Universität Potsdam
Institut für Informatik
Lehrstuhl Maschinelles Lernen

Suche im Web und Ranking

Tobias Scheffer

World Wide Web

- 1990 am CERN von Tim Berners Lee zum besseren Zugriff auf Papers entwickelt.
  - HTTP, URLs, HTML, Webserver.
- Verbindet FTP mit der Idee von Hypertext.
- Multilingual (ca. 75% englisch, je 5% japanisch und deutsch).
- Groß, verteilt.
- Volatil: Dokumente erscheinen und verschwinden.
- Unstrukturiert, heterogen.

World Wide Web
Entwicklung

- Hypertext: 1960er Jahre.
- ARPANET / DARPANET / Internet: 1970er.
- FTP: 80er Jahre.
  - Archie: Crawler sammelt Dateinamen von FTP-Servern und ermöglicht Regex-Suche.
    - Suchmaschine, die speziell zum Indizieren von FTP-Archiven entwickelt wurde.
    - Problem: Da nur die Datei- und Verzeichnisnamen für die Indizierung verwendet werden, sind die Suchanfragen auf diesen Index beschränkt.
  - Gopher.

World Wide Web
Entwicklung - Gopher

- Ähnelt dem frühen WWW.
- Wurde an der Universität von Minnesota von Mark P. McCahill entwickelt.
- **Idee**: Umständliche Handhabung des FTP-Protokolls umgehen; Schaffung eines einfach zu administrierenden Informationssystems.
  - Verzeichniswechsel durch Konsolenbefehle, um gewünschte Dateien herunter laden zu können.

World Wide Web
Entwicklung

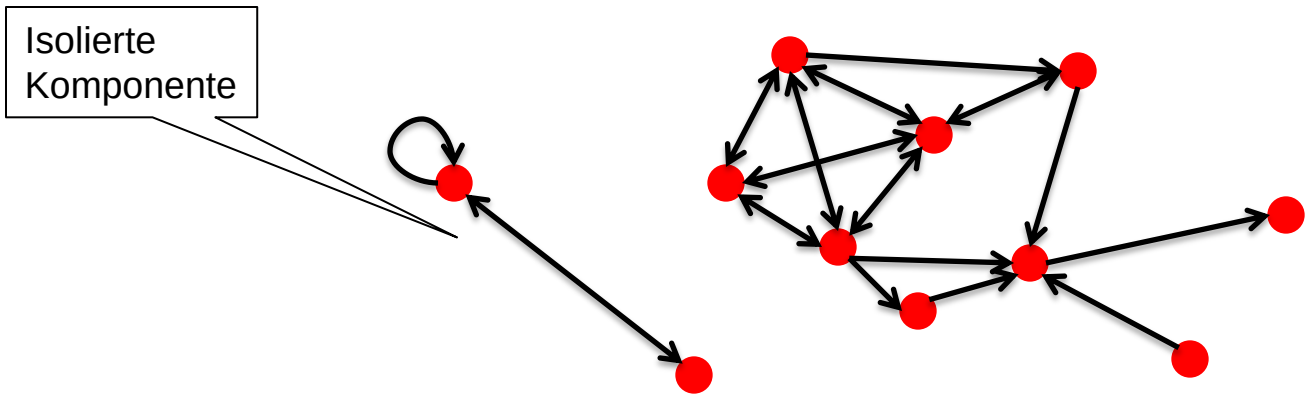
- Web-Browser: Anfang der 90er von Doktoranden.
  - Mosaic: Marc Andreessen & Eric Bina.
  - Netscape und IE aus Mosaic hervorgegangen.
- Web-Directories: Verzeichnis „Yahoo“, 1994, zwei Stanford-Doktoranden.
  - Bilden Einstiegspunkt, um im Web zu surfen.
  - Meist kombiniert mit Suchmaschine.

World Wide Web
Suche

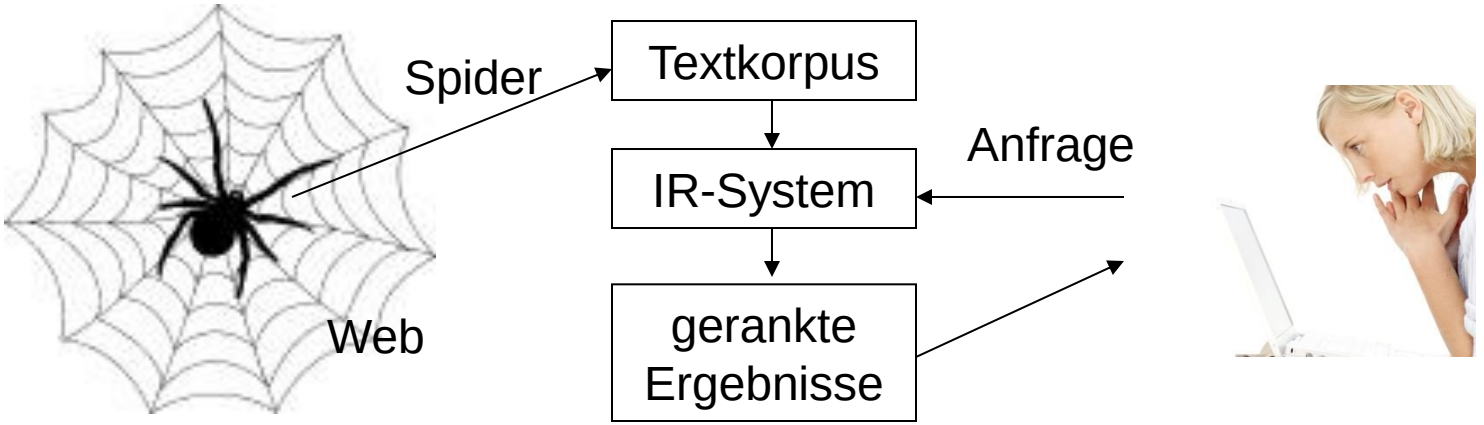
- WebCrawler: Programmier-Projekt, Uni Washington, 1994. → Excite, AOL.
- 1994: Doktorand an CMU entwickelt Lycos.
- 1995: DEC entwickelt Altavista.
- 1998: Doktoranden aus Stanford entwickeln Google.
  - Idee:
    - Analyse der Linkstruktur;
    - Relevantere Ergebnisse extrahieren.

World Wide Web
Graph-Struktur

- Große zentrale, stark verknüpfte Komponente.
  - Jede Seite von jeder Seite erreichbar.
- Randgebiete die nur
  - Auf die zentrale Komponente verweisen;
  - Von zentraler Komponente aus verwiesen werden.
- Isolierte Komponenten.



World Wide Web
Suche



- Unterschied zu anderen IR-Systemen:
  - Spider / Crawler sammelt Texte für Indexstruktur.
  - Analyse der Linkstruktur liefert Relevanzinformation.
  - Suchmaschinen sind stark verteilte Systeme.

Crawler

- **Crawling**: Sammeln von Internetseiten, um sie indexieren zu können.
- **Aufgaben**:
  - Schnell neue Internetseiten besuchen.
  - Effizient arbeiten.
    - So viele brauchbare Internetseiten wie möglich in kurzer Zeit bearbeiten;
    - Relevante Informationen extrahieren und speichern.
- Implementierung von lauffähigen Crawlern sind im Netz frei verfügbar.

Crawler
Eigenschaften

- **Stabilität**: Crawler muss in der Lage sein, Spider-traps zu umgehen.
  - Zyklen, dynamische Internetseiten.
- **Fair**: Crawler darf eine Seite nicht beliebig oft in kurzen Zeitabständen besuchen.
  - „Denial of Service Attack“.
- **Verteilt**: Crawler sollte auf mehreren Maschinen verteilt sein.
  - Mehrere Threads laufen parallel.
- **Skalierbar**: Crawler sollte erweiterbar sein.
  - Mehrere Maschinen, höhere Bandbreite.

Crawler
Eigenschaften

- **Effizienz**: Intelligente Nutzung des verfügbaren Speichers, der Bandbreite und des Prozessors.
  - Möglichst wenige Prozesse die „idle“ sind.
- **Qualität**: Die wichtigsten Seiten sollten zuerst besucht werden.
  - Internetseiten mit hohem Informationsgehalt wichtiger als private Internetseiten.
- **Aktualität**: Der Crawler sollte das Internet kontinuierlich crawlen.
  - Die Frequenz, mit der eine Internetseite besucht wird, sollte sich an ihrer Änderungsfrequenz orientieren.
- **Erweiterbar**: Sollte um neue Dateiformate erweitert werden können (XML,…).

Crawler
Suchstrategien

- **Strategien**:
  - Breadth first,
  - Depth first,
  - Hoher PageRank first.

- Crawlhäufigkeit in Abhängigkeit von:
  - Änderungshäufigkeit und PageRank (WallstreetJournal.com vs. HottesSockenShop.de).
  - Der Zugriffshäufigkeit auf die Seite.

- Crawling zu den richtigen Zeitpunkten bei zyklischen Updates.

Crawler
Arbeitsweise

Algorithmus:

1. Beginne mit einem Pool von URLs.
2. Besuche die URLs des Pools und extrahiere den Text und die Links der besuchten URLs.
3. Übergebe den Text einer URL an den Indexer und füge gefundene Links zum Pool hinzu.
4. Füge bereits besuchte URLs wieder in den Pool ein, um sie später erneut besuchen zu können.
   - Inhalte von Internetseiten können sich ändern.

- Der Graph des Internets wird dabei mehrmals traversiert.

Crawler
Arbeitsweise

- Um eine Milliarde Internetseiten in einem Monat zu crawlen, müssen Hunderte Internetseiten pro Sekunde besucht werden.
- Dabei ist zu beachten:
  - Links, die während des Crawlens gefunden wurden, können relative Links sein.
    - Normalisierung aller relativen URLs.
  - Links können mehrmals auftauchen.
    - Duplikaterkennung.
  - Links können auf Webseiten verweisen, die nicht automatisch besucht werden dürfen.
    - robot.txt

Crawler
Architektur

- **URL Frontier**: Liste von URLs die besucht werden sollen.
- **DNS-Auflösung**: Bestimmung der IP-Adresse für eine gegebene URL.
- **Fetching-Modul**: Lädt Internetseiten herunter, damit sie bearbeitet werden können.
- **Parsing-Modul**: Extrahiert Text und Links einer Internetseite.
- **Dublikaterkennung**: Erkennt URLs und Inhalte, die bereits kurze Zeit vorher bearbeitet wurden.

Crawler
Architektur

- Beim Parsen einer Internetseite werden die Ankertexte als Indexterme für die verlinkte Seite abgelegt.

```
<a href="http://example.com/">Ankertext</a>
```

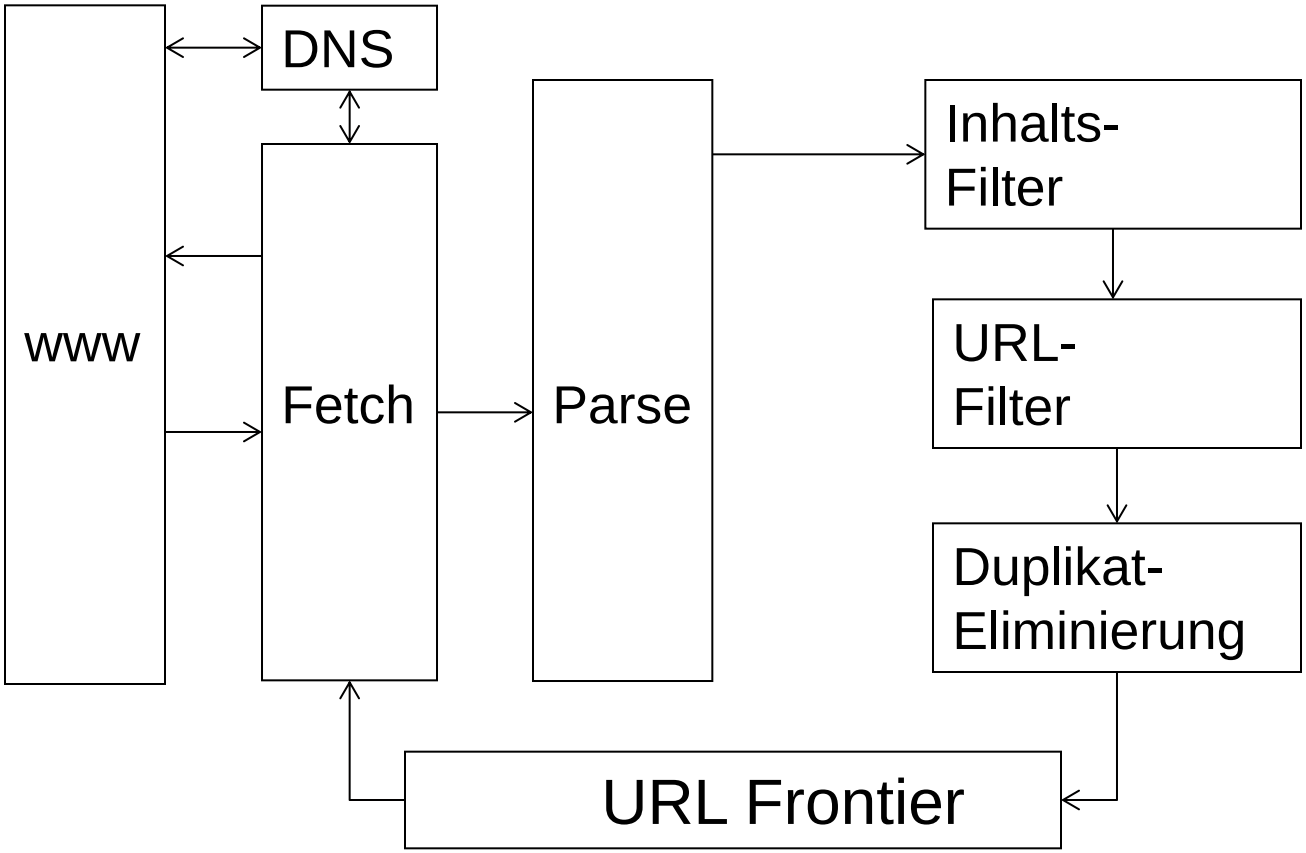
- Bevor ein Link zur URL Frontier hinzugefügt wird, muss überprüft werden:
  - Ob diese URL bereits vorhanden ist.
  - Ob die Internetseite ein inhaltliches Duplikat einer anderen Internetseite der URL Frontier ist.
  - Ob es erlaubt ist, die Internetseite zu crawlen (robot.txt).

Beispiel für robot.txt:

```
User-agent: mein-Robot
Disallow: /quellen/dtd/

User-agent: *
Disallow: /unsinn/
Disallow: /temp/
Disallow: /newsticker.shtml
```

Crawler
Architektur



Crawler
URL Frontier

- **Anforderungen**:
  - Jede URL besitzt Score der angibt, wie relevant eine URL ist.
  - Es darf nur eine Verbindung zu einer Domain zur selben Zeit aufgebaut werden.
  - Zwischen zwei Anfragen auf eine Domain, muss eine bestimmte Zeit gewartet werden.
    - „DOS-Attacken“.
  - Soll alle Threads mit URLs versorgen.
    - Keine „idle“ Threads.

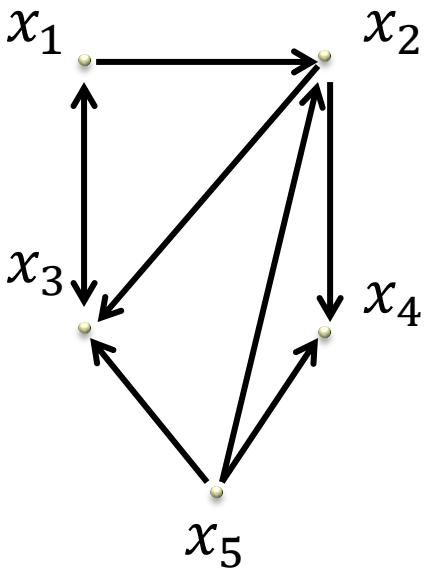
Ranking

- Ende der 90er wurden alle Suchmaschinen in Web-Portale mit breitem Informations- und Unterhaltungsangebot umgebaut.
- Gegenannahme von Google: nur gute Suchfunktion ist wichtig, der Rest interessiert niemanden.
- Idee von Kleinberg (HITS) und Page & Brin (PageRank): Verweisstruktur zeigt, welche Webseiten irgend jemanden interessieren.
- Suchergebnisse werden nach PageRank bzw. HITS sortiert.

Webgraph

- Darstellung des Webgraphen als Adjazenzmatrix

$$A_{ij} = 1 \text{ falls Link von Seite } x_i \text{ nach } x_j$$
$$A_{ij} = 0 \text{ sonst}$$



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

Authority-Ranking
PageRank

- Ranking der Seiten.
- PageRank ist umso höher, je mehr andere Seiten auf die Seiten verweisen, und je höher deren PageRank ist.
- Wird auf das ganze Web angewandt.

Authority-Ranking
PageRank

- Random Surfer:
  - Beginnend bei einem beliebigen Knoten folgt der Surfer mit Wahrscheinlichkeit 1-ε einem zufälligen Link.
  - Mit Wahrscheinlichkeit ε startet er neu an Zufallsknoten.
  - Wahrscheinlichkeit für Aufenthalt an einem Knoten = globaler Authority Score.

23

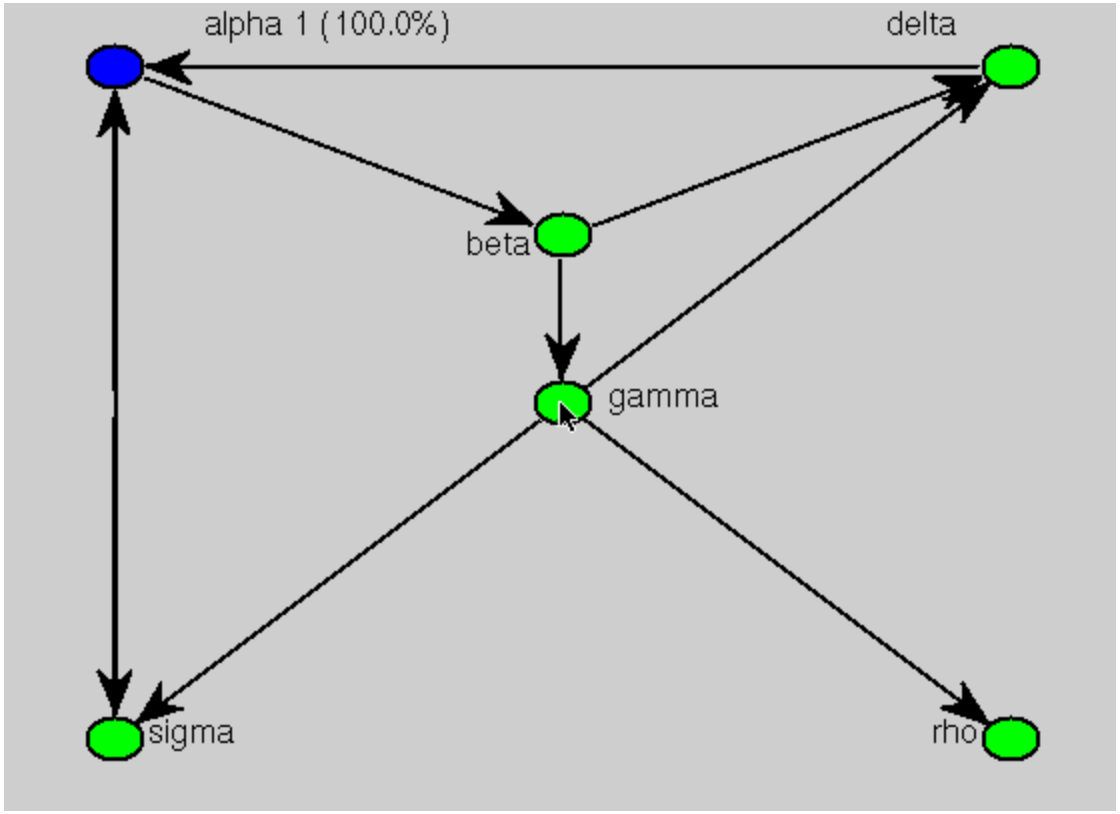
Authority-Ranking
PageRank

- **PageRank**: Random Surfer Modell zum Ranking von Webseiten.
  - (Ursprünglicher) Ranking-Algorithmus von Google.
  - Abhängig von Query werden relevante Webseiten gefunden und nach ihrem globalen Authority Score sortiert.
- **Annahmen**:
  - Link auf Webseite $\mathbf{x}_i$ verweist auf „kompetente“ Webseite $\mathbf{x}_j$, d.h. aus der Adjazenzmatrix des Webgraphen kann eine Authority Matrix erstellt werden mit $A_{ij} > 0$.
  - Mit Wahrscheinlichkeit $1 - \varepsilon$ folgt der Nutzer (Random Surfer) einem Link auf der Webseite.
  - Mit Wahrscheinlichkeit ε wechselt er auf eine zufällige Webseite.

24

Authority-Ranking
PageRank

- Beispiel:



25

Authority-Ranking
PageRank - Formal

- **Gegeben**: Instanzen (Webseiten) $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ mit gegebenen lokalen *Authority Scores* (Kompetenz-Bewertungen):
  - Authority Score $A_{ij} = a(\mathbf{x}_i, \mathbf{x}_j)$ gibt an wie „kompetent“ $\mathbf{x}_j$ aus Sicht von $\mathbf{x}_i$ ist.
  - Link oder kein Link, Link-Position.

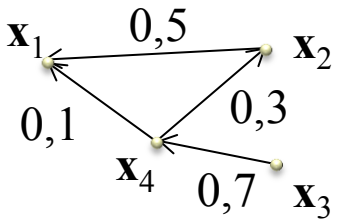
Authority Matrix

- **Gesucht**: Modell $f : A \in R^{n \times n} \mapsto s \in R^n$ welches für Instanzen $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ globale Authority Scores (Ranking) s_i liefert.
  - Annahme: Je kompetenter $\mathbf{x}_i$ und je höher der Authority Score A_{ij}, desto kompetenter $\mathbf{x}_j$.

26

Authority-Ranking
PageRank - Modellierung

- Modellierung der (nicht-symmetrischen) Kompetenz-Bewertungen als gerichteter Graph:
  - Instanzen sind Knoten, Authority Scores sind Kanten-Gewichte
    $\Rightarrow$ Authority Matrix = (gewichtete) Adjazenzmatrix $\mathbf{A}$.



$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0,5 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0,7 \\ 0,1 & 0,3 & 0 & 1 \end{bmatrix} \qquad \mathbf{D} = \begin{bmatrix} d_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & d_n \end{bmatrix} \qquad d_i = \sum_{j=1}^{n} A_{ij}$$

  - Intuition: (normierter) Graph beschreibt mit welcher Wahrscheinlichkeit Knoten $\mathbf{x}_i$ Knoten $\mathbf{x}_j$ als „Experten“ nennen würde.
  - Beispiel: $\mathbf{x}_4$ hält $\mathbf{x}_2$ für 3-mal so kompetent wie $\mathbf{x}_1$, $\mathbf{x}_3$ ist aus seiner Sicht kein Experte.

27

Authority-Ranking
PageRank

- **Gegeben**: Wahrscheinlichkeit dafür, dass der Nutzer von Webseite $\mathbf{x}_i$ zu Webseite $\mathbf{x}_j$ wechselt ist

$$P_{ij} = p(\mathbf{x}_j \mid \mathbf{x}_i) = (1-\varepsilon)\frac{A_{ij}}{\sum_{k=1}^{n} A_{ik}} + \varepsilon\frac{1}{n}$$

Transitionswahrscheinlichkeit

und somit $\mathbf{P} = (1-\varepsilon)\mathbf{D}^{-1}\mathbf{A} + \varepsilon\mathbf{U}$ mit $U_{ij} = \frac{1}{n}$.

- **Gesucht**: Wahrscheinlichkeit dafür, dass man auf Webseite $\mathbf{x}_i$ ist, d.h. $s_i = p(\mathbf{x}_i)$.

Aufenthaltswahrscheinlichkeit

28

Authority-Ranking
PageRank

- **Algorithmus**: Beginnend mit initialem Ranking Scores $\mathbf{s}$ iterativ neue Scores bestimmen mit

$$\mathbf{s}' = \frac{1}{c}\mathbf{P}^{\mathrm{T}}\mathbf{s} \quad \text{wobei} \quad c = \left\|\mathbf{P}^{\mathrm{T}}\mathbf{s}\right\| \Rightarrow \left\|\mathbf{s}'\right\| = 1$$

- Konvergenz von PageRank bei $\mathbf{s}' = \mathbf{s}$, sodass gilt

$$\mathbf{s} = \frac{1}{\lambda}\mathbf{P}^{\mathrm{T}}\mathbf{s}$$

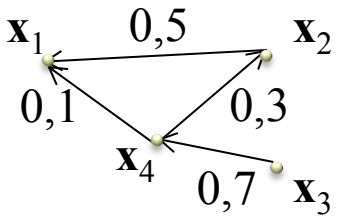
d.h. $\boldsymbol{s}$ ist ein Eigenvektor von $\mathbf{P}^{\mathrm{T}}$ mit Eigenwert λ.

- Man kann zeigen, dass $\mathbf{s}$ der Eigenvektor mit größtem Eigenwert λ ist.

29

Authority-Ranking
PageRank

- Beispiel:



$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0{,}5 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0{,}7 \\ 0{,}1 & 0{,}3 & 0 & 1 \end{bmatrix} \quad D^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{2}{3} & 0 & 0 \\ 0 & 0 & \frac{10}{17} & 0 \\ 0 & 0 & 0 & \frac{5}{7} \end{bmatrix}$$

$$P = (1-\varepsilon)D^{-1}A + \varepsilon U = \begin{bmatrix} \frac{397}{400} & \frac{1}{400} & \frac{1}{400} & \frac{1}{400} \\ \frac{133}{400} & \frac{53}{80} & \frac{1}{400} & \frac{1}{400} \\ \frac{1}{400} & \frac{1}{400} & \frac{417}{713} & \frac{865}{2109} \\ \frac{41}{560} & \frac{601}{2800} & \frac{1}{400} & \frac{655}{923} \end{bmatrix}, \text{ mit } \varepsilon = 0.01$$

$$s' = \frac{1}{\|P^T s\|} P^T s = \begin{bmatrix} 0{,}6711 \\ 0{,}4227 \\ 0{,}2838 \\ 0{,}5389 \end{bmatrix} \xRightarrow{rekursiv\ einsetzen} \begin{bmatrix} 0{,}997 \\ 0{,}0186 \\ 0{,}0062 \\ 0{,}0176 \end{bmatrix}, \text{ mit } s_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix},$$

30

Authority-Ranking
PageRank

- **Vorteile**:
  - Leicht und effizient berechenbar.
  - Existenz & Eindeutigkeit der Lösung sowie Konvergenz des Algorithmus ist garantiert für $0 < \varepsilon < 1$.

- **Nachteile**:
  - Links können schlechte Indikatoren für Kompetenz-Bewertung sein:
    - Kompetenz zweier Instanzen kann sich gegenseitig verstärken.
    - Automatisch generierte Links haben kaum Aussagekraft.
    - „Künstliche“ Links (z.B. Link-Spam) verändern Ranking.
  - Eigenschaften der Instanzen fließen nicht ins Ranking ein.

31

PageRank
Link Spam

- PageRank ist beeinflussbar.
- Link Farmen:
  - Felder künstlich generierter Seiten, deren PageRank „geerntet“ und zu Zielseite geleitet wird.
- Guestbook Spam:
  - Generierte Einträge in Gästebüchern und Blogs, die Verweis auf Zielseite enthalten.
- Link Exchange Services:
  - Seiten mit Links auf (thematisch nicht verwandte) Seiten, meist gegen Geld.
- Partner-Programme: z.B. Amazon, Ebay.
  - Link auf Produktseite gegen Umsatzbeteiligung.

Link Spam
BadRank

- Wenn PageRank beeinflussbar wird, dann verliert er seine Korrelation zur Relevanz der Seiten.
- Link Spam sollte bei der Berechnung des Page Ranks nicht so berücksichtigt werden wie „natürliche“ Links.
- Suchmaschinenbetreiber haben „Blacklists“.
  - URLs von Link-Spam-Seiten.
  - Werden manuell erstellt.

Link Spam
BadRank

- Invertierter PageRank-Algorithmus, „bestraft“ Seiten, die auf Link Spam verweisen.
- Initialisierung: B(u) = 1, wenn u auf Blacklist.
- Für alle Seiten:

$$B'(u) = \sum_{v:u \to v} \frac{B(v)}{N_u}$$

Gemittelter BadRank der Links eines Knoten

$$B(u) = \frac{B'(u)}{|B'(u)|}$$

Normalisierung

- BadRank wirkt wie negativer PageRank.

World Wide Web
Benutzerschnittstellen

- Advanced Search:
  - Wird fast nie benutzt.
  - Ähnliche Seiten finden,
  - Links auf Seite finden,
  - Maschinelle Übersetzung, cross-language retrieval.
- Clusterung von Seiten

World Wide Web
Retrieval - Websuche

- Vereinfachter möglicher Ablauf einer Websuche
  - Suchterme werden eingeben
  - Index liefert Liste von URLs
  - Ranking der Ergebnisse mit Hilfe von PageRank

- Ranking der Ergebnisse nicht nur auf Grund des Webgraphen und anderer Linkinformationen.

- Idee: Erstelle Ranking der URLs durch Verwenden vieler Merkmale und Gewichtung der einzelnen Komponenten.

World Wide Web
Retrieval - Websuche

- Aufbau eines Ähnlichkeitsvektors der URL x mit verschiedenen Merkmalen

$$\Phi(x,q) = \begin{pmatrix} sim(x,q) \text{ im Vektorraummodell} \\ \#\text{ gleicher Terme in } \langle \text{h1} \rangle\text{-Tags} \\ \dots \\ \text{PageRank}(x) \end{pmatrix}$$

- Ranking der Seiten auf Grund von Scores $w\Phi(x,q)$
- *w* kann manuell konstruiert werden.
- Besser: Lernen von *w* aus Klickdaten.

RankSVM

- **Idee**: Lernen des Gewichtsvektors durch Klickdaten.

1. SVM-Support Vector Machines
www.support-vector-machines.org/
2. SVM-Light Support Vector Machine
svmlight.joachims.org/
3. Kernel-*Machines*.Org — Kernel *Machines*
www.kernel-machines.org/
***4. Support Vector Machines* - The Book**
www.support-vector.net/
5. *Support Vector Machines*
www.svms.org/
6. SVM - Support Vector Machines
www.dtreg.com/svm.htm

$$l_1 > l_2$$
$$l_1 > l_3$$
$$l_1 > l_5$$
$$l_1 > l_6$$
$$l_4 > l_2$$
$$l_4 > l_3$$
$$l_4 > l_5$$
$$l_4 > l_6$$

RankSVM

- **Eingabe**:
  - Paare von Suchanfragen mit partiellem **Ranking**:
    - $(q_1, r_1'), (q_2, r_2'), \ldots, (q_n, r_n')$
    - $r_i' = \{(d_1, d_2), (d_1, d_3), \ldots, (d_j, d_k)\}$

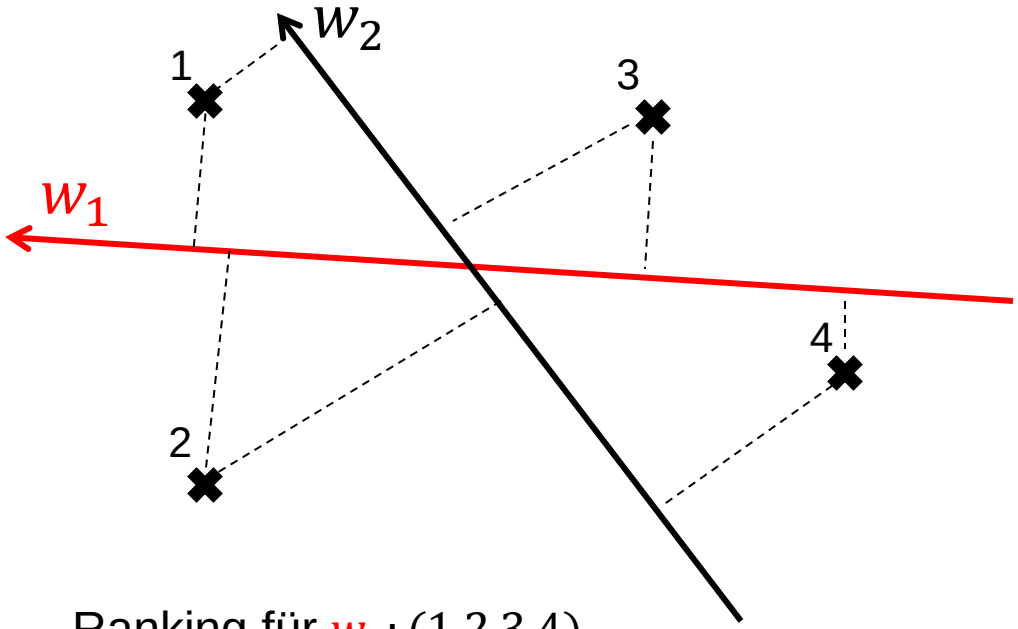
$$d_1 > d_2$$
$$d_1 > d_3$$
$$\vdots$$
$$d_j > d_k$$

- **Ziel**: Gewichtsvektor w, so dass für alle Relationen (d_i, d_j) einer Anfrage q gilt:
  - $w^T \Phi(d_i, q) > w^T \Phi(d_j, q)$

$$w^T \Phi(d_1, q_i) > w^T \Phi(d_2, q_i)$$
$$w^T \Phi(d_1, q_i) > w^T \Phi(d_3, q_i)$$
$$\vdots$$
$$w^T \Phi(d_j, q_i) > w^T \Phi(d_k, q_i)$$

RankSVM

- Beispiel:



Ranking für w_1: $(1,2,3,4)$

Ranking für w_2: $(1,3,2,4)$

RankSVM

- RankSVM Optimierungsproblem:

Minimiere: $\frac{1}{2}||w||^2 + C\sum\xi_{i,j,k}$

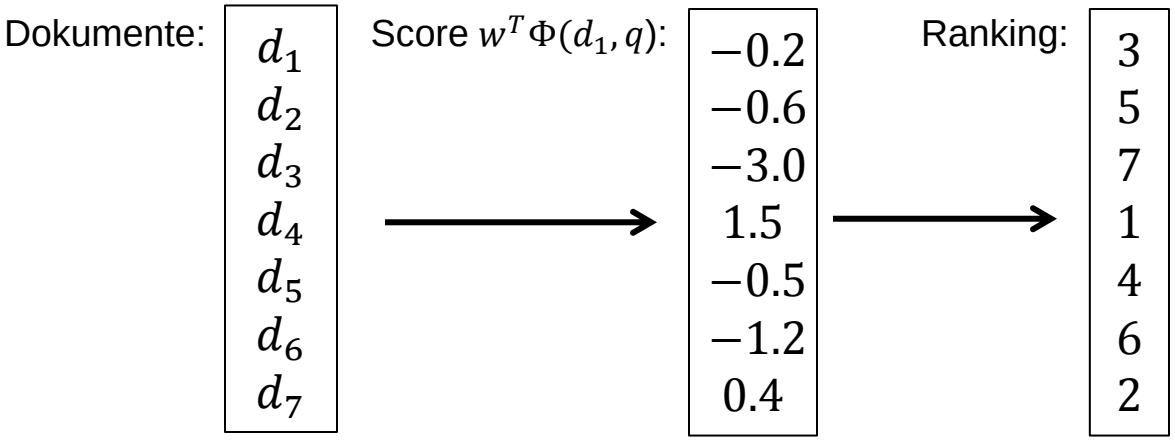
so, dass: $\forall(d_i, d_j) \in r_1' : w^T\Phi(d_i, q_1) > w^T\Phi(d_j, q_1) + 1 - \xi_{i,j,1}$

$\vdots$

$\forall(d_i, d_j) \in r_k' : w^T\Phi(d_i, q_k) > w^T\Phi(d_j, q_k) + 1 - \xi_{i,j,k}$

$\forall i \forall j \forall k : \xi_{i,j,k} \geq 0$

- Bestimmung des optimalen Rankings für Anfrage q:



RankSVM
Evaluation

- Kendall‘s τ: Bestimmt den Grad der Übereinstimmung zweier Rankings.
- $\tau = \frac{2P}{\frac{1}{2}n(n-1)} - 1$

  $2P$: Anzahl von Übereinstimmungen

  $\frac{1}{2}n(n-1)$: Anzahl von Relationen
- τ kann Werte von -1 (keine Übereinstimmung) bis +1 (perfekte Übereinstimmung) annehmen.
- Beispiel:
  - Ranking 1: A,B,C,D

    $A > B; A > C; A > D; B > C; B > D; C > D$
  - Ranking 2: B,C,A,D

    $B > C; B > A; B > D; C > A; C > D; A > D$
  - $\tau = \frac{2*4}{\frac{1}{2}4(4-1)} - 1 = \frac{1}{3}$

Zusammenfassung

- Crawling
- Ranking
  - PageRank
  - BadRank
- Lernen von Rankingfunktionen