

PageRank im Detail

Alexander Kandelberg

06.12.2007

Inhaltsverzeichnis

① Einführung

Inhaltsverzeichnis

① Einführung

② PageRank

Definition

Markoff-Modell des Webs

Dangling Nodes

Inhaltsverzeichnis

- ① Einführung
- ② PageRank
 - Definition
 - Markoff-Modell des Webs
 - Dangling Nodes
- ③ PageRank Berechnung
 - Potenziteration
 - Konvergenz und Eindeutigkeit
 - Linearsystem Formulierung
 - Personalisierungsvektor
 - Effiziente Techniken

Inhaltsverzeichnis

- ① Einführung
- ② PageRank
 - Definition
 - Markoff-Modell des Webs
 - Dangling Nodes
- ③ PageRank Berechnung
 - Potenziteration
 - Konvergenz und Eindeutigkeit
 - Linearsystem Formulierung
 - Personalisierungsvektor
 - Effiziente Techniken
- ④ Effizienz und Stabilität
 - Dämpfungsfaktor
 - Stabilität
 - Komplexitätsanalyse

Zielsetzung

Das *World Wide Web*

- wird immer komplexer
- Suchanfrage liefert große Menge von *relevanten* Webseiten

Frage: Wie kann man die Relevanz einer Webseite beurteilen?

Google benutzt:

- Hypertext-Matching-Analyse
- PageRanking

PageRanking

- absolute Wichtigkeit von Webseiten

PageRanking

- absolute Wichtigkeit von Webseiten
- anfrageunabhängig

PageRanking

- absolute Wichtigkeit von Webseiten
- anfrageunabhängig
- Rangfolge der relevanten Webseiten

PageRanking: Idee

Zitat-Analyse wissenschaftlicher Literatur

Die entscheidende Rolle spielen:

- Häufigkeit des Zitierens

PageRanking: Idee

Zitat-Analyse wissenschaftlicher Literatur

Die entscheidende Rolle spielen:

- Häufigkeit des Zitierens
- Wichtigkeit der Quelle

Erster Versuch

Gegeben: Webgraph $G = (V, E)$

- Definiere PageRank einer Seite als *Summe von PageRank-Werten aller ihrer backlinks*

Beobachtung:

- Seiten können mehrere ausgehende Links haben!

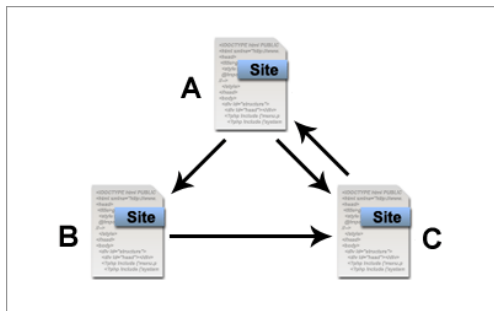
⇒ das gesamte PageRank einer Seite wird an allen ausgehenden Links gleichverteilt.

Seien $u, v \in V$, x_i PageRank der Seite i . Dann gilt:

$$x_v = \sum_{(u,v) \in E} \frac{x_u}{\text{outdeg}(u)}$$

Beispiel

$$x_v = \sum_{(u,v) \in E} \frac{x_u}{\text{outdeg}(u)}$$



$$\begin{aligned} x_A &= x_C \\ x_B &= \frac{x_A}{2} \\ x_C &= \frac{x_A}{2} + x_B \end{aligned}$$

Random Walk im Webgraphen

Abbildung des Verhaltens eines Websurfers:

- Starte an einem zufällig gewähltem Knoten im Webgraphen
- Für jede Seite i :
wähle zufällig einen der ausgehenden Links der Seite i und
verfolge diesen zu einer neuen Seite j .

Hinweis: Inhalte von Seiten werden nicht beachtet!

Problem: *Zufallssurfer* kann immer die gleichen Seiten aufrufen!

⇒ Iteration auf diesen Seiten unendlich

⇒ andere Seiten werden nie aufgerufen!

In Wirklichkeit: der Websurfer ist irgendwann gelangweilt und wählt zufällig eine andere Seite im Webgraphen, die nicht von der aktuell betrachteten verlinkt ist.

Dämpfungsfaktor

Lösung des Problems:

- Definiere die Wahrscheinlichkeit d für Verfolgen der Links.
- Dann: $(1 - d)$ die Wahrscheinlichkeit dafür, dass *Random Walk* abgebrochen wird.

Random Walk:

- Starte an einem zufällig gewähltem Knoten im Webgraphen
- Für jede Seite i :
 - mit **Wkeit d** : wähle zufällig einen der ausgehenden Links der Seite i und verfolge diesen zu einer neuen Seite j .
 - mit **Wkeit $(1 - d)$** : „Neustart“: rufe zufällig gleichverteilt eine andere Webseite auf.

PageRank: vollständige Definition

Seien $u, v \in V$, x_i PageRank der Seite i . Dann gilt:

$$x_v = \frac{1-d}{N} + d \sum_{(u,v) \in E} \frac{x_u}{\text{outdeg}(u)},$$

wobei

- N Anzahl der Seiten im Web und
- d Dämpfungsfaktor ist.

In dieser Definition ist

- x_v die globale Wahrscheinlichkeit des Aufrufes der Seite v ,
- PageRank die Wahrscheinlichkeitsverteilung über alle Seiten des Webs,
- die Summe aller PageRank-Werte 1.

Markoff-Kette

Definition(Stochastischer Prozess)

Eine Familie $\{X_t | t \in T\}$ von Zufallsvariablen nennt man einen *stochastischen Prozess*.

Definition(Markoff-Kette)

Sei $S = \{0, 1, \dots, t-1\}$ die Zustandsmenge und X_0, X_1, \dots, X_{t-1} die Folge von Zufallsvariablen über S . Eine Markoff-Kette ist ein stochastischer Prozess $(X_t)_{t \in \mathbb{N}}$, der folgende Bedingung erfüllt:

$\forall t \in \mathbb{N}$ und $i_0, \dots, i_t \in I$ mit $P(X_0 = i_0, \dots, X_t = i_t) > 0$ gilt:
 $P(X_{t+1} = i_{t+1} | X_0 = i_0, \dots, X_t = i_t) = P(X_{t+1} = i_{t+1} | X_t = i_t)$.

Interpretation 1/2

Verhalten des Zufallssurfers \rightarrow Markoff-Kette

- X_t ist zum Zeitpunkt t besuchte Seite
- $I = V$ die Menge der von Google indexierten Seiten
- Markoff-Eigenschaft: die Wahrscheinlichkeit des Aufrufes einer Seite zum Zeitpunkt $(t + 1)$ ist abhängig
 - von der zum Zeitpunkt t betrachteten Seite
 - und vom Zeitpunkt $t \rightarrow$ unerwünscht!

Eine Markoff-Kette heißt *homogen*, wenn $\forall t \in \mathbb{N}$ und $i, j \in I$ gilt:

$$P(X_{t+1} = j | X_t = i) = p_{ij}$$

Interpretation 2/2

Verhalten des Zufallssurfers \rightarrow Markoff-Kette

- X_t ist zum Zeitpunkt t besuchte Seite
- $I = V$ die Menge der von Google indexierten Seiten
- einen gerichteten Pfad im Webgraphen \rightarrow Sequenz der besuchten Zustände der Markoff-Kette.

Markoff-Kette. Repräsentation:

- einen gewichteten Digraph
- Übergangsmatrix

Übergangsmatrix

Definition(Markoff-Matrix)

Eine *stochastische Matrix* M ist die Übergangsmatrix einer endlichen Markoff-Kette, falls

- Elemente der Matrix reelle Zahlen aus $[0, 1]$ sind und
- die Summe der Zeileneinträge 1 ist $\Rightarrow M$ stochastisch

Für unsere *erste Definition* vom PageRank:

Sei $A = (a_{ij})$ die Adjazenzmatrix des Webgraphen.
Übergangsmatrix $P = (p_{ij})_{i,j \in V}$ ist definiert durch

$$p_{ij} = \begin{cases} \frac{a_{ij}}{\text{outdeg}(i)}, & \text{wenn } \text{outdeg}(i) > 0 \\ 0, & \text{sonst} \end{cases}$$

Ursache: *Dangling Nodes!*

Behandlung von *Dangling Nodes*

Definition (Dangling Nodes)

Dangling Nodes sind Knoten v im Webgraphen mit $\text{outdeg}(v) = 0$.

Beispiele: word-, pdf-Dokumente

Behandlung von Dangling Nodes:

- Sei P die oben definierte gewichtete Adjazenzmatrix,
- $e = (\frac{1}{n}, \dots, \frac{1}{n})$, wobei $n = \text{dim}(P)$

Wir erhalten die Übergangsmatrix der Markoff-Kette P' wie folgt:

- Ersetze jede Nullzeile der Matrix P durch e

$$p'_{ij} = \begin{cases} \frac{a_{ij}}{\text{outdeg}(i)}, & \text{wenn } \text{outdeg}(i) > 0 \\ \frac{1}{n}, & \text{sonst} \end{cases}$$

PageRank Iteration

Sei $r_i^{(t)}$ die Wahrscheinlichkeit für einen Websurfer, sich zum Zeitpunkt t im Knoten i zu befinden. Die Wahrscheinlichkeit, dass er sich zum Zeitpunkt $t + 1$ im Knoten j befindet, ist dann:

$$r_j^{(t+1)} = \sum_i p'_{ij} r_i^{(t)}$$

oder in Matrix-Vektor-Notation:

$$r^{(t+1)} = r^{(t)} P'$$

Potenziteration

Gegeben ist ein stochastische Prozess

Gesucht: ein Vektor $r^{(t)}$, gegen welchen die Iteration konvergiert

→ stationäre Verteilung der Markoff-Kette!

Stationäre Verteilung der Markoff-Kette

Stationäre Verteilung

Stationäre Verteilung einer Markoff-Kette M ist eine Wahrscheinlichkeitsverteilung π , so dass gilt:

$$\pi = M^T \pi$$

Vektor der stationären Verteilung:

- Einträge sind die Wahrscheinlichkeiten dafür, dass der Zufallssurfer eine bestimmte Seite aufruft.
- Wahrscheinlichkeitsverteilung für alle Seiten des Webs (\rightarrow *PageRank*)

Eigenvektorproblem

Vektor der stationären Verteilung

⇒ Eigenvektor der stochastischen Matrix zum (betragsmäßig größten) Eigenwert.

Eigenvektorproblem

Eine Zahl $\alpha \in \mathbb{R}$ heißt *Eigenwert* einer Matrix M , wenn einen Vektor $v \neq 0$ existiert, so dass

$$Mv = \alpha v$$

gilt und jeder Vektor $v \neq 0$, der diese Gleichung erfüllt, heißt *Eigenvektor* von M zum Eigenwert α .

⇒ $\pi = M^T \pi$ ist somit identisch mit Eigensystem

$$M^T v = \alpha v \text{ mit } \alpha = 1$$

Eigenwert einer stochastischen Matrix

Satz

Matrix M stochastisch \Rightarrow (größter) Eigenwert von M ist 1.

Beweis.

Sei

- A eine stochastische $n \times n$ -Matrix
- $e = (1, \dots, 1)^T$ n -dimensionalen Vektor

Dann gilt:

$$Ae = e$$



Es stellen sich folgende Fragen:

- 1 Existiert eine Lösung?

Es stellen sich folgende Fragen:

- 1 Existiert eine Lösung?
- 2 Ist die Lösung eindeutig?

Es stellen sich folgende Fragen:

- 1 Existiert eine Lösung?
- 2 Ist die Lösung eindeutig?
- 3 Ist das Ergebnis der Berechnung sinnvoll?

Eindeutigkeit der Lösung 1/2

Um Eindeutigkeit der Lösung zu garantieren, muss die Matrix

- stochastisch
- irreduzibel
- und **aperiodisch** sein.

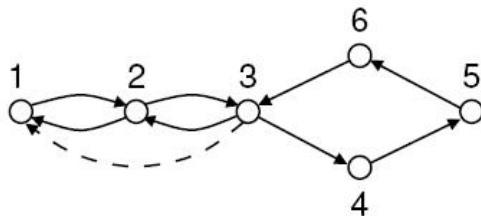
Aperiodizität der Markoff-Kette

Die *Periode* eines Zustands der Markoff-Kette ist das größte p , so dass gilt:

$$\forall k, n \in \mathbb{N}. P(X_{k+n} = x | X_k = x) > 0 \Rightarrow p \text{ teilt } n$$

Falls $p = 1$, dann heißt der Zustand aperiodisch. Eine Markoff-Kette heißt aperiodisch, falls jeder Zustand aperiodisch ist.

Beispiel



Knoten:	1	2	3	4	5	6
Periode:	2	2	2	4	4	4

Aperiodizität

Wieso brauchen wir die Aperiodizität?

Beispiel:

Gegeben ist

- ein stark zusammenhängender, periodischer Graph $G = (V, E)$
- und Knoten $u, v \in V$

PageRank Iteration:

$$r^{(i)} = (q, 1 - q)$$

$$r^{(j)} = (1 - q, q)$$

mit $i, j \geq 0$ und $j > i$.

⇒ Iteration konvergiert nicht!

Eindeutigkeit der Lösung 2/2

Um Eindeutigkeit der Lösung zu garantieren, muss die Matrix

- stochastisch
- irreduzibel
- und aperiodisch sein.

Definition(ergodische Markoff-Kette)

Eine Markoff-Kette heißt *ergodisch*, falls sie irreduzibel und aperiodisch ist.

Grenzwertsatz für ergodische Markoff Ketten

Jede (endliche homogene) Markoff-Kette besitzt eine **eindeutige** stationäre Verteilung π und es gilt:

$$\lim_{t \rightarrow \infty} q_0 M^T = \pi \text{ (und } \pi \text{ von } q_0 \text{ unabhängig ist).}$$

Umformung der Matrix

Ziel: irreduzible und aperiodische Matrix.

Gegeben: eine stochastische Matrix P'

Idee:

- Für alle Einträge in P' werden Mindestwahrscheinlichkeiten d eingeführt!

$$P'' = dP' + (1 - d)E,$$

wobei

- $E = ev^T$ (Störmatrix),
- $e = (1, \dots, 1)$,
- v Personalisierungsvektor
- und $0 < d < 1$

Linearsystem

PageRank als Eigenvektorproblem:

⇒ Vektor π^T in

$$\pi^T P'' = \pi^T$$

kann gefunden werden durch das Lösen von

$$\pi^T (I - P'') = 0$$

Einsetzen von P'' in die obere Gleichung liefert:

$$\pi^T (dP' + (1-d)ev^T) = \pi^T$$

$$\Rightarrow \pi^T I - \pi^T dP' = (1-d) \underbrace{\pi^T e}_{=1} v^T$$

$$\pi^T (I - dP') = (1-d)v^T$$

$$\pi^T(I - dP') = (1 - d)v^T$$

$$P' = P + av^T \text{ mit}$$

$$a_i = \begin{cases} 1, & \text{wenn i-te Zeile in P nur Nullen enthält} \\ 0, & \text{sonst} \end{cases}$$

$$\text{und } v = \left(\frac{1}{n}, \dots, \frac{1}{n}\right)$$

$$\pi^T(I - dP - dav^T) = (1 - d)v^T$$

$$\text{Sei } \pi^T a = \gamma$$

$$\Rightarrow \pi^T(I - dP) = (1 - d + d\gamma)v^T$$

$$\text{Wegen } \pi^T e = 1 \text{ wähle } \gamma = 1$$

$$\pi^T(I - dP) = v^T$$

PageRank Personalisierung 1/2

PageRank

- *globale* Wichtigkeit von Webseiten
- und somit ein **subjektiver** Begriff!
- Suchanfrage hat keinen Einfluss auf die Berechnung

Aus der Sicht des Webbenutzers:

Interesse an **Personalisierung** des Webs!

PageRank Personalisierung 2/2

Idee (Haveliwala):

- berechne für jedes spezielle Themengebiet (z.B. Sport, Wirtschaft) ein PageRank-Vektor
- berechne das gesamte PageRank als Linearkombination des themenspezifischen PageRanks

Problem:

- PageRank-Berechnung für nur einen Vektor v^T kann mehrere Tagen dauern!

Lösung: skalierbare personalisierte PageRank-Methode

Skalierbare personalisierte PageRank-Methode

Arbeit: Jeh, Widom.

Idee: Ausnutzung der linearen Abhängigkeit zwischen v^T und dem dazugehörigen PageRank-Vektor

- Erzeuge eine Menge von linear unabhängigen Personalisierungsvektoren (Basisvektoren)
→ Ansatz der dynamischen Programmierung
- berechne daraus die Approximation des Vektors v^T

Spam-Kontrolle

Definition (Link Farm)

Als Linkfarm wird eine Ansammlung von Webseiten oder ganzen Domänen im Web bezeichnet, die dem Zweck dient, möglichst viele Hyperlinks auf eine andere Webseite zu legen.

- besitzen oft Seiten mit einem hohen PageRank
- enthalten mehrere untereinander verbundene Knoten

Ansatz: geeigneter Personalisierungsvektor!

PageRank Berechnung

- „*The World's Largest Matrix Computation*“
- kann mehrere Tagen dauern!

Für die Minimierung der Zeitdifferenz von der Fertigstellung eines Crawl bis zum Beginn der Suche braucht man Techniken zur

- Reduzierung der Arbeit in jeder Iteration
- Reduzierung der Anzahl der Iterationen

Reduzierung der Arbeit in jeder Iteration

Adaptiver PageRank Algorithmus

Beobachtung: Konvergenzrate der PageRank-Werte einzelner Webseiten verläuft ungleichmäßig

Idee:

- ermittle in jeder Iteration die bereits konvergierten Seiten
- berechne in der nachfolgenden Iteration PageRank nur für die noch nicht konvergierten Seiten

Adaptiver PageRank Algorithmus

Seien

- C die Menge der Seiten, die bereits konvergiert sind,
- N die Menge der noch nicht konvergierten Seiten,
- A zum Webgraph gehörige Matrix,
- und A_N und A_C Submatrizen von A.

Vorschrift einer Iteration:

$$x_N^{(k+1)} = A_N x_N^{(k)}$$

$$x_C^{(k+1)} = x_C^{(k)}$$

⇒ Reduzierung der Berechnung um ca. 30 % !

Reduzierung der Anzahl der Iterationen

BlockRank Algorithmus

Beobachtung: Mehrzahl an Links befindet sich zwischen den Seiten derselben Domain.

Idee: statt der Berechnung eines *globalen PageRanks*

- berechne für jeden Block einen *lokalen* PageRank-Vektor
- berechne für alle Blöcke einen *BlockRank*-Vektor
- approximiere der *globale* PageRank-Vektor mit Hilfe von *lokalen* PageRanks und einem *BlockRank*-Vektor
- benutze die Approximation als Startvektor für die Standard-PageRank-Methode

⇒ Reduzierung der Anzahl der Iterationen bis zu Faktor 2!

Interpretation

Interpretation im Random-Surfer-Modell:

- mit Wahrscheinlichkeit d wähle die nächste Seite
- mit Wahrscheinlichkeit $(1 - d)$ „Neustart“ bei einer beliebigen Seite

$$P'' = dP' + (1 - d)E \text{ mit } E = ev^T$$

- $d = 1 \rightarrow$ *reale* Linkstruktur des Webgraphen
- $d = 0 \rightarrow$ Gleichverteilung für alle Seiten des Webs

Wert eines Dämpfungsfaktors

Von dem Wert eines Dämpfungsfaktors hängt die Geschwindigkeit der Konvergenz ab:

Anzahl der benötigten Iterationen

$$\frac{\log_{10} \tau}{\log_{10} d},$$

wobei τ *Konvergenzgenauigkeit* ist:

$$\tau = r^{(i)T} P'' - r^{(i)T} = r^{(i+1)T} - r^{(i)T}$$

Für $\tau = 10^{-6}$, $d = 0,85$ ergeben sich

$$\frac{-6}{\log_{10} 0,85} \approx 85 \text{ Iterationen}$$

\Rightarrow 50 - 100 Iterationen für $d = 0,85$ und τ von 10^{-3} bis 10^{-7}

World Wide Web

Webgraph

Der Webgraph ist ein endlicher gerichteter Graph $G = (V, E)$ mit

- $V = \{x_1, \dots, x_n\}$ ist statisch
- $E \subseteq V \times V$,
so dass $(v_i, v_j) \in E$ gdw. Seite v_i hat einen Link auf Seite v_j

Das **World Wide Web** ist jedoch **dynamisch**:

- Es werden stets die neuen Seiten erzeugt, geändert, gelöscht.
- Einige Seiten sind von der Suchmaschine nicht erreichbar oder werden übergangen

⇒ **Beeinflussung des PageRanks!**

PageRank Beeinflussung

Satz (Ng, Zheng, Jordan)

Sei r der PageRank-Vektor, $d < 1$ sei Dämpfungsfaktor und r' eine stationäre Verteilung nach der Veränderung der Linksstruktur des Webs. S sei die Menge der modifizierten Seiten.

Dann gilt:

$$\|r' - r\|_1 \leq \frac{2}{1-d} \sum_{i \in S} r_i$$

Also ist die Änderung des PageRanks nicht so groß, wenn

- d nicht zu nah an 1 gewählt wird und
- PageRank-Werte der geänderten Seiten nicht zu groß sind.

Google Dance

- PageRank-Werte sind stabil gegen relativ kleine (*tägliche*) Änderungen
- WWW ändert sich ständig

⇒ nach einiger Zeit sind PageRank-Werte der einzelnen Seiten nicht mehr aktuell!

Aktualisierung des PageRanks (vor 2003):

- erneute PageRank-Berechnung monatlich

Aktualisierung des PageRanks (ab 2003):

- kontinuierliches Index-Update
- erneute PageRank-Berechnung durchschnittlich einmal in 3 Monaten

Laufzeit des Algorithmus 1/2

$$P'' = dP' + (1 - d)E \text{ mit } E = ev^T$$

Laufzeit des Algorithmus hängt ab von:

- den Kosten der einzelnen Matrix-Vektor-Berechnung $K_{(MV)}$,
- der Größe der Matrix n (= Anzahl Knoten im Webgraphen)
- und der Anzahl benötigter Iterationen Cnt_i

Also:

$$O(K_{(MV)} n Cnt_i)$$

Laufzeit des Algorithmus 2/2

Eigenschaft: Matrix P' ist dünn besetzt!

Dünn besetzte Matrix (engl. sparse matrix)

Eine Matrix heißt *dünn besetzt*, falls sie nur sehr wenige von Null verschiedene Einträge besitzt.

⇒ Einzelne Matrix-Vektor-Berechnung in der Zeit $O(n)$ möglich!

Anzahl der benötigter Iterationen:

$$\frac{\log_{10} \tau}{\log_{10} d}$$

Für $\tau = 10^{-6}$, $d = 0,85$ ergeben sich

$$\frac{-6}{\log_{10} 0,85} \approx 85 \text{ Iterationen}$$

⇒ Laufzeit des Algorithmus ist $O(n^2)$.