

Duplikatfilterung und Sampling von Webseiten

**Seminar Suchmaschinen,
Wintersemester 2007/2008**

Martin Sauerhoff

Lehrstuhl 2, Universität Dortmund

1. Duplikatfilterung:

1.1 Gleichheitstest mit Fingerabdrücken

1.2 Min-Hashing

2. Sampling von Webseiten:

2.1 Anfragebasiertes Sampling (kurz)

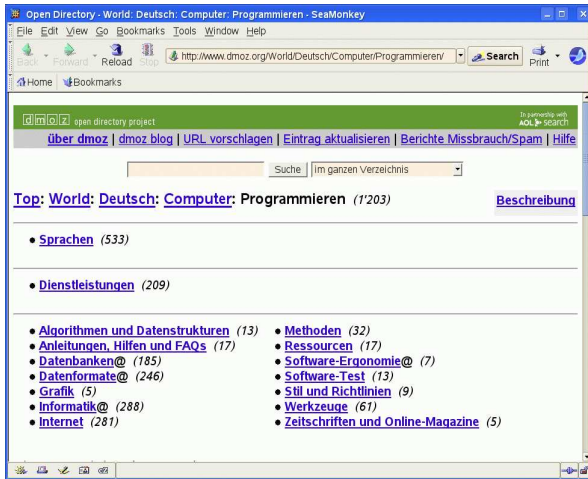
2.2 Random-Walk-basiertes Sampling

1. Duplikatfilterung – Einleitung

Problem für Web-Tools:

Viele Beinahe-Kopien von Seiten im Web.

Beispiel: Seite im dmoz-Verzeichnis (Original)

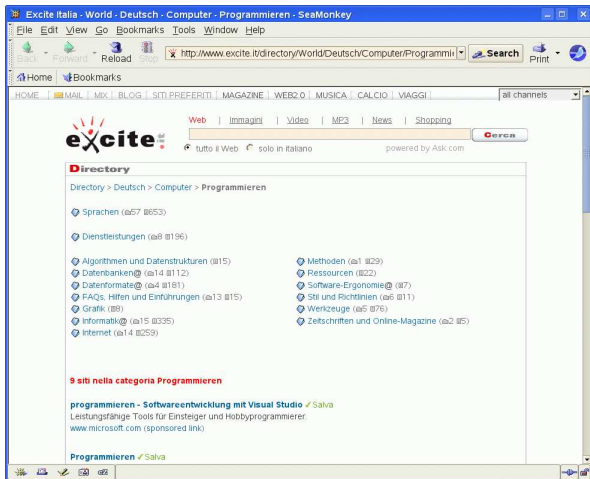


1. Duplikatfilterung – Einleitung

Problem für Web-Tools:

Viele Beinahe-Kopien von Seiten im Web.

Beispiel: Seite im dmoz-Verzeichnis (Beinahe-Kopie)



1. Duplikatfilterung – Einleitung

Problem für Web-Tools:

Viele Beinahe-Kopien von Seiten im Web.

Beispiel: Seite im dmoz-Verzeichnis (Beinahe-Kopie)

The screenshot shows a Netscape browser window with the address bar displaying `http://www.gesundheitsseiten24.de/webkatalog.html?no_cache=1`. The page content is a directory listing for 'Sprachen' (Languages) under the path 'Webkatalog / World / Deutsch / Computer / Programmieren /'. The listing includes categories like 'Algorithmen und Datenstrukturen', 'Anleitungen, Hilfen und FAQs', 'Datenbanken', 'Datenformate', 'Grafik', 'Informath', 'Internet', 'Methoden', 'Ressourcen', 'Software-Ergonomie', 'Software-Test', 'Stil und Richtlinien', and 'Werkzeuge'. A sidebar on the left contains navigation links such as 'Arztuche Online', 'Fachgebiete', 'Gesundheit aktuell', 'Gesundheitsberichte', 'Gesundheit auf Reisen', 'Thema der Woche', 'Gesund leben', 'Fitness & Wellness', 'Schönheit & Ästhetik', 'Impfung', 'Insektenstiche', 'Ernährung', 'Gesunde Ernährung', '5 Elemente Ernährung', 'Diäten & Abnehmen', 'Fastenkuren', 'Körperrehabilitation', 'Infos zur Kur', 'Kur beantragen', 'Kurorte', and 'Kurlhotels'. A sidebar on the right features an advertisement for 'SONNENLAB 39,99 € 55 € 1 MONAT' and text about satellite internet access.

Webkatalog / World / Deutsch / Computer / Programmieren / - SeaMonkey

File Edit View Go Bookmarks Tools Window Help

Back Forward Reload Stop `http://www.gesundheitsseiten24.de/webkatalog.html?no_cache=1` Search Print

Home Bookmarks

Arztuche Online

Fachgebiete

Gesundheit aktuell

Gesundheitsberichte

Gesundheit auf Reisen

Thema der Woche

Gesund leben

Fitness & Wellness

Schönheit & Ästhetik

Impfung

Insektenstiche

Ernährung

Gesunde Ernährung

5 Elemente Ernährung

Diäten & Abnehmen

Fastenkuren

Körperrehabilitation

Infos zur Kur

Kur beantragen

Kurorte

Kurlhotels

Webkatalog

Top / World / Deutsch / Computer / Programmieren /

- [Sprachen](#) (547)
- [Dienstleistungen](#) (199)

• [Algorithmen und Datenstrukturen](#) (12)

- [Anleitungen, Hilfen und FAQs](#) (13)
- [Datenbanken@](#) (191)
- [Datenformate@](#) (244)
- [Grafik](#) (5)
- [Informath@](#) (291)
- [Internet](#) (253)

- [Methoden](#) (29)
- [Ressourcen](#) (13)
- [Software-Ergonomie@](#) (7)
- [Software-Test](#) (15)
- [Stil und Richtlinien](#) (9)
- [Werkzeuge](#) (59)
- [Zeitschriften und Online-Magazine](#) (5)

Diese Kategorie in anderen Sprachen:

Arabisch (29)	Aserbaidschanisch (0)	Englisch (19'095)
Bosnisch (1)	Chinesisch vereinfacht (189)	Chinesisch (65)
Dänisch (124)	Esperanto (7)	Farsi (3)
Finnisch (58)	Französisch (803)	Hebräisch (12)
Indonesisch (8)	Italienisch (290)	Japanisch (358)
Katalanisch (19)	Koreanisch (36)	Kroatisch (6)
Litauisch (18)	Malaisch (1)	Mazedonisch (0)
Niederländisch (54)	Norwegisch (19)	Polnisch (176)
Portugiesisch (100)	Rumänisch (25)	Russisch (355)
Schwedisch (49)	Slowakisch (52)	Spanisch (444)
Thailändisch (21)	Tschechisch (136)	Türkisch (100)
Ungarisch (6)	Vietnamesisch (0)	

Flächen-deckend verfügbar!

Unbegrenzt surfen - Überall!

Highspeed Internet via Satellit!

Keine Anschluss-Wartzeit

SONNENLAB 39,99 € 55 € 1 MONAT

Betrachte hier folgende Probleme:

- Test, ob zwei Dokumente exakt gleich sind.
- Test, ob zwei Dokumente „ähnlich“ sind.

Anwendungen:

- Web-Crawling (Mercator: „Content-seen-Test“);
- Spam-Entdeckung;
- Entdecken von Plagiaten.

Wie messen wir Ähnlichkeit? (Beispiele)

- **Für Zeichenketten (Bitstrings, Texte...):**

Zeichenweiser Vergleich, Anzahl Unterschiede.
(Hammingabstand)

- **Für Mengen (z. B. vorkommende Stichwörter):**

Mengen X, Y mit $X \cup Y \neq \emptyset$:

$$J(X, Y) := \frac{|X \cap Y|}{|X \cup Y|} \in [0, 1].$$

(Jaccard-Koeffizient)

(Natürlich gibt es viele weitere Ähnlichkeitsmaße!)

Typisches Szenario:

Vergleiche „neues“ Dokument T mit großer Kollektion $C = \{T_1, \dots, T_k\}$ von „gespeicherten“ Dokumenten.

Naiver Ansatz: Direkter Vergleich mit allen Dokumenten.

Falls alle Dokumente mit Länge n : Zeit für Vergleich mit allen Dokumenten mindestens $\Theta(|C| \cdot n)$.

$|C| = 10^{10}$, $n = 10^3$ Wörter \rightarrow zu viel!

Idee:

Vergleiche nur viel kürzere **Hashwerte** der Dokumente.

Hashingbasierter Dokumentvergleich

- **Extrahiere für Vergleich relevante Features:**

Dokument \rightarrow Menge / Vektor, Eintrag für Vorhandensein / Gewicht des jeweiligen Features.

Beispiele:

- Trivialer Ansatz: vorkommende (Stich-)Wörter.
- Gewichtsvektor mit TF-IDF-Werten.

Dokument $T \mapsto$ Menge / Vektor F_T .

- **Hashing:**

$F_T \mapsto$ Hashwert $h(F_T) \in M$, M „klein“.

- **Vergleich von Dokumenten T_1, T_2 :**

- Exakte Gleichheit: $h(F_{T_1}) = h(F_{T_2})?$
- Ähnlichkeit: $h(F_{T_1}) \approx h(F_{T_2})?$

Hashingbasierter Dokumentvergleich (Forts.)

- Statt mindestens linearer Zeit in Dokumentlänge / Featureanzahl nur noch lineare Zeit in Bitlänge der Hashwerte (bei geeigneter Hashfunktion).
- Preis: Kollisionen. Verschiedene Dokumente mit gleichen Hashwerten (*False Positives*).
- Je nach Vorgehensweise auch ähnliche Dokumente, die nicht entlarvt werden (*False Negatives*).

Wesentliche Frage: Geeignete Hashfunktionen?

1.1 Gleichheitstest mit Fingerabdrücken

Problem: Teste exakte Gleichheit für zwei Objekte aus „großem“ Universum $U = \{0, \dots, 2^n - 1\}$.

Z. B. $n = 2^{13}$ für 1 KB große HTML-Seiten.

Klassische Lösung: [Rabin \(1981\)](#).

Algorithmus (einfache Version):

- Wähle p aus ersten m Primzahlen zufällig gleichverteilt \rightarrow Abbildung $h_p: \{0, \dots, 2^n - 1\} \rightarrow \{0, \dots, p - 1\}$:
$$h_p(x) := x \bmod p.$$
- Speichere anstatt Wert $x \in U$ nur $h_p(x)$.
- Test „ $x \stackrel{?}{=} y$ “: Vergleiche $h_p(x)$ und $h_p(y)$.

Satz:

Sei Menge S von Elementen aus $U = \{0, \dots, 2^n - 1\}$ abzuspeichern. Dann gilt für $x, y \in S$:

- $x = y$: $h_p(x) = h_p(y)$.
- $x \neq y$: $\Pr_p\{h_p(x) = h_p(y)\} \leq \binom{|S|}{2} \cdot \frac{n}{m}$.

Also:

- Keine False Negatives.
- Für $m = O((1/\varepsilon) \cdot |S|^2 \cdot n)$ Wskt. für False Positives durch $\varepsilon > 0$ beschränkt.

Beweis:

Fall $x = y$: Dann gilt für alle p :

$$h_p(x) = x \bmod p = y \bmod p = h_p(y).$$

Fall $x \neq y$:

- Jedes $z \in \{0, \dots, 2^n - 1\}$ hat höchstens n Primteiler:

$$z = p_1 \cdot \dots \cdot p_k, \quad p_1, \dots, p_k \geq 2.$$

- Seien $x, y \in U$ mit $x \neq y$ und $h_p(x) = h_p(y)$, d. h.,

$$x \equiv y \bmod p \Leftrightarrow (x - y) \equiv 0 \bmod p.$$

Also p teilt $|x - y|$. Wskt. dafür höchstens n/m .

- Insgesamt $|S|$ Elemente gespeichert:

Wskt., dass einer von $\binom{|S|}{2}$ möglichen Tests fehlschlägt:

$$\leq \binom{|S|}{2} \cdot n/m \text{ (Vereinigungsschranke).}$$



Ressourcen:

- Bitlänge für abzuspeichernde Werte:
 $O(\log m) = O(\log n + \log |S| + \log(1/\varepsilon))$.
(Benutze, dass Größe der m -ten Primzahl $O(m \log m)$ gemäß Primzahlsatz.)
- Rechenzeit für Hashwert-Berechnung:
 $O((\log m)^2)$.

Rabin: Besseres Verfahren mit Zeit $O(\log m)$.

Fazit:

Rabin-Fingerabdrücke liefern effizienten, randomisierten Gleichheitstest.

Anwendung für Ähnlichkeitstest?

Funktioniert nicht!

Beispiel: $n = 10$, $p = 173$

x :	$x \bmod p$:
0101011011	00000001
1101011011	10100111
0001011011	01011011

Arbeit: [Broder \(1998\)](#).

Techniken wie diese erforscht und patentiert u. a. von Altavista und Google.

Übersicht:

- Shingling:
Dokument \rightarrow Menge von Dokumentfragmenten, genannt [Shingles](#) (Dachziegel).
- Ähnlichkeitsmaß:
Jaccard-Koeffizient von Shingle-Mengen.
- Approximation des Jaccard-Koeffizienten durch Hashingtechnik.

Shingling:

Beispiel: 2-Shingles

spam spam spam lovely spam wonderful spam lovely spam
spam spam
 spam spam
 spam lovely
 lovely spam
 spam wonderful
 wonderful spam
 spam lovely
 lovely spam

Shingle-Menge:

{ spam spam, spam lovely, lovely spam, spam wonderful,
wonderful spam}.

Ähnlichkeitsmaß:

- Betrachte allgemein *k-Shingles*:
Fenster der Länge k über Wortfolge schieben.
Wie k wählen? Experimente: $k = 5 \dots 10$.
- Ähnlichkeit von T_1 und T_2 :

$$J(S(T_1), S(T_2)) = \frac{|S(T_1) \cap S(T_2)|}{|S(T_1) \cup S(T_2)|},$$

dabei $S(T)$ zu T gehörige Shingle-Menge.

Speicherung von Shingle-Mengen / Berechnung des Jaccard-Koeffizienten zu aufwendig, da $|S(T)| \approx |T|$.

Min-Hashing – idealer Algorithmus:

- 1 Wähle Permutation π über Universum der Shingles zufällig gleichverteilt.
- 2 Für Shingle-Menge S berechne $h_\pi(S) := \min(\pi(S))$.

Beispiel: $\pi = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 4 & 1 & 2 \end{pmatrix}$, $S = \{2, 3, 4\}$:

$$\pi(S) = \{4, 1, 2\}, h_\pi(S) = 1.$$

Satz:

Für Mengen S_1, S_2 gilt:

$$\Pr_\pi \{h_\pi(S_1) = h_\pi(S_2)\} = J(S_1, S_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}.$$

Also: Shingle-Menge $S \rightarrow$ einzelner Shingle $\min(\pi(S))$.
Weitere Reduktion durch Hashing der einzelnen Shingles.

Algorithmus RABIN-MIN-HASHING:

Wähle unabhängige, zufällige Primzahlen p_1, \dots, p_r für Rabin-Fingerabdruck von Shingles.

Für jedes Dokument T :

- 1 Berechne Shingle-Menge S von T .
- 2 Für $i = 1, \dots, r$: Berechne $m_i := \min(h_{p_i}(S))$.
- 3 Speichere $m(T) := (m_1, \dots, m_r)$ als **Skizze** für T .

Ähnlichkeit von Dokumenten T_1, T_2 :

Anzahl gleicher Komponenten in $m(T_1), m(T_2)$.

Problem grob:

Zufällig gleichverteilte Auswahl einer Webseite

- aus dem Web oder
- aus dem Index einer Suchmaschine.

Randbedingung: Keine internen Infos von Suchmaschinen-Betreibern.

Motivation

- Relative Größe von Suchmaschinen-Indexen:
→ „Suchmaschinen-Kriege“.

©2005 Google - Searching 8,168,684,336 web pages

Mit zusätzlichen absoluten Zahlen zur Indexgröße
auch **absolute Größe des indizierten Webs**.

- Qualität von Suchmaschinen-Indexen,
z. B. Spam-Anteil (mit TrustRank o. ä. kombinieren).
- diverse Statistiken.

Wie macht man's?

Schweres Problem:

- Zugriff auf Suchmaschinen nur über Benutzerschnittstelle.
- Zugriff auf das Web: Gleiche Probleme wie beim Crawling. (Was kann man überhaupt ohne vollständigen Crawl tun?)

Ansätze:

- Anfragebasiertes Sampling:
Typischerweise für Sampling von Suchmaschinen-Index.
[Bharat und Broder \(1998\)](#)
 - Random Walks:
Typischerweise für Sampling aus gesamtem Web.
[Henzinger \(2000\)](#), [Bar-Yossef u. a. \(2000\)](#).
- Fortgeschrittene Techniken: [Bar-Yossef u. a. \(2006\)](#).

Verfahren von Bharat und Broder (1998):

Sampling aus Suchmaschinen-Index.

- Lexikon mit Suchbegriffen (konkret: 400.000 Wörter aus Yahoo!-Crawl).
- **Vorgehensweise:**
 - Anfrage zufällig aus Lexikon wählen.
 - Aus Ergebnismenge der Suchmaschine zufällig gleichverteilt unter ersten k Ergebnissen Seite wählen (konkret: $k = 100$).
- **Ziel:** Nicht zu große, aber nichtleere Ergebnismengen.
Suchanfrage. . .
zu grob: Zu großer Einfluss des Rankings.
zu eng: Zu großer Einfluss der Anfrage.

Anwendung: Größe von Suchmaschinen-Indexen vergleichen

Vergleich von Suchmaschinen-Indexen S_1 , S_2 :

- Bestimme approximativ

$$\Pr\{P \in S_2 \mid P \in S_1\}, \Pr\{P \in S_1 \mid P \in S_2\}.$$

Overlap zwischen den Suchmaschinen. Dazu:

- Sample Seite aus Index S_1 ; checke, ob sie in S_2 .
- Sample Seite aus Index S_2 ; checke, ob sie in S_1 .

- Es gilt:

$$\frac{\Pr\{P \in S_1 \mid P \in S_2\}}{\Pr\{P \in S_2 \mid P \in S_1\}} = \frac{\Pr\{P \in S_1\}}{\Pr\{P \in S_2\}}.$$

Relative Größe der Indexe.

Experimente (Gulli und Signorini 2005):

Einige Ergebnisse:

- **Relative Größen:**

Google / Yahoo!: 1,22

Google / MSN: 1,41

Google / Ask/Teoma: 1,65

- **Overlap:**

- Google indiziert durchschnittlich 68,2 % aller anderen;

- MSN 49,2 %;

- Ask/Teoma 43,5 %; und

- Yahoo! 59,1 %.

- **Absolute Größe des indizierten Webs:** 11,5 Mrd. Seiten.

Probleme:

- Ungleichgewicht durch Anfragen:
Umfangreiche Dokumente viel häufiger
(z. B. Wörterbücher).
- Ungleichgewicht durch Ranking:
Seiten mit sehr kleinem Rang werden nie ausgewählt.
(Suchmaschinen zeigen die evtl. gar nicht an.)
- Gutes Lexikon schwer beschaffbar:
Soll repräsentativ sein für ganzes Web
(zumindest für alle Suchmaschinen).

Abhilfe: Korrektur der Verteilung durch Simulationstechniken
([Bar-Yossef u. a. 2006](#)).

2.2 Random-Walk-basiertes Sampling

Hier: WebWalker (Bar-Yossef u. a. 2000).

Idee: Konvergenztheorie für Markoffketten:

- Graph ungerichtet, nicht bipartit, zusammenhängend \Rightarrow Konvergenz gegen von Startverteilung unabhängige stationäre Verteilung π .

Nicht bipartit \Rightarrow aperiodisch.

Behauptung dann aus Perron-Frobenius-Theorem.

- Graph zudem d -regulär, d. h. jeder Knoten mit genau d Nachbarn: Dann ist π Gleichverteilung.

Beweis: Dies ist Eigenvektor zum Eigenwert 1.

Aber: Webgraph gerichtet und alles andere als regulär!?

Neue Idee:

Verwaltete gepatchte Version des Webgraphen, die Anforderungen erfüllt und benutze diese für Steuerung des Random Walks.

- **Gerichtet** → **ungerichtet**:

Bei Schritt des Walks von Knoten aus Vorwärts- und Rückwärtskanten benutzen.

Rückwärtskanten von Suchmaschinen beschaffen (`link:`). Problem: Liefert nur ein Bruchteil der realen Kantenmenge. Ungelöst, muss mit Approximation leben.

- **Regularität**:

Für große Konstante d , z. B. $d = 10.000.000$:

Falls Knoten $d' < d$ Nachbarn hat, ergänze $d - d'$ Schleifen.

Algorithmus WEBWALKER:

Verwalte Graph $G_t = (V_t, E_t)$ für t -ten Schritt des Walks.

Zu Anfang: $V_0 := E_0 := \emptyset$, $t := 1$,

aktueller Knoten irgendein geeigneter Startknoten.

Schritt $t \geq 1$, aktueller Knoten sei v .

Falls Knoten v neu entdeckt:

- $V_t := V_{t-1} \cup \{v\}$.
- E_t : Füge folgende Kanten ungerichtet zu E_{t-1} hinzu:
 - v hat Grad $d' < d$: $d - d'$ Schleifen für v ;
 - alle ausgehenden Kanten von v ;
 - zufällig ausgewählte r eingehende Kanten von v .

Wähle neuen aktuellen Knoten zufällig gleichverteilt aus Nachbarn von v in $G_t = (V_t, E_t)$.

Kommentare:

- Schleifen müssen nicht wirklich durchlaufen werden:
Offline simulieren oder direkt Erwartungswert für
Anzahl Durchläufe einsetzen.
- Zufällige Auswahl der eingehenden Kanten:
Verhindere zu starken Einfluss von Suchmaschinen.
Konkrete Werte in Experimenten: $r = 0, 3, 10$.
- Nach Erstbesuch von Knoten v wird Nachbarschaft fixiert.

Analyse:

Beachte: Zwei Arten von Zufallsentscheidungen:

- Zufällige Fixierung der Nachbarschaft für Knoten.
- Zufällige Auswahl der jeweils begangenen Kante.

Annahmen im Folgenden:

- Realer Webgraph statisch;
- vorkommende Graphen für Walk nicht bipartit (trivial erfüllt, falls mindestens eine Schleife).

Satz:

Für beliebigen Startknoten v_0 konvergiert der von WebWalker generierte Random Walk gegen die Gleichverteilung auf der Vereinigung der starken Zusammenhangskomponente von v_0 und der Menge der von dort aus erreichbaren Knoten ($\text{SCC}(v_0)$ bzw. $\text{OUT}(v_0)$).

Beweis:

- Nach endlich vielen Schritten werden keine neuen Knoten mehr entdeckt. Grund: Statischer Webgraph ist endlich.
- Danach sind Nachbarschaften der Knoten (irgendwie!) fixiert.

Erreichter Graph sei dann G_t . Dann gilt:

- $V_t = \text{SCC}(v_0) \cup \text{OUT}(v_0)$: Sei $v \in \text{SCC}(v_0) \cup \text{OUT}(v_0)$. Dann existiert im Webgraphen gerichteter Weg $v_0 \rightsquigarrow v$. Annahme: Kante $e = (x, y)$ erste auf diesem Weg nicht in G_t . Dann $x \in V_t$, aber nicht alle ausgehenden Kanten aufgenommen. Widerspruch.
- G_t erfüllt Voraussetzungen für Konvergenztheorie \rightarrow restlicher Walk konvergiert gegen Gleichverteilung. □

Konvergenzgeschwindigkeit:

Anzahl Schritte i. W. bestimmt durch $1/(1 - |\lambda_2|)$,
 λ_2 zweitgrößter Eigenwert der Adjazenzmatrix des betrachteten Graphen.

Experimente: $\lambda_2 \approx 1 - 10^{-6}$.

Mögliche Quellen für Ungleichgewicht bei Verteilung:

- Bevorzugung von Knoten mit hohem Grad.
- Bevorzugung von Knoten im Index von benutzten Suchmaschinen.
- Bevorzugung der Knoten in der Nachbarschaft des Startknotens.

Mögliche Abhilfe: Siehe wieder [Bar-Yossef u. a. \(2006\)](#).