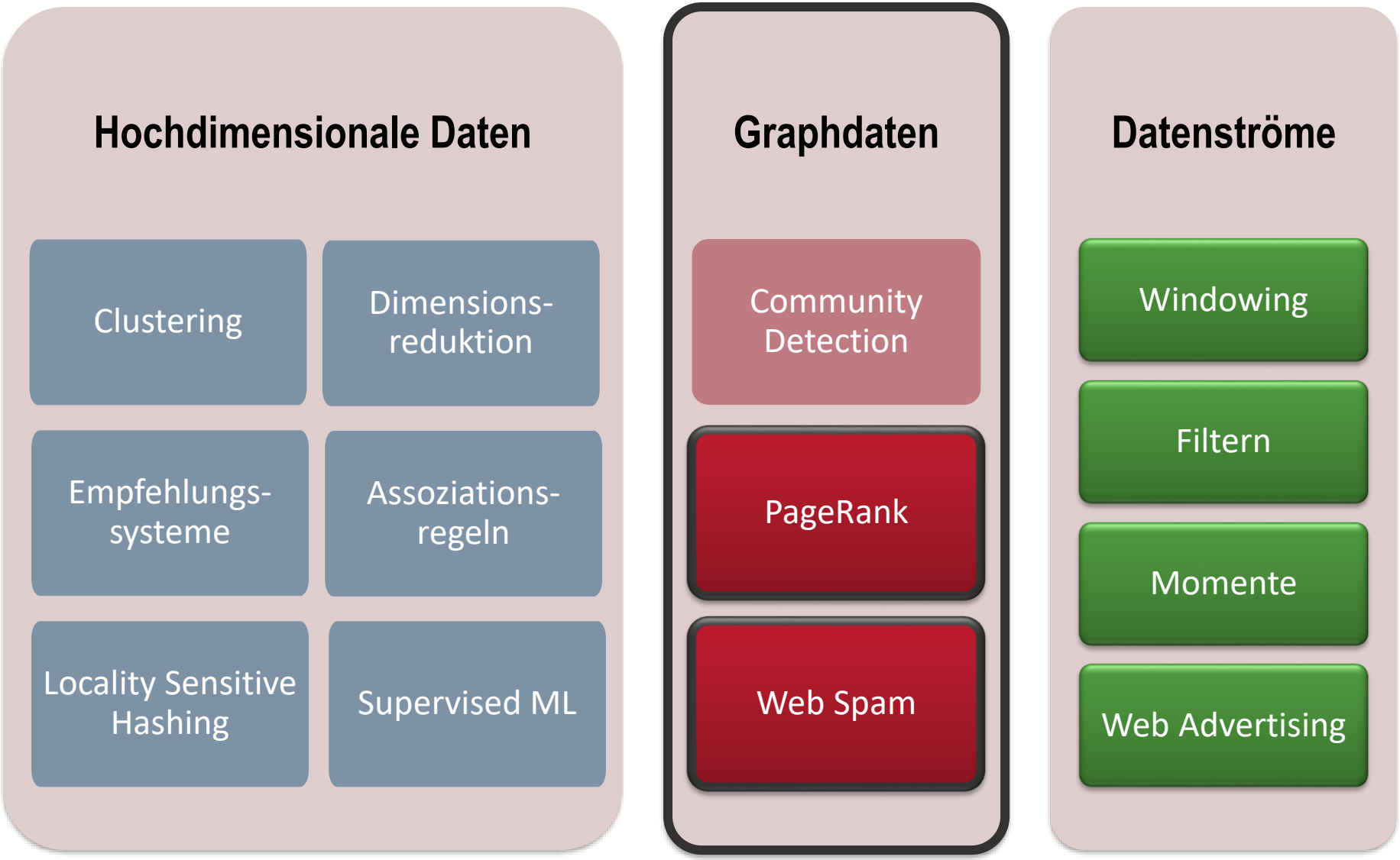
Data Mining

Linkanalyse

Johannes Zschache
Wintersemester 2019

Abteilung Datenbanken, Universität Leipzig
http://dbs.uni-leipzig.de

Übersicht

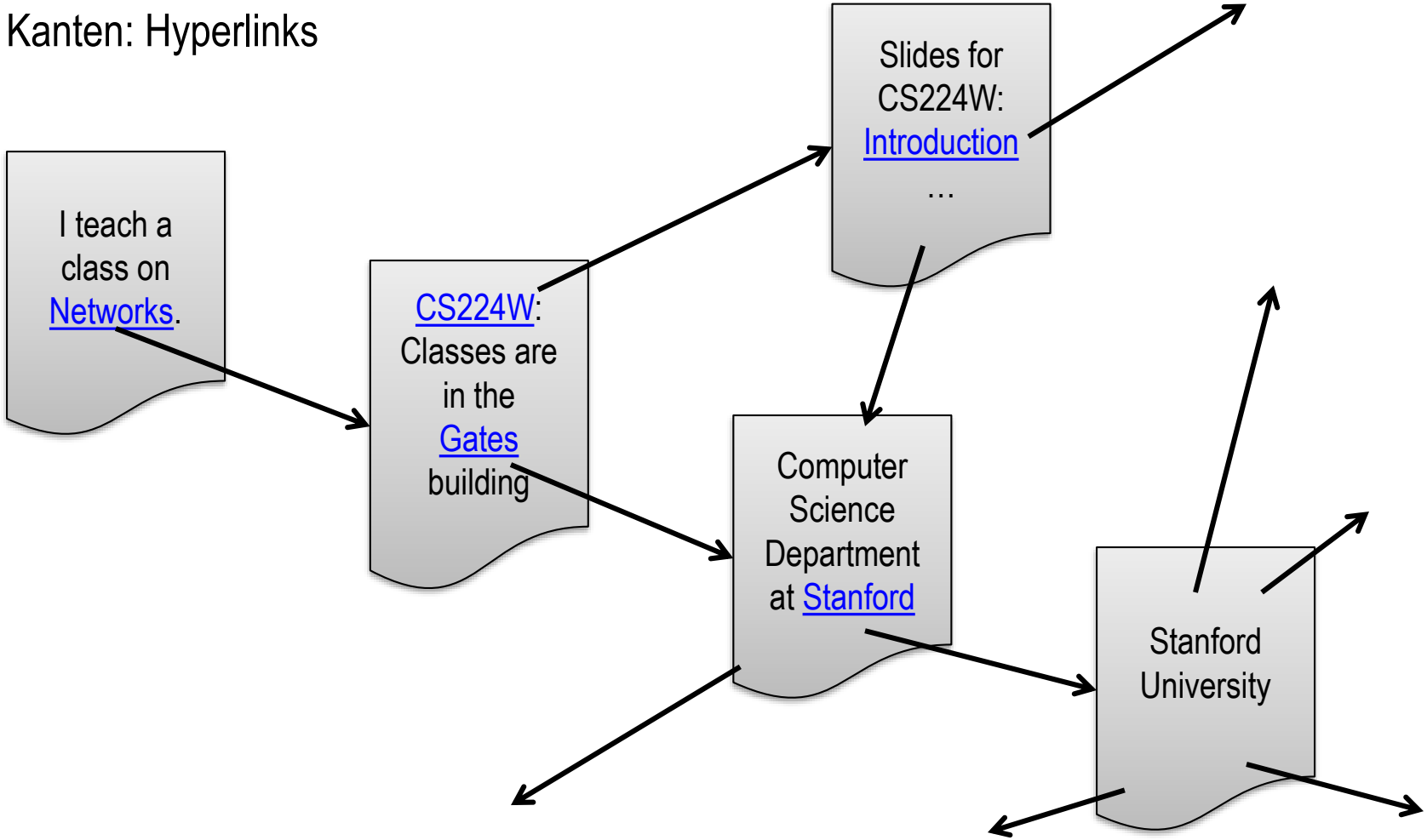

Hochdimensionale Daten
Clustering
Dimensions-reduktion
Empfehlungs-systeme
Assoziations-regeln
Locality Sensitive Hashing
Supervised ML
Graphdaten
Community Detection
PageRank
Web Spam
Datenströme
Windowing
Filtern
Momente
Web Advertising

Inhaltsverzeichnis

Literatur: Kapitel 5 aus „Mining of Massive Datasets“: http://www.mmds.org/

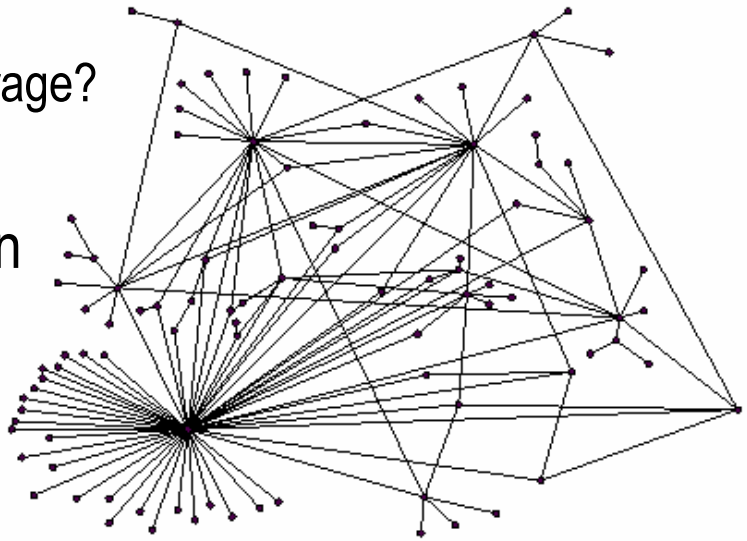
Graphdaten: World Wide Web

- Das Web als ein gerichteter Graph:
  - Knoten: Webseiten
  - Kanten: Hyperlinks



World Wide Web

- Wie kann das Web organisiert werden?
- 1. Versuch: Webverzeichnis: Yahoo, Curlie (DMOZ)
- 2. Versuch: Suchmaschine
  - Webcrawler durchlaufen Web
  - Inverted Index für Suchanfragen
- Herausforderungen für Suchmaschinen
  1. Welche Seiten sind vertrauenswürdig und enthalten tatsächlich die relevanten Informationen zu einer Anfrage?
  2. Welches ist die „beste“ Antwort auf eine Anfrage?
- Verwendung der **Hyperlink-Struktur** um die Bedeutung einer Webseite zu ermitteln



Inhaltsverzeichnis

- **Einführung**
- **PageRank**
  - Probleme
  - Effiziente Berechnung
  - Themenspezifischer PageRank
- **Link-Spam**
- **Übungen**

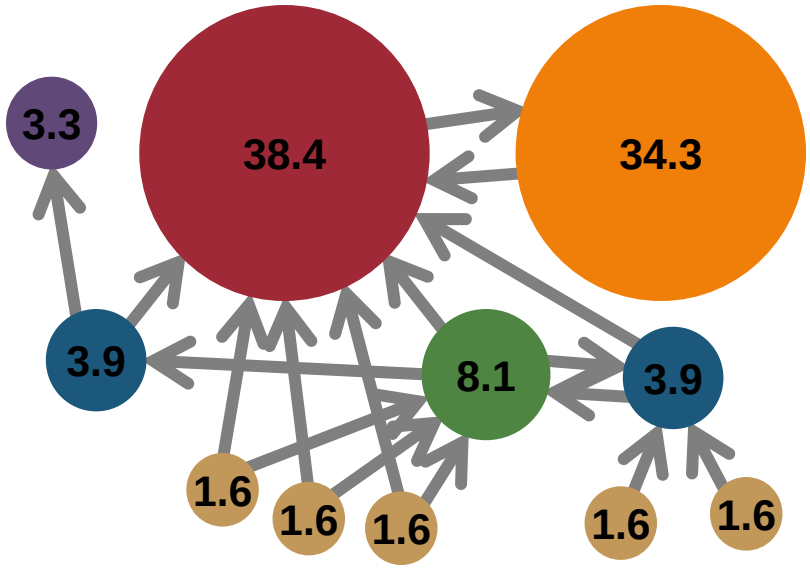
Literatur: Kapitel 5 aus „Mining of Massive Datasets“: http://www.mmds.org/

PageRank

- **Idee**: Webseite ist wichtig, falls sie von vielen Nutzern besucht wird
- Anstatt das Verhalten direkt zu beobachten, wird angenommen, dass Nutzer den Hyperlinks zufällig folgen: **Random-Surfer-Model**
  - Beginne auf einer beliebigen Seite
  - Wiederhole: Folge einem zufällig ausgewählten Hyperlink dieser Seite

- **PageRank** einer Webseite: „*Neigung*“, *dass ein Random-Surfer diese Webseite besucht*

- Folgerungen:
  - Webseite ist *wichtig (hoher PageRank)*, falls viele Hyperlinks auf sie verweisen
  - Hyperlinks von *wichtigen* Webseiten haben höheres Gewicht

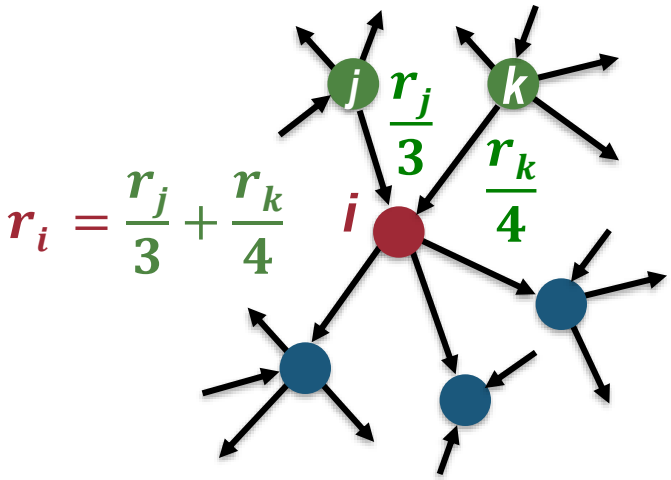


Einfacher PageRank

- **Einfacher PageRank:**

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

  - d_j: Anzahl der Webseiten mit mind. einem Hyperlink ausgehend von j (Outdegree)
  - $j \to i$: Summe läuft über alle auf i verweisenden Webseiten j



- Sei n die Anzahl der Knoten (Webseiten)
- Lineares Gleichungssystem aus n Gleichungen und n Unbekannten
- Eindeutige Lösung falls Einschränkung, z.B. auf $\sum_i r_i = n$
- *Gaußsches Eliminationsverfahren* nur bei Graphen mit kleinem n möglich
- *Alternative Berechnung für **riesigen** Webgraph notwendig*

Matrixformulierung

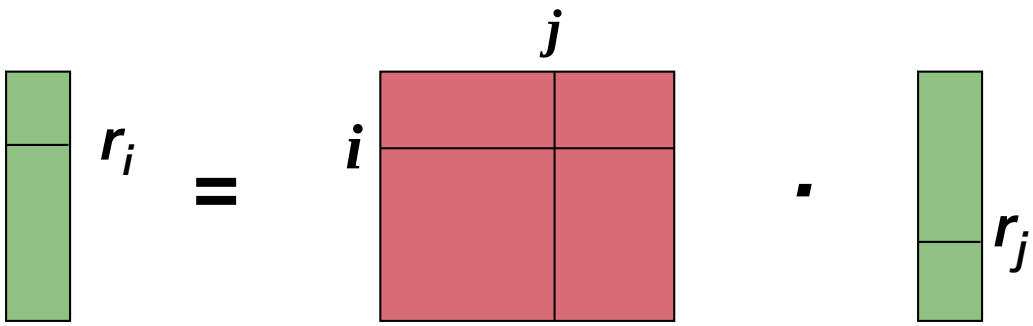
- Stochastische Adjazenzmatrix M $(n \times n)$

$$M_{ij} = \begin{cases} \frac{1}{\mathrm{d}_j}, & j \to i \\ 0, & \text{sonst} \end{cases}$$

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

- Für den PageRank-Vektor $\boldsymbol{r} = (r_1, r_2, \ldots, r_n)^T$ gilt:

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



Power Iteration Method

- *Power Iteration Method*:
  - Zu Beginn: $\boldsymbol{r}^{(\mathbf{0})} = (1,1, \ldots, 1)^T$
  - Iteration: $\boldsymbol{r}^{(\boldsymbol{t+1})} = M \cdot \boldsymbol{r}^{(\boldsymbol{t})}$
  - Stopp, falls $\left\| \boldsymbol{r}^{(\boldsymbol{t+1})} - \boldsymbol{r}^{(\boldsymbol{t})} \right\| < \boldsymbol{\varepsilon}$

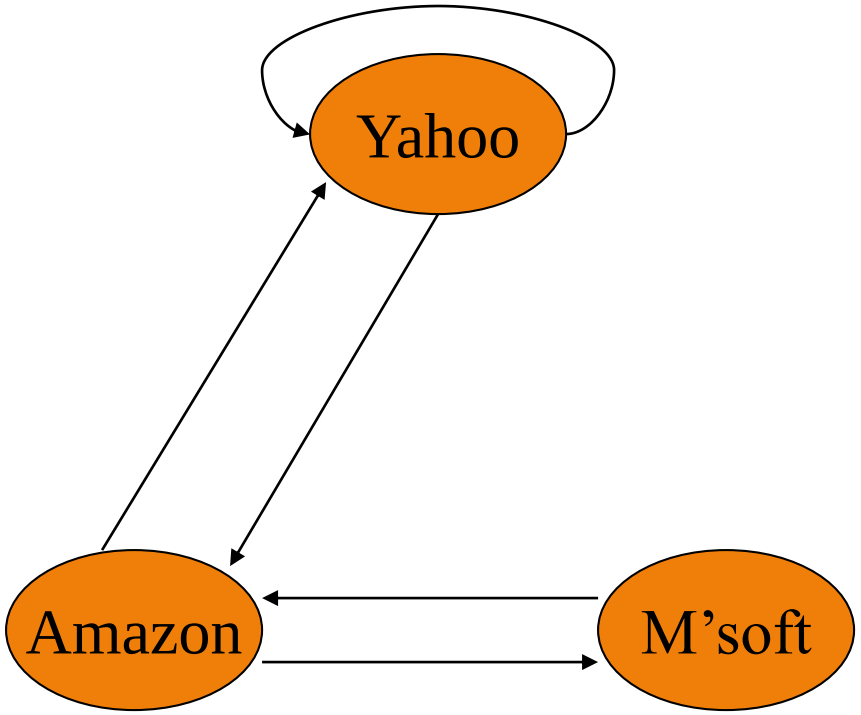
- Vektor $\boldsymbol{r}^{(t)} = M^t \cdot \boldsymbol{r}^{(\mathbf{0})}$ gibt mit $\frac{r_i^{(t)}}{n}$ die Wahrscheinlichkeit, dass sich der Random-Surfer zum Zeitpunkt t auf einer Seite $i = 1, \ldots, n$ befindet

- *Falls Algorithmus konvergiert,* gilt mit $\boldsymbol{r} \coloneqq \log_{t \to \infty} M^t \cdot \boldsymbol{r}^{(\mathbf{0})}$

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

- Vektor $\boldsymbol{r}$ gibt den PageRank

Beispiel: Mini-WWW



M	y	a	m
y	1/2	1/2	0
a	1/2	0	1
m	0	1/2	0

$$y = \frac{1}{2}\,y + \frac{1}{2}\,a$$
$$a = \frac{1}{2}\,y + m$$
$$m = \frac{1}{2}\,a$$

Beispiel: Mini-WWW

- Iteration $\boldsymbol{r}^{(t+1)} = M \cdot \boldsymbol{r}^{(t)}$ mit $\boldsymbol{r}^{(t)} = \left(y^{(t)}, a^{(t)}, m^{(t)}\right)$
  - $y^{(t+1)} = \frac{1}{2} y^{(t)} + \frac{1}{2} a^{(t)}$
  - $a^{(t+1)} = \frac{1}{2} y^{(t)} + m^{(t)}$
  - $m^{(t+1)} = \frac{1}{2} a^{(t)}$

	y	a	m
y	1/2	1/2	0
a	1/2	0	1
m	0	1/2	0

$$\begin{array}{cccccccc} y & & 1 & 1 & 5/4 & 9/8 & & 6/5 \\ a & = & 1 & 3/2 & 1 & 11/8 & \ldots & 6/5 \\ m & & 1 & 1/2 & 3/4 & 1/2 & & 3/5 \end{array}$$

Inhaltsverzeichnis

- **Einführung**
- **PageRank**
  - Probleme
  - Effiziente Berechnung
  - Themenspezifischer PageRank
- **Link-Spam**
- **Übungen**

Literatur: Kapitel 5 aus „Mining of Massive Datasets“: http://www.mmds.org/

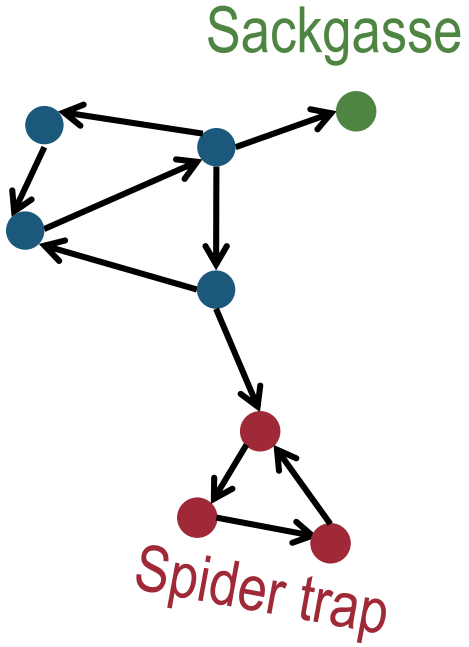
PageRank: Probleme

- Konvergenz der Power-Iteration-Method ist nicht sicher

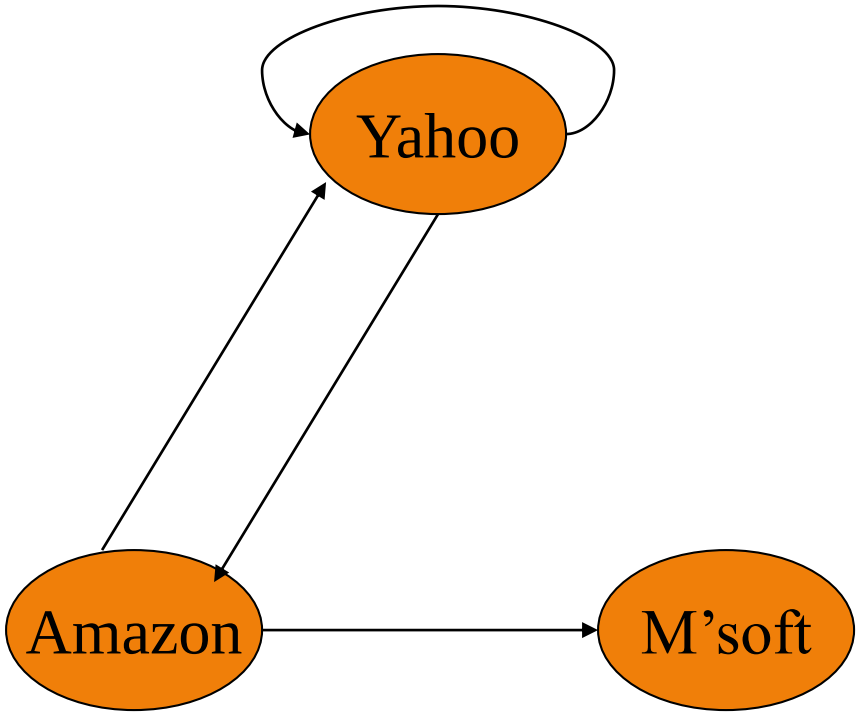
- Das reale WWW ist komplex

- **Sackgassen**: Webseiten ohne Hyperlinks
  - Random-Surfer kann Sackgasse nicht verlassen
  - Gewichtungen „verschwinden“ über Sackgasse

- **Spider Traps**: Gruppen von Webseiten ohne ausgehende Hyperlinks
  - Random-Surfer kann Gruppe nicht verlassen
  - Gewichtungen konzentrieren sich auf Gruppe



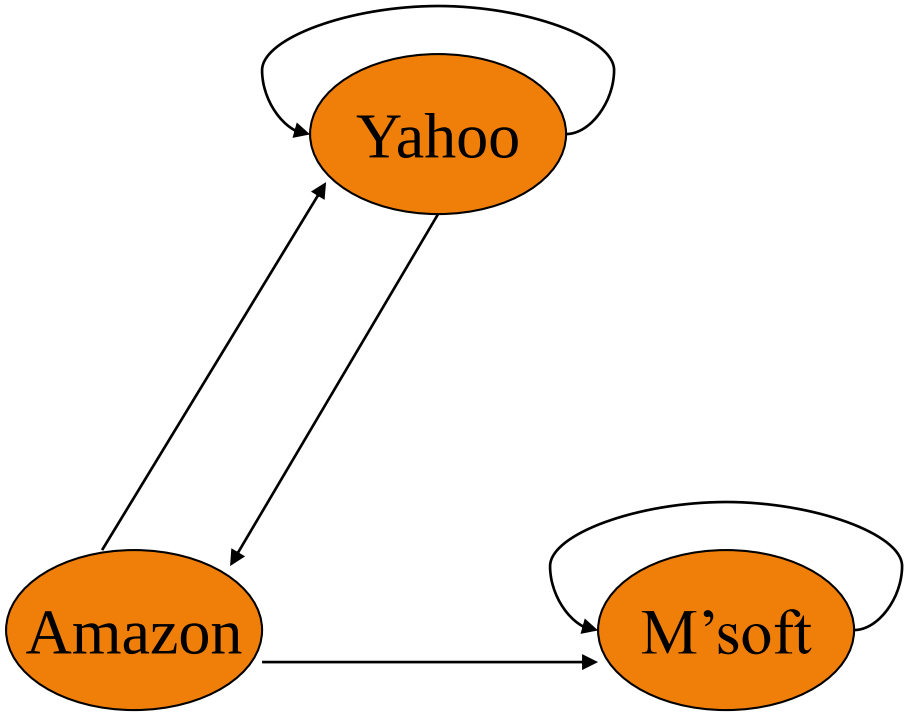
Problem: Sackgasse



M	y	a	m
y	1/2	1/2	0
a	1/2	0	0
m	0	1/2	0

$$y = \frac{1}{2}\,y + \frac{1}{2}\,a$$
$$a = \frac{1}{2}\,y$$
$$m = \frac{1}{2}\,a$$

Problem: Spider Trap



M	y	a	m
y	1/2	1/2	0
a	1/2	0	0
m	0	1/2	1

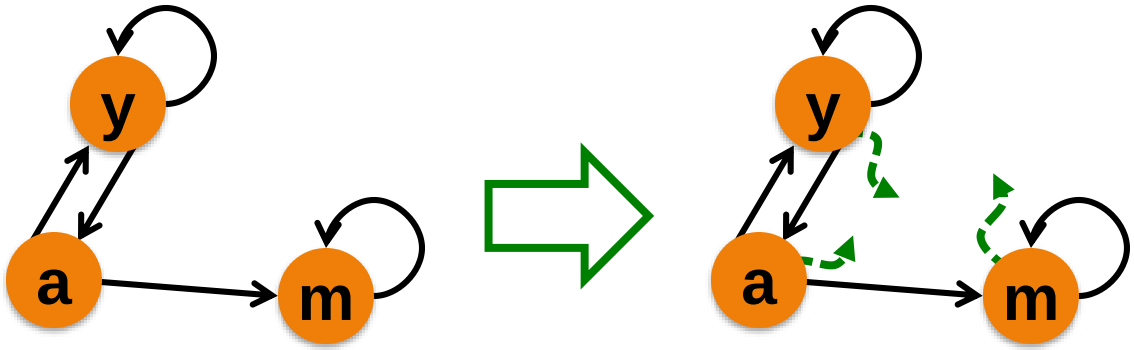
$$y = \frac{1}{2}y + \frac{1}{2}a$$
$$a = \frac{1}{2}y$$
$$m = \frac{1}{2}a + m$$

PageRank (allgemeine Form)

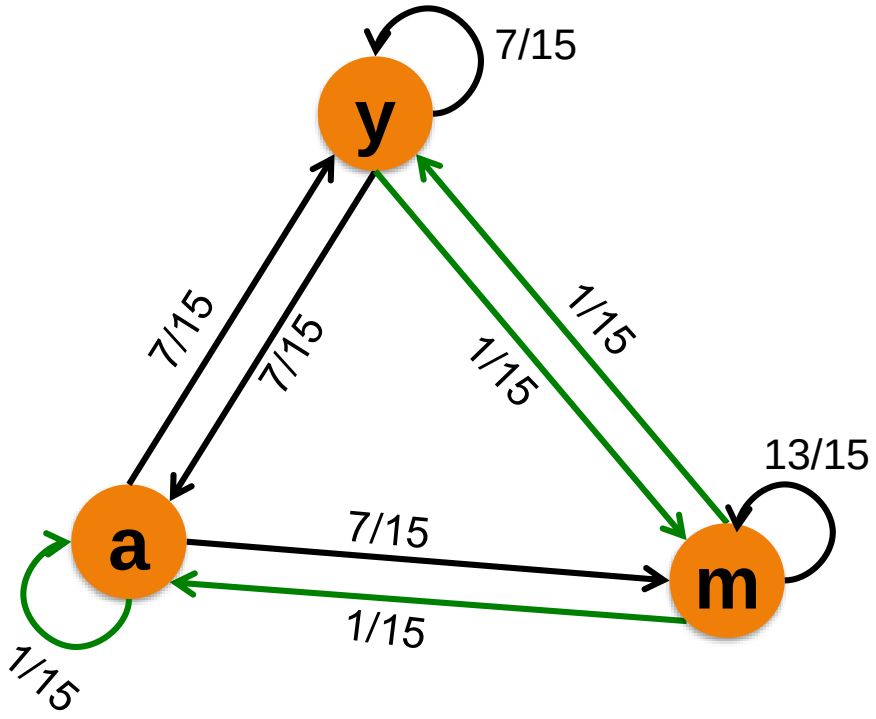
- PageRank [Brin & Page, 1998, http://infolab.stanford.edu/~backrub/google.html]

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

- Interpretation:
  - Mit Wahrscheinlichkeit β, folge einem zufällig gewählten Hyperlink
  - Mit Wahrscheinlichkeit $1-\beta$, wechsel zu einer zufälligen Seite
- Gewöhnlich setzt man β auf einen Wert zwischen 0.8 und 0.9
- *Annahme: Keine Sackgassen*



Beispiel



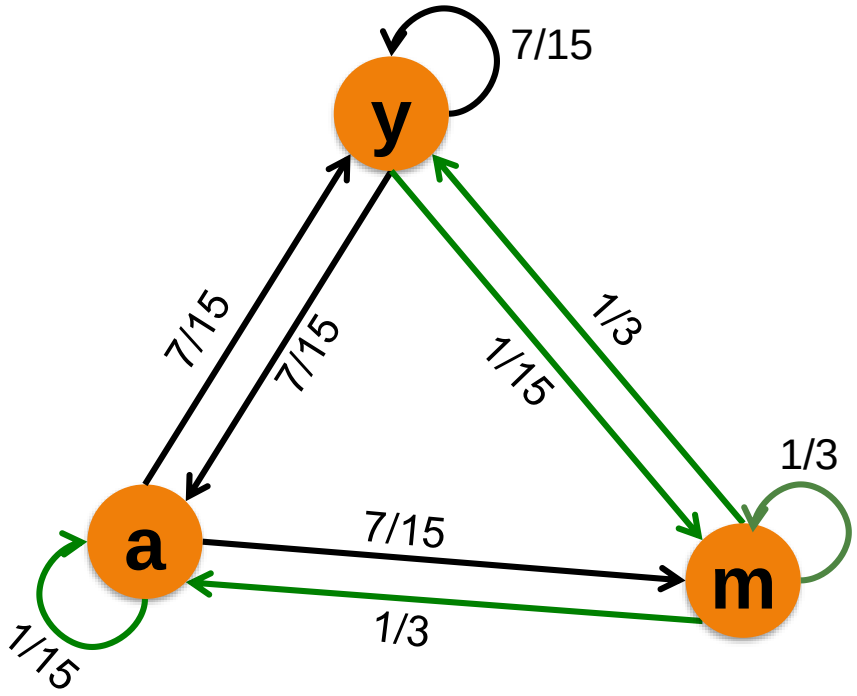
$$0.8 \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix} + 0.2 \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}$$

M $\qquad$ $[1/n]_{nxn}$

$$= \begin{array}{c} \\ y \\ a \\ m \end{array} \begin{bmatrix} 7/15 & 7/15 & 1/15 \\ 7/15 & 1/15 & 1/15 \\ 1/15 & 7/15 & 13/15 \end{bmatrix}$$

$$\begin{array}{cc} y & \\ a & = \\ m & \end{array} \begin{array}{ccccc} 1 & 1 & 0.84 & 0.78 & \\ 1 & 0.6 & 0.60 & 0.54 & \ldots \\ 1 & 1.4 & 1.56 & 1.69 & \end{array} \begin{array}{c} 7/11 \\ 5/11 \\ 21/11 \end{array}$$

Beispiel: mit Sackgasse

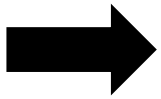


M

$$0.8 \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \end{bmatrix} + 0.2\,?$$

$$= \begin{matrix} y \\ a \\ m \end{matrix} \begin{bmatrix} 7/15 & 7/15 & \mathbf{1/3} \\ 7/15 & 1/15 & \mathbf{1/3} \\ 1/15 & 7/15 & \mathbf{1/3} \end{bmatrix}$$

Annahme: falls Knoten eine Sackgasse ist, wechsel mit Wahrscheinlichkeit 1 zu einer zufälligen Seite



Alternative Berechnung des PageRank

PageRank: Vollständiger Algorithmus

- Eingabe: Gerichteter Graph $\boldsymbol{G}$ aus n Knoten (inkl. Sackgassen und Spider Traps) und Parameter $\boldsymbol{\beta}$
- Ausgabe: PageRank Vector $\boldsymbol{r^{new}}$

– Intitialisiere: $\forall i: r_i^{old} = 0,\ r_i^{new} = 1$

– Wiederhole

1. Berechne $\sum_j \left| r_j^{new} - r_j^{old} \right|$ und Abbruch, falls $< \varepsilon$
2. $\boldsymbol{r}^{old} \leftarrow \boldsymbol{r}^{new}$
3. $\forall i: r_i^{new} \leftarrow \sum_{j \to i} \beta \frac{r_j^{old}}{d_j}$
4. $S \leftarrow \sum_i r_i^{new}$
5. $\forall i: r_i^{new} \mathrel{+}= 1 - \frac{S}{n}$

Inhaltsverzeichnis

- **Einführung**
- **PageRank**
  - Probleme
  - Effiziente Berechnung
  - Themenspezifischer PageRank
- **Link-Spam**
- **Übungen**

Literatur: Kapitel 5 aus „Mining of Massive Datasets“: http://www.mmds.org/

Berechnung des PageRank

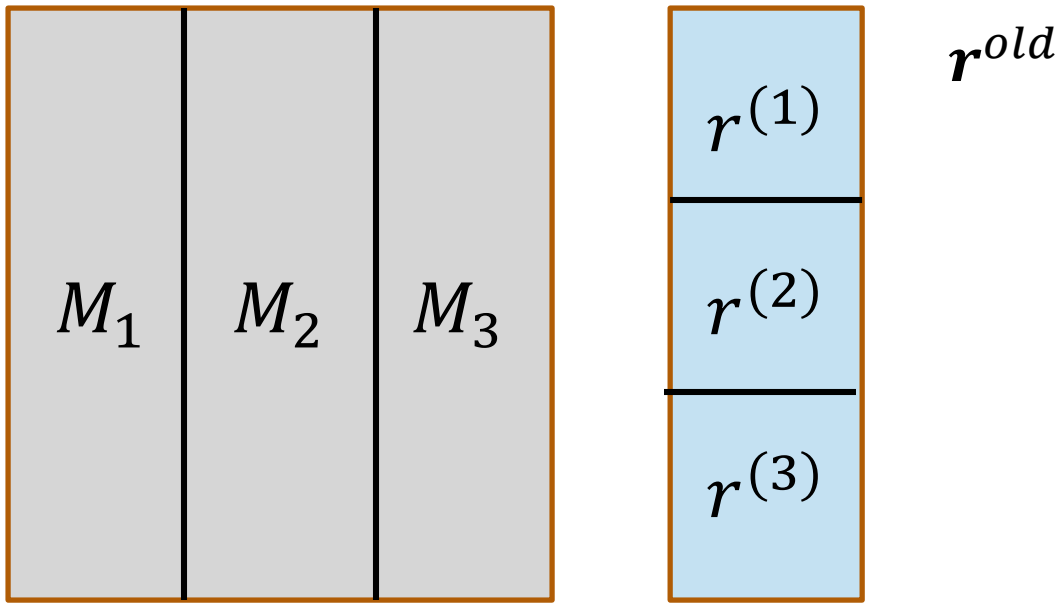
- Stochastische Adjazenzmatrix $\boldsymbol{M}$ ist sehr groß und liegt verteilt vor
- Initialisierung von $\boldsymbol{r^{old}}$ $\left(\forall i: r_i^{old} = 0\right)$ und Vektor $\boldsymbol{r^{new}}$ $(\forall i: r_i^{new} = 1)$ im Hauptspeicher

Wiederhole:

1. Berechne $\sum_j \left| r_j^{new} - r_j^{old} \right|$ und Abbruch, falls $< \varepsilon$
2. $\boldsymbol{r}^{old} \leftarrow \boldsymbol{r}^{new}$
3. **Matrix-Vektor-Produkt**: $\boldsymbol{r}^{new} \leftarrow \beta \cdot \boldsymbol{M} \cdot \boldsymbol{r}^{old}$
4. $S \leftarrow \sum_j r_j^{new}$
5. $\forall i:\ r_i^{new} \mathrel{+}= 1 - \frac{S}{n}$

Verteilte Berechnung des PageRank

- Matrix-Vektor-Produkt über **MapReduce**
- *Problem*: PageRank-Vektor $\boldsymbol{r}^{old}$ passt nicht in Hauptspeicher
  - Mind. 1 Billion (10^{12}) Webseiten
  - Einfache/doppelte Genauigkeit für Wert: 4 Byte/8 Byte
  - 4TB/8TB Speicherbedarf für Vektor
- *Unterteilung des Vektors* r^{old} *in Abschnitte* (Siehe auch Folie 1-26)



Verteilte Berechnung des PageRank

- Jeder Map-Task ist nur für einen Streifen M_k zuständig und benötigt auch nur den dazugehörigen Teil $r^{(k)}$ von $\boldsymbol{r}^{old}$

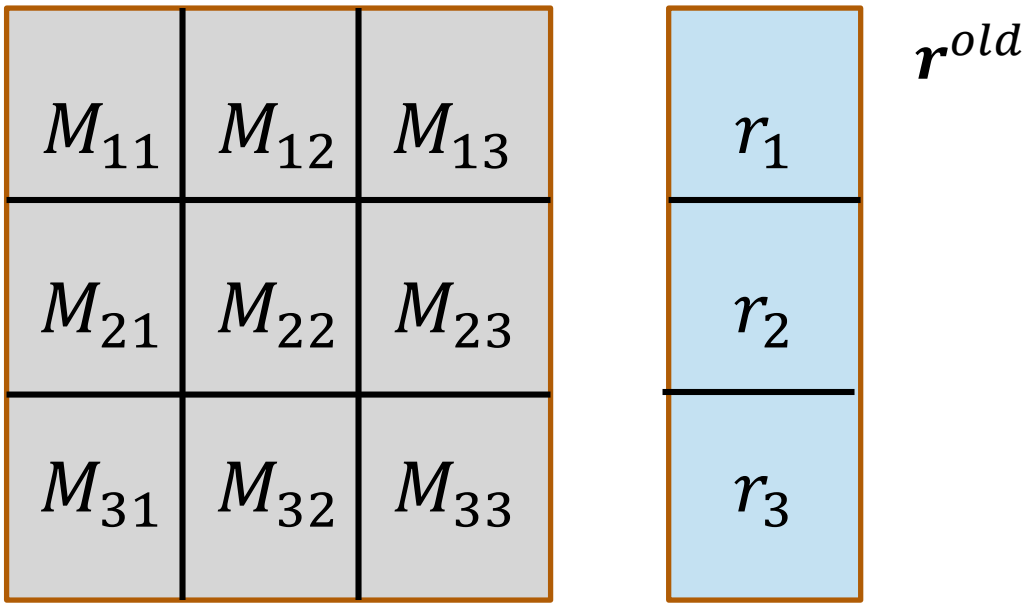
Wiederhole:

1. Berechne $\sum_j \left| r_j^{new} - r_j^{old} \right|$ und Abbruch, falls $< \varepsilon$
2. $\boldsymbol{r}^{old} \leftarrow \boldsymbol{r}^{new}$ (**nur relevanter Teil $\boldsymbol{r}^{(k)}$**)
3. **Matrix-Vektor-Produkt**: $\forall i\colon\ r_i^{new} \leftarrow \sum_{j \to i} \beta \ \frac{r_j^{old}}{d_j}$
   - Eingabe: Elemente (i, j, m_{ij}) des Streifens $M_{\boldsymbol{k}}$
   - Map: Elemente (i, j, m_{ij}) auf $(i,\ m_{ij} r_j)$
   - Reducer für Schlüssel i bekommt Liste $[m_{i1} r_1, \ldots, m_{in} r_n]$ und berechnet $\left(i, \sum_j \beta m_{ij} r_j\right)$ also r_i^{new}
4. $S \leftarrow \sum_j r_j^{new}$
5. $\forall i\colon\ r_i^{new} \mathrel{+}= 1 - \frac{S}{n}$

4 MapReduce-Prozeduren

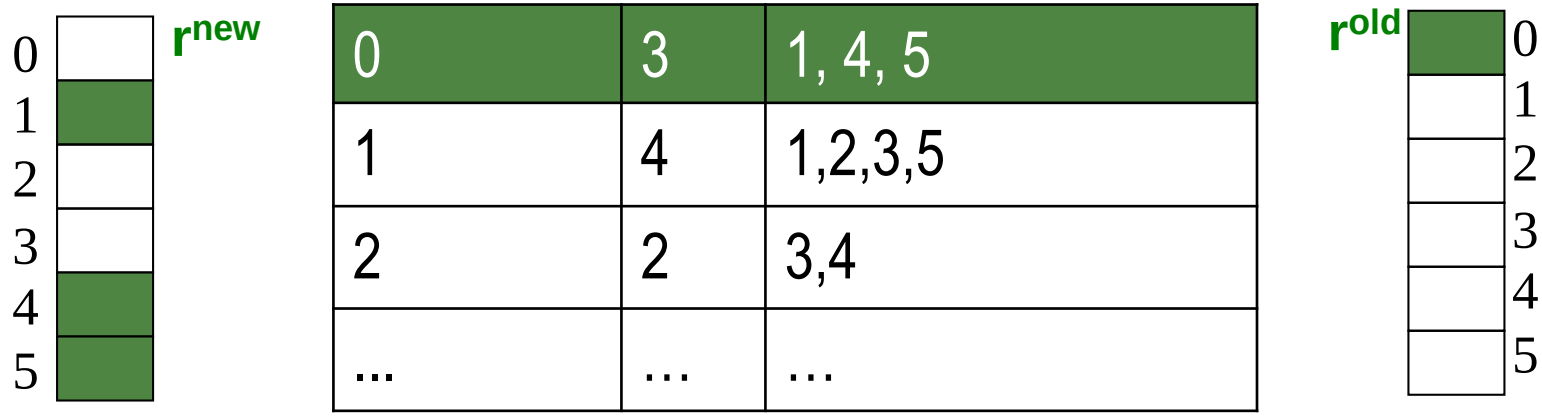
Blockmatrix

- Unterteilung von $\boldsymbol{r}^{old}$ in k Blöcke der Länge l
- **Zusätzliche Unterteilung der Matrix in Quadrate der Länge l**
- *Vorteil*: Ein Map-Task ist nur für ein Quadrat verantwortlich und somit schneller fertig
- *Nachteile*:
  - Insg. k^2 (statt nur k) Map-Tasks
  - Ein Block von $\boldsymbol{r}^{old}$ wird von insgesamt k Map-Tasks gelesen

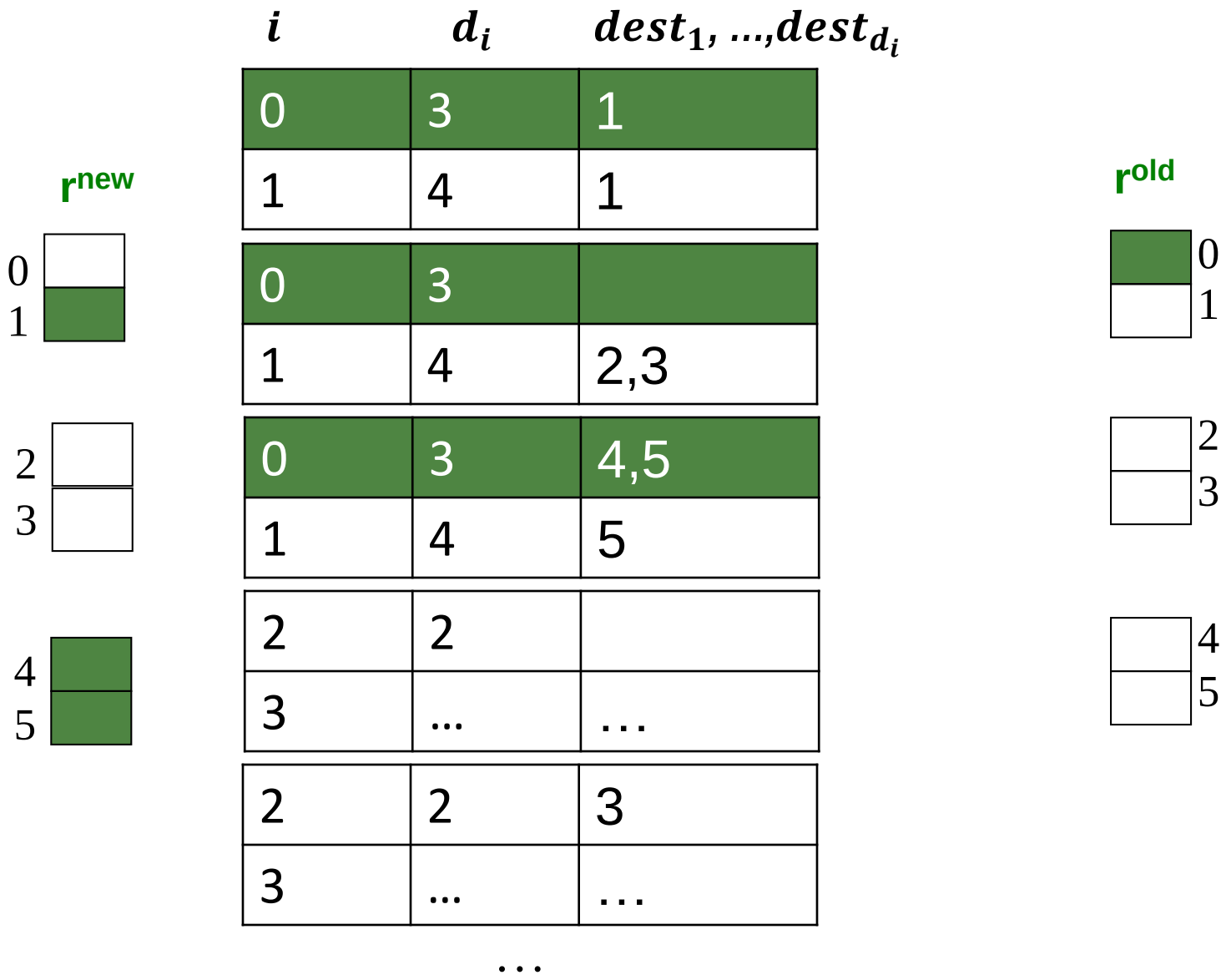


Effiziente Repräsentation der Matrix M

- Effizientere Repräsentation der Matrix M möglich, da spärlich besetzt und nur zwei verschiedene Werte pro Spalte
- Für jede Spalte: Speichere d_i (Anzahl der Hyperlinks) und Liste der Zeilen mit einem Wert ungleich Null: $dest_1, \ldots, dest_{d_i}$
- Unterteilung von M in Spalten:



Blockmatrix aus 3x3 Quadrate



Inhaltsverzeichnis

- **Einführung**
- **PageRank**
  - Probleme
  - Effiziente Berechnung
  - Themenspezifischer PageRank
- **Link-Spam**
- **Übungen**

Literatur: Kapitel 5 aus „Mining of Massive Datasets“: http://www.mmds.org/

Themenspezifischer PageRank

- PageRank misst die allgemeine Beliebtheit einer Webseite und vernachlässigt somit evtl. wichtige themenspezifische Quellen

- **Ziel**: Bewertung von Webseiten nicht nur nach Popularität, sondern auch nach Thema, z.B. Wissenschaft, Kunst, Natur, Motorsport, …
  - Berücksichtigung des Kontexts oder der Interessen eines Nutzers bei Suchanfragen
  - Beispiel: Anfrage nach „Jaguar“ soll, je nach Interesse des Nutzers, entweder zu Webseiten mit Tieren oder mit Autos führen

- **Idee**: Höhere Gewichtung für themenspezifische Seiten
  - Menge $\boldsymbol{S}$ mit themenspezifischen Seiten
  - Einen PageRank-Vektor $\boldsymbol{r}^{(\boldsymbol{S})}$ für jedes $\boldsymbol{S}$
  - Falls $\boldsymbol{i} \boldsymbol{\epsilon} \boldsymbol{S}$: $\boldsymbol{r_i^{(S)}} = \sum_{j \to i} \beta \, \frac{r_j}{d_j} + (1-\beta) \frac{n}{|S|}$; Sonst: $\boldsymbol{r_i^{(S)}} = \sum_{j \to i} \beta \, \frac{r_j}{d_j}$
  - Mit Wahrscheinlichkeit $1-\beta$: Wähle zufällig eine Seite aus $\boldsymbol{S}$
  - Auch Webseiten „in der Nähe“ von Elementen aus $\boldsymbol{S}$ werden höher gewichtet

Zusammenstellen der Menge *S*

- Verwendung der Seiten eines Webverzeichnisses, z.B. Curlie (DMOZ)
- Erweiterung mit *ähnlichen* Webseiten unter Verwendung *typischer* Worte
  - Typische Wörter = Wörter, die generell selten aber relative häufig in einem Dokument eines bestimmten Themas vorkommen
  - z.B. über Tf-idf-Maß eines Wortes i in Dokument j: $TF.IDF_{ij} = TF_{ij} \cdot IDF_i$
    - Absolute Häufigkeit f_{ij} eines Wortes i in Dokument j
    - Relative Häufigkeit des Wortes i in Dokument j: $TF_{ij} = \frac{f_{ij}}{\max_k f_{kj}}$
    - Sei n_i die Anzahl der insgesamt N Dokumente, die das Wort i enthalten
    - Inverse Dokumenthäufigkeit: $IDF_i = \log_2 \frac{N}{n_i}$
- Auch der Kontext bzw. die Interessen eines Nutzers können über diese Methode erschlossen werden:
  - Wörter in vergangenen Anfragen
  - Wörter auf Seiten mit Bookmark
  - Wörter auf aktueller Seite (von wo Anfrage gesendet)

Inhaltsverzeichnis

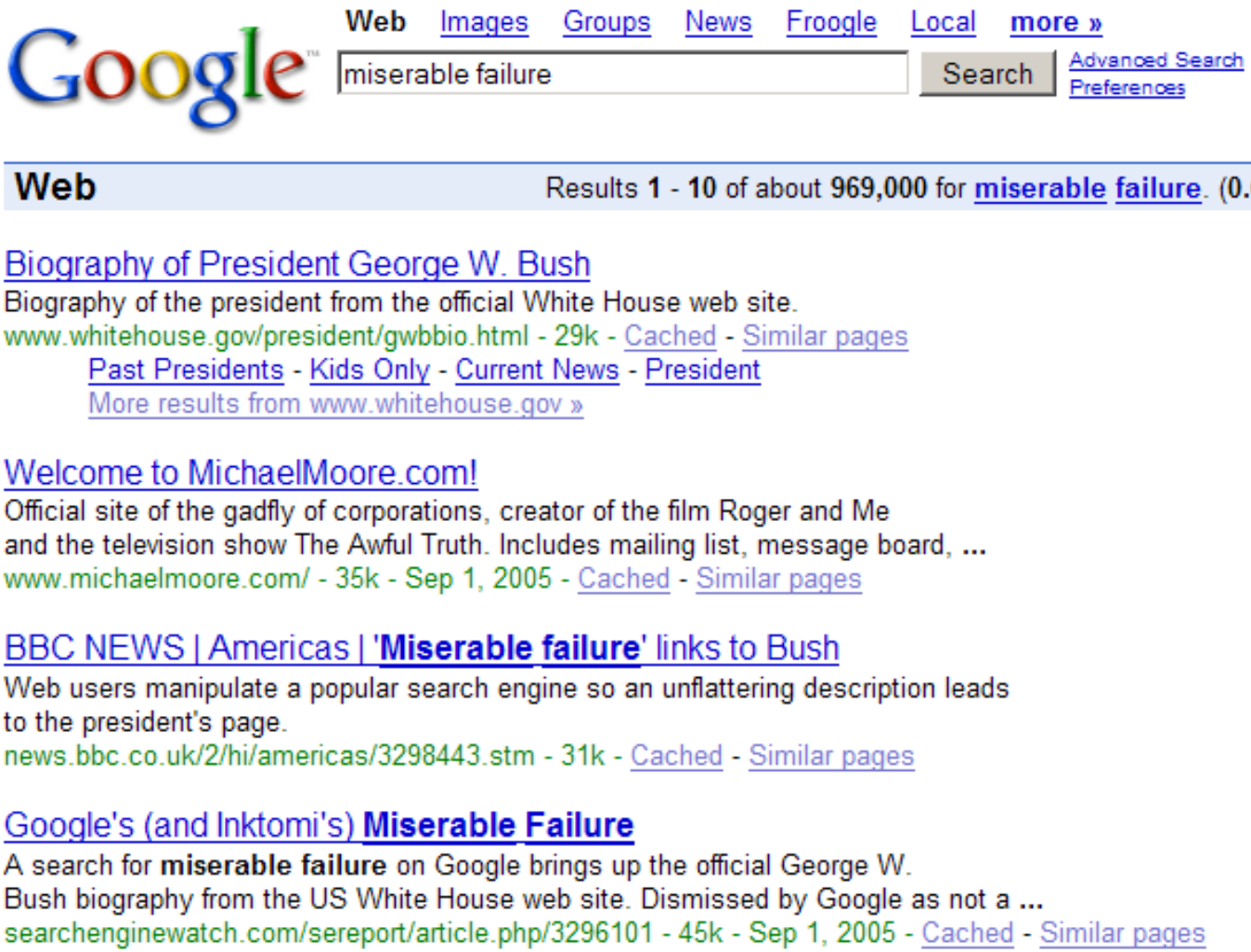
- **Einführung**
- **PageRank**
  - Probleme
  - Effiziente Berechnung
  - Themenspezifischer PageRank
- **Link-Spam**
- **Übungen**

Literatur: Kapitel 5 aus „Mining of Massive Datasets“: http://www.mmds.org/

Web Spamming

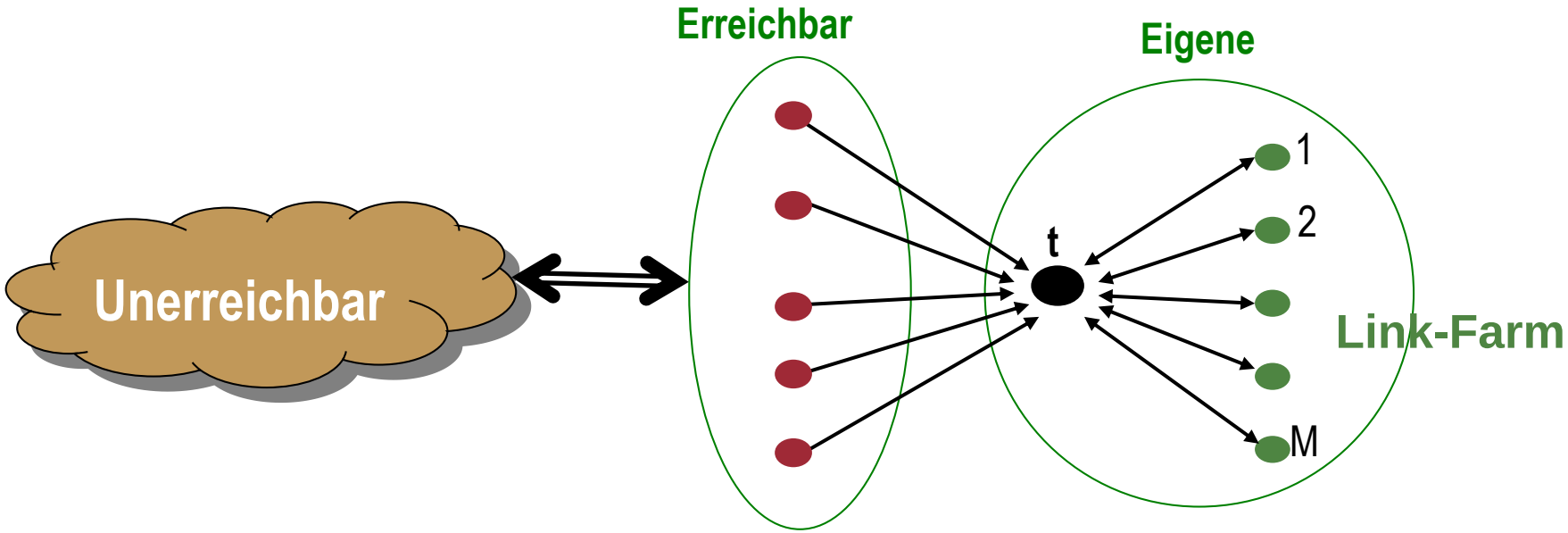
- **Web Spamming**: alle Unternehmungen um eine Webseite im Ranking einer Suchmaschine zu verbessern ***ohne*** die „Bedeutung“ der Seite für eine Suchanfrage zu erhöhen
- **Term-Spamming**: Manipulation des Inhalts einer Webseite
  - Beispiel: ein Verkäufer von T-Shirts lässt seine Webseite so aussehen, als ob interessante Information zu Sport/Filmen zu finden sind
  - Einfügen von relevanten Wörtern bzw. einer vollständigen Kopie einer relevanten Webseite und ändern der Textfarbe auf Hintergrundfarbe
- Lösung von Suchmaschinen: Anstelle der Textes einer Webseite, Verwendung des Textes in und um Verlinkungen zu einer Webseite
- PageRank verhindert den Versuch diese Lösung zu umgehen, indem man tausende irrelevante Seiten mit Verlinkungen auf Zielseite erstellt
- **Link-Spamming:** Erstellen einer Struktur von Verlinkungen um PageRank zu erhöhen

Google-Bombe (2006)



Link-Spamming

- Aus Sicht des Spammer: 3 Arten von Webseiten
  - Unerreichbare Seiten
  - Erreichbare/editierbare Seiten (z.B. über Kommentare auf Blogs/Nachrichtenseiten)
  - Eigene Seiten
- Vorgehen:
  - Möglichst viele Verlinkungen von erreichbaren Seiten zur Zielseite **t**
  - Erstellen einer **Link-Farm** um PageRank weiter zu erhöhen



Funktionsweise der Link-Farm

- PageRank-Gewicht aus den erreichbaren Seiten: $\boldsymbol{x}$
- PageRank der Zielseite t: $\boldsymbol{y}$
- PageRank einer der insgesamt M Farm-Seiten: $\frac{\beta \boldsymbol{y}}{M} + 1 - \beta$
- Dann:

$$\boldsymbol{y} = \boldsymbol{x} + \beta M \left[\frac{\beta \boldsymbol{y}}{M} + 1 - \beta\right] + 1 - \beta \; = \boldsymbol{x} + \beta^2 \boldsymbol{y} + \beta(1-\beta)M + 1 - \beta$$

$$\Rightarrow \boldsymbol{y} = \boxed{\frac{x}{1-\beta^2}} + \boxed{\frac{\beta}{1+\beta} M} + \boxed{\frac{1}{(1+\beta)}}$$

Vervielfachung von x

Beliebig großes y durch großes M

Vernachlässigbar

TrustRank

- Google erkennt das klassische Design einer Link-Farm und wird die Webseiten aus dem Index entfernen
  - Außerdem versucht Google neue Designs zu erkennen
  - *Krieg* zwischen Google und Spammer

- Alternative: Verwendung des **TrustRank:** Themenspezifischer PageRank mit einer Auswahl vertrauenswürdiger Seiten als Menge $\boldsymbol{S}$
  - **Annahme:** Vertrauenswürdige Seiten verlinken selten auf Spam
  - Vertrauen in beliebige Seite nimmt ab mit Entfernung zu vertrauenswürdiger Seite
  - Vertrauen wird über alle Verlinkungen aufgeteilt da viele Verlinkungen oft mit einer ungenaueren Prüfung dieser einhergeht
- Auswahl der vertrauenswürdigen Seiten (ohne menschliche Inspektion):
  - Seiten mit sehr hohem PageRank
  - Verwendung vertrauenswürdiger Domains, z.B. .edu, .gov, …

Inhaltsverzeichnis

- **Einführung**
- **PageRank**
  - Probleme
  - Effiziente Berechnung
  - Themenspezifischer PageRank
- **Link-Spam**
- **Übungen**

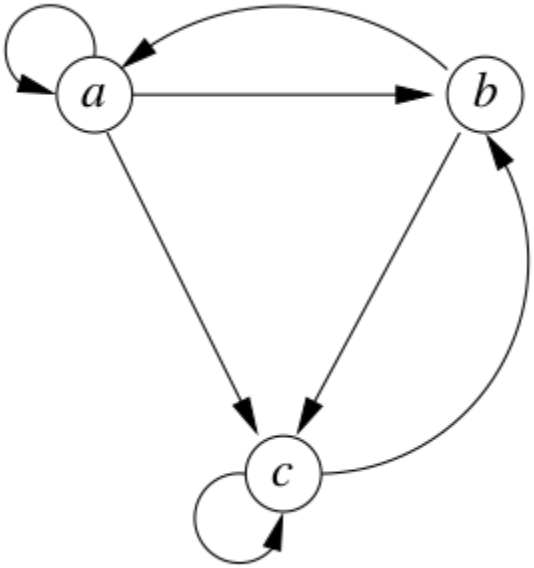
Literatur: Kapitel 5 aus „Mining of Massive Datasets“: http://www.mmds.org/

Übung 1

Berechnen Sie den PageRank für das abgebildete Netzwerk!

a) Setzen Sie $\beta = 1$.

b) Setzen Sie $\beta = 0.8$.

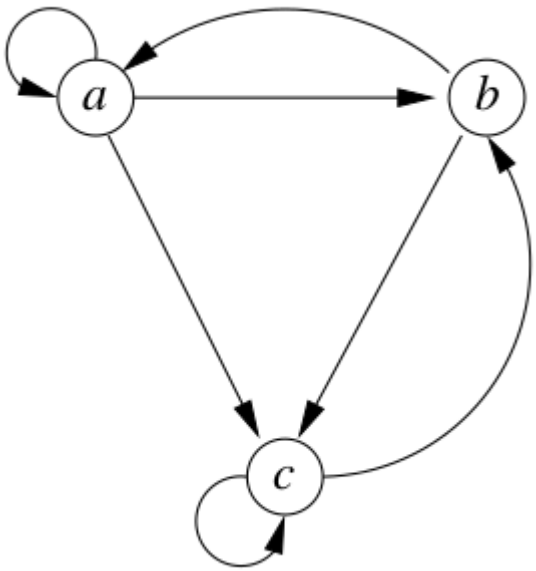


Hinweis: Für dieses kleine Netzwerk muss der PageRank nicht über die Iterationsmethode approximiert werden. Sie können das Gleichungssystem direkt lösen, indem Sie einen Wert festsetzen (z.B. $r_a = 1$).

Übung 1: Lösung

Berechnen Sie den PageRank für das abgebildete Netzwerk!

a) $\beta = 1$



$$r_a = \frac{1}{3} r_a + \frac{1}{2} r_b$$
$$r_b = \frac{1}{3} r_a + \frac{1}{2} r_c$$
$$r_c = \frac{1}{3} r_a + \frac{1}{2} r_b + \frac{1}{2} r_c$$

$$\downarrow \quad r_a = 1$$

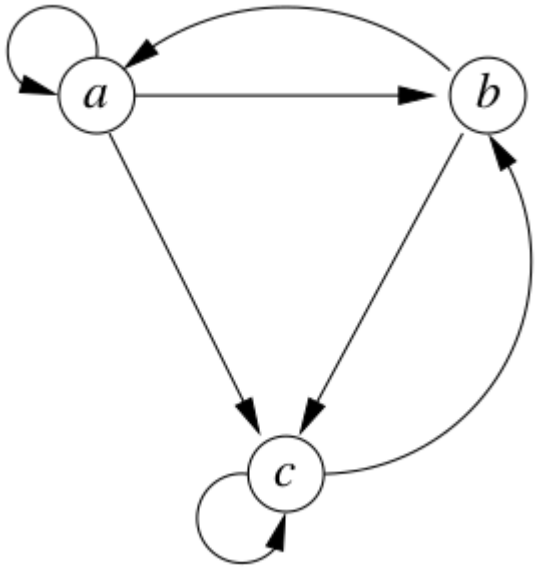
$$r_b = \frac{4}{3}$$
$$r_c = 2$$

Übung 1: Lösung

Berechnen Sie den PageRank für das abgebildete Netzwerk!

b) $\beta = 0.8$

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



$$r_a = \frac{1}{3} r_a + \frac{7}{15} r_b + \frac{1}{15} r_c$$
$$r_b = \frac{1}{3} r_a + \frac{1}{15} r_b + \frac{7}{15} r_c$$
$$r_c = \frac{1}{3} r_a + \frac{7}{15} r_b + \frac{7}{15} r_c$$

$$r_a = 1$$
$$r_b = \frac{25}{21}$$
$$r_c = \frac{5}{3}$$

Übung 2: Lösung

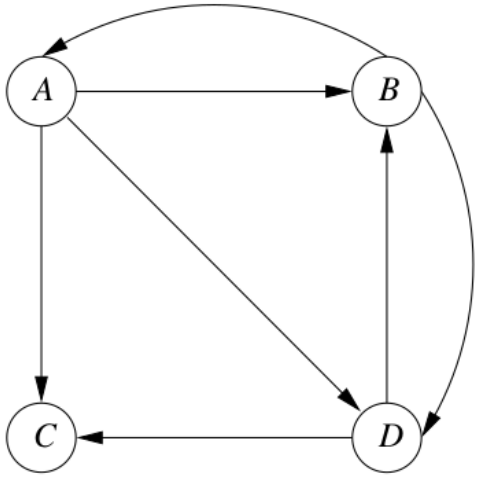
a) Angenommen Sie wollen eine Adjazenzmatrix (Zellen entweder 0 oder 1) speichern. Anstatt der Repräsentation als Matrix könnten die Einträge auch über eine Liste von Paaren gespeichert werden (die Paare bezeichnen die Indizes der Einträge mit einer Eins). Wie spärlich müsste die Adjazenzmatrix mit Einsen besetzt sein, damit die Repräsentation über eine Liste effizienter als die Matrixrepräsentation ist?

- Bit-Repräsentation: n^2 Bit Speicherplatz
- Anzahl der Einsen: e
- Spärliche Repräsentation: $e \cdot 2 \cdot \log_2 n$Bit
- Liste ist effizienter, falls $e \cdot 2 \cdot \log_2 n < n^2$ bzw. $e < \frac{n^2}{2 \cdot \log_2 n}$

n	$\frac{n^2}{2 \cdot \log_2 n}$
100	752
1 000	50 172
10 000	3 762 875
100 000	301 029 996

Übung 2: Lösung

b) Stellen Sie die stochastische Adjazenzmatrix des abgebildeten Netzwerks als Blöcke der Größe 2x2 dar!



A	3	B
B	2	A

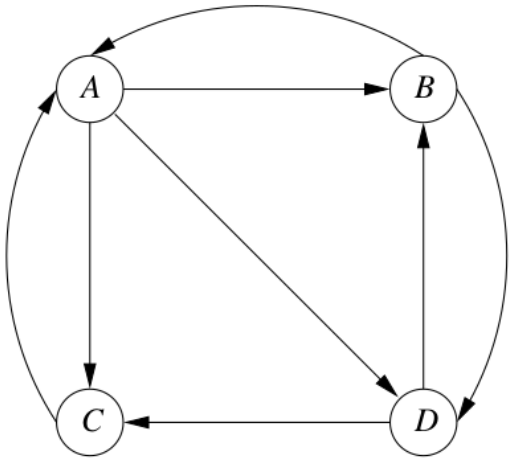
A	3	C,D
B	2	D

C	0	
D	2	B

C	0	
D	2	C

Übung 3: Lösung

Berechnen Sie den themenspezifischen PageRank für das abgebildete Netzwerk mit $S = \{A\}$ und $\beta = 0.8$!



M

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

$$= \begin{bmatrix} 1/5 & 3/5 & 1 & 1/5 \\ 4/15 & 0 & 0 & 2/5 \\ 4/15 & 0 & 0 & 2/5 \\ 4/15 & 2/5 & 0 & 0 \end{bmatrix}$$

$r_A = 1$

$$r_B = r_C = r_D = \frac{4}{9}$$