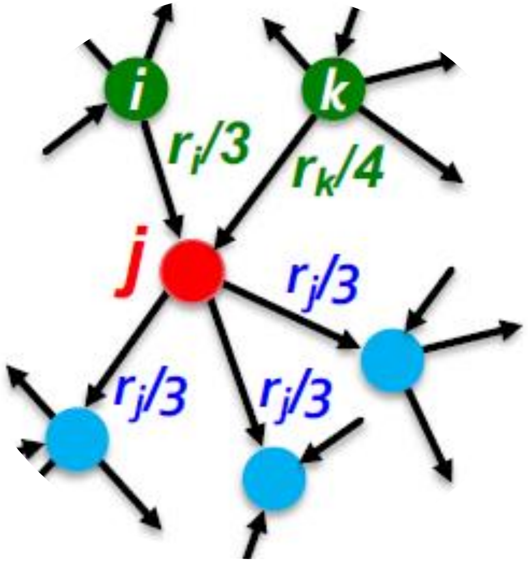
Solve the recursive equation:

"importance of a page j
=
its share of the importance of each of its predecessor pages"

Idea: Each link's vote is proportional to the importance of its source page

- page j has importance r_j and n out-links
  $\Rightarrow$ each link gets $\frac{r_j}{n}$ votes
- Page j's own importance is the sum of the votes on its in-links

$$r_j = \frac{r_i}{3} + \frac{r_k}{4}$$

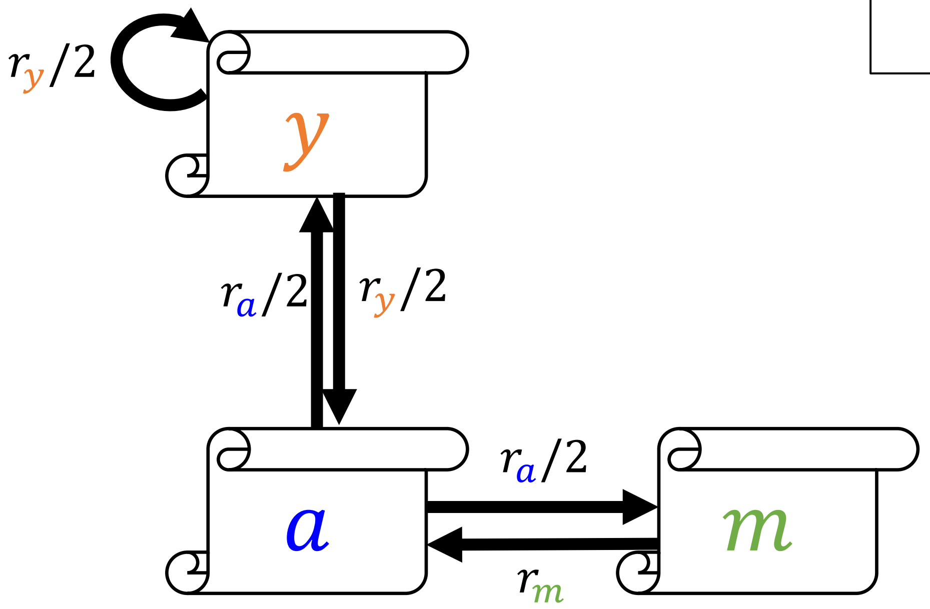


PageRank: the „Flow“ model

- A “vote” from an important page is worth more
- A page is important if it is pointed to by other important pages

Definition (Rank for page j)

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



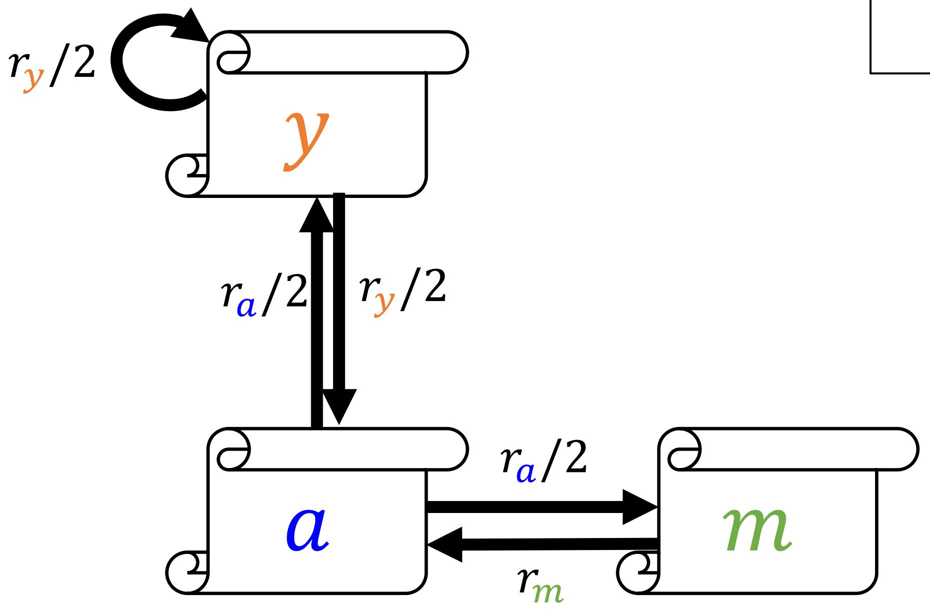
PageRank: the „Flow“ model

- A “vote” from an important page is worth more
- A page is important if it is pointed to by other important pages

Definition (Rank for page j)

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

nodes $\boldsymbol{i}$ pointing to node j



PageRank: the „Flow“ model

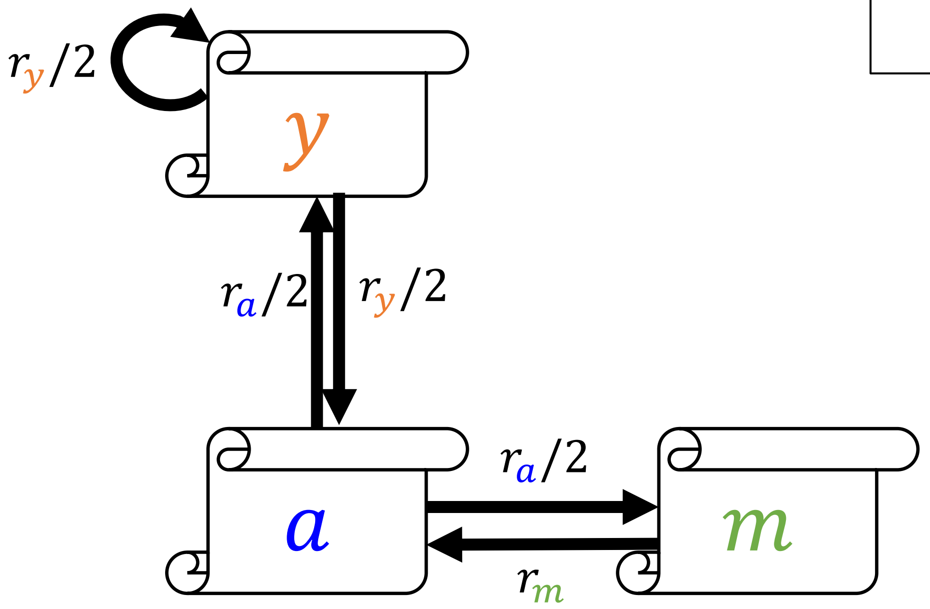
- A “vote” from an important page is worth more
- A page is important if it is pointed to by other important pages

Definition (Rank for page j)

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

d_i = out-degree of node ***i***

nodes ***i*** pointing to node j



PageRank: the „Flow“ model

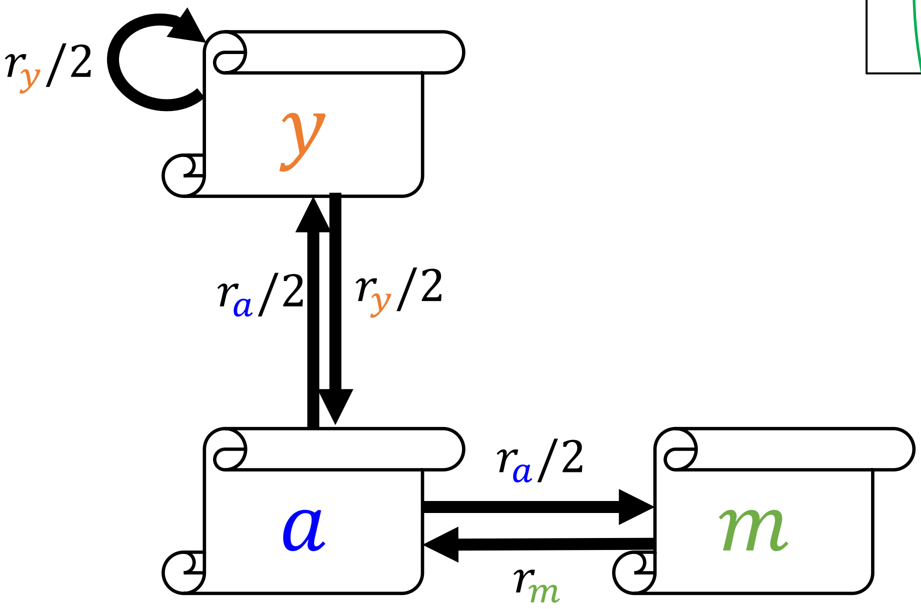
- A “vote” from an important page is worth more
- A page is important if it is pointed to by other important pages

Definition (Rank for page j)

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

d_i = out-degree of node ***i***

nodes ***i*** pointing to node j



solutions to the

„Flow“ Equations

$$\mathbf{r}_y = \frac{1}{2} \cdot r_y + \frac{1}{2} \cdot r_a$$

$$\mathbf{r}_a = \frac{1}{2} \cdot r_y + \mathbf{r}_m$$

$$\mathbf{r}_m = \frac{1}{2} \cdot r_a$$

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

Definition (Rank for page j)

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

d_i = out-degree of node ***i***

nodes ***i*** pointing to node j

solutions to the

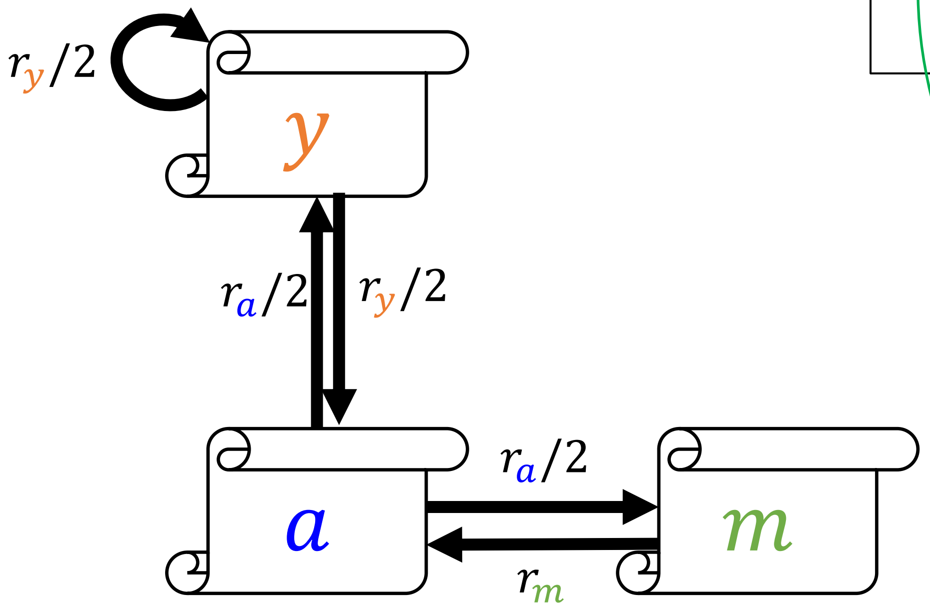
„Flow“

Equations

$$\mathbf{r}_y = \frac{1}{2}\cdot r_y + \frac{1}{2}\cdot r_a + 0\cdot r_m$$

$$\mathbf{r}_a = \frac{1}{2}\cdot r_y + 0\cdot r_a + 1\cdot r_m$$

$$\mathbf{r}_m = 0\cdot r_y + \frac{1}{2}\cdot r_a + 0\cdot r_m$$



PageRank: the „Flow“ model

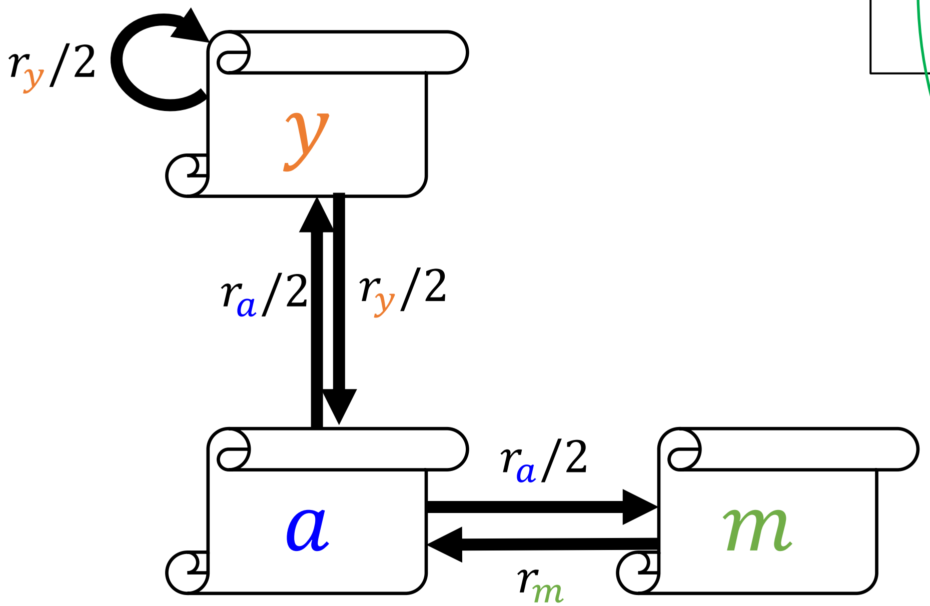
- A “vote” from an important page is worth more
- A page is important if it is pointed to by other important pages

Definition (Rank for page j)

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

d_i = out-degree of node $\boldsymbol{i}$

nodes $\boldsymbol{i}$ pointing to node j



solutions to the

„Flow“ Equations

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

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

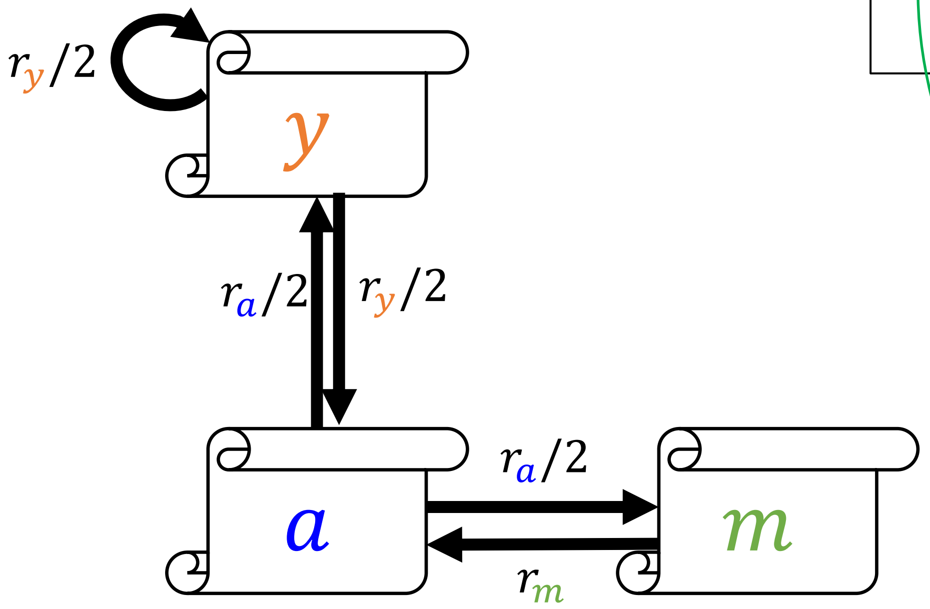
Definition (Rank for page j)

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

d_i = out-degree of node ***i***

nodes ***i*** pointing to node j

solutions to the



„Flow“

Equations

$$\begin{bmatrix} \mathbf{r}_y \\ \mathbf{r}_a \\ \mathbf{r}_m \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{r}_y \\ \mathbf{r}_a \\ \mathbf{r}_m \end{bmatrix}$$

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

PageRank: Matrix Formulation

- Define stochastic adjacency matrix $\mathbf{M}$
  - Let page i has d_i out-links
  - If $i \to j$, then $\mathbf{M_{ji}} = \frac{\mathbf{1}}{d_i}$, else $\mathbf{M_{ji}} = \mathbf{0}$

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

Definition (rank vector)

The rank vector is a vector with one entry per page; it captures importance of the page.

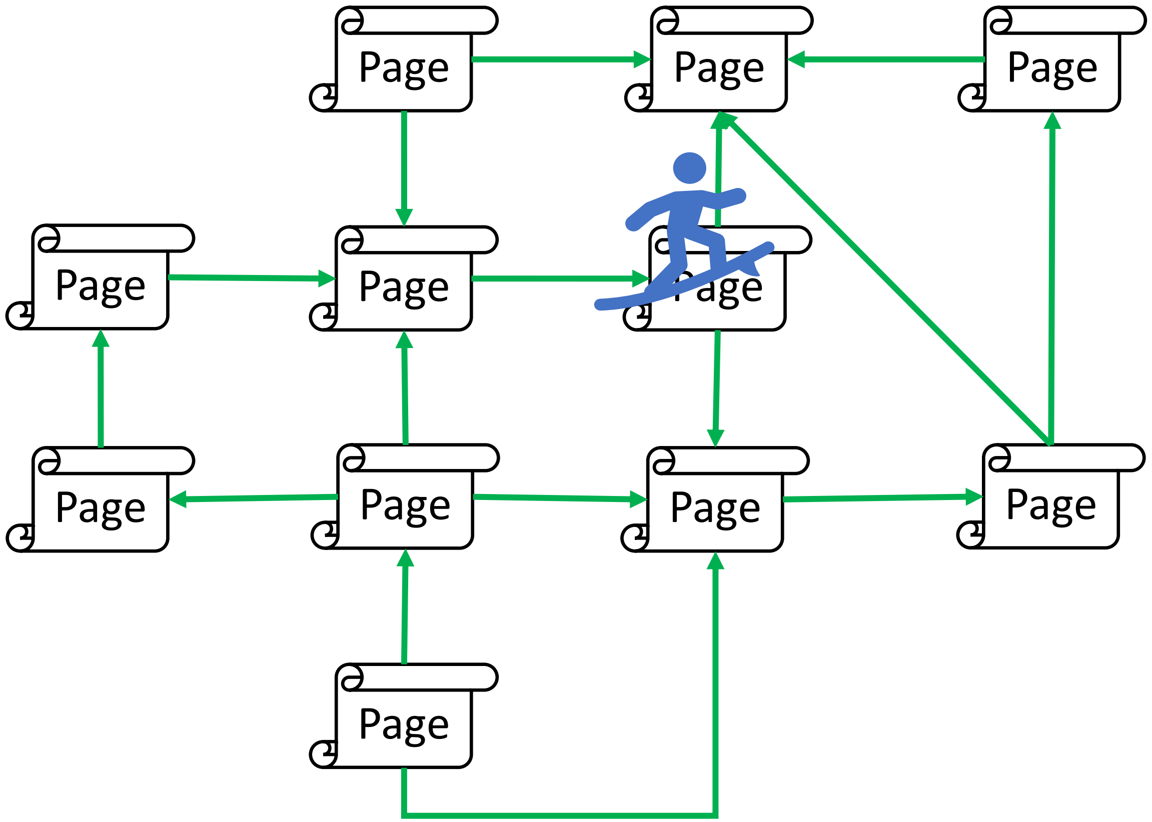
- importance score of page i
- $\sum_i \boldsymbol{r_i} = \mathbf{1}$

„Flow“ Equations can be written

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

$\mathbf{M}$ is a **column stochastic matrix:**
each column sums to 1

Random Walk interpretation



- At any time t, surfer is on page i.
- At time t+1, surfer follows out-link randomly (uniform prob)
- Surfer ends on page j, linked from i
- Process repeats

i-th coordinate of vector $p(t)$ represents probability, that surfer is on page i at time t.

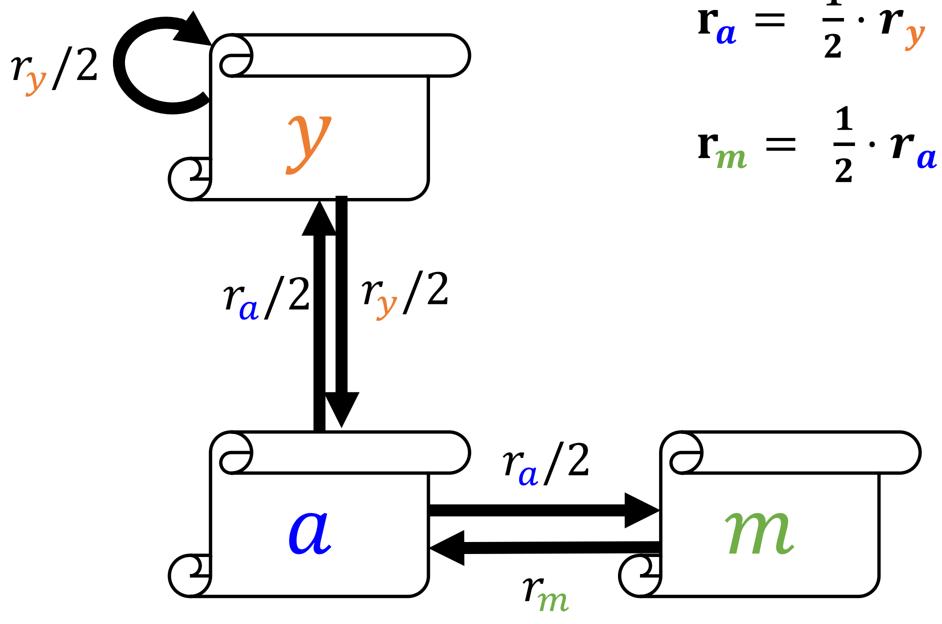
$\Rightarrow$ $p(t)$ is a probability distribution over pages

If we have

$$p(t+1) = Mp(t) = p(t)$$

=> then $p(t)$ is a stationary distribution

Flow Equations and Matrix M



$$\mathbf{r}_y = \frac{1}{2}\cdot r_y + \frac{1}{2}\cdot r_a$$

$$\mathbf{r}_a = \frac{1}{2}\cdot r_y + \mathbf{r}_m$$

$$\mathbf{r}_m = \frac{1}{2}\cdot r_a$$

$$\begin{bmatrix}\mathbf{r}_y\\ \mathbf{r}_a\\ \mathbf{r}_m\end{bmatrix} = \begin{bmatrix}1/2 & 1/2 & 0\\ 1/2 & 0 & 1\\ 0 & 1/2 & 0\end{bmatrix}\cdot\begin{bmatrix}\mathbf{r}_y\\ \mathbf{r}_a\\ \mathbf{r}_m\end{bmatrix}$$

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

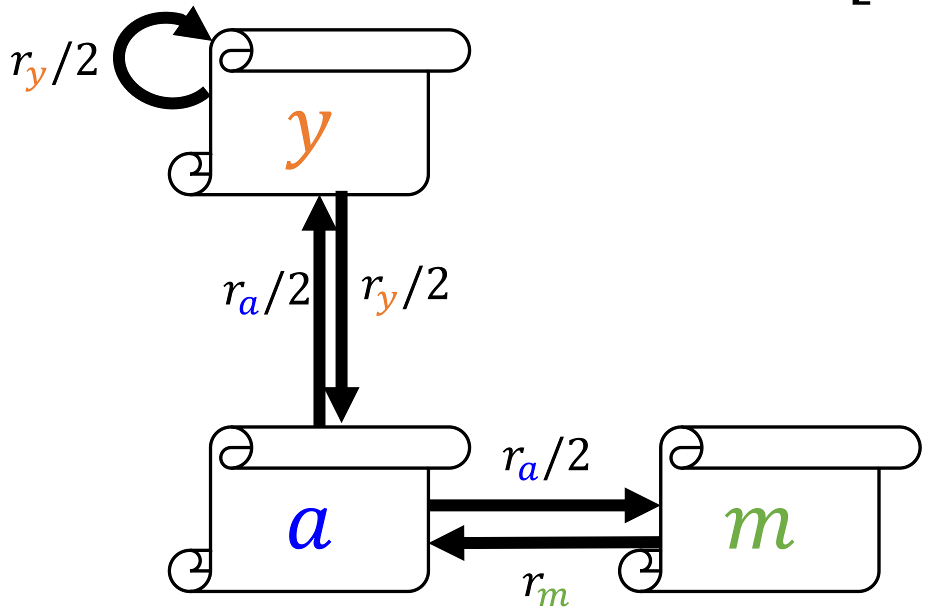
Solution

$$r_y = \frac{2}{5},\ r_a = \frac{2}{5},\ r_m = \frac{1}{5}$$

Eigenvector Formulation

$$\boxed{\mathbf{r} = \mathbf{M} \cdot \mathbf{r}}$$

$$\begin{bmatrix} \mathbf{r}_y \\ \mathbf{r}_a \\ \mathbf{r}_m \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{r}_y \\ \mathbf{r}_a \\ \mathbf{r}_m \end{bmatrix}$$

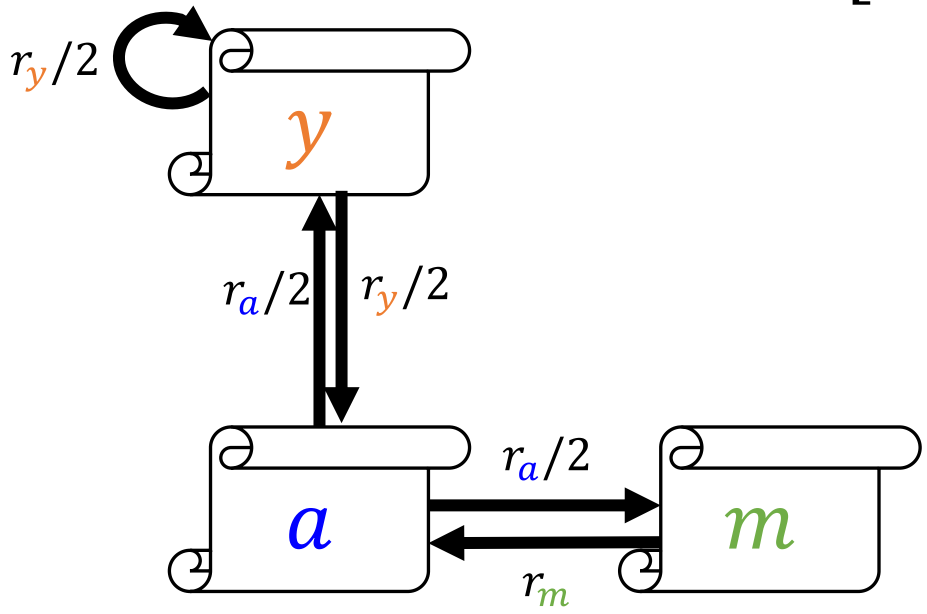


Eigenvector Formulation

$$\boxed{\mathbf{r} = \mathbf{M}\cdot\mathbf{r}}$$

$$\begin{bmatrix}\mathbf{r}_y\\ \mathbf{r}_a\\ \mathbf{r}_m\end{bmatrix} = \begin{bmatrix}\mathbf{1/2} & \mathbf{1/2} & \mathbf{0}\\ \mathbf{1/2} & \mathbf{0} & \mathbf{1}\\ \mathbf{0} & \mathbf{1/2} & \mathbf{0}\end{bmatrix}\cdot\begin{bmatrix}\mathbf{r}_y\\ \mathbf{r}_a\\ \mathbf{r}_m\end{bmatrix}$$

rank vector r is an **eigenvector** of the stochastic web matrix $\mathbf{M}$

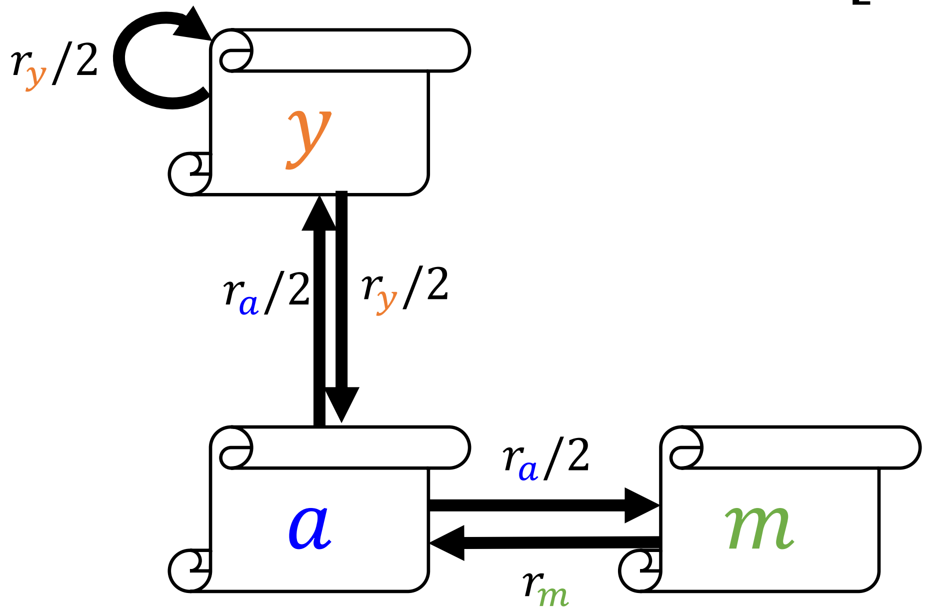


Eigenvector Formulation

$$\begin{bmatrix} \mathbf{r}_y \\ \mathbf{r}_a \\ \mathbf{r}_m \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{r}_y \\ \mathbf{r}_a \\ \mathbf{r}_m \end{bmatrix}$$

$$\boxed{\mathbf{r} = \mathbf{M} \cdot \mathbf{r}}$$

rank vector $\mathbf{r}$ is an **eigenvector** of the stochastic web matrix $\mathbf{M}$



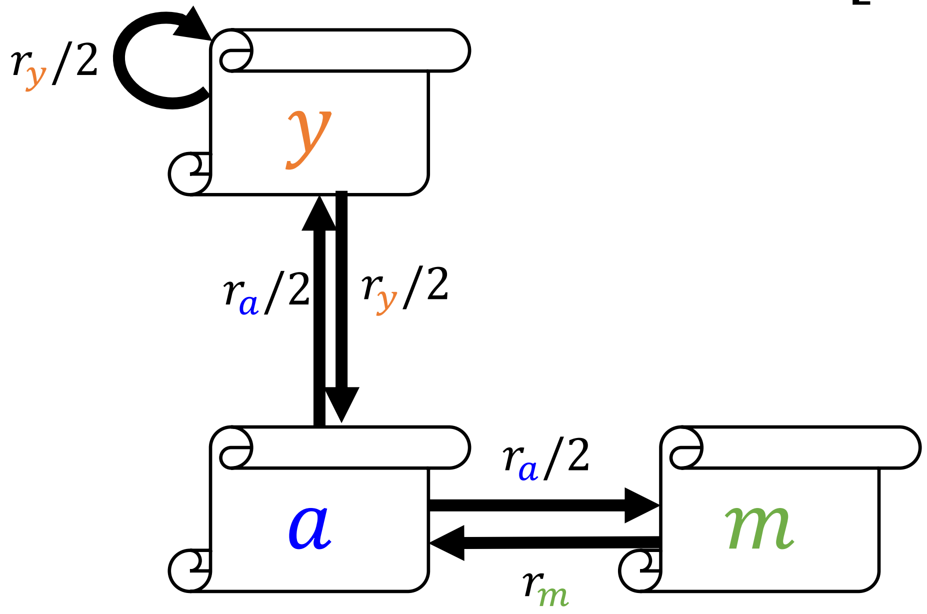
NOTE:
$\boldsymbol{x}$ is an **eigenvector** with the corresponding **eigenvalue** λ if:
$\boldsymbol{Ax} = \boldsymbol{\lambda x}$

Eigenvector Formulation

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

$$\begin{bmatrix} \mathbf{r}_y \\ \mathbf{r}_a \\ \mathbf{r}_m \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{r}_y \\ \mathbf{r}_a \\ \mathbf{r}_m \end{bmatrix}$$

rank vector r is an **eigenvector** of the stochastic web matrix $\mathbf{M}$



Starting from any stochastic vector $\boldsymbol{u}$, the limit

$$\mathbf{M}(\mathbf{M}(\ldots\ \mathbf{M}(\mathbf{M}\ \boldsymbol{u})))$$

is the long-term distribution of the surfers.

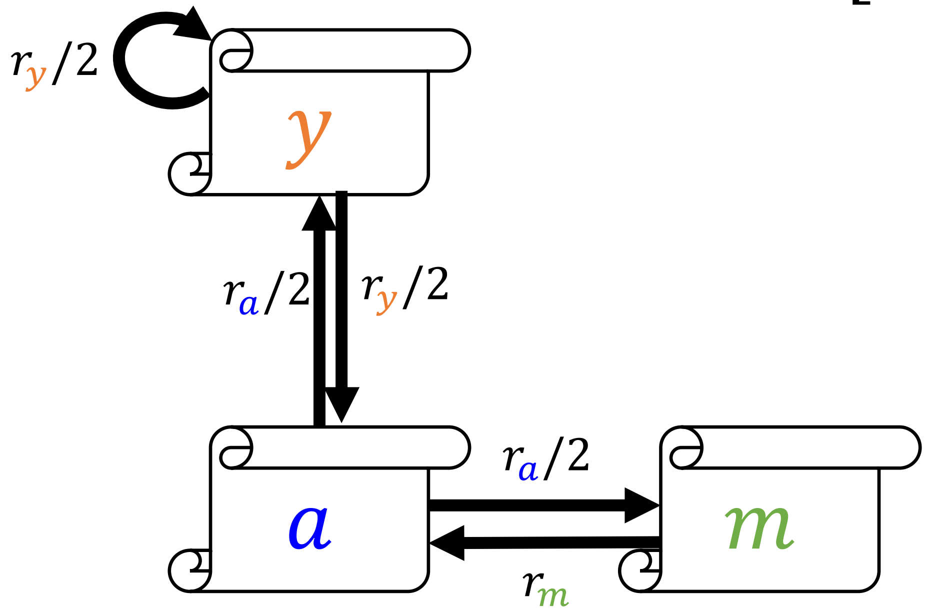
NOTE:
$\boldsymbol{x}$ is an **eigenvector** with the corresponding **eigenvalue** λ if:
$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{\lambda}\boldsymbol{x}$

Eigenvector Formulation

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

$$\begin{bmatrix} \mathbf{r}_y \\ \mathbf{r}_a \\ \mathbf{r}_m \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{r}_y \\ \mathbf{r}_a \\ \mathbf{r}_m \end{bmatrix}$$

rank vector r is an **eigenvector** of the stochastic web matrix $\mathbf{M}$



Starting from any stochastic vector $\boldsymbol{u}$, the limit

$$\mathbf{M}(\mathbf{M}(\ldots\ \mathbf{M}(\mathbf{M}\ \boldsymbol{u})))$$

is the long-term distribution of the surfers.

i.e.: **limiting distribution = principal eigenvector** of $\mathbf{M}$

= **PageRank**.

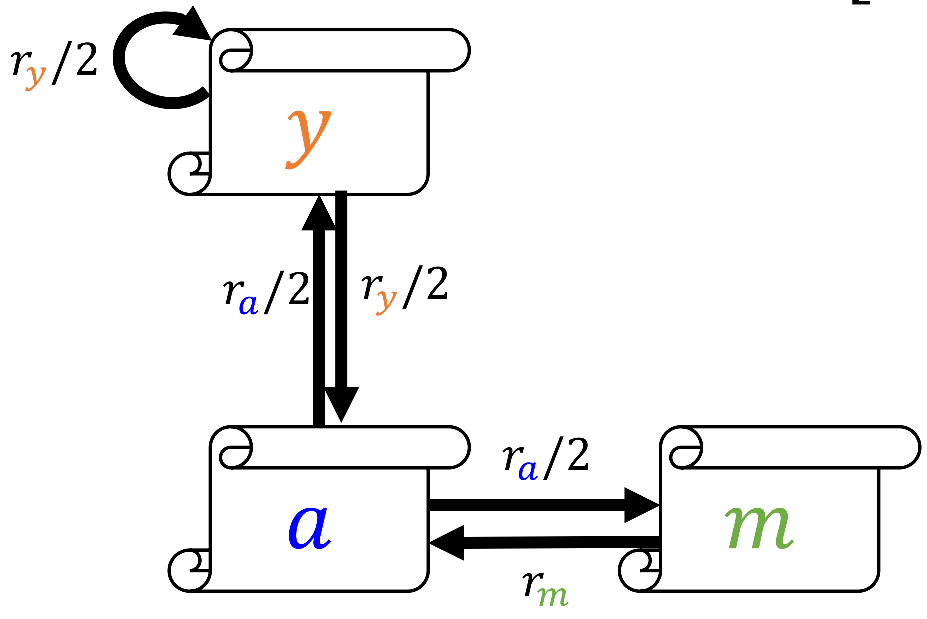
NOTE:
$\boldsymbol{x}$ is an **eigenvector** with the corresponding **eigenvalue** λ if:
$\boldsymbol{Ax} = \boldsymbol{\lambda x}$

Eigenvector Formulation

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

$$\begin{bmatrix} \mathbf{r}_y \\ \mathbf{r}_a \\ \mathbf{r}_m \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{r}_y \\ \mathbf{r}_a \\ \mathbf{r}_m \end{bmatrix}$$

rank vector r is an **eigenvector** of the stochastic web matrix $\mathbf{M}$



Starting from any stochastic vector $\boldsymbol{u}$, the limit

$$\mathbf{M}(\mathbf{M}(\ldots\, \mathbf{M}(\mathbf{M}\, \boldsymbol{u})))$$

is the long-term distribution of the surfers.

i.e.: **limiting distribution** = **principal eigenvector** of $\mathbf{M}$
= **PageRank**.

If $\boldsymbol{r}$ is the limit of $\mathbf{M}(\mathbf{M}(\ldots\, \mathbf{M}(\mathbf{M}\, \boldsymbol{u})))$, then $\boldsymbol{r}$ satisfies the equation $\boldsymbol{r} = \boldsymbol{M}\boldsymbol{r}$, so $\boldsymbol{r}$ is an eigenvector of $\boldsymbol{M}$ with eigenvalue 1

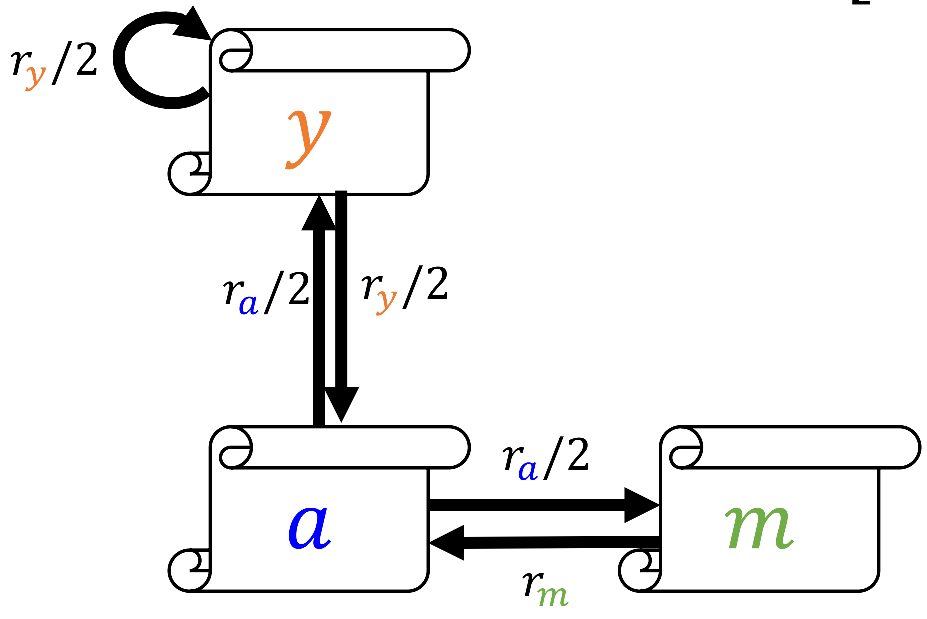
NOTE:
$\boldsymbol{x}$ is an **eigenvector** with the corresponding **eigenvalue** λ if:
$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{\lambda}\boldsymbol{x}$

Eigenvector Formulation

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

$$\begin{bmatrix} \mathbf{r}_y \\ \mathbf{r}_a \\ \mathbf{r}_m \end{bmatrix} = \begin{bmatrix} \mathbf{1/2} & \mathbf{1/2} & \mathbf{0} \\ \mathbf{1/2} & \mathbf{0} & \mathbf{1} \\ \mathbf{0} & \mathbf{1/2} & \mathbf{0} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{r}_y \\ \mathbf{r}_a \\ \mathbf{r}_m \end{bmatrix}$$

rank vector r is an **eigenvector** of the stochastic web matrix $\mathbf{M}$



Starting from any stochastic vector $\boldsymbol{u}$, the limit

$$\mathbf{M}(\mathbf{M}(\ldots\ \mathbf{M}(\mathbf{M}\ \boldsymbol{u})))$$

is the long-term distribution of the surfers.

i.e.: **limiting distribution** = **principal eigenvector** of $\mathbf{M}$
= **PageRank**.

If $\boldsymbol{r}$ is the limit of $\mathbf{M}(\mathbf{M}(\ldots\ \mathbf{M}(\mathbf{M}\ \boldsymbol{u})))$, then $\boldsymbol{r}$ satisfies the equation $\boldsymbol{r} = \boldsymbol{M}\boldsymbol{r}$, so $\boldsymbol{r}$ is an eigenvector of $\boldsymbol{M}$ with eigenvalue 1

We can now efficiently solve for $\boldsymbol{r}$.
The method is called ***"Power iteration"***

NOTE:
$\boldsymbol{x}$ is an **eigenvector** with the corresponding **eigenvalue** λ if:
$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{\lambda}\boldsymbol{x}$

Power Iteration Method

Given a web graph with N nodes, where the nodes are pages and edges are hyperlinks

Algorithm

- Initialize: $\mathbf{r}^{(0)} = \left[\frac{1}{N}, \dots, \frac{1}{N}\right]^T$
- Iterate: $\mathbf{r}^{(\mathrm{t}+1)} = \mathbf{M} \cdot \mathrm{r}^{(\mathrm{t})}$ (t = timestep)
- Stop when: $\left|\mathbf{r}^{(\mathrm{t}+1)} - \mathbf{r}^{(\mathrm{t})}\right|_1 < \varepsilon$

Power Iteration Method

Given a web graph with N nodes, where the nodes are pages and edges are hyperlinks

<u>Algorithm</u>

- Initialize: $\mathbf{r}^{(0)} = \left[\frac{1}{N}, \ldots, \frac{1}{N}\right]^T$
- Iterate: $\mathbf{r}^{(\mathrm{t+1})} = \mathbf{M} \cdot \mathrm{r}^{(\mathrm{t})}$ (t = timestep)
- Stop when: $\left|\mathbf{r}^{(\mathrm{t+1})} - \mathbf{r}^{(\mathrm{t})}\right|_1 < \varepsilon$

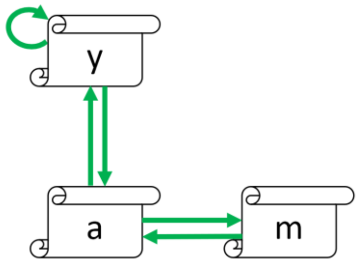
$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{d_i}$$

$|\mathbf{x}|_1 = \sum_{1 \le i \le N} |x_i|$ is the L_1-Norm

Power Iteration: Example

Algorithm

- Initialize: $\mathbf{r}^{(0)} = \left[\frac{1}{N}, \ldots, \frac{1}{N}\right]^T$
- Iterate: $\mathbf{r}^{(t+1)} = \mathbf{M} \cdot \mathrm{r}^{(t)}$



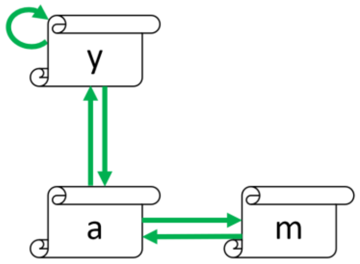
$$\mathrm{r} = \begin{bmatrix} \mathrm{r}_y \\ \mathrm{r}_a \\ \mathrm{r}_m \end{bmatrix} = \begin{matrix} 1/3 \\ 1/3 \\ 1/3 \end{matrix}$$

Iteration	0

Power Iteration: Example

Algorithm

- Initialize: $\mathbf{r}^{(0)} = \left[\frac{1}{N}, \dots, \frac{1}{N}\right]^T$
- Iterate: $\mathbf{r}^{(t+1)} = \mathbf{M} \cdot \mathrm{r}^{(t)}$



$$\mathrm{r} = \begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{matrix} 1/3 \\ 1/3 \\ 1/3 \end{matrix} \quad \begin{matrix} 1/3 \\ 3/6 \\ 1/6 \end{matrix}$$

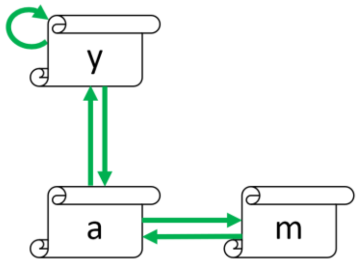
Iteration	1

$$\boldsymbol{r}^{(1)} = \boldsymbol{M} \cdot \boldsymbol{r}^{(0)}$$

Power Iteration: Example

Algorithm

- Initialize: $\mathbf{r}^{(0)} = \left[\frac{1}{N}, \ldots, \frac{1}{N}\right]^T$
- Iterate: $\mathbf{r}^{(t+1)} = \mathbf{M} \cdot \mathrm{r}^{(t)}$



$$r = \begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{matrix} 1/3 & 1/3 & 5/12 \\ 1/3 & 3/6 & 1/3 \\ 1/3 & 1/6 & 3/12 \end{matrix}$$

Iteration **2**

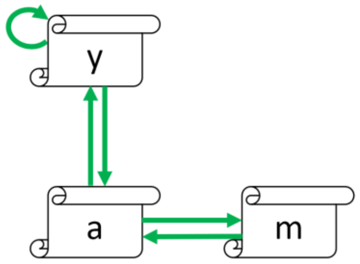
$$r^{(1)} = M \cdot r^{(0)}$$

$$\boldsymbol{r}^{(2)} = \boldsymbol{M} \cdot \boldsymbol{r}^{(1)} = \boldsymbol{M}\left(\boldsymbol{M}\boldsymbol{r}^{(0)}\right) = \boldsymbol{M}^2 \cdot \boldsymbol{r}^{(0)}$$

Power Iteration: Example

Algorithm

- Initialize: $\mathbf{r}^{(0)} = \left[\frac{1}{N}, \ldots, \frac{1}{N}\right]^T$
- Iterate: $\mathbf{r}^{(t+1)} = \mathbf{M} \cdot \mathrm{r}^{(t)}$



$$\mathrm{r} = \begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{matrix} 1/3 & 1/3 & 5/12 & 9/24 \\ 1/3 & 3/6 & 1/3 & 11/24 \\ 1/3 & 1/6 & 3/12 & 1/6 \end{matrix}$$

Iteration 3

$$r^{(1)} = M \cdot r^{(0)}$$

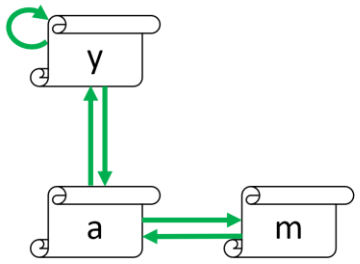
$$r^{(2)} = M \cdot r^{(1)} = M\left(Mr^{(0)}\right) = M^2 \cdot r^{(0)}$$

$$\boldsymbol{r}^{(3)} = \boldsymbol{M} \cdot \boldsymbol{r}^{(2)} = \boldsymbol{M}\left(\boldsymbol{M}^2 \boldsymbol{r}^{(0)}\right) = \boldsymbol{M}^3 \cdot \boldsymbol{r}^{(0)}$$

Power Iteration: Example

Algorithm

- Initialize: $\mathbf{r}^{(0)} = \left[\frac{1}{N}, \dots, \frac{1}{N}\right]^T$
- Iterate: $\mathbf{r}^{(t+1)} = \mathbf{M} \cdot \mathrm{r}^{(t)}$



$$
\mathrm{r} = \begin{pmatrix} \mathrm{r_y} \\ \mathrm{r_a} \\ \mathrm{r_m} \end{pmatrix} = \begin{matrix} 1/3 & 1/3 & 5/12 & 9/24 & & 6/15 \\ 1/3 & 3/6 & 1/3 & 11/24 & \dots & 6/15 \\ 1/3 & 1/6 & 3/12 & 1/6 & & 3/15 \end{matrix}
$$

Iteration $\quad\quad\quad\quad\quad\quad\quad\quad$ $\boldsymbol{k}$

$$\boldsymbol{M}^k \cdot \boldsymbol{r}^{(0)}$$

Claim: The Sequence $\left(\mathbf{M}^k \boldsymbol{r}^{(0)}\right)_{k \in \mathbb{N}_0}$ approaches the dominant eigenvector of $\boldsymbol{M}$.

Why does Power Iteration work?

Claim: The Sequence $\left(\mathbf{M}^{\boldsymbol{k}}\boldsymbol{r}^{(\mathbf{0})}\right)_{\boldsymbol{k}\in\mathbb{N}_{\mathbf{0}}}$ approaches the dominant eigenvector of $\boldsymbol{M}$.

Why does Power Iteration work?

Claim: The Sequence $\left(\mathbf{M}^{k}\boldsymbol{r}^{(\mathbf{0})}\right)_{\boldsymbol{k}\in\mathbb{N}_{\mathbf{0}}}$ approaches the dominant eigenvector of $\boldsymbol{M}$.

Proof:

Assume M has n linearly independent **eigenvectors**, $x_1, x_2, \ldots, x_n$ with corresponding eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$, sorted in descending order: $\lambda_1 > \lambda_2 > \ldots > \lambda_n$.

NOTE:
$\boldsymbol{x}$ is an **eigenvector** with the corresponding **eigenvalue** λ if:
$\boldsymbol{Ax} = \boldsymbol{\lambda x}$

Why does Power Iteration work?

Claim: The Sequence $\left(\mathbf{M}^{\boldsymbol{k}}\boldsymbol{r}^{(\mathbf{0})}\right)_{\boldsymbol{k}\in\mathbb{N}_{\mathbf{0}}}$ approaches the dominant eigenvector of $\boldsymbol{M}$.

Proof:

Assume M has n linearly independent **eigenvectors**, $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n$ with corresponding eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_n$, sorted in descending order: $\lambda_1 > \lambda_2 > \dots > \lambda_n$.

Vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n$ form a basis, hence we can write: $\boldsymbol{r}^{(0)} = c_1\boldsymbol{x}_1 + c_2\boldsymbol{x}_2 + \dots + c_n\boldsymbol{x}_n$

Why does Power Iteration work?

Claim: The Sequence $\left(\mathbf{M}^{k}\boldsymbol{r}^{(0)}\right)_{k\in\mathbb{N}_0}$ approaches the dominant eigenvector of $\boldsymbol{M}$.

Proof:

Assume M has n linearly independent **eigenvectors**, $x_1, x_2, \dots, x_n$ with corresponding eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_n$, sorted in descending order: $\lambda_1 > \lambda_2 > \dots > \lambda_n$.

Vectors $x_1, x_2, \dots, x_n$ form a basis, hence we can write: $\boldsymbol{r}^{(0)} = c_1 x_1 + c_2 x_2 + \dots + c_n x_n$

Then: $\mathbf{M}\boldsymbol{r}^{(0)} = \mathbf{M}(c_1 x_1 + c_2 x_2 + \dots + c_n x_n)$

Why does Power Iteration work?

Claim: The Sequence $\left(\mathbf{M}^{k}\boldsymbol{r}^{(\mathbf{0})}\right)_{k\in\mathbb{N}_0}$ approaches the dominant eigenvector of $\boldsymbol{M}$.

Proof:

Assume M has n linearly independent **eigenvectors**, $x_1, x_2, \dots, x_n$ with corresponding eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_n$, sorted in descending order: $\lambda_1 > \lambda_2 > \dots > \lambda_n$.

Vectors $x_1, x_2, \dots, x_n$ form a basis, hence we can write: $\boldsymbol{r}^{(0)} = c_1 x_1 + c_2 x_2 + \dots + c_n x_n$

Then: $\mathbf{M}\boldsymbol{r}^{(0)} = \mathbf{M}(c_1 x_1 + c_2 x_2 + \dots + c_n x_n)$

$= c_1(\mathbf{M}x_1) + c_2(\mathbf{M}x_2) + \dots + c_n(\mathbf{M}x_n)$

Why does Power Iteration work?

Claim: The Sequence $\left(\mathbf{M}^k \boldsymbol{r}^{(0)}\right)_{k \in \mathbb{N}_0}$ approaches the dominant eigenvector of $\boldsymbol{M}$.

Proof:

Assume M has n linearly independent **eigenvectors**, $x_1, x_2, \dots, x_n$ with corresponding eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_n$, sorted in descending order: $\lambda_1 > \lambda_2 > \dots > \lambda_n$.

Vectors $x_1, x_2, \dots, x_n$ form a basis, hence we can write: $\boldsymbol{r}^{(0)} = c_1 x_1 + c_2 x_2 + \dots + c_n x_n$

Then: $\mathbf{M}\boldsymbol{r}^{(0)} = \mathbf{M}(c_1 x_1 + c_2 x_2 + \dots + c_n x_n)$

$= c_1(\mathbf{M}x_1) + c_2(\mathbf{M}x_2) + \dots + c_n(\mathbf{M}x_n)$

$= c_1(\lambda_1 x_1) + c_2(\lambda_2 x_2) + \dots + c_n(\lambda_n x_n)$

Why does Power Iteration work?

Claim: The Sequence $\left(\mathbf{M}^{\boldsymbol{k}}\boldsymbol{r}^{(\mathbf{0})}\right)_{\boldsymbol{k}\in\mathbb{N}_{\mathbf{0}}}$ approaches the dominant eigenvector of $\boldsymbol{M}$.

Proof:

Assume M has n linearly independent **eigenvectors**, $\boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_n}$ with corresponding eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_n$, sorted in descending order: $\lambda_1 > \lambda_2 > \dots > \lambda_n$.

Vectors $\boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_n}$ form a basis, hence we can write: $\boldsymbol{r}^{(\mathbf{0})} = c_1\boldsymbol{x_1} + c_2\boldsymbol{x_2} + \dots + c_n\boldsymbol{x_n}$

Then:
$$\begin{aligned}\mathbf{M}\boldsymbol{r}^{(\mathbf{0})} &= \mathbf{M}(c_1\boldsymbol{x_1} + c_2\boldsymbol{x_2} + \dots + c_n\boldsymbol{x_n}) \\ &= c_1(\mathbf{M}\boldsymbol{x_1}) + c_2(\mathbf{M}\boldsymbol{x_2}) + \dots + c_n(\mathbf{M}\boldsymbol{x_n}) \\ &= c_1(\lambda_1\boldsymbol{x_1}) + c_2(\lambda_2\boldsymbol{x_2}) + \dots + c_n(\lambda_n\boldsymbol{x_n})\end{aligned}$$

Repeatedly, i.e. $\boldsymbol{k}-\mathbf{1}$ times, multiplying $\mathbf{M}$ on both sides yields:

$$\mathbf{M}^{\boldsymbol{k}}\boldsymbol{r}^{(\mathbf{0})} = c_1\left(\lambda_1^k\boldsymbol{x_1}\right) + c_2\left(\lambda_2^k\boldsymbol{x_2}\right) + \dots + c_n\left(\lambda_n^k\boldsymbol{x_n}\right)$$

NOTE:
$\boldsymbol{x}$ is an **eigenvector** with the corresponding **eigenvalue** λ if:
$\boldsymbol{Ax} = \boldsymbol{\lambda x}$

Why does Power Iteration work?

Claim: The Sequence $\left(\mathbf{M}^{\boldsymbol{k}}\boldsymbol{r}^{(0)}\right)_{\boldsymbol{k}\in\mathbb{N}_0}$ approaches the dominant eigenvector of $\boldsymbol{M}$.

Proof:

Assume M has n linearly independent **eigenvectors**, $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ with corresponding eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$, sorted in descending order: $\lambda_1 > \lambda_2 > \ \ldots > \lambda_n$.

Vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ form a basis, hence we can write: $\boldsymbol{r}^{(0)} = c_1\boldsymbol{x}_1 + c_2\boldsymbol{x}_2 + \ \ldots + c_n\boldsymbol{x}_n$

Then:
$$\begin{aligned}\mathbf{M}\boldsymbol{r}^{(0)} &= \mathbf{M}(c_1\boldsymbol{x}_1 + c_2\boldsymbol{x}_2 + \ \ldots + c_n\boldsymbol{x}_n) \\ &= c_1(\mathbf{M}\boldsymbol{x}_1) + c_2(\mathbf{M}\boldsymbol{x}_2) + \ \ldots + c_n(\mathbf{M}\boldsymbol{x}_n) \\ &= c_1(\lambda_1\boldsymbol{x}_1) + c_2(\lambda_2\boldsymbol{x}_2) + \ \ldots + c_n(\lambda_n\boldsymbol{x}_n)\end{aligned}$$

Repeatedly, i.e. $\boldsymbol{k}-\mathbf{1}$ times, multiplying $\mathbf{M}$ on both sides yields:

$$\mathbf{M}^{\boldsymbol{k}}\boldsymbol{r}^{(0)} = c_1\left(\lambda_1^k\boldsymbol{x}_1\right) + c_2\left(\lambda_2^k\boldsymbol{x}_2\right) + \ \ldots + c_n\left(\lambda_n^k\boldsymbol{x}_n\right)$$

$$\mathbf{M}^{\boldsymbol{k}}\boldsymbol{r}^{(0)} = \lambda_1^k\left[c_1\boldsymbol{x}_1 + c_2\left(\frac{\lambda_2}{\lambda_1}\right)^k\boldsymbol{x}_2 + \ \ldots + c_n\left(\frac{\lambda_n}{\lambda_1}\right)^k\boldsymbol{x}_n\right]$$

Since $\forall i > 1$: $\lambda_1 > \lambda_i \Rightarrow \frac{\lambda_i}{\lambda_1} < 1$

Why does Power Iteration work?

Claim: The Sequence $\left(\mathbf{M}^{k}\boldsymbol{r}^{(0)}\right)_{k\in\mathbb{N}_0}$ approaches the dominant eigenvector of $\boldsymbol{M}$.

Proof:

Assume M has n linearly independent **eigenvectors**, $x_1, x_2, \ldots, x_n$ with corresponding eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$, sorted in descending order: $\lambda_1 > \lambda_2 > \ldots > \lambda_n$.

Vectors $x_1, x_2, \ldots, x_n$ form a basis, hence we can write: $\boldsymbol{r}^{(0)} = c_1 x_1 + c_2 x_2 + \ldots + c_n x_n$

Then:

$$\begin{aligned}\mathbf{M}\boldsymbol{r}^{(0)} &= \mathbf{M}(c_1 x_1 + c_2 x_2 + \ldots + c_n x_n)\\ &= c_1(\mathbf{M}x_1) + c_2(\mathbf{M}x_2) + \ldots + c_n(\mathbf{M}x_n)\\ &= c_1(\lambda_1 x_1) + c_2(\lambda_2 x_2) + \ldots + c_n(\lambda_n x_n)\end{aligned}$$

Repeatedly, i.e. $\boldsymbol{k}-\mathbf{1}$ times, multiplying $\mathbf{M}$ on both sides yields:

$$\mathbf{M}^{k}\boldsymbol{r}^{(0)} = c_1\left(\lambda_1^k x_1\right) + c_2\left(\lambda_2^k x_2\right) + \ldots + c_n\left(\lambda_n^k x_n\right)$$

$$\mathbf{M}^{k}\boldsymbol{r}^{(0)} = \lambda_1^k\left[c_1 x_1 + c_2 \underbrace{\left(\frac{\lambda_2}{\lambda_1}\right)^k}_{\to 0} x_2 + \ldots + c_n \underbrace{\left(\frac{\lambda_n}{\lambda_1}\right)^k}_{\to 0} x_n\right]$$

Since $\forall i > 1$: $\lambda_1 > \lambda_i \Rightarrow \frac{\lambda_i}{\lambda_1} < 1$

As $k \to \infty$

Why does Power Iteration work?

Claim: The Sequence $\left(\mathbf{M}^{\boldsymbol{k}}\boldsymbol{r}^{(\mathbf{0})}\right)_{\boldsymbol{k}\in\mathbb{N}_{\mathbf{0}}}$ approaches the dominant eigenvector of $\boldsymbol{M}$.

Proof:

Assume M has n linearly independent **eigenvectors**, $x_1, x_2, \ldots, x_n$ with corresponding eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$, sorted in descending order: $\lambda_1 > \lambda_2 > \ldots > \lambda_n$.

Vectors $x_1, x_2, \ldots, x_n$ form a basis, hence we can write: $\boldsymbol{r}^{(0)} = c_1 x_1 + c_2 x_2 + \ldots + c_n x_n$

Then:
$$\begin{aligned}\mathbf{M}\boldsymbol{r}^{(0)} &= \mathbf{M}(c_1 x_1 + c_2 x_2 + \ldots + c_n x_n)\\ &= c_1(\mathbf{M}x_1) + c_2(\mathbf{M}x_2) + \ldots + c_n(\mathbf{M}x_n)\\ &= c_1(\lambda_1 x_1) + c_2(\lambda_2 x_2) + \ldots + c_n(\lambda_n x_n)\end{aligned}$$

Repeatedly, i.e. $\boldsymbol{k}-\mathbf{1}$ times, multiplying $\mathbf{M}$ on both sides yields:

$$\mathbf{M}^{\boldsymbol{k}}\boldsymbol{r}^{(0)} = c_1\left(\lambda_1^k x_1\right) + c_2\left(\lambda_2^k x_2\right) + \ldots + c_n\left(\lambda_n^k x_n\right)$$

$$\mathbf{M}^{\boldsymbol{k}}\boldsymbol{r}^{(0)} = \lambda_1^k\left[c_1 x_1 + c_2 \underbrace{\left(\frac{\lambda_2}{\lambda_1}\right)^k}_{\to 0} x_2 + \ldots + c_n \underbrace{\left(\frac{\lambda_n}{\lambda_1}\right)^k}_{\to 0} x_n\right]$$

Since $\forall i > 1\colon \lambda_1 > \lambda_i \Rightarrow \frac{\lambda_i}{\lambda_1} < 1$

As $k \to \infty$

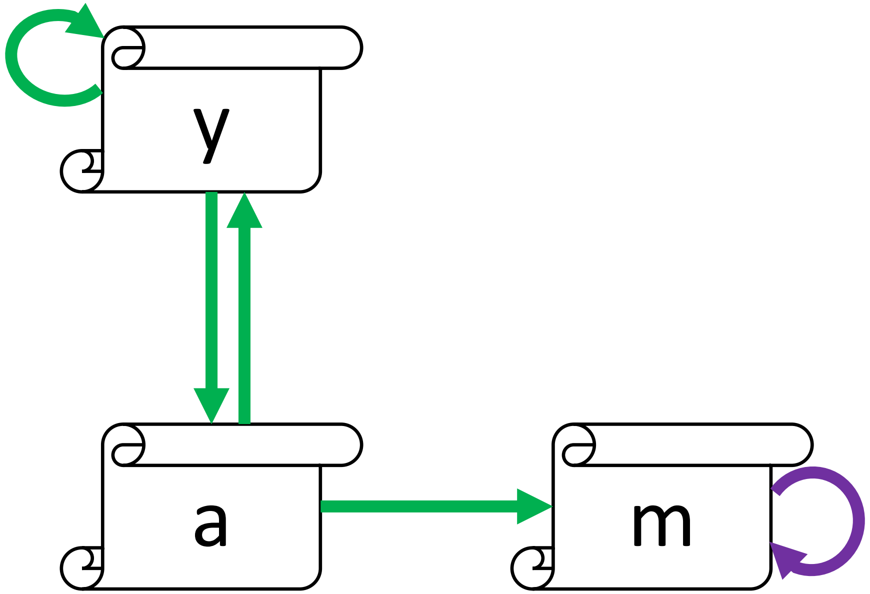
$$\Rightarrow \mathbf{M}^{\boldsymbol{k}}\boldsymbol{r}^{(\mathbf{0})} \approx c_1 \lambda_1^k \boldsymbol{x}_{\mathbf{1}}$$

If c_1 = 0, then the method won't converge.

Problem: Spider Traps

Power Iteration

- Set $r_j = \frac{1}{N}\ \forall j$
- $r_j = \sum_{i \to j} \frac{r_i}{d_i}$
  - iterate



	y	a	m
y	½	½	0
a	½	0	0
m	0	½	1

$$\mathbf{r_y} = \frac{r_y}{2} + \frac{r_a}{2}$$

$$\mathbf{r}_a = \frac{r_y}{2}$$

$$\mathbf{r}_m = \frac{r_a}{2}$$

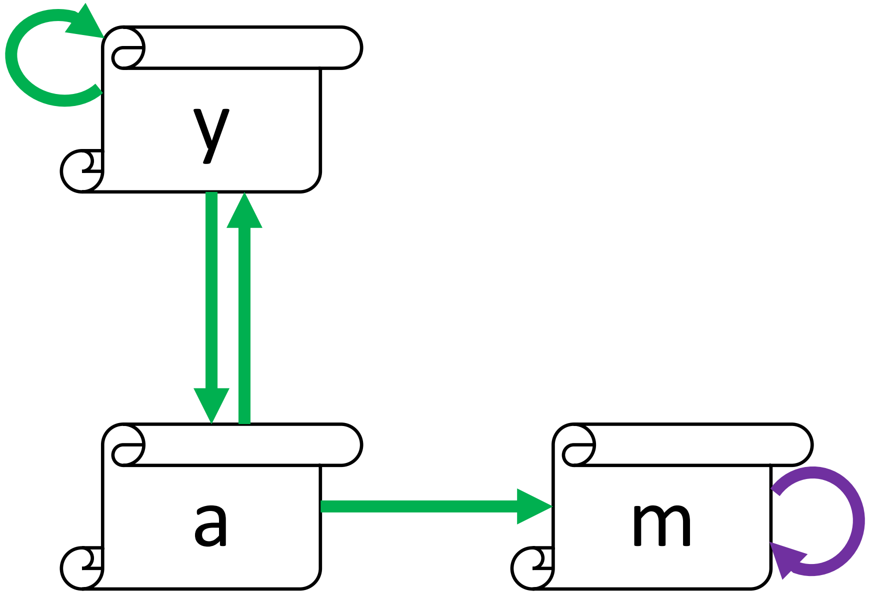
$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{matrix} 1/3 \\ 1/3 \\ 1/3 \end{matrix}$$

Iteration T	0

Problem: Spider Traps

Power Iteration

- Set $r_j = \frac{1}{N}\ \forall j$
- $r_j = \sum_{i \to j} \frac{r_i}{d_i}$
  - iterate



$$\begin{array}{c|ccc|} & y & a & m \\ \hline y & \frac{1}{2} & \frac{1}{2} & 0 \\ a & \frac{1}{2} & 0 & 0 \\ m & 0 & \frac{1}{2} & \boxed{1} \\ \hline \end{array}$$

$$r_y = \frac{r_y}{2} + \frac{r_a}{2}$$

$$r_a = \frac{r_y}{2}$$

$$r_m = \frac{r_a}{2}$$

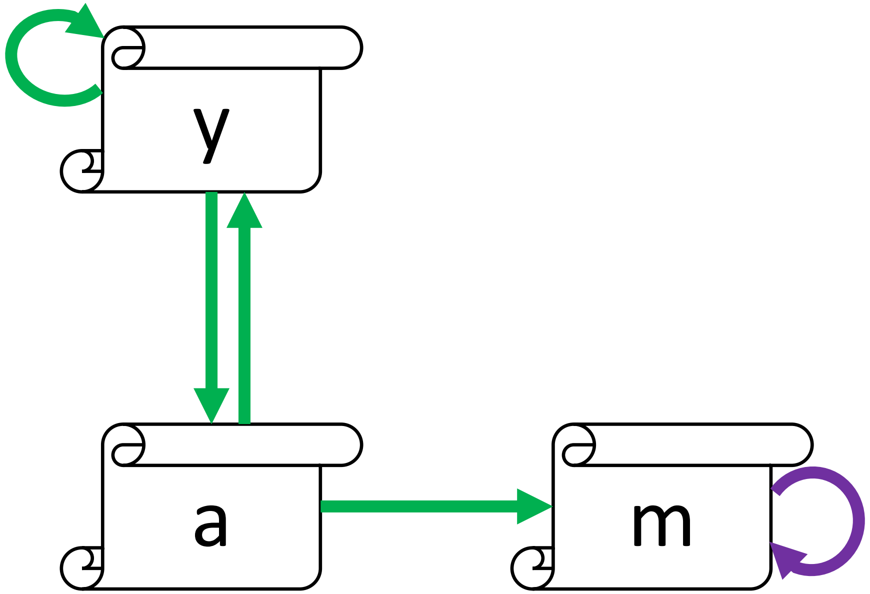
$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{matrix} 1/3 & 2/6 \\ 1/3 & 1/6 \\ 1/3 & 3/6 \end{matrix}$$

Iteration T	0	1

Problem: Spider Traps

Power Iteration

- Set $r_j = \frac{1}{N}\ \forall j$
- $r_j = \sum_{i \to j} \frac{r_i}{d_i}$
  - iterate



$$\begin{array}{c|ccc} & y & a & m \\ \hline y & ½ & ½ & 0 \\ a & ½ & 0 & 0 \\ m & 0 & ½ & 1 \end{array}$$

$$\mathbf{r_y} = \frac{r_y}{2} + \frac{r_a}{2}$$

$$\mathbf{r}_a = \frac{r_y}{2}$$

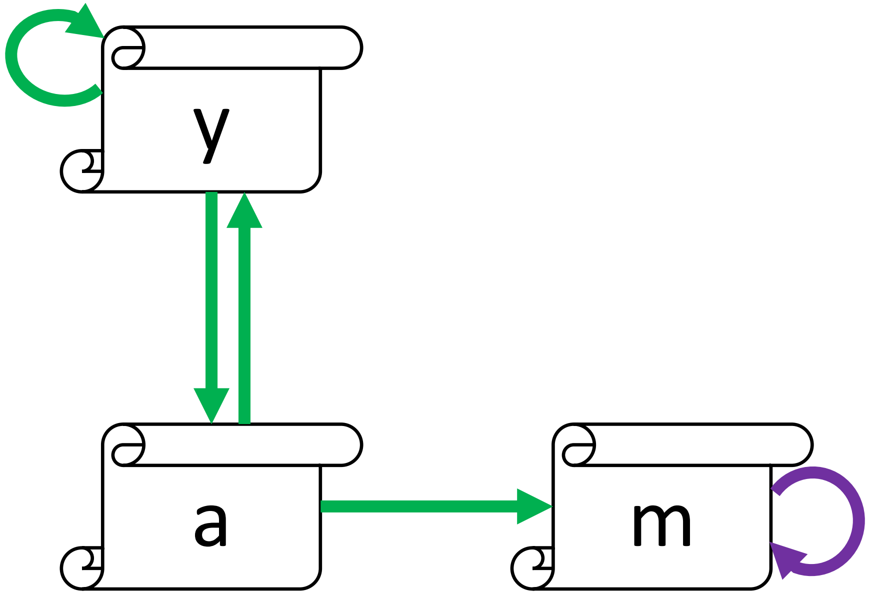
$$\mathbf{r}_m = \frac{r_a}{2}$$

r_y	1/3	2/6	3/12
r_a	1/3	1/6	2/12
r_m	1/3	3/6	7/12
Iteration T	0	1	2

Problem: Spider Traps

Power Iteration

- Set $r_j = \frac{1}{N}\ \forall j$
- $r_j = \sum_{i \to j} \frac{r_i}{d_i}$
  - iterate



	y	a	m
y	½	½	0
a	½	0	0
m	0	½	1

$$\mathbf{r_y} = \frac{r_y}{2} + \frac{r_a}{2}$$

$$\mathbf{r}_a = \frac{r_y}{2}$$

$$\mathbf{r}_m = \frac{r_a}{2}$$

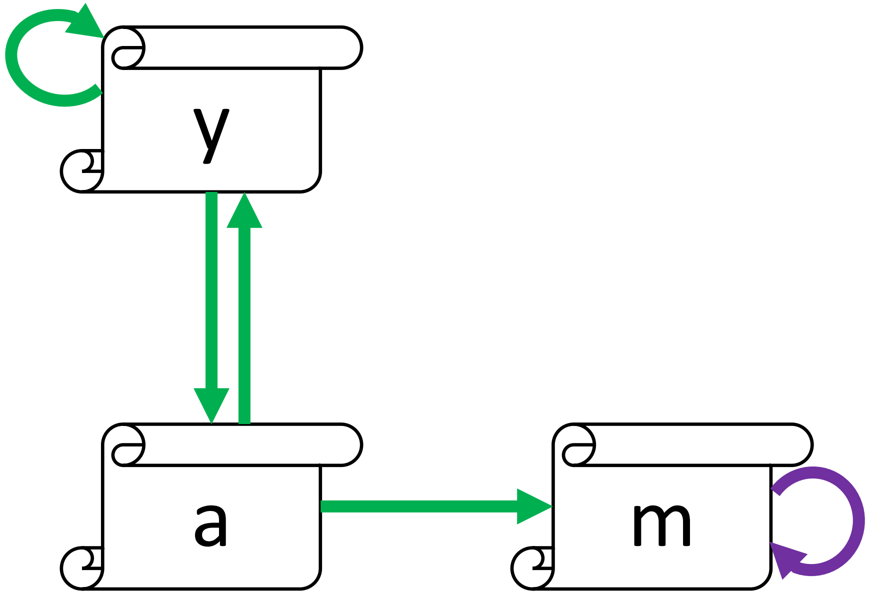
$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{matrix} 1/3 & 2/6 & 3/12 & 5/24 \\ 1/3 & 1/6 & 2/12 & 3/24 \\ 1/3 & 3/6 & 7/12 & 16/24 \end{matrix}$$

Iteration T	0	1	2	3

Problem: Spider Traps

Power Iteration

- Set $r_j = \frac{1}{N}\ \forall j$
- $r_j = \sum_{i \to j} \frac{r_i}{d_i}$
  - iterate



	y	a	m
y	½	½	0
a	½	0	0
m	0	½	1

$$\mathbf{r}_y = \frac{r_y}{2} + \frac{r_a}{2}$$

$$\mathbf{r}_a = \frac{r_y}{2}$$

$$\mathbf{r}_m = \frac{r_a}{2}$$

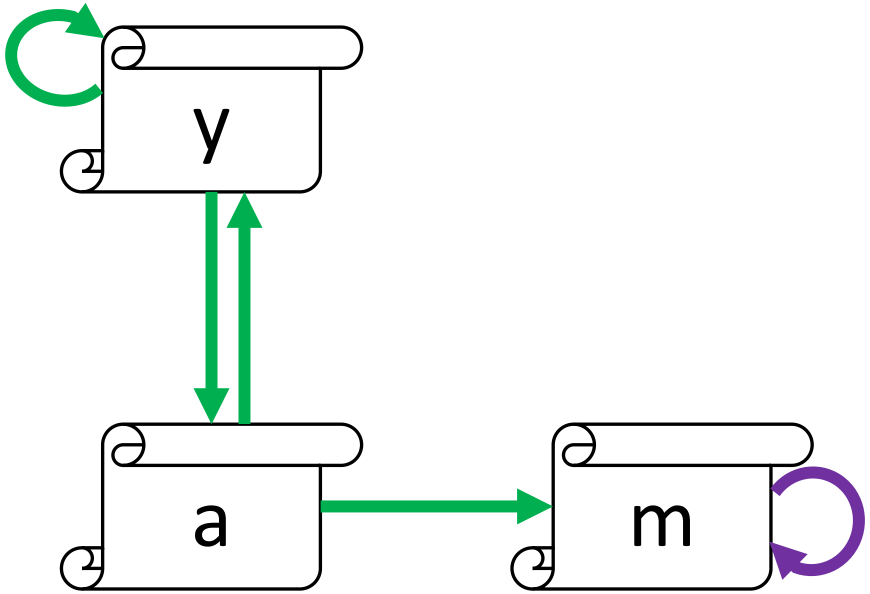
$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{matrix} 1/3 & 2/6 & 3/12 & 5/24 & & 0 \\ 1/3 & 1/6 & 2/12 & 3/24 & \ldots & 0 \\ 1/3 & 3/6 & 7/12 & 16/24 & & 1 \end{matrix}$$

Iteration T	0	1	2	3	…	X

Problem: Spider Traps

Power Iteration

- Set $r_j = \frac{1}{N}\ \forall j$
- $r_j = \sum_{i \to j} \frac{r_i}{d_i}$
  - iterate



	y	a	m
y	½	½	0
a	½	0	0
m	0	½	1

$$r_y = \frac{r_y}{2} + \frac{r_a}{2}$$

$$r_a = \frac{r_y}{2}$$

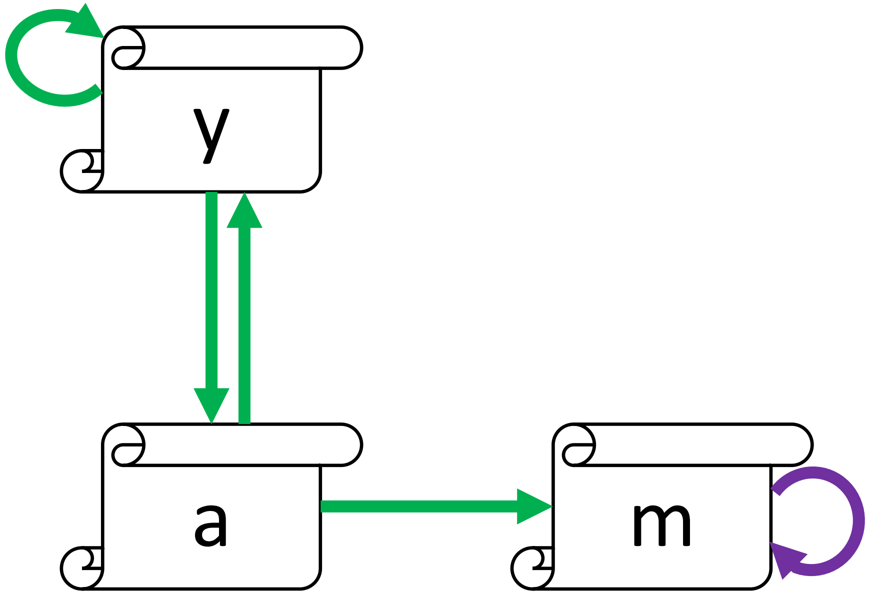
$$r_m = \frac{r_a}{2}$$

After a while, the random surfer will land on a page (“spider trap”) and never leave it with probability 1

Problem: Spider Traps

Power Iteration

- Set $r_j = \frac{1}{N}\ \forall j$
- $r_j = \sum_{i \to j} \frac{r_i}{d_i}$
    - iterate



	y	a	m
y	½	½	0
a	½	0	0
m	0	½	1

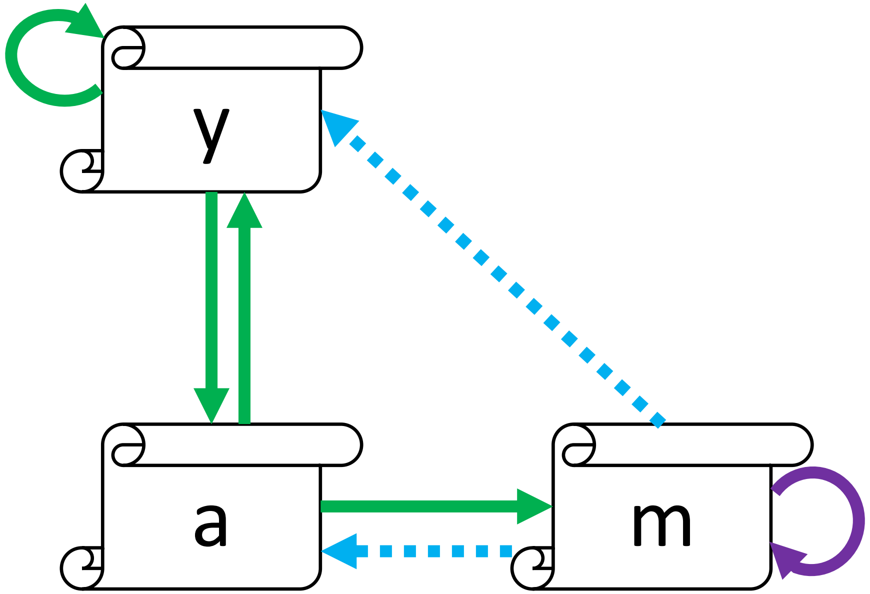
$$\mathbf{r_y} = \frac{r_y}{2} + \frac{r_a}{2}$$

$$\mathbf{r}_a = \frac{r_y}{2}$$

$$\mathbf{r}_m = \frac{r_a}{2}$$

Eventually spider traps absorb all importance (that's not what we want!)

Solution (to Spider Trap): Teleportation



Within a few time steps, random surfer will teleport out of spider trap

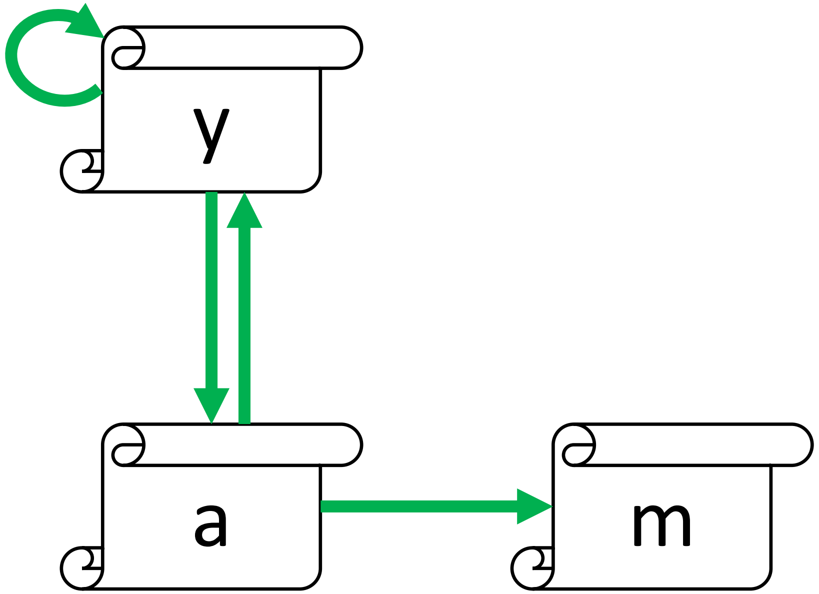
At each time step, the random surfer has two options:

- With probability β, follow link at random
- With prob. $(1-\beta)$, jump to some random page

Common value:
$\beta \in [0.8, 0.9]$

Problem: Dead Ends

	y	a	m
y	½	½	0
a	½	0	0
m	0	½	0



$$\mathbf{r_y} = \frac{r_y}{2} + \frac{r_a}{2}$$

$$\mathbf{r}_a = \frac{r_y}{2}$$

$$\mathbf{r}_m = \frac{r_a}{2}$$

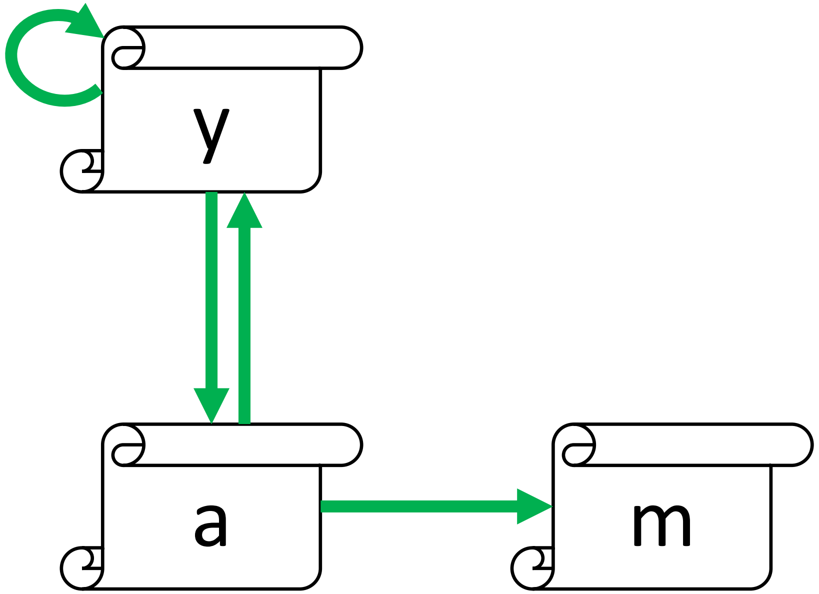
Power Iteration

- Set $r_j = \frac{1}{N}\ \forall j$
- $r_j = \sum_{i \to j} \frac{r_i}{d_i}$
  - iterate

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{matrix} 1/3 \\ 1/3 \\ 1/3 \end{matrix}$$

Iteration T	0

Problem: Dead Ends



$$\begin{array}{c|ccc} & y & a & m \\ \hline y & ½ & ½ & 0 \\ a & ½ & 0 & 0 \\ m & 0 & ½ & 0 \end{array}$$

$$\mathbf{r}_y = \frac{r_y}{2} + \frac{r_a}{2}$$

$$\mathbf{r}_a = \frac{r_y}{2}$$

$$\mathbf{r}_m = \frac{r_a}{2}$$

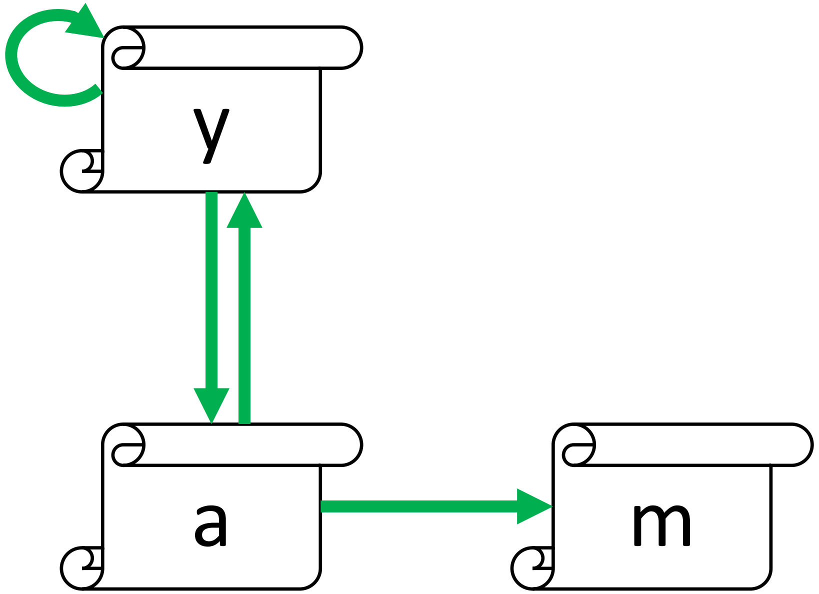
Power Iteration

- Set $r_j = \frac{1}{N}\ \forall j$
- $r_j = \sum_{i \to j} \frac{r_i}{d_i}$
  - iterate

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{array}{cc} 1/3 & 2/6 \\ 1/3 & 1/6 \\ 1/3 & 1/6 \end{array}$$

Iteration T	0	1

Problem: Dead Ends



	y	a	m
y	½	½	0
a	½	0	0
m	0	½	0

$$\mathbf{r_y} = \frac{r_y}{2} + \frac{r_a}{2}$$

$$\mathbf{r}_a = \frac{r_y}{2}$$

$$\mathbf{r}_m = \frac{r_a}{2}$$

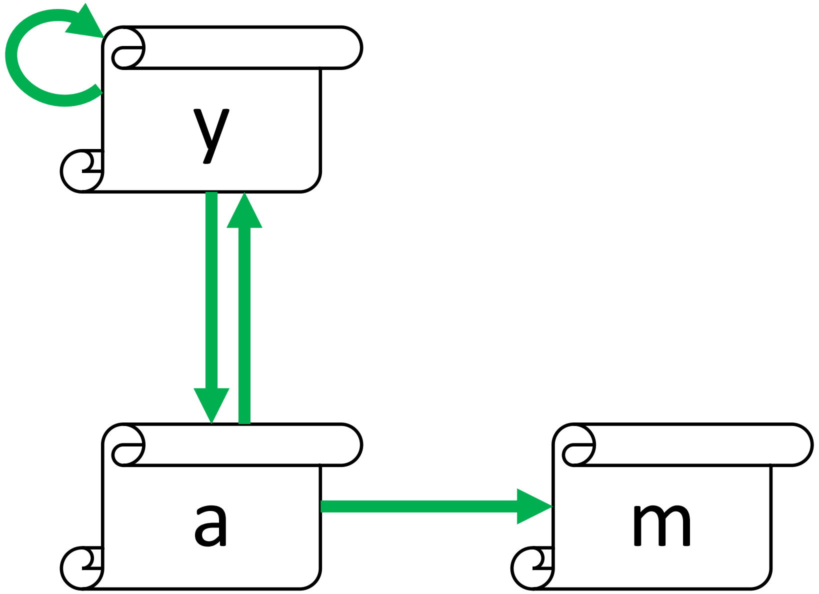
Power Iteration

- Set $r_j = \frac{1}{N} \; \forall j$
- $r_j = \sum_{i \to j} \frac{r_i}{d_i}$
  - iterate

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{matrix} 1/3 & 2/6 & 3/12 \\ 1/3 & 1/6 & 2/12 \\ 1/3 & 1/6 & 1/12 \end{matrix}$$

Iteration T	0	1	2

Problem: Dead Ends



$$\begin{array}{c|ccc} & y & a & m \\ \hline y & ½ & ½ & 0 \\ a & ½ & 0 & 0 \\ m & 0 & ½ & 0 \end{array}$$

$$\mathbf{r_y} = \frac{r_y}{2} + \frac{r_a}{2}$$

$$\mathbf{r}_a = \frac{r_y}{2}$$

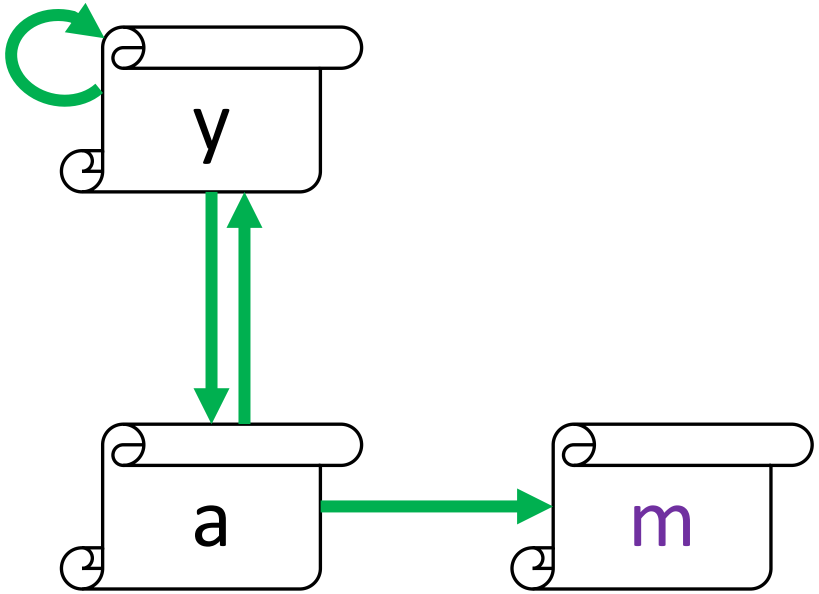
$$\mathbf{r}_m = \frac{r_a}{2}$$

Power Iteration

- Set $r_j = \frac{1}{N}\ \forall j$
- $r_j = \sum_{i \to j} \frac{r_i}{d_i}$
  - iterate

$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} =$	1/3 1/3 1/3	2/6 1/6 1/6	3/12 2/12 1/12	5/24 3/24 2/24
Iteration T	0	1	2	3

Problem: Dead Ends



	y	a	m
y	½	½	0
a	½	0	0
m	0	½	0

$$\mathbf{r}_y = \frac{r_y}{2} + \frac{r_a}{2}$$

$$\mathbf{r}_a = \frac{r_y}{2}$$

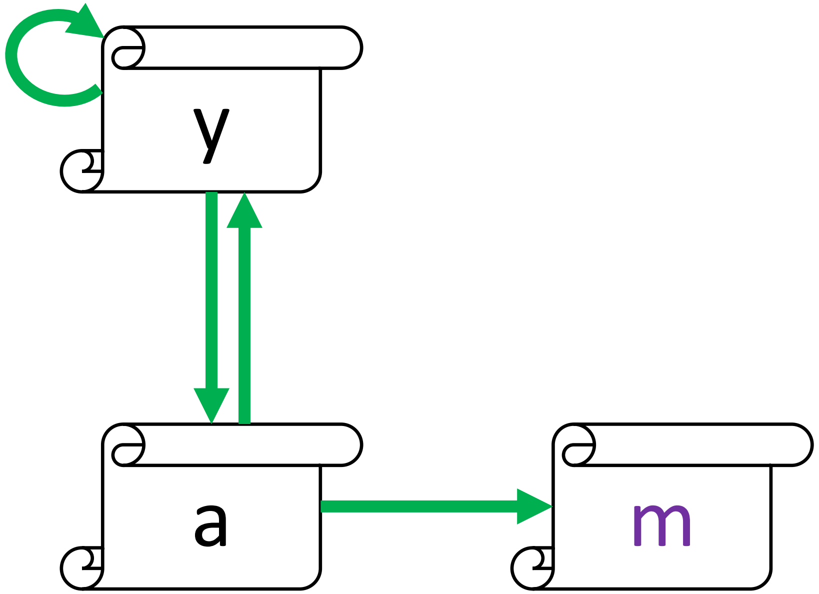
$$\mathbf{r}_m = \frac{r_a}{2}$$

Power Iteration

- Set $r_j = \frac{1}{N}\ \forall j$
- $r_j = \sum_{i \to j} \frac{r_i}{d_i}$
  - iterate

$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} =$	$\begin{matrix} 1/3 \\ 1/3 \\ 1/3 \end{matrix}$	$\begin{matrix} 2/6 \\ 1/6 \\ 1/6 \end{matrix}$	$\begin{matrix} 3/12 \\ 2/12 \\ 1/12 \end{matrix}$	$\begin{matrix} 5/24 \\ 3/24 \\ 2/24 \end{matrix}$	…	$\begin{matrix} 0 \\ 0 \\ 0 \end{matrix}$
Iteration T	0	1	2	3	…	X

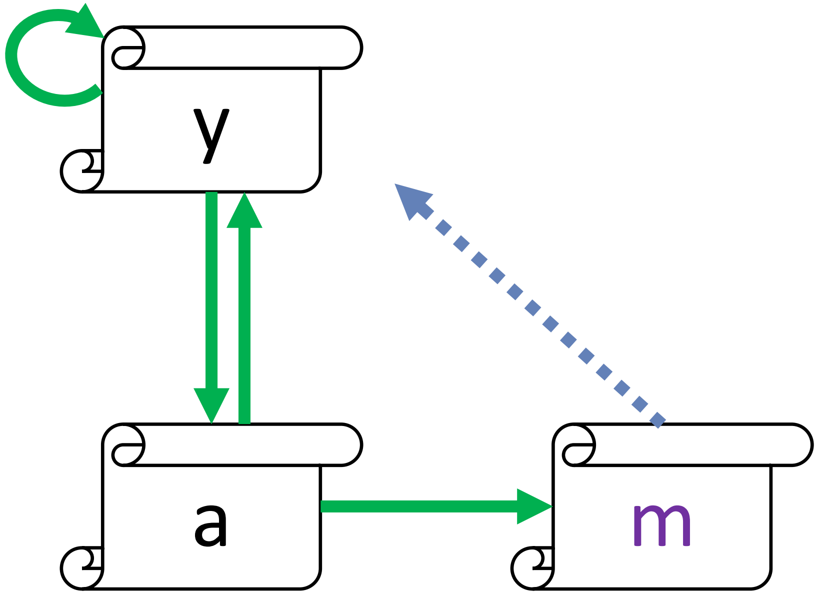
Solution (to Dead Ends): *Always* Teleport!



	y	a	m
y	½	½	0
a	½	0	0
m	0	½	0

- The matrix is not column stochastic so our initial assumptions are not met.
- Such pages cause importance to “leak out”

Solution (to Dead Ends): *Always* Teleport!



Follow random teleport links with probability 1.0 from dead-ends

	y	a	m
y	½	½	0
a	½	0	0
m	0	½	0

	y	a	m
y	½	½	⅓
a	½	0	⅓
m	0	½	⅓

Google's Solution: Random Teleports

At each time step, the random surfer has two options:

- With probability β, follow link at random
- With prob. $(1-\beta)$, jump to some random page

Google‘s Solution: Random Teleports

At each time step, the random surfer has two options:

- With probability β, follow link at random
- With prob. $(1-\beta)$, jump to some random page

- PageRank equation [Brin-Page, 98]

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

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.

Google‘s Solution: Random Teleports

At each time step, the random surfer has two options:

- With probability β, follow link at random
- With prob. $(1-\beta)$, jump to some random page

- PageRank equation [Brin-Page, 98]

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

Google matrix

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

Power method still works!

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.

Rearranging equation

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

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

$$\boldsymbol{r} = \boldsymbol{A}\boldsymbol{r}$$

Rearranging equation

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

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

$$\boldsymbol{r} = \boldsymbol{A}\boldsymbol{r}$$

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

Rearranging equation

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

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

$$\boldsymbol{r} = \boldsymbol{A}\boldsymbol{r}$$

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

$$\Leftrightarrow \boldsymbol{r} = \boldsymbol{r}\,\beta M + \left[\frac{1-\beta}{N}\right]_{N \times N}$$

To verify this, compute the j-th row of $\boldsymbol{r}$:

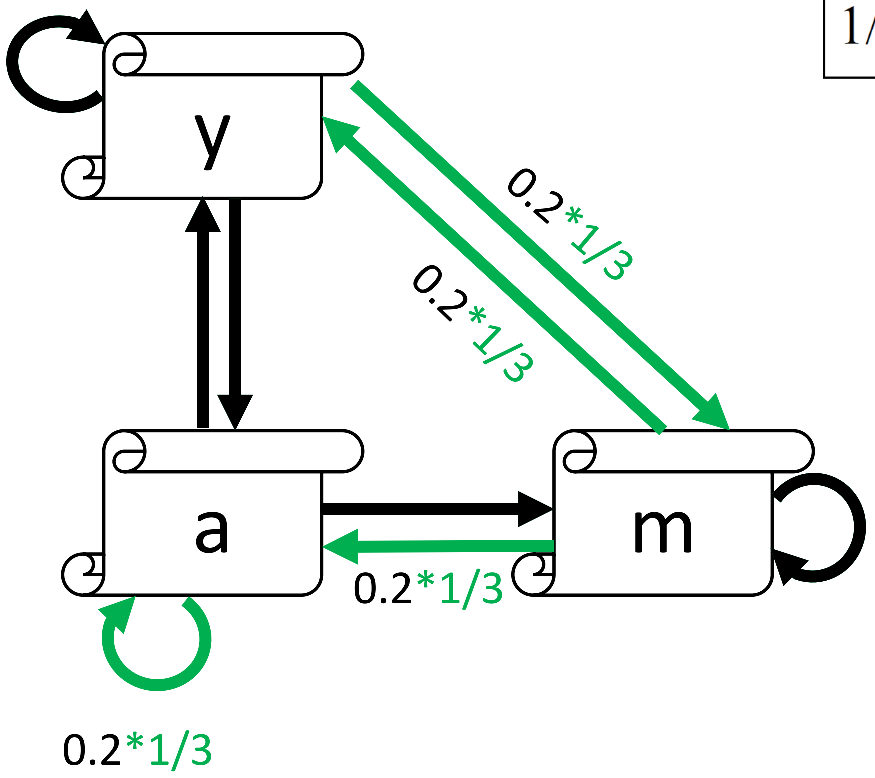
$$\boldsymbol{r}_j = \sum_{j=1}^{n} \boldsymbol{A}_{ij}\boldsymbol{r}_i$$

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.

Example

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

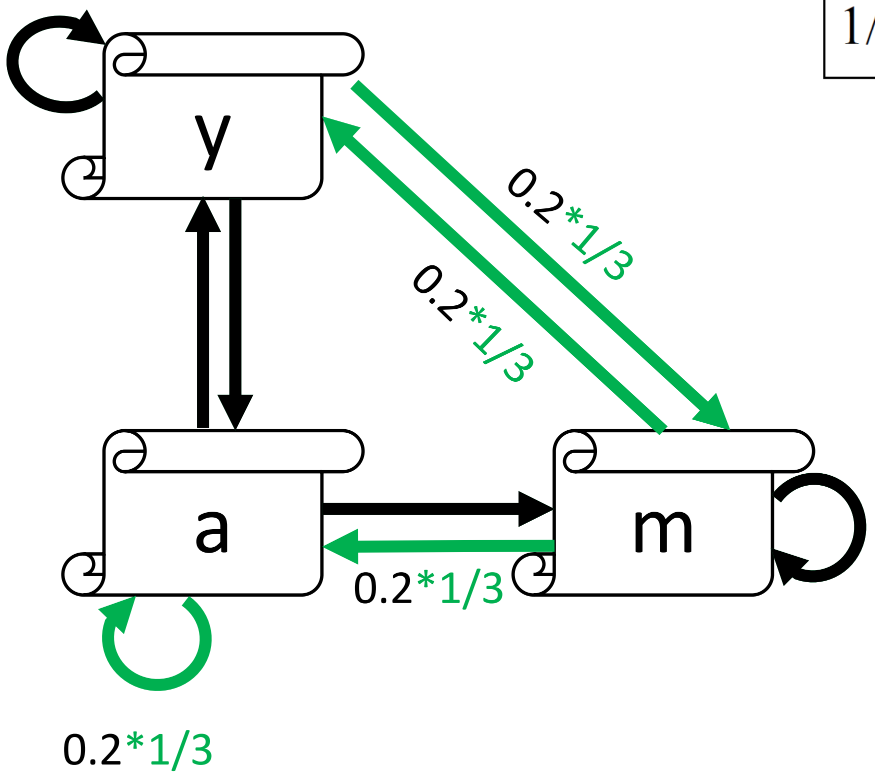
$$\begin{bmatrix} 7/15 & 7/15 & 1/15 \\ 7/15 & 1/15 & 1/15 \\ 1/15 & 7/15 & 13/15 \end{bmatrix} = 0.8 \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix} + 0.2 \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}$$



Example

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

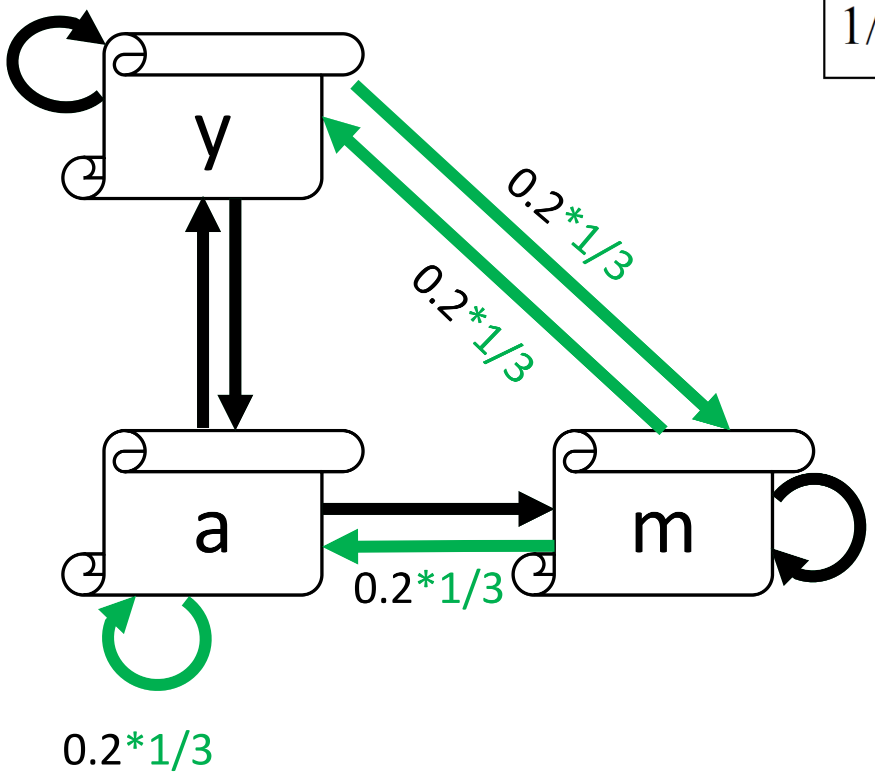
$$\begin{bmatrix} 7/15 & 7/15 & 1/15 \\ 7/15 & 1/15 & 1/15 \\ 1/15 & 7/15 & 13/15 \end{bmatrix} = 0.8 \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix} + 0.2 \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}$$



Example

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

$$\boxed{\begin{matrix} 7/15 & 7/15 & 1/15 \\ 7/15 & 1/15 & 1/15 \\ 1/15 & 7/15 & 13/15 \end{matrix}} = 0.8 \boxed{\begin{matrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{matrix}} + 0.2 \boxed{\begin{matrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{matrix}}$$



$$\begin{matrix} y \\ a \\ m \end{matrix} = \begin{matrix} 1/3 & 0.33 & 0.24 & 0.26 & & 7/33 \\ 1/3 & 0.20 & 0.20 & 0.18 & \dots & 5/33 \\ 1/3 & 0.46 & 0.52 & 0.56 & & 21/33 \end{matrix}$$

PageRank: Complete Algorithm

Input: Directed Graph $\boldsymbol{G}$ (can contain spider traps and dead ends) and parameter $\boldsymbol{\beta}$

- **Set:** $r_j^{old} = \frac{1}{N}$
- **repeat until convergence:** $\sum_j \left| r_j^{new} - r_j^{old} \right| < \varepsilon$
  - $\forall j$: $\boldsymbol{r}'^{new}_j = \sum_{i \to j} \boldsymbol{\beta} \frac{r_i^{old}}{d_i}$

    $\boldsymbol{r}'^{new}_j = \mathbf{0}$ if in-degree of $\boldsymbol{j}$ is **0**
  - **Now re-insert the leaked PageRank:**

    $\forall \boldsymbol{j}$: $\boldsymbol{r}_j^{new} = \boldsymbol{r}'^{new}_j + \frac{\mathbf{1} - \boldsymbol{S}}{\boldsymbol{N}}$ where: $S = \sum_j r'^{new}_j$
  - $\boldsymbol{r}^{old} = \boldsymbol{r}^{new}$

Output: PageRank vector **r**

Problems with PageRank

- **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

More about computation: see *Stanford Lecture Stanford C246* from Jure Leskovec & Mina Ghashami