Informationssysteme

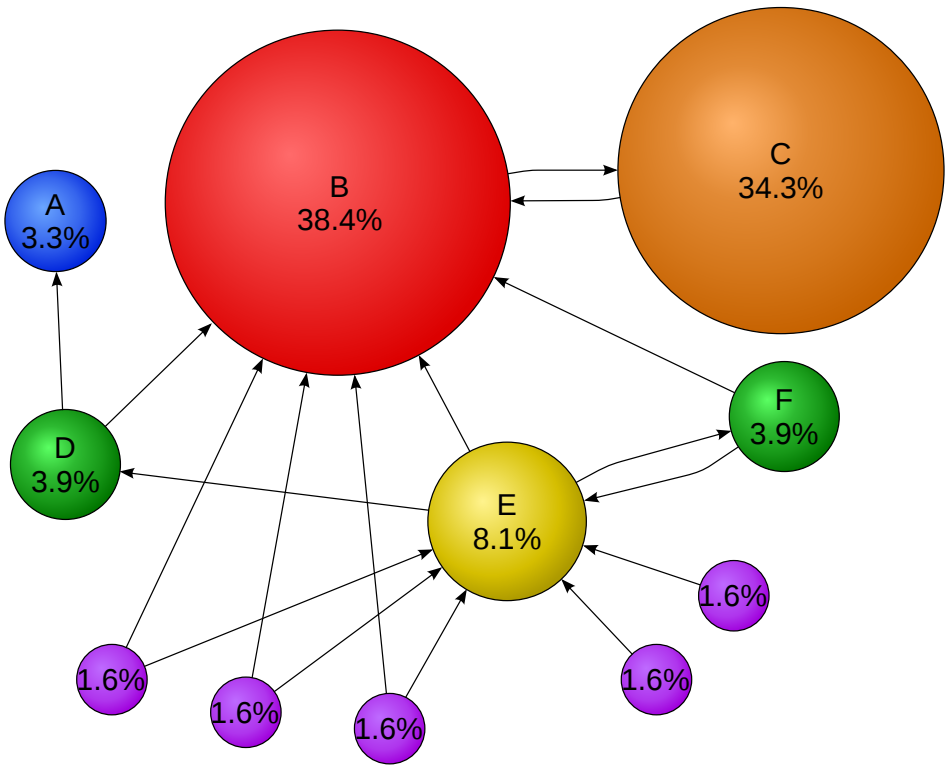
Sommersemester 2016

Prof. Dr.-Ing. Sebastian Michel
TU Kaiserslautern

smichel@cs.uni-kl.de

PageRank

- **PageRank sagt, dass eine Webseite v wichtig ist, wenn viele wichtige Seiten auf v verweisen**
- Beschrieben in einem Papier von **Brin & Page aus 1998**: **The PageRank Citation Ranking: Bringing Order to the Web** `http://ilpubs.stanford.edu:8090/422/1/1999-66.pdf`.



Quelle der Abbildung Wikipedia

PageRank

- **PageRank sagt, dass eine Webseite v wichtig ist, wenn viele wichtige Seiten auf v verweisen**

Random-Surfer-Modell

- **folgt zufällig den Links, die von einer Seite ausgehen**, mit Wahrscheinlichkeit $(1 - \varepsilon)$ oder
- **springt zu einer zufällig ausgewählten Seite**, mit Wahrscheinlichkeit ε

Intuition: Wichtige Webseiten werden häufiger besucht.

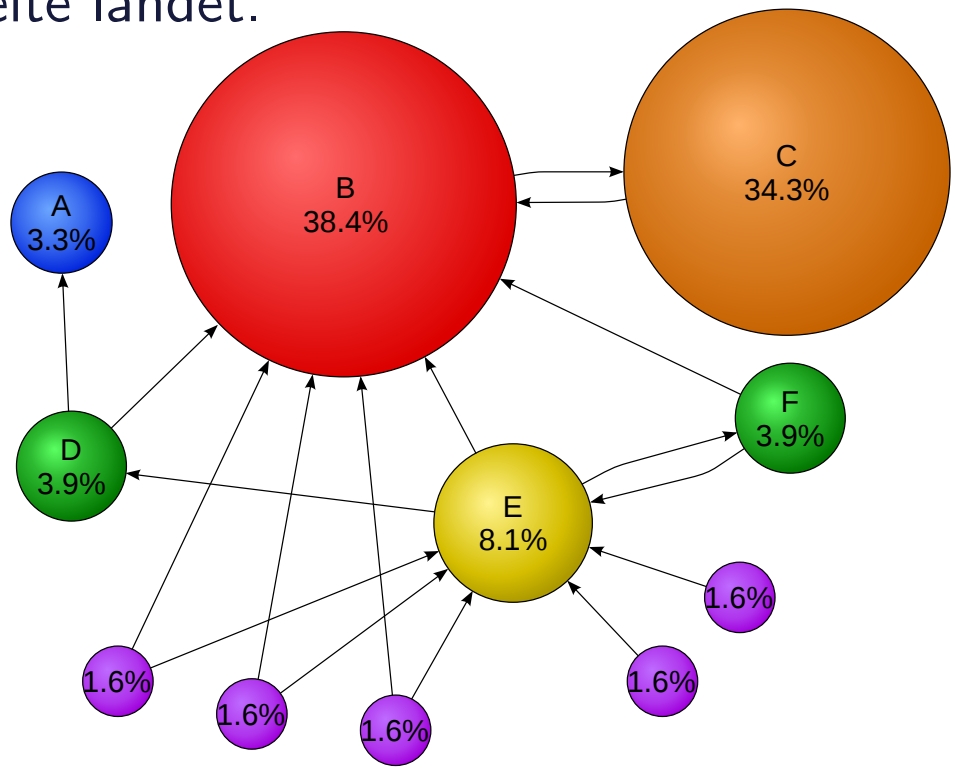
$$\mathrm{PR}(p) = (1 - \varepsilon) \times \sum_{q \to p} \frac{\mathrm{PR}(q)}{out(q)} + \varepsilon \times \frac{1}{N}$$

Schreibweise: $\sum_{q \to p}$, summiert über alle Knoten q, die auf p verweisen.

$out(q)$ ist die Anzahl der von q ausgehenden Links ("outdegree")

$$\mathrm{PR}(p) = (1-\varepsilon) \times \sum_{q \to p} \frac{\mathrm{PR}(q)}{out(q)} + \varepsilon \times \frac{1}{N}$$

PageRank beschreibt die Wahrscheinlichkeitsverteilung, dass ein Random Surfer auf einer Seite landet.

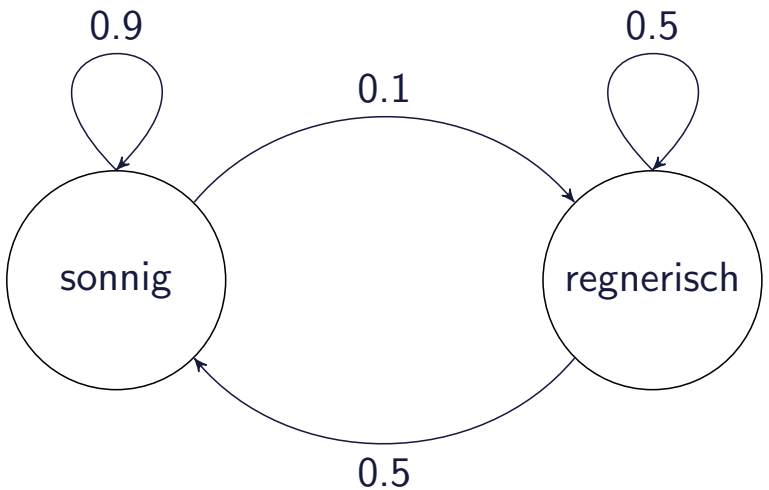


Verwandte Frage: Wie hoch ist die Wahrscheinlichkeit, dass ein Random Surfer auf Seite X landet, wenn er vorher auf Seite Y war?

Quelle der Abbildung: Wikipedia

Markov-Ketten: Wetter Beispiel

Wie wird das Wetter morgen sein, wenn es heute regnet?



$$\mathbf{P} = \begin{pmatrix} 0.9 & 0.1 \\ 0.5 & 0.5 \end{pmatrix}$$

$(P)_{ij}$ ist die Wahrscheinlichkeit, dass der folgende Tag vom Typ j ist, wenn der heutige Tag vom Typ i ist.

Sagen wir zu Beginn haben wir einen sonnigen Tag: $x^{(0)} = \begin{pmatrix} 1 & 0 \end{pmatrix}$
Dann ist das Wetter am nachfolgenden Tag:

$$x^{(1)} = x^{(0)}P = \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} 0.9 & 0.1 \\ 0.5 & 0.5 \end{pmatrix} = \begin{pmatrix} 0.9 & 0.1 \end{pmatrix}$$

Stochastischer Prozess & Markov-Ketten

- **Ein disktreter stochastischer Prozess ist eine Familie von Zufallsvariablen**

$$\{X_t | t \in T\}$$

wobei $T = \{0, 1, 2, \ldots\}$ die diskrete Zeit-Domäne ist.

- Ein stochastischer Prozess ist eine **Markov-Kette** falls gilt

$$P[X_t = x | X_{t-1} = w, \ldots, X_0 = a] = P[X_t = x | X_{t-1} = w]$$

D.h., der Prozess ist **gedächtnislos** (English: memoryless).

- Eine Markov-Kette ist **zeithomogen**, falls für alle Zeitpunkte t

$$P[X_{t+1} = x | X_t = w] = P[X_t = x | X_{t-1} = w]$$

gilt, d.h. die **Übergangswahrscheinlichkeiten hängen nicht von der Zeit ab**.

Zustandsraum & Übergangswahrscheinlichkeitsmatrix

- **Zustandsraum** einer Markov-Kette $\{X_t | t \in T\}$
- Markov-Kette ist in Zustand s zur Zeit t falls $X_t = s$
- Markov-Kette $\{X_t | t \in T\}$ ist endlich, falls der Zustandsraum **endlich** ist
- Falls eine Markov-Kette $\{X_t | t \in T\}$ endlich und zeithomogen ist, so können ihre **Zustandsübergangswahrscheinlichkeiten** durch eine Matrix $P = (p_{ij})$ mit

$$P = (p_{ij}) \text{ mit } p_{ij} = P[X_t = j | X_{t-1} = i]$$

  beschrieben werden.
- Für n Zustände ist die **Zustandsübergangsmatrix** P eine $n \times n$ zeilenstochastische **Matrix** (d.h. ihre Zeilen summieren sich auf zu 1):

$$\forall i : \sum_j p_{ij} = 1$$

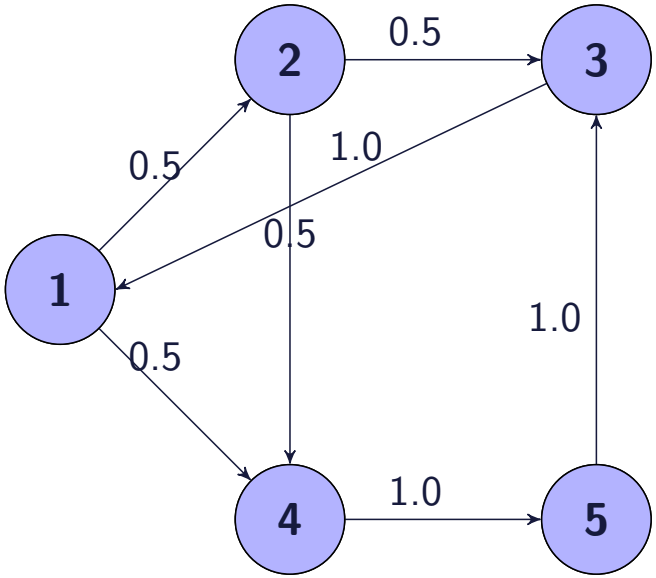
Eigenschaften von Markov-Ketten

Es gibt noch ein paar weitere Eigenschaften, die eine Markov-Kette haben kann, auf die wir aber in dieser Vorlesung nicht weiter eingehen.

Man sagt eine Markov-Kette ist **ergodisch** falls sie irreduzibel, positiv rekurrent und aperiodisch ist.

Theorem: Falls eine Markov-Kette endlich und ergodisch ist, dann existiert eine stationäre Zustandsverteilung π

Die Markov-Kette, die wir später für PageRank definieren werden hat diese Eigenschaften!



$$P = \begin{pmatrix} 0.0 & 0.5 & 0.0 & 0.5 & 0.0 \\ 0.0 & 0.0 & 0.5 & 0.5 & 0.0 \\ 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \\ 0.0 & 0.0 & 1.0 & 0.0 & 0.0 \end{pmatrix}$$

$$\pi^{(0)} = \begin{pmatrix} 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{pmatrix}$$

$$\pi^{(1)} = \begin{pmatrix} 0.0 & 0.5 & 0.0 & 0.5 & 0.0 \end{pmatrix}$$

$$\pi^{(2)} = \begin{pmatrix} 0.0 & 0.0 & 0.25 & 0.25 & 0.5 \end{pmatrix}$$

$$\pi^{(3)} = \begin{pmatrix} 0.25 & 0.0 & 0.5 & 0.0 & 0.25 \end{pmatrix}$$

$$\pi^{(4)} = \begin{pmatrix} 0.5 & 0.125 & 0.25 & 0.125 & 0.0 \end{pmatrix}$$

$$\pi^{(5)} = \begin{pmatrix} 0.25 & 0.25 & 0.0625 & 0.3125 & 0.125 \end{pmatrix}$$

$$\ldots.$$

$$\pi = \begin{pmatrix} 0.25 & 0.125 & 0.25 & 0.1875 & 0.1875 \end{pmatrix}$$

Berechnung von π mit "Power Iteration"-Methode

Idee: Berechne schrittweise Zustandsverteilungen π bis diese konvergieren

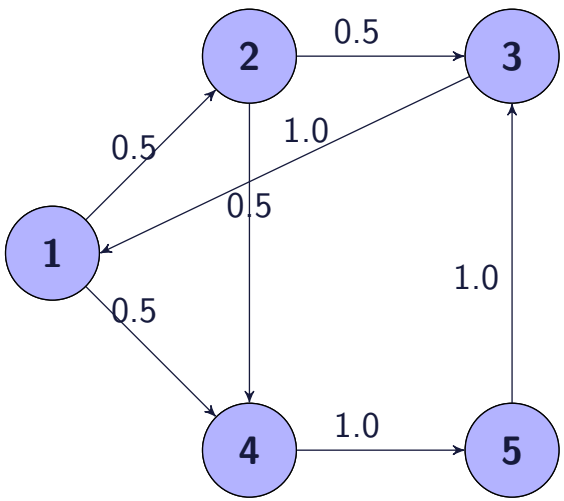
Algorithmus

- wähle initiale Zustandsverteilung $\pi^{(0)}$ mit $||\pi^{(0)}||_1 = 1$
- berechne $\pi^{(k)} = \pi^{(k-1)}P$ bis zur Konvergenz (z.B. bis $||\pi^{(k)} - \pi^{(k-1)}||_1 < \xi$ mit $\xi = 0.000001$)
- gebe die zuletzt berechnete Verteilung $\pi^{(k)}$ als stationäre Zustandsverteilung π zurück

Wir berechnen hier also einen **Linkseigenvektor**.

PageRank als Markov-Kette

- **Random-Surfer-Modell**
  - Folgt einem zufälligen ausgehenden Verweis mit Wahrscheinlichkeit $(1-\varepsilon)$
  - Springt zu einer zufälligen Webseite mit Wahrscheinlichkeit ε
- Sei $\mathbf{A}$ die Adjazenzmatrix des Webgraphen $G=(V,E)$ mit Kanten E und Knoten V.



$$A = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Zur Erinnerung: $\mathsf{PR}(p) = (1-\varepsilon) \times \sum_{q \to p} \frac{\mathsf{PR}(q)}{out(q)} + \varepsilon \times \frac{1}{N}$

Daher: Aus Matrix $\mathbf{A}$ erzeugen wir Matrix $\mathbf{T}$ indem wir die Kantengewichte 1 durch die Anzahl der ausgehenden Links teilen, also

$$(\mathbf{T})_{ij} = \begin{cases} 1/out(i) & \text{falls } (i,j) \in E \\ 1/|V| & \text{falls } out(i) = 0 \\ 0 & \text{sonst} \end{cases}$$

$$\mathbf{T} = \begin{pmatrix} 0 & 1/2 & 0 & 1/2 & 0 \\ 0 & 0 & 1/2 & 1/2 & 0 \\ 1/1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/1 \\ 0 & 0 & 1/1 & 0 & 0 \end{pmatrix}$$

Nun fügen wir noch die "random jumps" hinzu....

Zur Erinnerung: $\mathrm{PR}(p) = (1-\varepsilon) \times \sum_{q \to p} \frac{\mathrm{PR}(q)}{out(q)} + \varepsilon \times \frac{1}{N}$

Also haben wir für unser Beispiel mit 5 Seiten für $\varepsilon = 0.15$

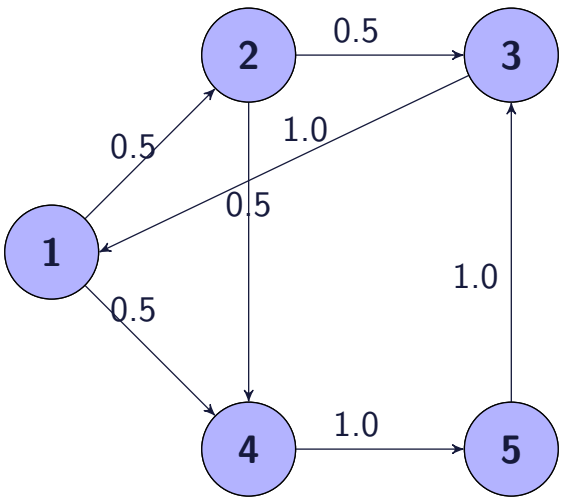
$$\mathbf{P} = (1-\varepsilon)\mathbf{T} + \varepsilon \begin{pmatrix} 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \end{pmatrix}$$

oder allgemein ausgedrückt

$$\mathbf{P} = (1-\varepsilon)\mathbf{T} + \varepsilon[1 \ldots 1]^T \mathbf{j}$$

mit $(\mathbf{j})_i = 1/|V|$

Wir erhalten also für unseren Webgraphen die Matrix P



$$P = \begin{pmatrix} 0.03 & 0.455 & 0.030 & 0.455 & 0.03 \\ 0.03 & 0.030 & 0.455 & 0.455 & 0.03 \\ 0.88 & 0.030 & 0.030 & 0.030 & 0.03 \\ 0.03 & 0.030 & 0.030 & 0.030 & 0.88 \\ 0.03 & 0.030 & 0.880 & 0.030 & 0.03 \end{pmatrix}$$

Wir beginnen mit Startvektor $p^{(0)} = (0.2, 0.2, 0.2, 0.2, 0.2)$ und berechnen $p^{(k)} = p^{(k-1)}P$ bis $||p^{(k)} - p^{(k-1)}||_1 \leq 0.0001$

Bemerkungen

$$(0.2, 0.2, 0.2, 0.2, 0.2) * \begin{pmatrix} 0.03 & 0.455 & 0.030 & 0.455 & 0.03 \\ 0.03 & 0.030 & 0.455 & 0.455 & 0.03 \\ 0.88 & 0.030 & 0.030 & 0.030 & 0.03 \\ 0.03 & 0.030 & 0.030 & 0.030 & 0.88 \\ 0.03 & 0.030 & 0.880 & 0.030 & 0.03 \end{pmatrix}$$

Für Webseite 1 haben wir also nach dieser Iteration den neuen Wert im ersten Eintrag des Vektors berechnet durch
$0.2 \times 0.03 + 0.2 \times 0.03 + 0.2 \times 0.88 + 0.2 \times 0.03 + 0.2 \times 0.03$
da in **Spalte 1** der Matrix (rot markiert) steht wie viel Anteil des PR-Wertes der einzelnen Seiten auf Seite 1 propagiert wird.

Die Summe der Zeilen von P ist 1, d.h. P ist zeilenstochastisch.
Man kann auch P^T betrachten und $p^{(k)^T} = P^T p^{(k-1)^T}$ berechnen. P^T ist entsprechend spaltenstochastisch.

```
T<-matrix(c(0,0.5,0,0.5,0,0,0,0.5,0.5,0,1,0,0,
0,0,0,0,0,0,1,0,0,1,0,0), nrow=5, ncol=5, byrow=TRUE)

#die Matrix für die zufälligen Sprünge
R <- c(rep(1/5,5)) %*% (t(c(rep(1,5))))
eps <- 0.15        #random jump Wahrscheinlichkeit
P <- (1-eps)*T + (eps)*R    #die Matrix P
p <- c(rep(1/5,5))      #so geht es los

stop <- 0.0001    #stop wenn kaum noch Aenderung
diff <- 1
p_old = p
while (diff>=stop) {
    p <- p_old %*% P
    diff <- (sum(abs(p_old-p)))
    p_old <-p
}
p    #finalen Vektor ausgeben
```

$$p^{(0)} = (0.2, 0.2, 0.2, 0.2, 0.2)$$
$$p^{(1)} = (0.2, 0.115, 0.285, 0.2, 0.2)$$
$$p^{(2)} = (0.2723, 0.115, 0.2489, 0.1639, 0.2)$$
$$p^{(3)} = (0.2415, 0.1457, 0.2489, 0.1946, 0.1693)$$
$$p^{(4)} = (0.2415, 0.1327, 0.2358, 0.1946, 0.1954)$$
$$p^{(5)} = (0.2305, 0.1327, 0.2525, 0.189, 0.1954)$$
$$p^{(6)} = (0.2446, 0.1279, 0.2525, 0.1843, 0.1907)$$
$$p^{(7)} = (0.2446, 0.134, 0.2465, 0.1883, 0.1867)$$
$$p^{(8)} = (0.2395, 0.134, 0.2456, 0.1909, 0.1901)$$
$$p^{(9)} = (0.2388, 0.1318, 0.2485, 0.1887, 0.1923)$$
$$\ldots$$
$$p^{(19)} = (0.2409, 0.1323, 0.248, 0.1886, 0.1902)$$
$$p^{(20)} = (0.2408, 0.1324, 0.2479, 0.1886, 0.1903)$$
$$p^{(21)} = (0.2407, 0.1323, 0.248, 0.1886, 0.1903)$$

PageRank und Anfragen

PageRank berechnet ein statisches Ranking von Webseiten und ist unabhängig von Anfragen.

Eine Möglichkeit PageRank und "TF*IDF" Scores zu kombinieren ist eine einfache Linearkombination:

$$\alpha \times sim(q,d) + (1 - \alpha) \times \mathsf{PR}(d)$$

Zusammenfassung PageRank

- Idee hinter PageRank: Webseiten haben verschiedenes Maß an Autorität
- Webseiten sollten nicht nur anhand der Nähe zur Anfrage sortiert werden sondern auch anhand der Autorität.

- PageRank modelliert einen random surfer, der zufällig das Web durchstöbert, dabei zufällig Links folgt oder zufällig irgendwelche URLS eingeben kann (random jumps).
- Autorität einer Webseite ist dann die Wahrscheinlichkeit, dass dieser Surfer auf der Seite landet.
- Das Verhalten des Surfers wird als Markov-Kette beschrieben.
- Man kann zeigen, dass diese eine stationäre Zustandsverteilung besitzt, also konvergiert (Details haben wir nicht angeschaut).
- Durch z.B. Power-Iteration kann man diese Verteilung berechnen