Web Information Retrieval

Überblick

- Locality Sensitive Hashing
- Das Ranking Problem
- PageRank
- HITS (Hubs & Authorities)
- Markov Ketten und Random Walks
- PageRank und HITS Berechnung

Locality Sensitive Hashing (LSH)

- Eine spezielle Skizzierungsmethode
- H = { h | h: U → T }: eine Familie von Hash Funktionen
- H ist locality sensitive bezüglich sim falls für alle p,q ∈ U, Pr[h(p) = h(q)] = sim(p,q).
  - Wahrscheinlichkeit ist über eine zufällige Wahl von h aus H
  - Wahrscheinlichkeit einer Kollision = Ähnlichkeit zwischen p und q

Syntaktisches Clustering mittels LSH

1. P ← leere Tabelle der Größe |D|
2. G ← leerer Graphen mit |D| Knoten
3. für i = 1,…,|D|
4. lese Dokument p_i aus der Menge
5. P[i] ← $h(p_i)$
6. sortiere P und gruppiere nach Wert
7. Gebe Gruppen aus

Analyse

- Skizzen können in einem Durchlauf erzeugt werden
- Tabelle P kann in einer Datei auf einem Rechner gespeichert werden
- Sortieren und Gruppieren erfordert O(|S| log |S|) Vergleiche
- Eine Gruppe A besteht aus dem Dokumenten mit den selben Hashwert
  - Wegen LSH Eigenschaft, die Dokument sind mit hoher Wahrscheinlichkeit ähnlich

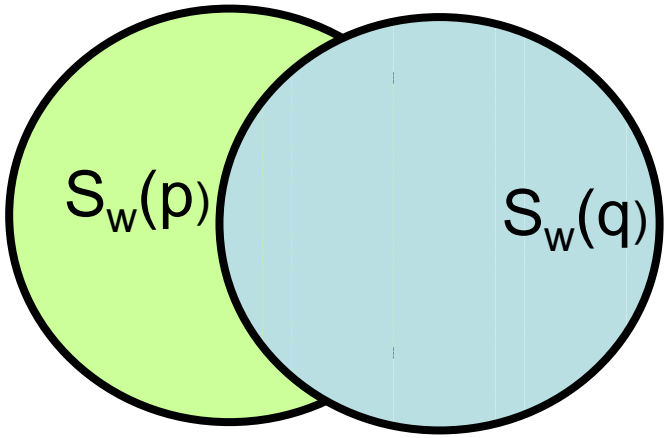
Shingling und Resemblance

- Token: Wörter, Zahlen, HTML tags, usw.
- Tokenisation(p): Sequenz der Token eines Dokuments p
- w: ein kleiner Integer
- $S_w(p)$ = w-shingling von p = Menge aller verschiedener fortlaufender Teilsequenzen von Tokenization(p) mit Länge w.
  - z.B.: p = “a rose is a rose is a rose”, w = 4
  - $S_w(p)$ = { (a rose is a), (rose is a rose), (is a rose is) }

- $\text{resemblance}_w(p,q) = \dfrac{|S_w(p) \cap S_w(q)|}{|S_w(p) \cup S_w(q)|}$

LSH für Resemblance

- $\text{resemblance}_w(p,q) = \dfrac{|S_w(p) \cap S_w(q)|}{|S_w(p) \cup S_w(q)|}$



- π = eine zufällige Permutation von Σ^w
  - π induziert eine zufällige Ordnung auf allen Tokensequenzen der Länge w
  - π induziert auch eine zufällige Ordnung auf einer beliebigen Teilmenge $X \subseteq \Sigma^w$
  - Für alle solche Teilmengen und für jedes $x \in X$, $\Pr(\min(\pi(X)) = x) = 1/|X|$
- LSH für resemblance: $h(p) = \min(\pi(S_w(p)))$

LSH für Resemblance

- Lemma: $\Pr[\min(\pi(S_w(p)) = \min(\pi(S_w(q))] = \frac{|S_w(p) \cap S_w(q)|}{|S_w(p) \cup S_w(q)|}$
- Proof:

$$
\begin{aligned}
&\Pr[\min(\pi(S_w(p)) = \min(\pi(S_w(q))] \\
&= \Pr[\min(\pi(S_w(p) \cup S_w(q)) \in S_w(p) \cap S_w(q)] \\
&= \sum_{x \in S_w(p) \cap S_w(q)} \Pr[\min(\pi(S_w(p) \cup S_w(q)) = x] \\
&= \sum_{x \in S_w(p) \cap S_w(q)} \frac{1}{|S_w(p) \cup S_w(q)|} \\
&= \frac{|S_w(p) \cap S_w(q)|}{|S_w(p) \cup S_w(q)|}
\end{aligned}
$$

The Ranking Problem

- Eingabe:
  - D: Dokumentkollektion
  - Q: Anfrageraum
- Ziel: Finde eine Ranking Funktion rank:D x Q → **R** s.d.

$$\forall q, d, d', \quad \text{if } \mathrm{rank}(d, q) > \mathrm{rank}(d', q)$$
$$\text{then } d \text{ ist relevanter fuer } q \text{ als } d'$$

  - Rank und q induzieren ein Ranking (partielle Ordnung) π_q auf D
  - Analog zu “Relevanz” im IR Kontext

Text-basiertes Ranking

- Klassische Ranking Funktionen:
  - Keyword-basiertes Boolesches Ranking
  - Cosinus Ähnlichkeit + TF-IDF Gewichte

- Grenzen im Kontext von Web Suche:
  - Das Problem das Überflusses
    - Recall ist nicht wichtig
  - Kurze Anfragen
  - Web Seiten sind kein durchgängiger Text
  - Synonyme (cars vs. autos)
  - Mehrdeutigkeit (java, “Michael Jordan”)
  - Spam

Link-basiertes Ranking

Hypertext IR Prinzip #1

If p → q, then q ist "relevant" für p

Hypertext IR Prinzip #2

If p → q, then p spricht q "Autorität" zu

- Hyperlinks haben wichtige Bedeutungen
  - Empfehlung
  - Kritik
  - Navigation

Statisches Ranking

- Statisches Ranking: rank: D → **R**, wobei rank(d) > rank(d') impliziert, dass d mehr "Autorität" als d' hat

- Nutze die Links um ein statisches Ranking aller Web Seiten zu berechnen.
- Für eine gegebene Anfrage q, nutze text-basiertes Ranking um ein Menge S mit relevanten Kandidatenseiten zu identifizieren.
- Ordne S nach dem statisches Rang.

- Vorteil: statisches Ranking kann als vorberechnet werden.
- Nachteil: Hypertext IR Prinzip #1 wird nicht verwendet.

Anfrage abhängiges Ranking

- Für eine gegebene Anfrage q, nutze text-basiertes Ranking um eine Menge S mit Kandidaten für relevante Seiten zu identifizieren.

- Nutze die Links innerhalb von S um ein Ranking rank: S → **R** zu finden, wobei rank(d) > rank(d') impliziert, dass d mehr Autorität als d' hat im Kontext der Anfrage q.

- Vorteil: beide Hypertext IR Prinzipien sind genutzt.
- Nachteil: weniger effizient.

Das Web als ein Graph

- V = eine Menge von Seiten
  - bei statischem Ranking, V = Web
  - bei Anfrage abhängigem Ranking, V = S
- Der Web Graph: G = (V,E), wobei
  - (p,q) ist eine Kante, gdw. p hat einen Link zu q
- A = Adjazenz Matrix von G

$$A_{p,q} = \begin{cases} 1 & \text{falls } p \rightarrow q \\ 0 & sonst \end{cases}$$

Popularitäts-Ranking

- rank(p) = in-degree(p)
- Vorteile
  - Wichtigsten Seiten aus Millionen von Trefffern hervorgehoben
  - Keine Notwendigkeit für textreiche Dokumente
  - Effizient berechnenbar
- Nachteile
  - Bias hinzu populären Seiten, unabhängig von der Anfrage
  - Kann leicht durch Spam beeinflußt werden

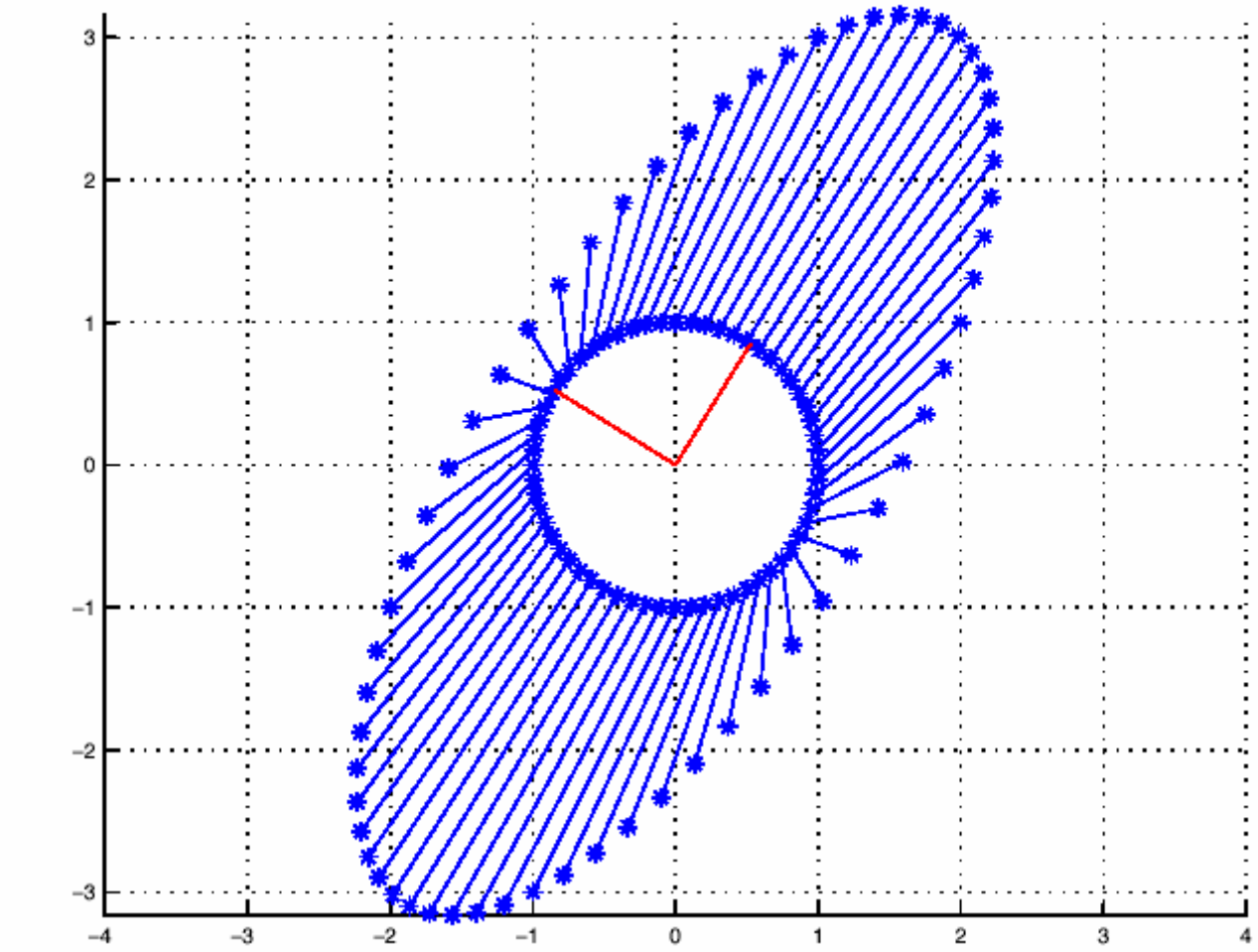
Eigenwerte, Eigenvektoren

- Sei A eine n x n Matrix und v ein Vektor mit

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$$

- Dann ist v ein Eigenvektor von A und lambda ist ein Eigenwert

Beispiel



PageRank

[Page, Brin, Motwani, Winograd 1998]

- Motivierende Prinzipien
  - Rang von p soll proportional zum Rang der Seiten sein, die auf p zeigen
    - Empfehlungen von Bill Gates & Steve Jobs vs. Andreas Both und Alexander Hinneburg
  - Rang von p soll abhängig von der Anzahl der Seiten sein, die mit p zusammen verlinkt sind
    - Vergleich: Bill Gates empfiehlt nur mich vs. Bill Gates empfiehlt alle Menschen auf der Erde

PageRank, 1. Versuch

$$\text{rank}(p) = \sum_{q:q\to p} \frac{\text{rank}(q)}{\text{out-degree(q)}}$$

- r = Rang Vektor
- B = normalisierte Adjazenz Matrix:

$$B_{p,q} = \begin{cases} 1/\text{out-degree}(p) & \text{if } p \to q \\ 0 & otherwise \end{cases}$$

- Dann: $r^T = r^T B$
  - r ist ein linker Eigenvektor von B
  - B muss 1 als einen Eigenwert haben
  - Weil einige Zeilen von B 0 Vektoren sind (Seiten ohne Links), ist 1 nicht zwingend ein Eigenwert
  - Rang geht in Senken “verloren”

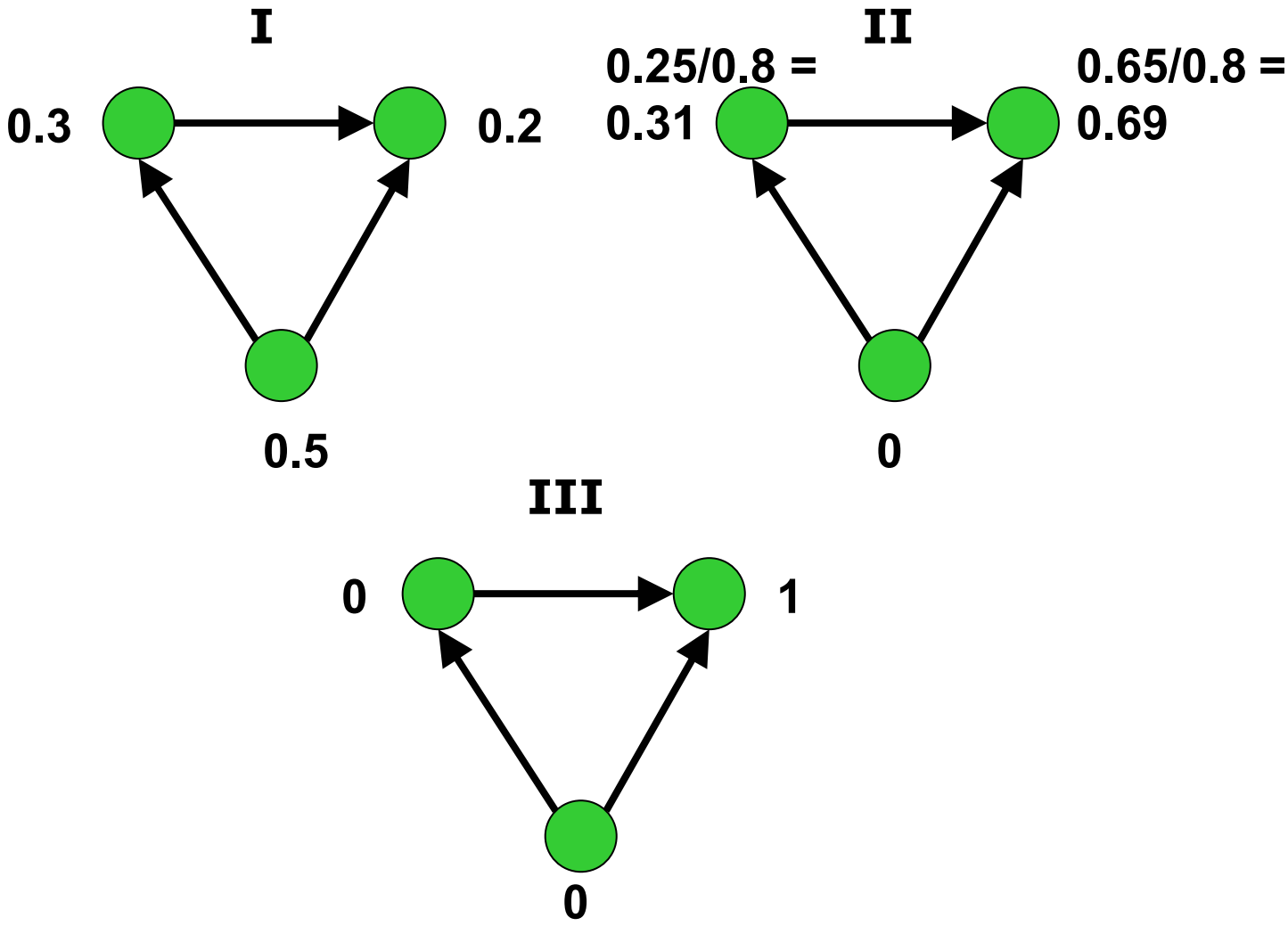
PageRank, 2. Versuch

$$\text{rank}(p) = \lambda \sum_{q:q\to p} \frac{\text{rank}(q)}{\text{out-degree(q)}}$$

wobei: $\lambda = \sum_{q} \text{rank}(q) \;\; / \sum_{q\,:\,\text{out-degree}(q)>0} \text{rank}(q)$

- Dann: $r^T = \lambda r^T B$
  - r ist ein linker Eigenvektor von B mit Eigenwert 1/λ
  - Jeder Eigenvektor ist für r einsetzbar.
  - Typischerweise nimmt man den normalisierten 1. Eigenvektor.
  - Rang akkumuliert sich in Senken und geschlossenen Gemeinschaften.

PageRank, 2. Versuch : Beispiel



PageRank, Endgültige Definition

$$\text{pagerank}(p) = \lambda \left(\sum_{q:q \to p} \frac{\text{pagerank}(q)}{\text{out-degree(q)}} + E(p) \right)$$

- E(p) = “Rang Quellen” Funktion
  - Standard Belegung: E(p) = $\beta/|V|$ für ein $\beta < 1$
- PageRank ist die normalisierte L_1 Norm für Matrizen
- e = Rang Quellen Vektor, r = PageRank Vektor
- **1** = alles 1 Vektor
- Dann: $r^T = \lambda(r^T B + e^T) = \lambda r^T (B + \mathbf{1} e^T)$
  - r ist ein linker Eigenvektor von (B + **1**e^T) mit Eigenwert1/λ
  - Nutze normalisierten 1. Eigenvektor (mit betragsmäßig größtem Eigenwert)

Das Random Surfer Modell

- Falls der “Random Surfer” die Seite p besucht:
  - mit Wahrscheinlichkeit 1 - α wählt er einen der ausgehenden Links p → q und geht nach q. (“fokusiertes Browsen”)
  - mit Wahrscheinlichkeit α springt er zu einer zufälligen Web-Seite q. (“Interesse verloren”)
  - Falls p keine ausgehenden Links hat, wird ein Link zu sich selbst angenommen.
- P: Wahrscheinlichkeitsübergangs Matrix:

$$P_{p,q} = \begin{cases} (1-\alpha)/\text{out-deg}(p) + \alpha/|V| & \text{falls } p \to q \\ \alpha/|V| & \text{sonst} \end{cases}$$

PageRank & Random Surfer Modell

$$P_{p,q} = \begin{cases} (1-\alpha)/\text{out-deg}(p) + \alpha/|V| & \text{falls } p \to q \\ \alpha/|V| & \text{sonst} \end{cases}$$

Angenommen: $E(p) = \frac{\alpha}{1-\alpha} \cdot \frac{1}{|V|}, \ \forall p$

Dann:

$$P = (1-\alpha)B + \alpha \mathbf{1}(1/|V|)^T = (1-\alpha)(B + \mathbf{1}e^T)$$

Deshalb gilt, **r** ist ein linker Eigenvektor von $(B + \mathbf{1}e^T)$ mit Eigenwert $1/(1 - \alpha)$, gdw. **r** ist ein linker Eigenvektor von P mit Eigenwert 1.

Einführung zu Markov Ketten

- V = Zustandsraum
- P = Wahrscheinlichkeitsübergangsmatrix
  - Nicht-negativ.
  - Summe jeder Zeile ist 1.
- q_0 = initiale Verteilung über V
- $q_t = q_0 P^t$: Verteilung über V nach t Schritten
- P ist ergodisch falls zutrifft:
  - Nichtreduzierbar (zugrundeliegender Graph ist stark zusammenhängend)
  - Aperiodisch (für alle Zustände u,v gilt: der GGT der Länge der Pfade von u nach v ist 1)
- **Theorem**

  Falls P ergodisch ist, dann hat es eine "stationäre Verteilung" π. Weiterhin, für alle q_0, $q_t \rightarrow \pi$ wobei t gegen unendlich geht.
- $\pi P = \pi$. π ist ein linker Eigenvektor von P mit Eigenwert 1.

PageRank & Markov Ketten

- **Schlußfolgerung:** Der PageRank Vektor r ist die stationäre Verteilung der Random Surfer Markov Kette.
- pagerank(p) = r_p = Wahrscheinlichkeit dass der Random Surfer p besucht.
- Bemerkung: “Zufällige Sprünge” garantieren der Markov Kette Nichtreduzierbarkeit und Aperiodizität.

PageRank Berechnung

$r_{old} \leftarrow (0, ..., 0)$

$r_{new} \leftarrow (1/|V|, ..., 1/|V|)$

while$(||r_{new} - r_{old}||_1 > \epsilon)$ do

$r_{old} \leftarrow r_{new}$

$r_{new} \leftarrow r_{new}P$

end while

In Praxis: etwa 50 Iterationen reichen aus

HITS: Hubs und Autoritäten

[Kleinberg, 1997]

- HITS: Hyperlink Induced Topic Search
- **Hauptprinzip:** Jede Seite p hat zwei Werte:
  - Autoritätswert: wieviel "Autorität" hat eine Seite für das Anfragethema
    - z.B. Anfrage: "IR"; Autoritäten: wissenschaftliche IR Artikel
    - z.B. Anfrage: "Autohersteller"; Autoritäten: Mercedes, VW, Toyota
  - Hub-Wert: Wie gut ist eine Seite als "Verweisliste" für das Anfragethema
    - z.B. Anfrage: "IR"; Hubs: Überblicksartikel und Bücher über IR
    - z.B. Anfrage: "Autohersteller"; Hubs: ADAC, Schwacke Liste

Gegenseitiges Antreiben

HITS Prinzipien:

- p ist eine hohe Autorität, wenn es von vielen guten Verzeichnissen (Hubs) gelistet wird.
- p ist ein guter Hub, wenn er auf viele hohe Autoritäten verweist.

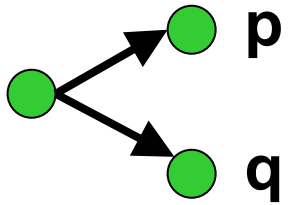
$$a(p) = \sum_{q:q \to p} h(q) \qquad h(p) = \sum_{q:p \to q} a(q)$$

HITS: Algebraische Form

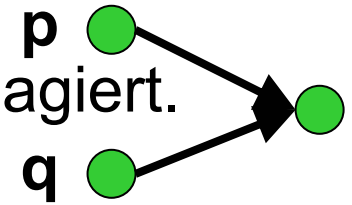
- a: Autoritäts Vektor
- h: Hub Vektor
- A: Adjazenz Matrix
- Dann: $a = A^T h \qquad h = Aa$
- Deshalb: $a = A^T Aa \qquad h = AA^T h$
- a ist der 1. Eigenvektor von A^TA
- h ist der 1. Eigenvektor von AA^T

Co-Citation und Kopplung

- A^TA: co-citation Matrix
  - $A^TA_{p,q}$ = Anzahl der Seiten, die zu beiden Seiten p und q zeigen.
  - Autoritätswerte werden durch co-citation propagiert.



- AA^T: Kopplungsmatrix
  - $AA^T_{p,q}$ = Anzahl der Seiten auf die von p und q verlinkt wird.
  - Hub-Werte werden durch Kopplung propagiert.



HITS Berechnung

$a \leftarrow (1/|V|, ..., 1/|V|)$

$h \leftarrow (1/|V|, ..., 1/|V|)$

while(a und h aendern sich signifikant) do

$h \leftarrow Aa$

$h \leftarrow h/||h||_1$

$a \leftarrow A^T h$

$a \leftarrow a/||a||_1$

end while

Berechnung des 1. Eigenvektors

- E: n x n Matrix
- $|\lambda_1| > |\lambda_2| >= |\lambda_3| \ldots >= |\lambda_n|$: Eigenwerte von E
- $v_1,\ldots,v_n$: korrespondierende Eigenvektoren
- Eigenvektoren sind linear unabhängig
- Eingabe:
  - Matrix E
  - (der 1. Eigenwert λ_1)
  - Ein Vektor u der Länge 1, der nicht orthogonal zu v_1 ist
- Ziel: berechne v_1

Die Power Methode

$w \leftarrow u$

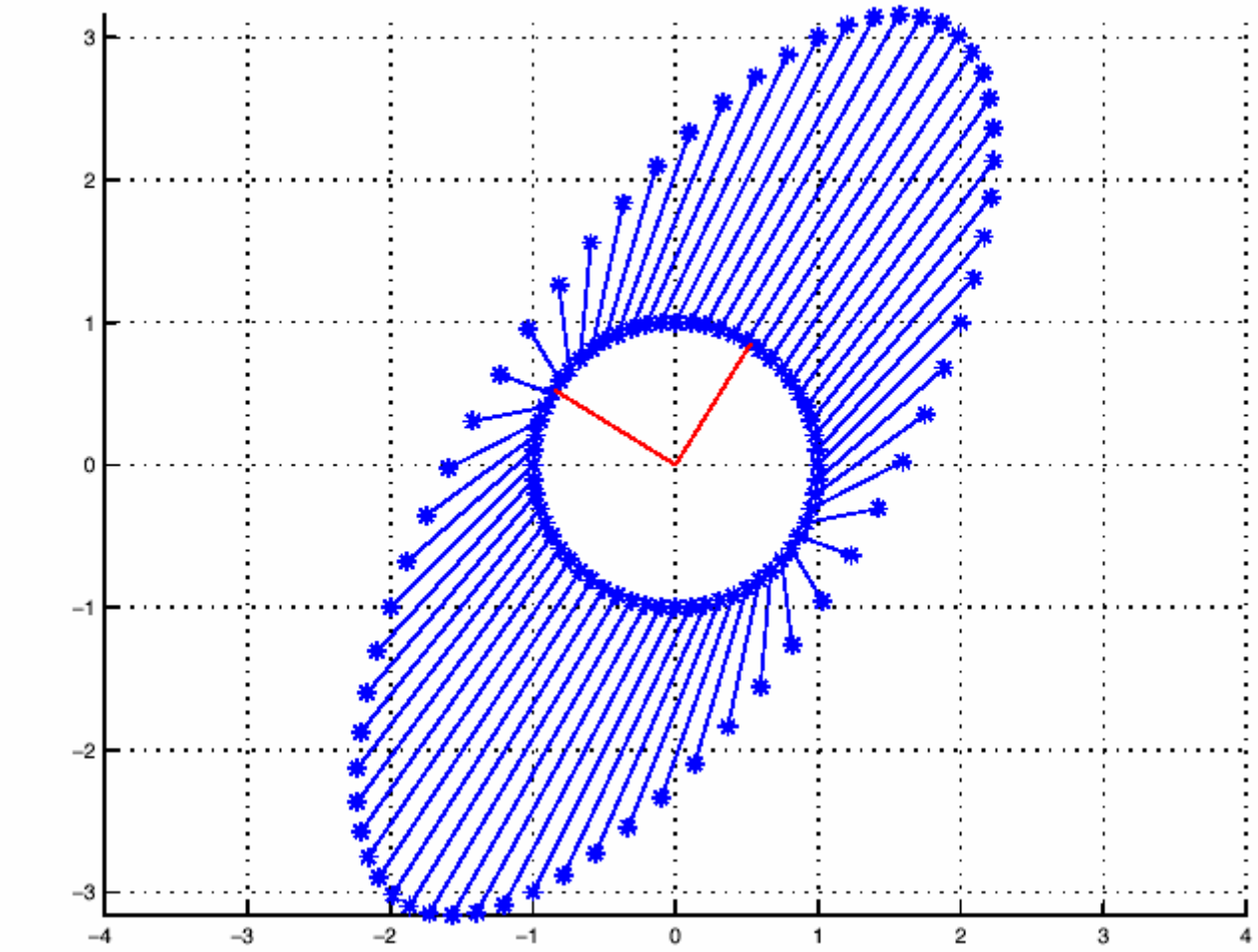
while(w aendert sich significant) do

$w \leftarrow Ew$

$w \leftarrow w/||w||_1$

end while

Beispiel



Warum funktioniert das?

$$w_0 = u = \sum_{i=1}^{n} c_i v_i$$

$$w_t = A^t w_0 / ||A^t w_0||_1 = \sum_{i=1}^{n} c_i \lambda_i^t v_i$$