PageRank-Algorithmus

Benedikt Wolters

Proseminar Algorithms and Data Structures

Gliederung

Motivation

Wir wollen eine Suchmaschine bauen, die das Web durchsucht.

Viele Probleme, darunter:

- Welche Suchergebnisse sind wichtiger als andere?
- Welches Maß gibt es, um die Wichtigkeit einer Webseite zu bestimmen?

naiver Ansatz

Idee: Wir benutzen als Maß für die Wichtigkeit einer Seite die absolute Häufigkeit der Vorkommen eines Schlüsselworts.

- wurde tatsächlich Anfang der neunziger Jahren so gemacht: vgl. Altavista, WebCrawler, World Wide Web Worm

Nachteile:

- **skaliert schlecht** mit zunehmender Größe des Webs
  Altavista bereits 1997 ca. 20 Mio. Anfragen pro Tag
  letzte bekannte Größe des Google Index 1 Billion (7/2008)
- **Junk-Ergebnisse, Werbung, Spam:**
  Suchergebnisse leicht zu manipulieren
  Hintergrund: steigendes Wachstum des Internets
  (akademisches Netz →kommerzielle Marketingplattform)

Wichtigkeit

- Wir suchen also ein möglichst universelles Kriterium, um die Wichtigkeit einer Seite zu bestimmen.
- **Aber:** Wichtigkeit ist subjektiv!
  Wir werden nie ein universell-objektives Kriterium für eine einzelne Seite finden.
- **Idee:** Betrachte die »**Stellung**« einer Webseite im Gesamtkontexts des Webs und lasse die Webseiten gegenseitig über ihre Wichtigkeit abstimmen.
- Dabei spielt der eigentliche Inhalt einer Seite eine untergeordnete Rolle.

Das Web als Graph

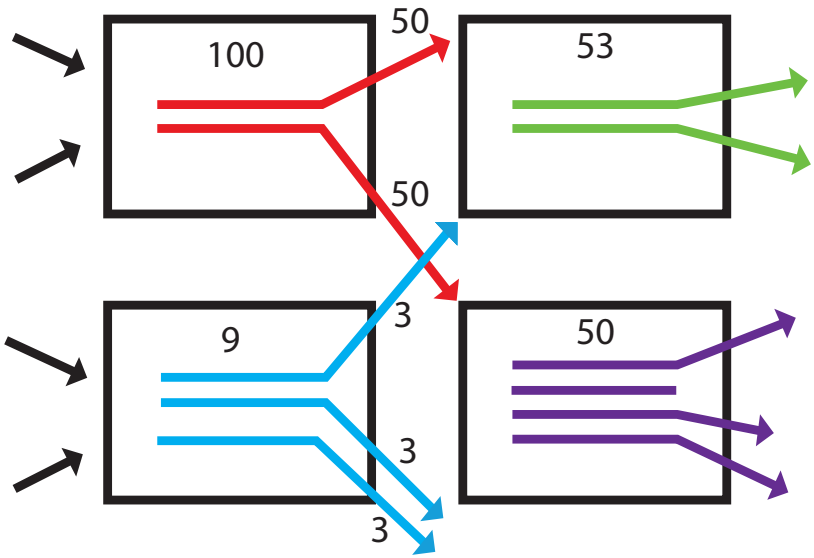
Wir betrachten das Web (bzw. Auschnitt/Schnappschuss) als gerichteten Graphen $G = (V, E)$

- Die Knoten $k \in V$ sind Seiten.
- Die Kanten sind Hyperlinks.
- $out(k)$ als den Ausgangsgrad (# der Seiten auf die k verlinkt)
- $in(k)$ als den Eingangsgrad (# der Seiten, die auf k verlinken)

Links als Stimmen 1

- Erster Versuch:
  - eine Seite ist wichtiger, wenn sie mehr Eingangslinks hat, d.h. wenn $in(k)$, $k \in V$ entsprechend groß ist.
- Betrachte Links als Stimmen:
  - `www.rwth-aachen.de` hat 12.365 Verlinkungen
  - `www.fh-aachen.de` hat 1.514 Verlinkungen (Alexa, 31.01.2012)
- Sind Links gleich »wichtig«?
  - Links von **wichtigen Seiten** zählen mehr.
  - Aber wer sind die wichtigen Seiten? **Rekursive Frage!**

Links als Stimmen 2



- Jede Stimme eines Links ist proportional zu der Wichtigkeit seiner Ausgangsseite.
- Wenn eine Seite k eine Anzahl von n Links hat, erhält jeder Link x/n Stimmen, wobei x die Wichtigkeit von k ist.
- Die Wichtigkeit von k ist die Summe der Stimmen seiner Eingangslinks.

1. Definition PageRank (fehlerhaft)

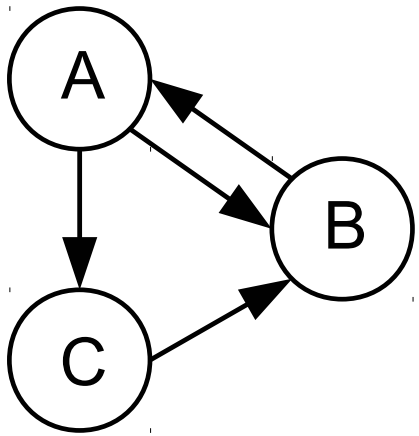
Für eine Seite k wird der PageRank $P(k)$ wie folgt definiert:

$$P(k) = \sum_{i \in L(u)} \frac{P(i)}{N_i}$$

wobei:

- $L(k)$ die Menge der Seiten ist, die einen Link auf k haben, und
- N_i die Anzahl der Seiten ist, auf die die Seite i verlinkt.

Beispiel



$$
\begin{aligned}
P(A) &= 0 + 1 \cdot P(B) + 0 \\
P(B) &= \tfrac{1}{2} \cdot P(A) + 0 + 1 \cdot P(C) \\
P(C) &= \tfrac{1}{2} \cdot P(A) + 0 + 0
\end{aligned}
$$

Internet mit 3 Seiten

- Algebraische Lösungsmethoden (Gauß über LR-Zerlegung etc.) funktionieren für kleine N ($N = |V|$),
  aber: $\mathcal{O}(N^3)$ und der Web Graph ist sehr groß!
- $\rightarrow$ Wir brauchen also einen besseren Lösungsweg!

Formulierung als Matrix

$$A \in \mathbb{R}^{N \times N}, A_{ij} = \begin{cases} \frac{1}{N_j} & \text{, falls } j \text{ auf } i \text{ verlinkt} \\ 0 & \text{, sonst} \end{cases}$$

Wir nennen A **Linkmatrix.**
A hat eine Zeile und eine Spalte für je eine Webseite.
N_j Anzahl der Seiten, auf die die Seite j verlinkt.

Durch die Konstruktion von A folgt:

$$\sum_{i=1}^{N} A_{ij} = 1,$$

$$\text{mit } 1 \leq j \leq N \text{ und } A_{ij} \geq 0 \text{ für alle } i, j.$$

So eine Matrix heißt **Markov-Matrix.**

Formulierung als Matrix

Weiter sei $v \in \mathbb{R}^N$ ein Vektor mit je einem Eintrag pro Seite.

- v_i ist die Wichtigkeit eines Seite i
- Wir nennen v den PageRank-Vektor.
- $\|v\|_1 = \sum_{i=1}^{N} |v_i| = 1$.

Formulierung als Matrix

- Wir können unser Gleichungssystem für den PageRank aller Seiten also auch wie folgt schreiben:

$$v = Av$$

- v ist also ein Eigenvektor zum Eigenwert 1.
- Jede Markov-Matrix $M \in \mathbb{R}^{N \times N}$ hat einen nicht negativen Eigenvektor zum Eigenwert 1.

Random Walk Interpretation

- Wir stellen uns einen Zufallssurfer vor.
- Zum Zeitpunkt t ist dieser auf einer Seite p.
- Zum Zeitpunkt $t+1$ folgt der Surfer zufällig einem Link von p und landet auf einer Seite q.
- Sei $p(t) \in \mathbb{R}^N$ ein Vektor und $p(t)_i$ sei die Wahrscheinlichkeit, dass der Surfer zum Zeitpunkt t auf Seite i ist.
- $p(t)$ ist eine also Wahrscheinlichkeitsverteilung von Seiten.

Stationäre Verteilung

- Wo ist der Surfer zum Zeitpunkt $t+1$?
  - Er folgt immer zufällig einem Link.
  - $p(t+1) = A \cdot p(t)$, A ist unsere Linkmatrix
- Gibt es einen Zustand, sodass: $p(t+1) = A \cdot p(t) = p(t)$?
  - Falls ja, nennen wir einen solchen Zustand $p(t)$ **Stationäre Verteilung.**
- Unser PageRank-Vektor v erfüllt diese Bedingung.
  - Ist v eine Stationäre Verteilung für den Zufallssurfer?
  - Wir wissen bereits, dass ein solcher Vektor existiert. **Aber ist er auch eindeutig?**

Perron-Frobenius-Theorem

Perron-Frobenius-Theorem

Ist M eine primitive Markov-Matrix, so ist der Eigenvektor $x = (x_1, \ldots, x_n)^T$ mit $\sum_{i=1}^{n} x_i = 1$ zum Eigenwert 1 **eindeutig bestimmt** und positiv. Weiter **existiert** $\lim_{t\to\infty} p(0) \cdot A^t = x$ und es gilt $M \cdot x = x$.

Was bedeutet das nun?
Wir können den PageRank-Vektor v approximieren, da wir wissen, dass ein Grenzwert existiert.

Power-Method

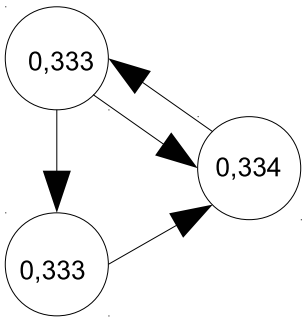
- Wir wissen also, dass ein Grenzwert $\lim_{t\to\infty} p(0)A^t = x$ existiert.
- Wir können diesen also approximieren:

Power-Method

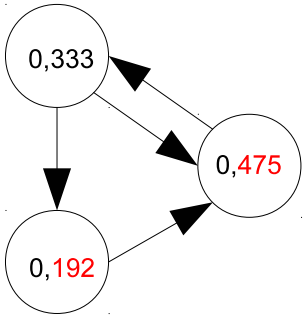
1. Initialisiere $v_0 = (\frac{1}{N}, \ldots \frac{1}{N})^T$
2. Iteriere $v_{t+1} = A \cdot v_t$
3. Stoppe, wenn $\| v_{t+1} - v_t \|_1 < \epsilon$
   - $\| x \|_1 = \sum_{1 \le i \le N} | x_i |$ ist die 1-Norm
   - $\epsilon > 0$ ist die gewünschte Genauigkeit

Beispiel

Startverteilung

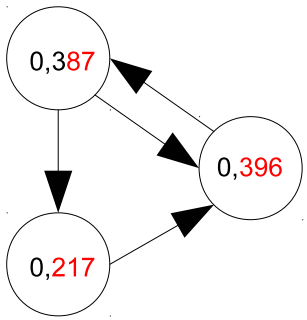


1. Iteration

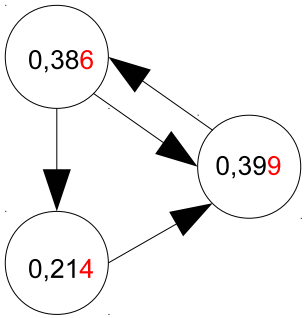


...

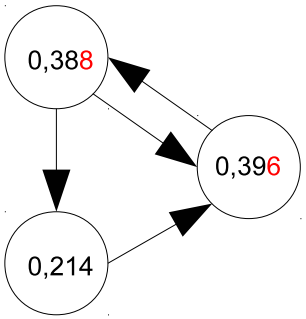
8. Iteration



9. Iteration

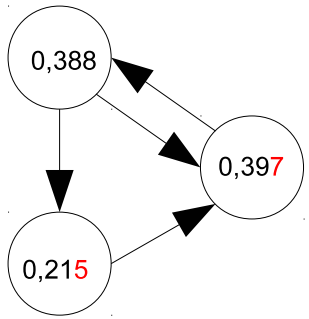


10. Iteration



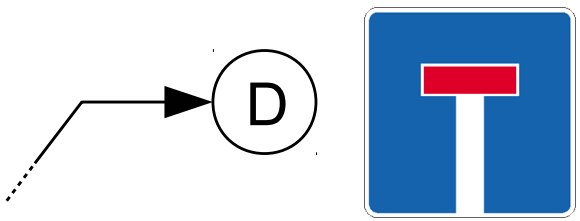
...

50. Iteration



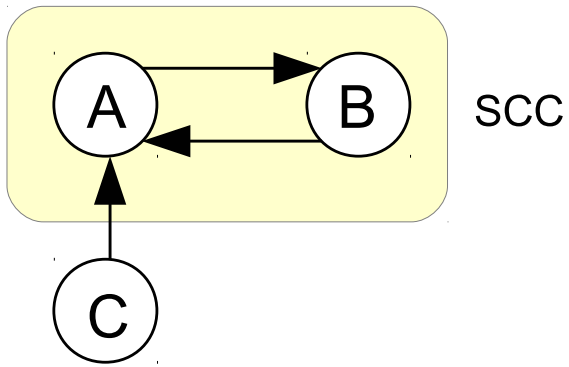
Probleme des Modells

Dead Ends



- Seiten, die nicht weiter verlinken
- unklar, wie der PageRank von dort aus weiterverteilt wird

Spider Traps



- „Konsumieren“ den PageRank in einer SCC
- Seiten, die auf diese SCC verlinken, verlieren ihren PageRank

Spider Traps - Lösung: Teleportation

- Wir modifizieren die PageRank-Formel:
- Bei jedem Schritt hat der Zufallssurfer zwei Optionen:
  - Mit der Wahrscheinlichkeit d folgt er einem zufälligen Link auf der Seite, auf der er gerade ist.
  - Mit Wahrscheinlichkeit $(1 - d)$ bricht er ab und springt gleichverteilt zu einer zufälligen Seite des Web Graphen.
- Der Zufallssurfer wird also irgendwann aus einer Spider Trap hinausteleportieren

2. Definition PageRank

Definition PageRank

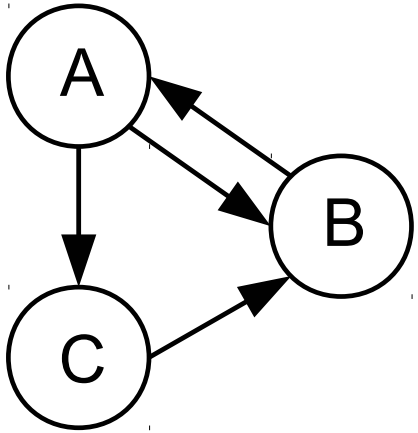
Sei u eine Webseite. Für u wird der PageRank $\tilde{P}(u)$ einer Seite u wie folgt definiert:

$$\tilde{P}(u) = \frac{1-d}{N} + d \cdot \sum_{i \in L(u)} \frac{\tilde{P}(i)}{N_i}$$

wobei:

- $L(u)$ die Menge der Seiten ist, die einen Link auf u haben, und
- N_i die Anzahl der Seiten ist, auf die die Seite i verlinkt, sowie
- N die Gesamtanzahl aller Seiten ist und
- $0 < d < 1$ ein Dämpfungsfaktor (typischerweise nahe 1, Google: $d \approx 0.85$)

modifiziertes Beispiel



Internet mit 3 Seiten

$$
\begin{aligned}
\tilde{P}(A) &= \frac{1-\mathbf{d}}{3} + \mathbf{d}\cdot\left(\quad 0 \quad + 1\cdot\tilde{P}(B) + \quad 0 \quad\right)\\
\tilde{P}(B) &= \frac{1-\mathbf{d}}{3} + \mathbf{d}\cdot\left(\tfrac{1}{2}\cdot\tilde{P}(A) + \quad 0 \quad + 1\cdot\tilde{P}(C)\right)\\
\tilde{P}(C) &= \frac{1-\mathbf{d}}{3} + \mathbf{d}\cdot\left(\tfrac{1}{2}\cdot\tilde{P}(A) + \quad 0 \quad + \quad 0 \quad\right)
\end{aligned}
$$

Folgen der neuen Definition

- Der Summand $\frac{(1-d)}{N}$ ist konstant und muss nur einmal berechnet werden, er kann ebenfalls aus der Matrix herausgezogen werden.
- $v = \frac{1-d}{N} \cdot e \cdot (e^T \cdot v) + d \cdot A \cdot v =$
  $\left(\frac{1-d}{N} \cdot (e \cdot e^T) \cdot v + d \cdot A \cdot v\right) = \underbrace{\left(\frac{1-d}{N} \cdot E + d \cdot A\right)}_{\tilde{A}} \cdot v$
- Wir erhalten nach wie vor eine Markov-Matrix und es gibt nach wie vor eine eindeutige Stationäre Verteilung:
- $\forall 1 \leq j \leq N : \sum_{i=1}^{N} \tilde{A}_{ij} = 1$

Dead Ends behandeln

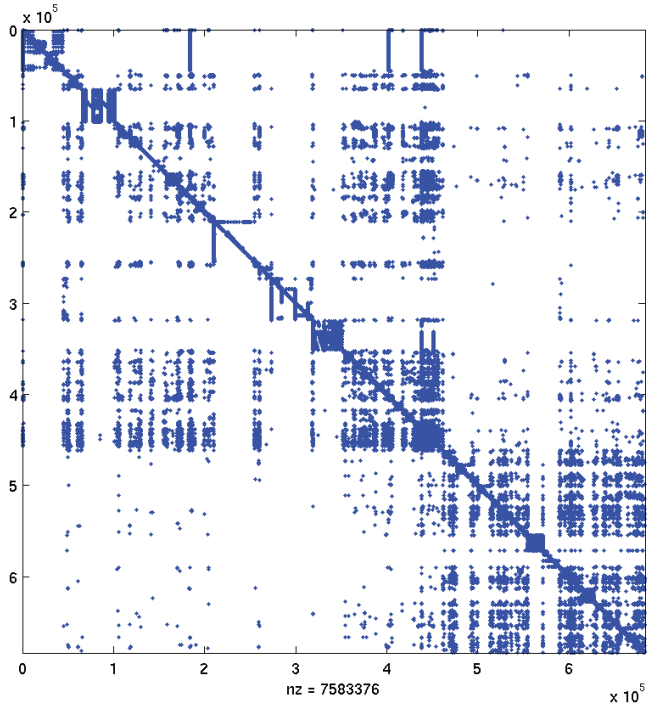
Es gibt verschiedene Ansätze, Dead Ends zu behandeln:

- **Teleport:** Falls der Zufallssurfer auf ein Dead End stößt, teleportiert er zufällig auf eine andere Seite.
  Nachteil: viele Nicht-Null-Einträge
- **»Prune and Propagate«:** Dead Ends in einer Vorverarbeitung erkennen und entfernen, später wieder hinzufügen und durch wenige Nachiterationen normalisieren
  Nachteil: benötigt unter Umständen mehrere Schritte (Overhead)

Wir nehmen im Folgenden an, dass Dead Ends in irgendeiner Weise bewältigt worden sind.

Contents

Ausnutzung der Blockstrukturen im Web Graph



Stanford/Berkeley

Bei näherer Betrachtung von Stichproben erkennt man Blockstrukturen im Web Graphen. Untersuchungen zeigen, dass der Verlinkungsgrad innerhalb einer Domain wesentlich höher ($\approx 80\,\%$) ist als der Verlinkungsgrad auf andere Domains.

BlockRank-Methode

Idee:

1. **Teile** den Web Graphen zunächst in **Blöcke von Domains** auf (kann bereits beim Untersuchen des Webs passieren).
2. Berechne zunächst **nur den PageRank einer Domain**. Die Matrix, die nur die Domains enthält, ist wesentlich kleiner!
3. Berechne dann für die **jeweiligen Domains einen** relativen **PageRank** für interne Seiten.

Man erhält dadurch eine bessere Approximation für den Startvektor v_0:
$\rightarrow$ weniger Iterationen notwendig
$\rightarrow$ etwa **doppelt so schnell**

Filter-Based-Adaptive PageRank 1

Beobachtung

Es gibt Seiten, die im Markov-Prozess schneller konvergieren als andere.

Annahme: Solche Seiten sind bereits gegen ihren PageRank konvergiert.
Sei C_k die Menge der Seiten, die im Iterationsschritt k bereits konvergiert ist. Wir definieren einen »**Filter**«:

$$v'(k)_j = \begin{cases} v(k)_j & \text{falls } j \in C_k \\ 0 & \text{sonst} \end{cases} \quad \text{und} \quad B_{ij} = \begin{cases} 0 & \text{falls } i \in C_k \\ \tilde{A}_{ij} & \text{sonst} \end{cases}$$

daraus folgt:

$$v(k+1) = B \cdot v(k) + v'(k)$$

Filter-Based-Adaptive PageRank 2

Warum ist das effizienter?

- Offensichtlich enthält unsere Matrix B mehr Null-Einträge als die ursprüngliche Matrix $\tilde{A}$.
- $nnz(B) \leq nnz(\tilde{A})$, wobei $nnz(M)$ die Nicht-Null-Einträge sind
- Die Laufzeit der Matrix-Vektor-Multiplikation hängt wesentlich von $nnz(B)$ ab!
- Verbesserung von $\mathbf{18 - 28\,\%}$

Contents

Zusammenfassung

- **Eindeutigkeit** und **Existenz** des Page Ranks als inhaltsunabhängiges **Maß für Wichtigkeit** einer Seite im Web Graph
- Probleme klassischer Lösungsansätze und Power Method
- Beispiele für effiziente Verbesserungsverfahren
- **Ausblick:** Vielseitiges Forschungsgebiet hinsichtlich Algorithmen, Datenstrukturen, Personalisierung, Hardware, Data-Mining, uvm.