Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam

Search Engines
Chapter 3 – Crawls & Feeds

21.4.2011

Felix Naumann

What to crawl

- Every document answers at least one question:
  - „Now where was that document again?“
- Poor quality documents
  - Swamp index
  - Slow down query answering
- But
  - Have at least some value
  - Have not hindered success of commercial search engines
- Thus: Crawl everything

- More important:
  - Keep documents up to date
  - Keep archive of older material

Overview

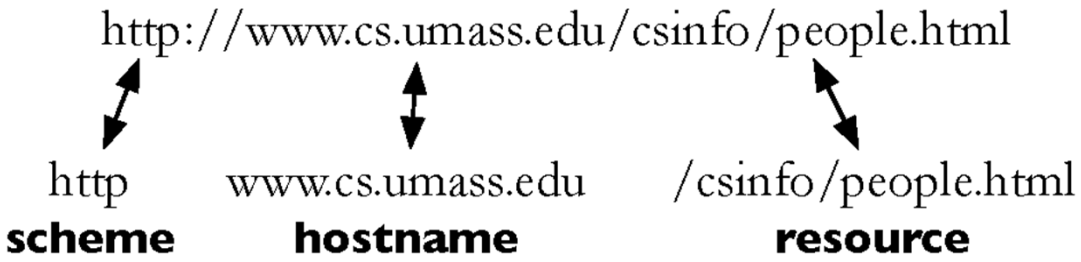
- Crawling the Web
- The Deep Web
- Document Conversion
- Document Storage
- Duplicate Detection
- Removing Noise

Web Crawler

- Synonym: A *spider* spiders the Web
- Web crawling is easy: Pages are meant to be found and retrieved
  - Other information is hard to come by.
- Finds and downloads web pages automatically
  - provides the collection for searching
- Web is huge and constantly growing
  - Google knows 1 trillion pages (2008)
    - Not countable – what constitutes a page?
  - Vast storage is needed
- Web is not under the control of search engine providers
  - Set of pages at a particular web server unknown
  - Owner might not want page to be copied
  - Parallel crawling is impolite
  - Pages are hidden behind form
- Web pages are constantly changing
- Crawlers also used for other types of data

Retrieving Web Pages

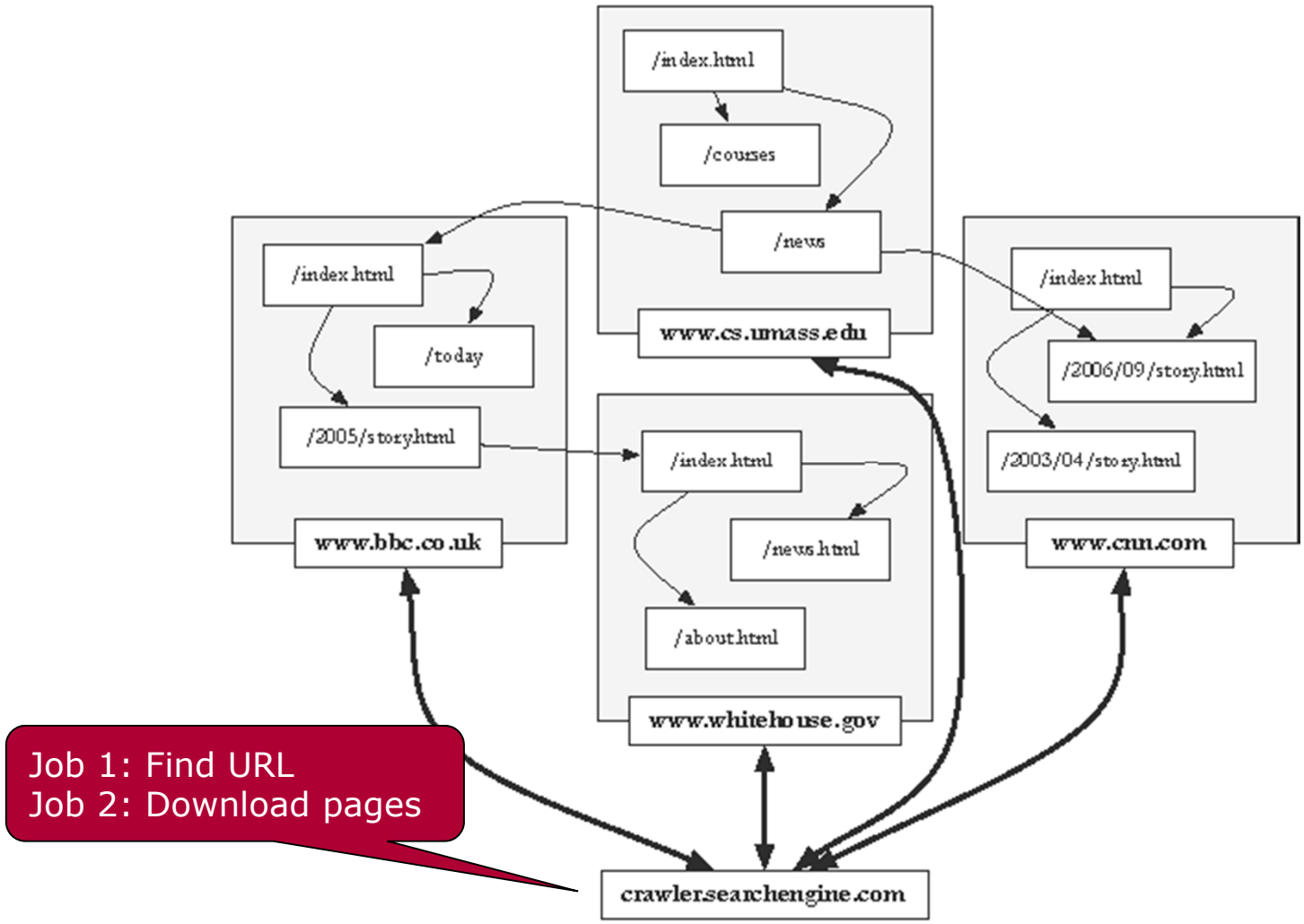
- Every page has a unique *uniform resource locator* (URL)
- Web pages are stored on *web servers* that use HTTP to exchange information with client software



Retrieving Web Pages

- Web crawler is similar to Web browser
  - Web clients that fetch web pages in the same way
- Web crawler client program connects to a *domain name system* (DNS) server
- DNS server translates the hostname into an *internet protocol* (IP) address
- Crawler then attempts to connect to server host using specific *port*
  - Usually port 80
- After connection, crawler sends an HTTP request to the web server to request a page
  - usually a GET request: **`GET /csinfo/peope.html HTTP/1.0`**

Crawling the Web



Web Crawler

- Start with a set of *seeds*
  - Set of URLs given as parameters
- Seeds are added to a URL request queue
- Crawler starts fetching pages from the request queue
- Downloaded pages are parsed to find link tags
  - Might contain other useful URLs to fetch
- New URLs added to the crawler's request queue, or *frontier*
  - Frontier: Standard queue or priority queue
- Continue until no more new URLs
  - or disk full

Web Crawling

- Web crawlers spend a lot of time waiting for responses to requests
  - DNS server response
  - Connection to web server
  - Page download
- To reduce this inefficiency, web crawlers use threads and fetch hundreds of pages at once
- Crawlers could potentially flood sites with requests for pages
- To avoid this problem, web crawlers use *politeness policies*
  - No parallel requests
  - Delay between requests to same web server
  - Logically: one queue per web server
- Large queue
  - Crawler fetches 100 pages / sec
  - At most 1 page per server per 30 sec
  - => Queue size > 3,000
  - Normally much larger (many URLs in queue from same server)

Controlling Crawling

- Even crawling a site slowly will anger some web server administrators, who object to any copying of their data
- **`robots.txt`** to control crawlers

```
User-agent: *
Disallow: /private/
Disallow: /confidential/
Disallow: /other/
Allow: /other/public/

User-agent: FavoredCrawler
Disallow:

Sitemap: http://mysite.com/sitemap.xml.gz
```

All crawlers

Special crawler

Reveal hidden web

HPI's robots.txt

http://www.hpi.u...m.de/robots.txt

```
# HPI Potsdam  -- http://www.hpi.uni-potsdam.de/
# Robot Exclusion File -- robots.txt
User-agent: *
Disallow: /fileadmin/hpi_settings
Disallow: /fileadmin/_temp_
Disallow: /fileadmin/template
Disallow: /fileadmin/user_upload
Disallow: /fileadmin/hpi/Intern
Disallow: /fileadmin/hpi/banner
Disallow: /fileadmin/hpi/FG_ITS/intern
Disallow: /uploads/secure
Disallow: /uploads/tf
Disallow: /uploads/tx_flvplayer
Disallow: /uploads/tx_gooffotoboek
Disallow: /uploads/tx_jshuniversity
Disallow: /uploads/tx_macinabanners
Disallow: /uploads/tx_rtehtmlarea
Disallow: /uploads/tx_srquicktime
Disallow: /uploads/tx_ttnews
Disallow: /typo3temp/
Disallow: /typo3conf/
Disallow: /typo3_src/
Disallow: /t3lib/
Disallow: /tslib/
Disallow: /error/
Disallow: /imgs/
Disallow: /imgs_menu/
Disallow: /imgs_navi/
```

Simple Crawler Thread


```
procedure CrawlerThread(frontier)
    while not frontier.done() do
        website ← frontier.nextSite()
        url ← website.nextURL()
        if website.permitsCrawl(url) then
            text ← retrieveURL(url)
            storeDocument(url, text)
            for each url in parse(text) do
                frontier.addURL(url)
            end for
        end if
        frontier.releaseSite(website)
    end while
end procedure
```

Politeness timer starts here

Freshness

- Web pages are constantly being added, deleted, and modified
- Web crawler must continually revisit pages it has already crawled to see if they have changed in order to maintain the *freshness* of the document collection
  - *Stale* copies no longer reflect the real contents of the web pages

Freshness

- HTTP protocol has a special request type called HEAD
  - Easy to check for page changes
  - Returns information about page, not page itself

Client request:

```
HEAD /csinfo/people.html HTTP/1.1
Host: www.cs.umass.edu
```

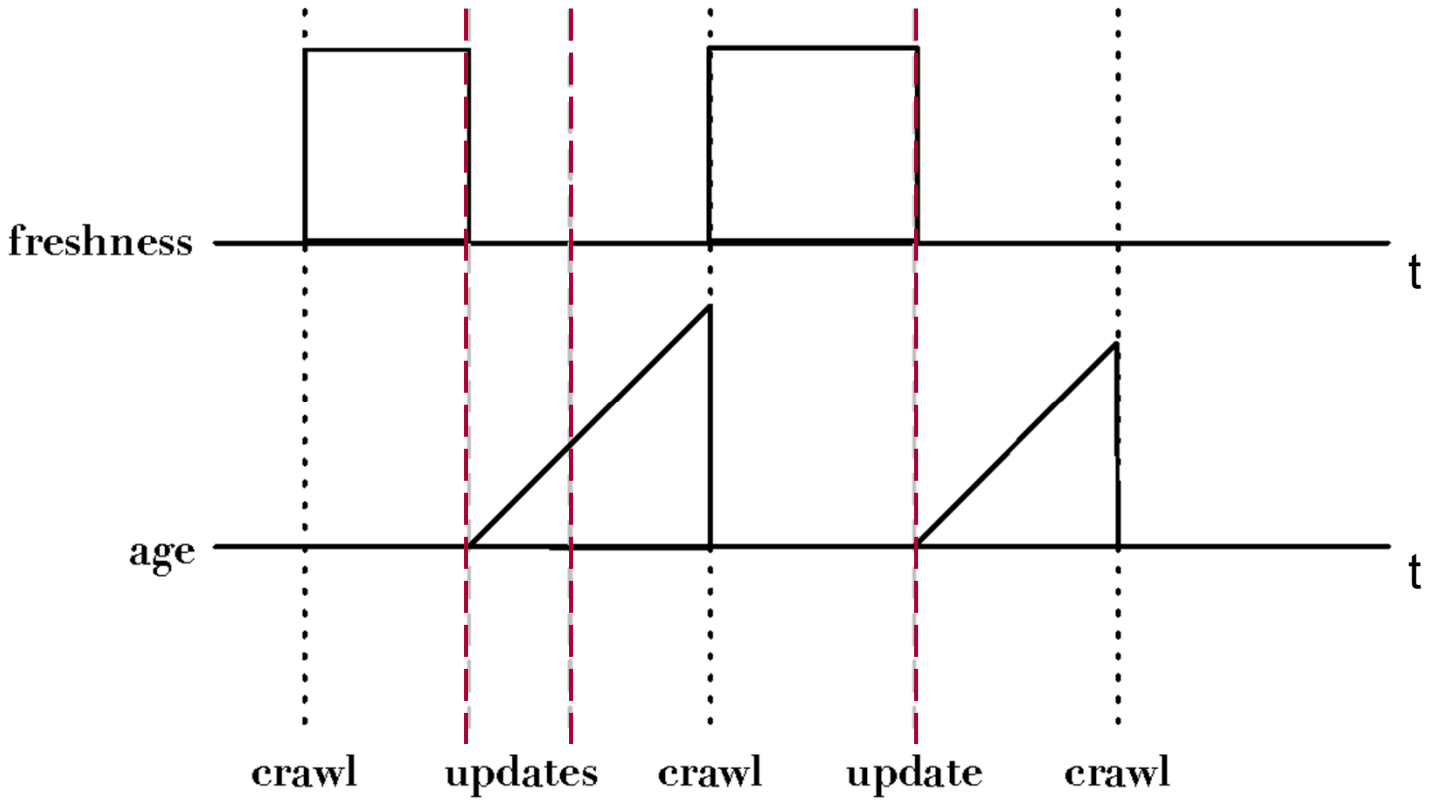
Server response:

```
HTTP/1.1 200 OK
Date: Thu, 03 Apr 2008 05:17:54 GMT
Server: Apache/2.0.52 (CentOS)
Last-Modified: Fri, 04 Jan 2008 15:28:39 GMT
ETag: "239c33-2576-2a2837c0"
Accept-Ranges: bytes
Content-Length: 9590
Connection: close
Content-Type: text/html; charset=ISO-8859-1
```

Freshness

- Not possible to constantly check all pages
  - must check important pages and pages that change frequently
  - Different types of pages update at different rates
    - Homepage vs. news site
  - Differences even within type: Active vs. inactive blog
- Freshness is the proportion of pages that are fresh
  - Fresh = most recent copy
- Optimizing for this metric can lead to bad decisions, such as not crawling popular sites
  - One can never achieve freshness there, thus waste of resources
- *Age* is a better metric

Freshness vs. Age



Age

- Expected age of a page, *t* days after it was last crawled:

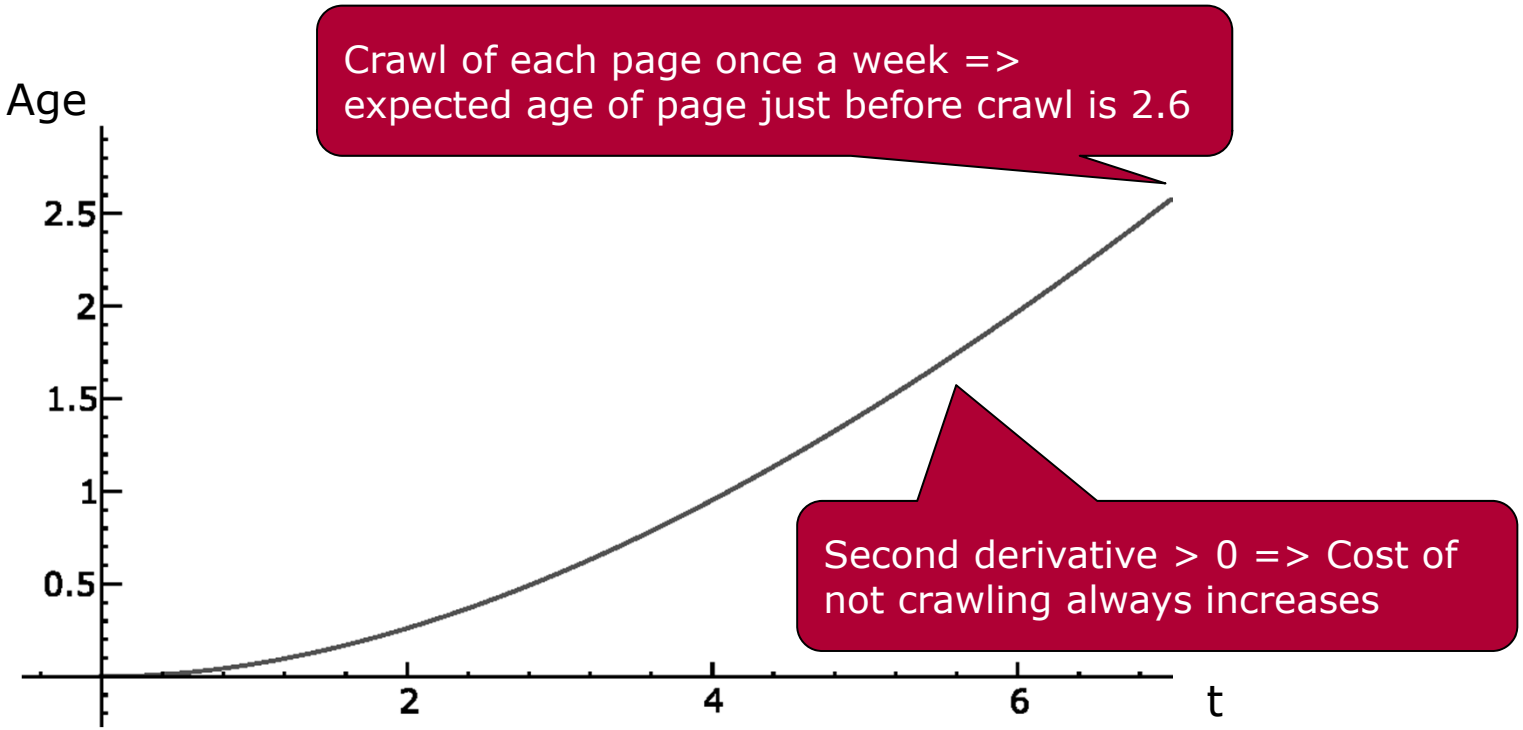
$$\text{Age}(\lambda, t) = \int_0^t P(\text{page changed at time } x)(t - x)dx$$

  - where λ is the change frequency per day.
  - Page crawled at time *t* and changed at time *x* => *(t – x)* is an age
- Web page updates follow the Poisson distribution on average
  - time until the next update is governed by an exponential distribution

$$\text{Age}(\lambda, t) = \int_0^t \lambda e^{-\lambda x}(t - x)dx$$

Age

- Older a page gets, the more it costs not to crawl it
  - e.g., expected age with mean change frequency $\lambda = 1/7$ (one change per week)

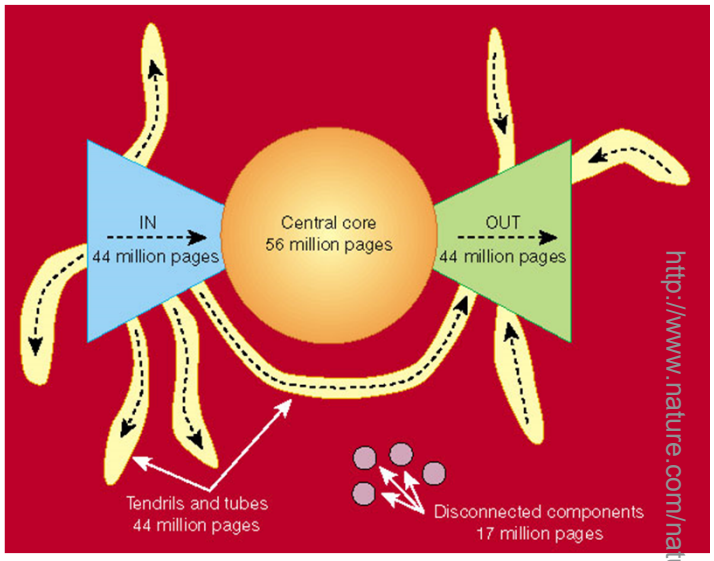


Focused Crawling

- Also: Topical crawling
- Attempts to download only those pages that are about a particular topic
  - Used by *vertical search* applications
- Promise: Higher accuracy
  - User preselects relevance by choosing vertical search engine
  - Index is not cluttered by irrelevant documents
- Rely on the fact that pages about a topic tend to have links to other pages on the same topic
  - Anchor texts help
  - Popular pages for a topic are typically used as seeds
- Crawler uses *text classifier* to decide whether a page is on topic
  - See Chapter 9

Deep Web

- Sites that are difficult for a crawler to find are collectively referred to as the *deep* (or *hidden*) *Web*
  - much larger than conventional Web
- Three broad categories:
  - Private sites
    - no incoming links, or may require log in with a valid account
    - Exception: News sites
  - Form results
    - Sites that can be reached only after entering some data into a form
    - Exception: Online stores
  - Scripted pages
    - Pages that use JavaScript, Flash, or another client-side language to generate links
    - Possible but difficult for crawler



http://www.nature.com/nature/journal/v405/n6783/full/405113a0.html

More in a minute…

Sitemaps

- Sitemaps contain lists of URLs and data about those URLs
  - modification time, modification frequency
- Generated by web server administrators
- Tells crawler about pages it might not otherwise find
- Gives crawler a hint about when to check a page for changes

Sitemap Example


```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
  <url>
    <loc>http://www.company.com/</loc>
    <lastmod>2008-01-15</lastmod>
    <changefreq>monthly</changefreq>
    <priority>0.7</priority>
  </url>
  <url>
    <loc>http://www.company.com/items?item=truck</loc>
    <changefreq>weekly</changefreq>
  </url>
  <url>
    <loc>http://www.company.com/items?item=bicycle</loc>
    <changefreq>daily</changefreq>
  </url>
</urlset>
```

always
hourly
daily
weekly
monthly
yearly
never

Between 0.0 and 1.0 (default 0.5)

https://www.google.com/webmasters/tools/docs/de/protocol.html

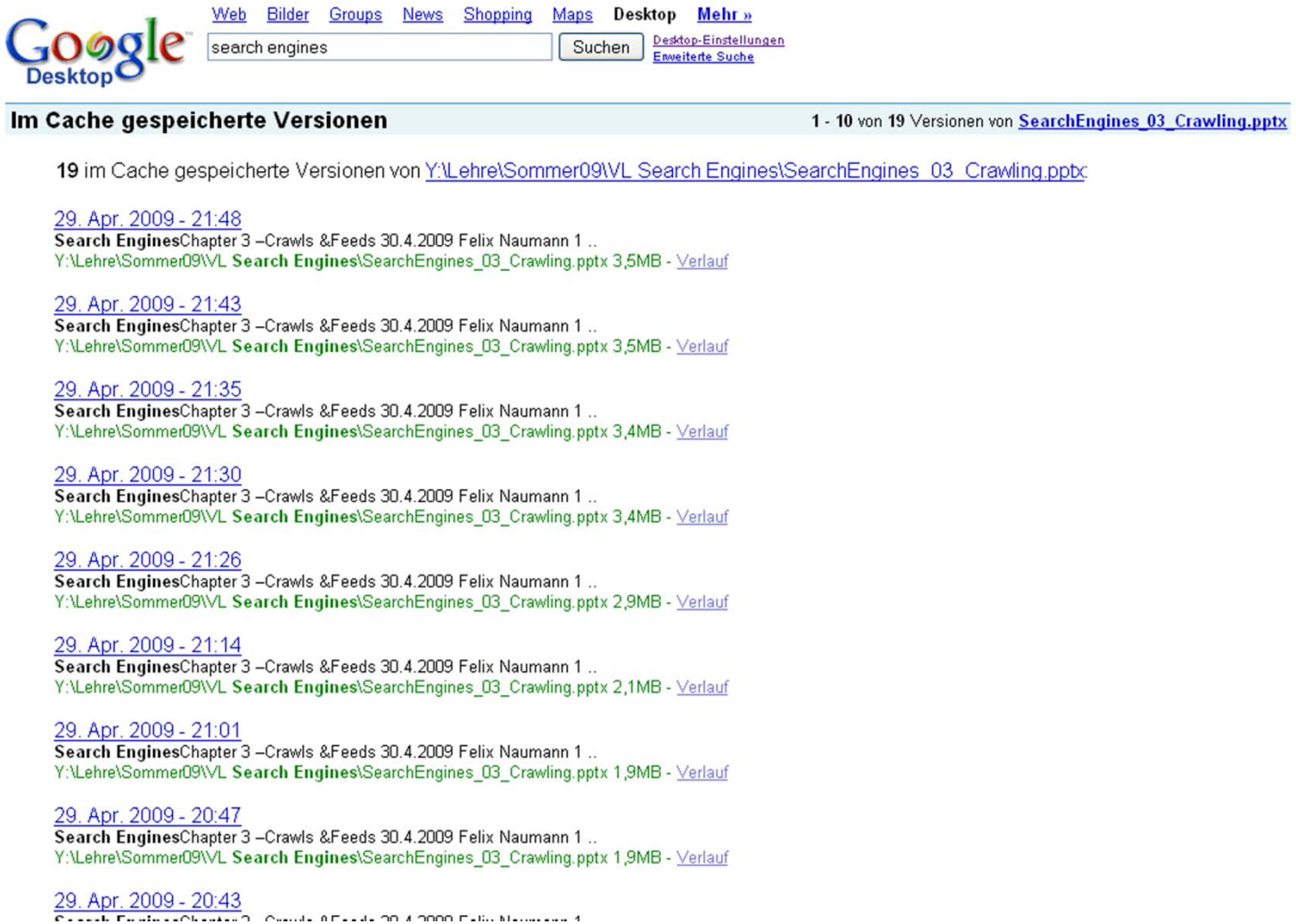
Distributed Crawling

- Three reasons to use multiple computers for crawling
  1. Helps to put the crawler closer to the sites it crawls
     - Low latency, high throughput
  2. Reduces the number of sites the crawler has to remember
     - In queue
     - In index
  3. Reduces computing resources required
     - Parsing, network bandwidth
- Distributed crawler uses a hash function to assign URLs to crawling computers
  - URLs are sent in batches
  - Hash function should be computed on the host part of each URL
  - Causes imbalance, but
    - Mostly site-internal links
    - Abides politeness anyway

Desktop Crawls

- Used for desktop search and enterprise search
- Differences to web crawling:
  - Much easier to find the data
  - Responding quickly to updates is more important
    - Different expectations than for Web
    - File systems can send change notifications
  - Many different document formats
    - → Document conversion
  - Data privacy very important
  - Must be conservative in terms of disk and CPU usage
    - Copies of documents need not be stored…
    - GDS?

Google Desktop Search



Document Feeds

- Many documents are *published*
  - Created at a fixed time and rarely updated again
  - e.g., news articles, blog posts, press releases, email
  - In general time-sensitive content
- Published documents from a single source can be ordered in a sequence called a *document feed*
  - New documents found by examining the end of the feed
  - Single place, no crawling necessary
  - using HTTP GET requests to web servers that host them
- Two types:
  - A *push feed* alerts the subscriber to new documents (phone)
    - Expensive; news agencies (“Ticker”)
  - A *pull feed* requires the subscriber to check periodically for new documents (email)
- Most common format for pull feeds is called *RSS*
  - Really Simple Syndication, RDF Site Summary, Rich Site Summary

RSS Example


```
<?xml version="1.0"?>
<rss version="2.0">
  <channel>
    <title>Search Engine News</title>
    <link>http://www.search-engine-news.org/</link>
    <description>News about search engines.</description>
    <language>en-us</language>
    <pubDate>Tue, 19 Jun 2008 05:17:00 GMT</pubDate>
    <ttl>60</ttl>

    <item>
      <title>Upcoming SIGIR Conference</title>
      <link>http://www.sigir.org/conference</link>
      <description>The annual SIGIR conference is coming!
        Mark your calendars and check for cheap
        flights.</description>
      <pubDate>Tue, 05 Jun 2008 09:50:11 GMT</pubDate>
      <guid>http://search-engine-news.org#500</guid>
    </item>
```

Time to live (minutes)

RSS Example


```
...
   <item>
      <title>New Search Engine Textbook</title>
      <link>http://www.cs.umass.edu/search-book</link>
      <description>A new textbook about search engines
        will be published soon.</description>
      <pubDate>Tue, 05 Jun 2008 09:33:01 GMT</pubDate>
      <guid>http://search-engine-news.org#499</guid>
    </item>
  </channel>
</rss>
```

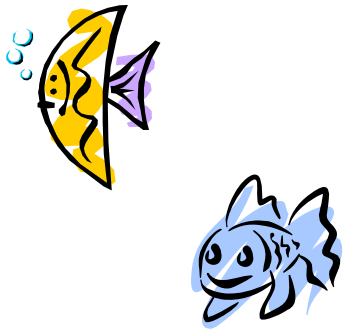
Description might contain entire text of news or web page

Overview

- Crawling the Web
- The Deep Web
- Document Conversion
- Document Storage
- Duplicate Detection
- Removing Noise

Surface web

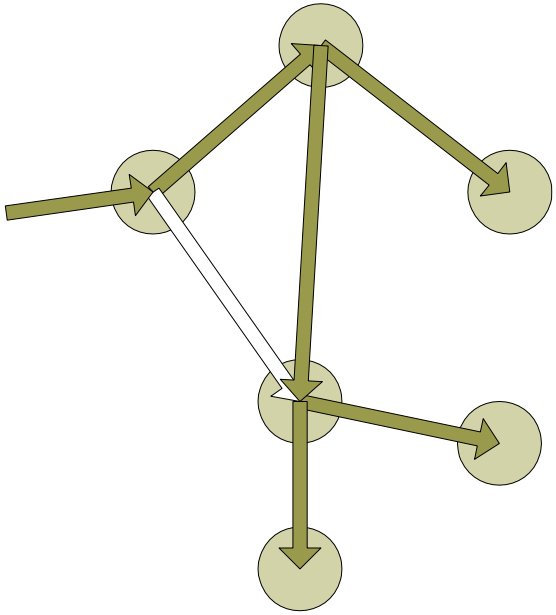
Shallow web



Deep web

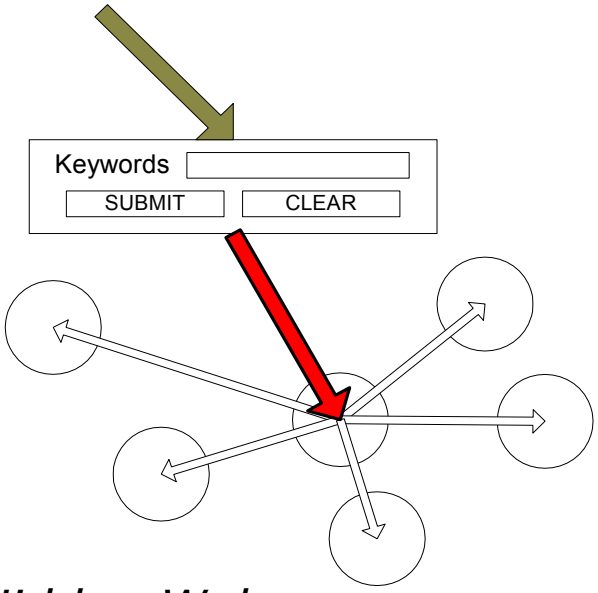
Quelle: [To01]

Surface Web vs. Hidden Web



Surface Web

- Link-Struktur
- Kann *gecrawled* werden
- Dokumente durch Suchmaschinen indiziert
- **Anfragen auf viele Websites gleichzeitig**



Hidden Web

- Keine Link-Struktur
- Dokumente verborgen in DBMS
- Dokumente nicht durch Internet-Suchmaschinen indiziert
- Dokumente eventl. durch Intranet-Suchmaschinen indiziert
- **Anfragen auf jede Sammlung einzeln**

Quelle: Folie aus [IGS01]

Begriffe / Synonyme

- Surface Web (Oberflächen-Web)
  - Inhalt für „normale“ Suchmaschinen sichtbar
- Shallow Web (Flaches Web)
  - „Normale“ Web-Seiten, die dynamisch generiert werden
  - Anfragen durch Klicken auf Links
- Hidden Web (verborgenes Web)
  - Inhalt für „normale“ Suchmaschinen unsichtbar
  - Invisible Web (unsichtbares Web)
    - Synonym mit Hidden web
  - Deep Web (tiefes Web)
    - nach BrightPlanet,
    - Synonym mit Hidden Web

Quelle: [To01]

Hidden Web: Beispiel

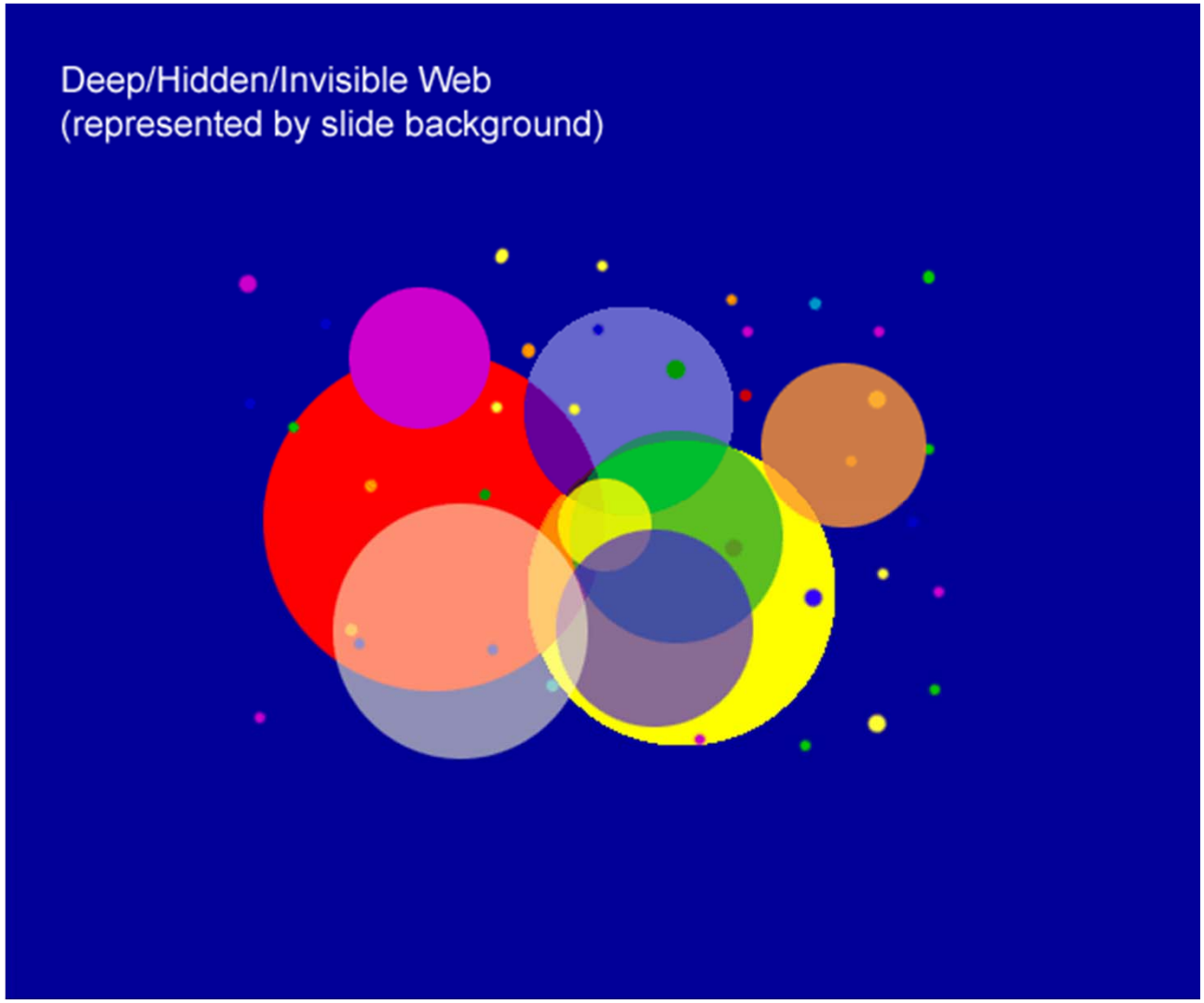
- Suche in PubMed nach "diabetes"
  - 178,975 Treffer
- Google Suche: "diabetes site:www.ncbi.nlm.nih.gov"
  - nur 119 matches
- Weitere Beispiele:

Database	Query	Matches	Google
PubMed	diabetes	178,975	119
U.S. Patents	wireless network	16,741	0
Library of Congress	visa regulations	>10,000	0
…	…	…	…

- Gegenbeispiel
  - Amazon: Hilft explizit bei Verlinkung

Quelle: Folie aus [IGS01]

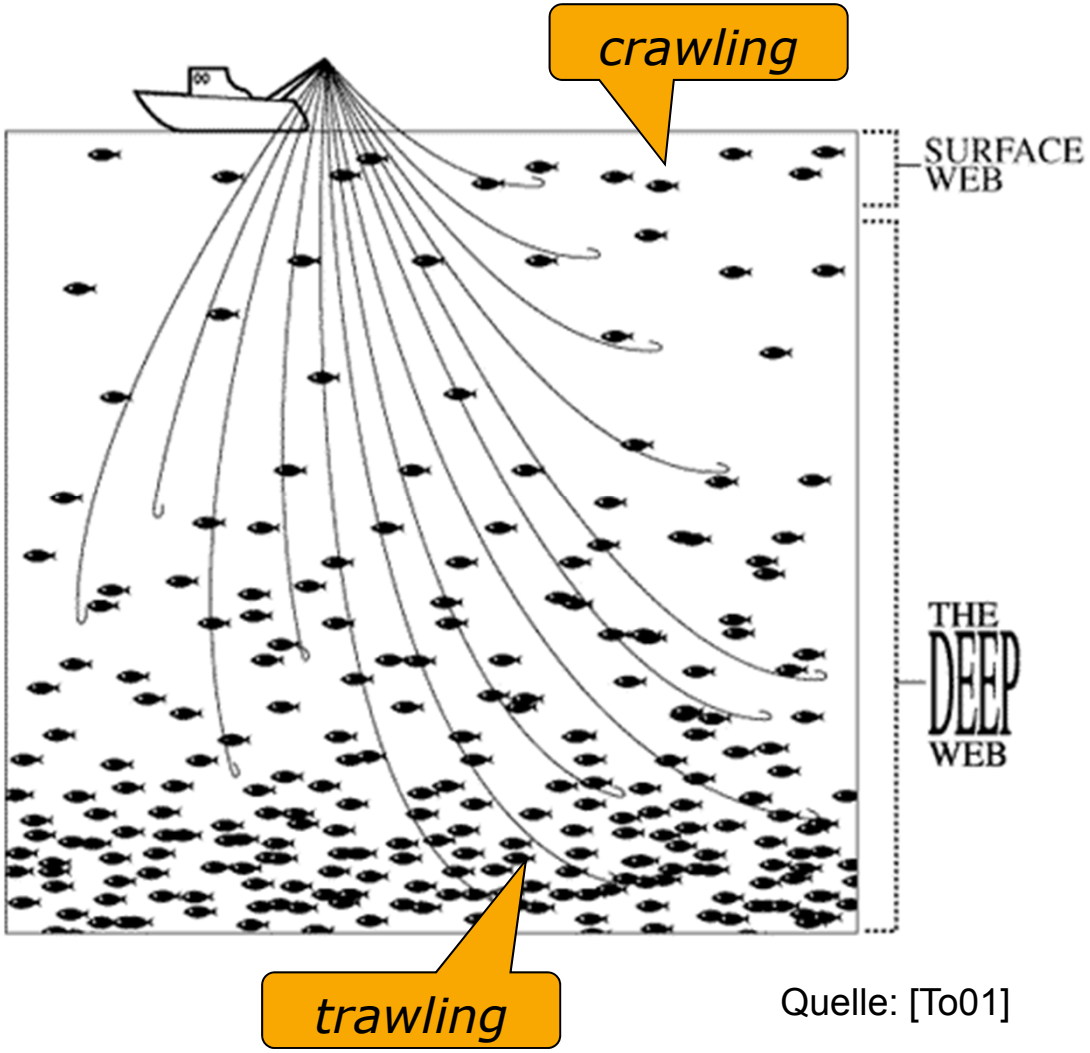
Invisible/Hidden/Deep Web



Quelle: [To01]

Surface vs. Hidden Web [Be01]

- "Der Inhalt des Surface Web ist persistent auf statischen Seiten, die mittels crawling von Suchmaschinen entdeckt werden kann. Inhalt des Hidden Web wird dynamisch präsentiert in Antwort auf eine konkrete Anfrage."
- "…der größte Anteil Inhalts des Hidden Web wird unterhalb der Oberfläche bleiben und kann nur im Kontext einer bestimmten Anfrage entdeckt werden."



Quelle: [To01]

Das Verborgene Web

- Der Teil des Webs, der nicht durch Suchmaschinen indiziert wird
  - Oft gespeichert in Datenbanken
  - Dynamisch generierte Web Seiten durch Anwendungen im Server
    - jsp, cgi, …
  - Sites und Seiten mit Passwort-geschütztem Inhalt
  - Früher: Inhalt von Dateien, die nicht in Standard-Formaten gespeichert werden
    - *.pdf, *.ppt, *.doc
    - Grafikformate

Quelle: [To01]

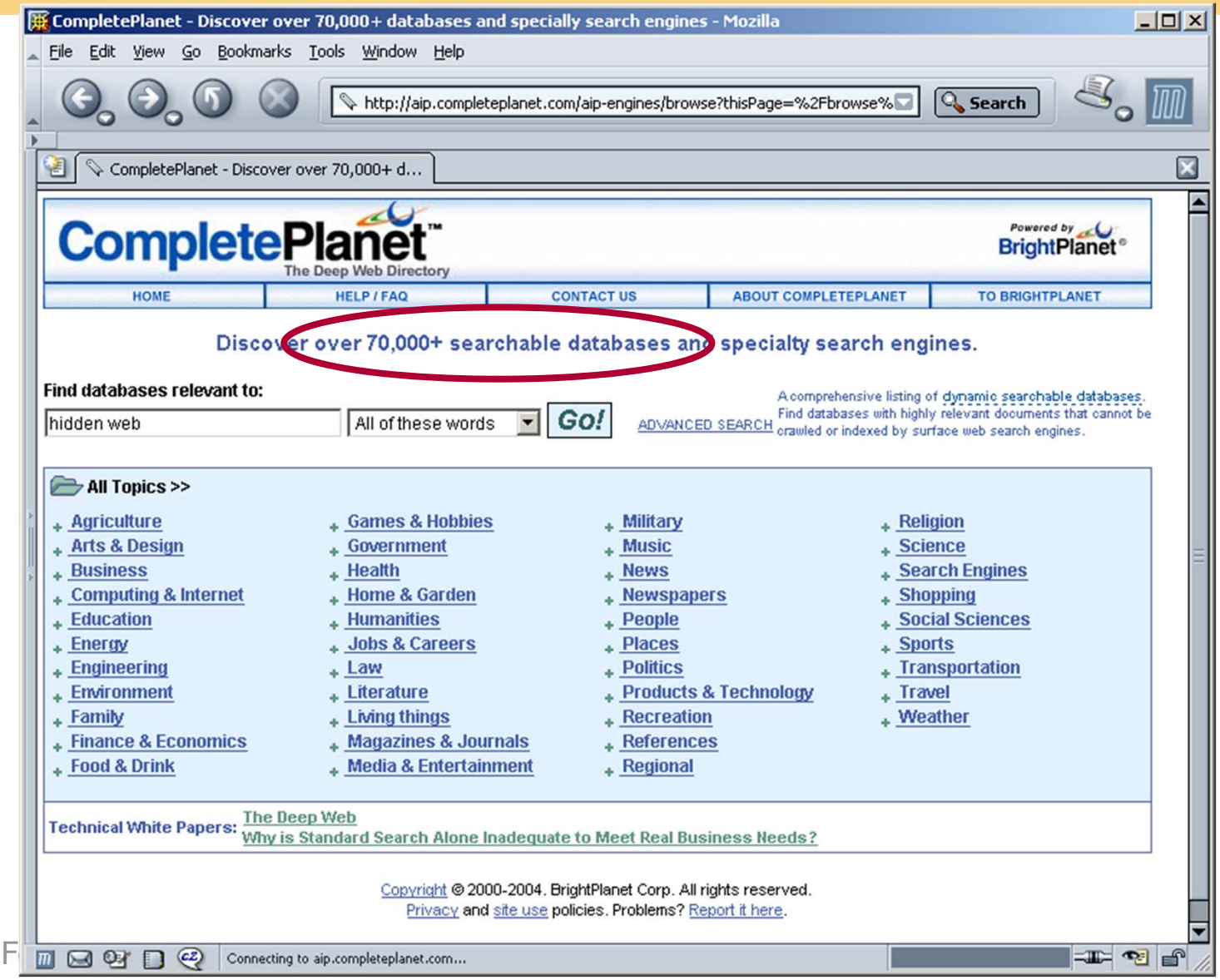
Statistiken [Be01]

- 400 bis 550 fach größer als Surface Web
  - 7,500 Terabytes Informationen im Hidden Web
  - 19 Terabytes Information im Surface Web
  - 550 Milliarden Dokumente im Hidden Web
  - 1 Milliarde Dokumente im Surface Web
  - je nach dem, was man zählt…
    - Dynamische Seiten...
- 100,000 Hidden Websites
- ca. 84% sind auf Text-Dokumente spezialisiert
- ca. 95% des Hidden Web ist öffentlich verfügbar.

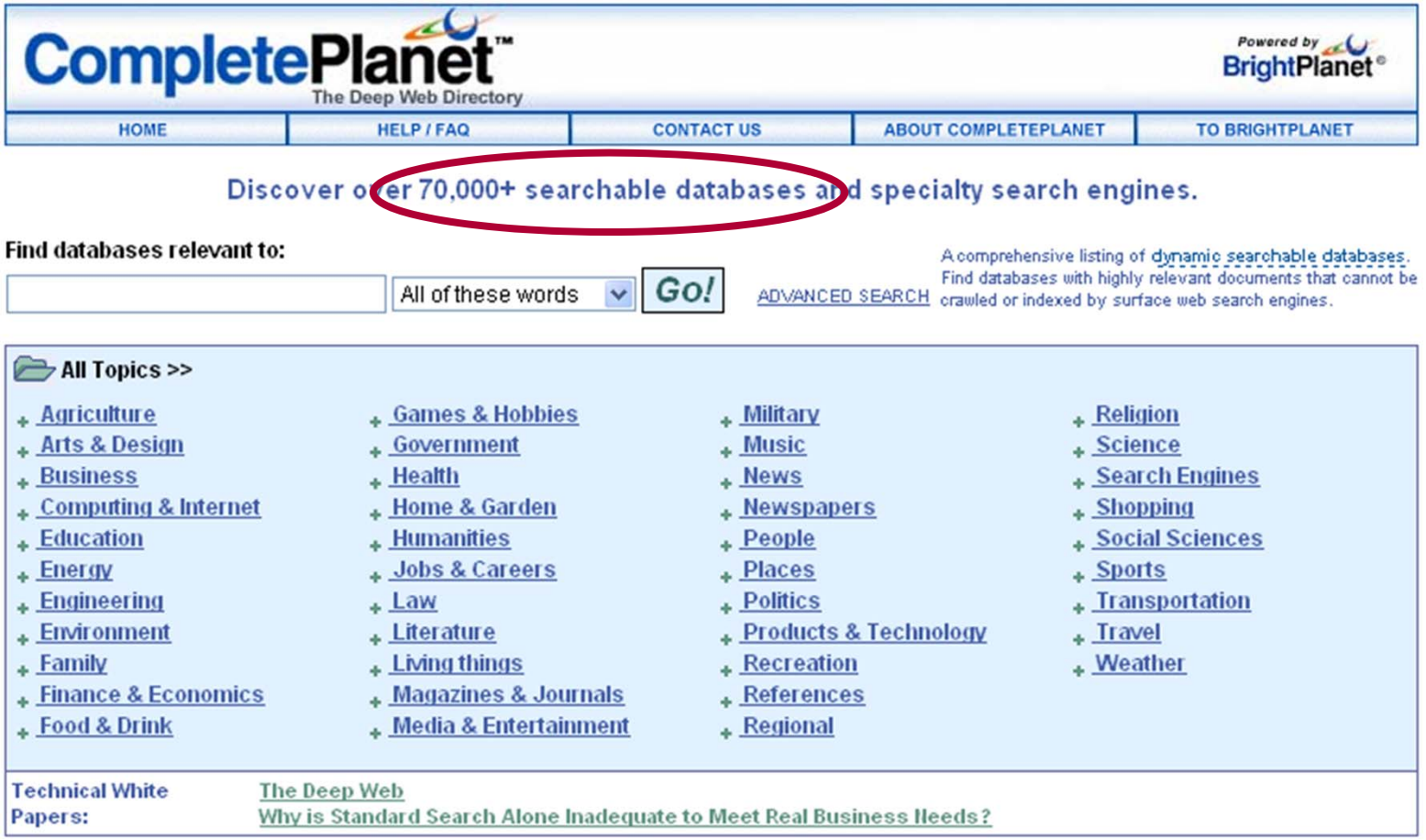
Eigenschaften [Be01]

- Hidden Websites haben thematisch oft „schmaleren“, aber „tieferen“ Inhalt.
- Oft qualitativ bessere Informationen
- Meist relevanter Inhalt
  - Kein Spam
- Über die Hälfte aller Hidden Websites sind thematisch spezialisiert.
- Am schnellsten wachsende Kategorie neuer Informationen im Internet

Beispiel: CompletePlanet.com



Beispiel: CompletePlanet.com



Beispiel: Search Engine Guide

Auffinden von Hidden Web Quellen [BC04]

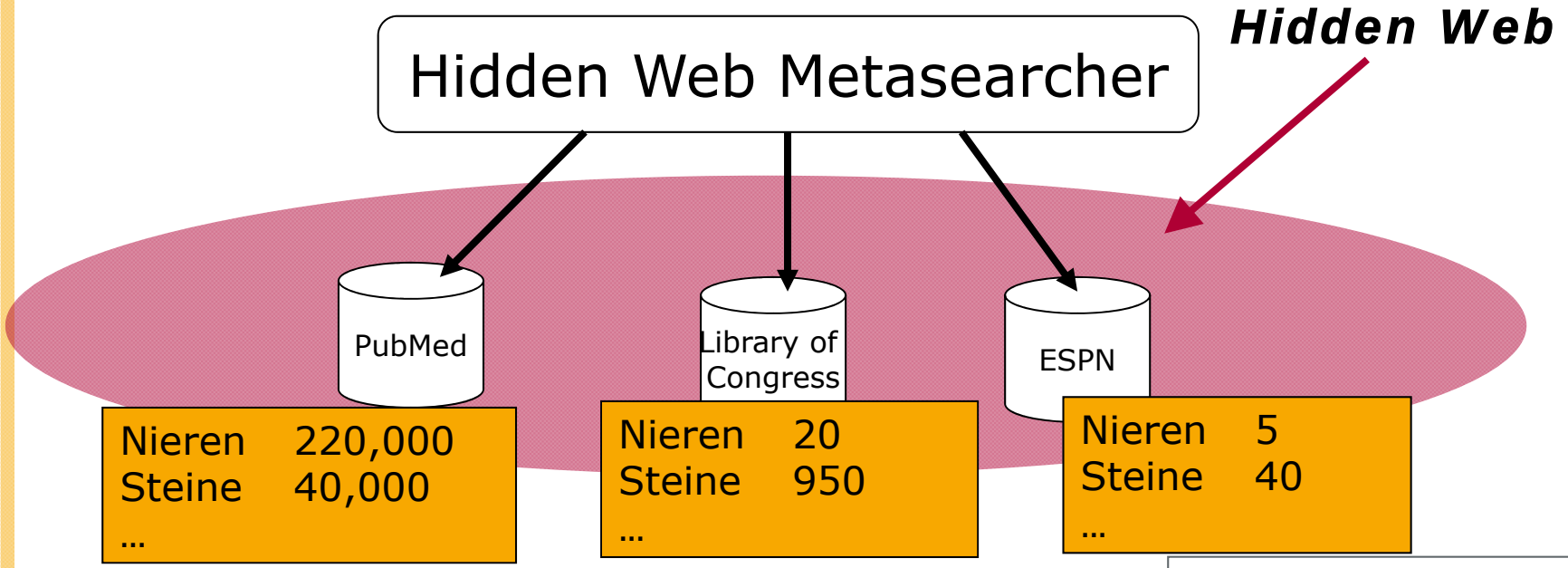
- Ziel: Finde Webseiten, die als Einstiegspunkt ins Hidden Web dienen.
  - Seiten mit HTML Formular
- Einschränkungen
  - Textuelle Formulare
    - mindestens ein Textinput
    - Gegenbeispiele?
      - Nur radio buttons, menus, checkboxen...
  - Anfrageformulare
    - Formulare, die Anfragen entgegennehmen und Informationen liefern
    - Gegenbeispiele?
      - Login Seiten
  - „Hidden Web Formulare“
    - Keine Seiten mit komplexen Formularen (mehr als ein Inputfeld)
- Aufgabe: Automatisches Finden und Erkennen von Hidden Web Formularen

Auffinden von Hidden Web Quellen

- Manuell ☹
- Automatisches Auffinden von Formularen
  1. Google-Suche (nach Themen)
  2. Lokales *breadth-first* Crawling bis Formular gefunden
     - Innerhalb einer Site
     - Bis zu einer festen Tiefe
- Automatisches Erkennen von Hidden Web Formularen (Heuristiken)
  - Testanfragen mit positiven und negativen Suchwörtern
    - Positiv: „passende" Worte
    - Negativ: Fantasieworte
  - Ergebnisse negativer Suchwörter immer gleich groß (Byte)
  - Ergebnisse positiver Suchworte immer größer als negative
    - Berechnung der Größe durch „Subtraktion" von Webseiten (als Baum)

Suche im Hidden Web – Probleme

- Auswahl relevanter Quellen für Anfrage
  - Themen extrahieren
    - *Content summary*
  - Nach Themen klassifizieren



Quelle: Folie aus [IGS01]

Content Summaries

- Statistiken, die den Inhalt einer Hidden Web Quelle beschreiben
- Document-cardinality *dc*
  - Anzahl der Dokumente insgesamt
- Document-frequency *df(w)*
  - Pro Wort: Anzahl der Dokumente, die dieses Wort enthalten

KrebsDB	
Document cardinality: 148.944	
Wort	Document frequency
Darm	121.134
Krebs	91.688
…	…

Vorschau zur Verwendung von content summaries
• Anfrage „Darm-Krebs“
• Anzahl Treffer = dc * *df*(Darm)/*dc* * *df*(Krebs)/*dc* = 74569

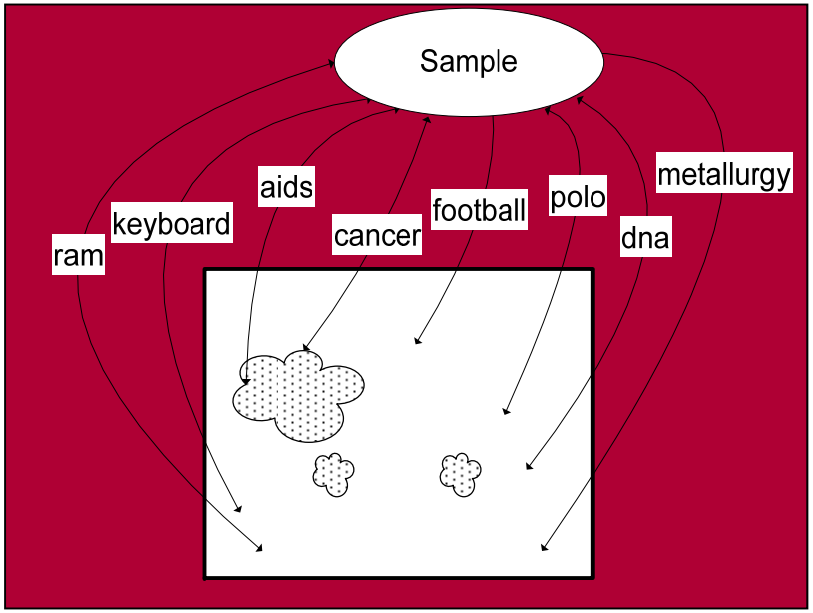
Extraktion von Content Summaries – Probleme

- Kein direkter Zugang zu den Dokumenten ohne konkrete Anfrage
  - Gebundene Variablen
- Deswegen: Anfrage-basiertes Dokument-Sampling:
  1. „Sinnvolle" Anfragen an Datenbank schicken (*focussed probing*)
     - Ergebnisliste mit Links
  2. Ergebnisdokumente aus Liste einholen (das „Sample")
  3. Sample verwenden um *content summary* zu erstellen

Quelle: Folie aus [IGS01]

“Zufälliges” Anfrage-basiertes Sampling

1. Start mit leerem *content summary*
   - Jedes Wort hat *df*(w) = 0.
2. Wähle ein Wort und schicke es als Anfrage an Hidden Web Quelle.
3. Wähle top-*k* Dokumente der Antwort (z.B. *k*=4).
4. Zähle *df(w)* für alle *w* in Sample um *content summary* zu füllen.
5. Wiederhole bis „genug“ (z.B. 300) Dokumente empfangen wurden

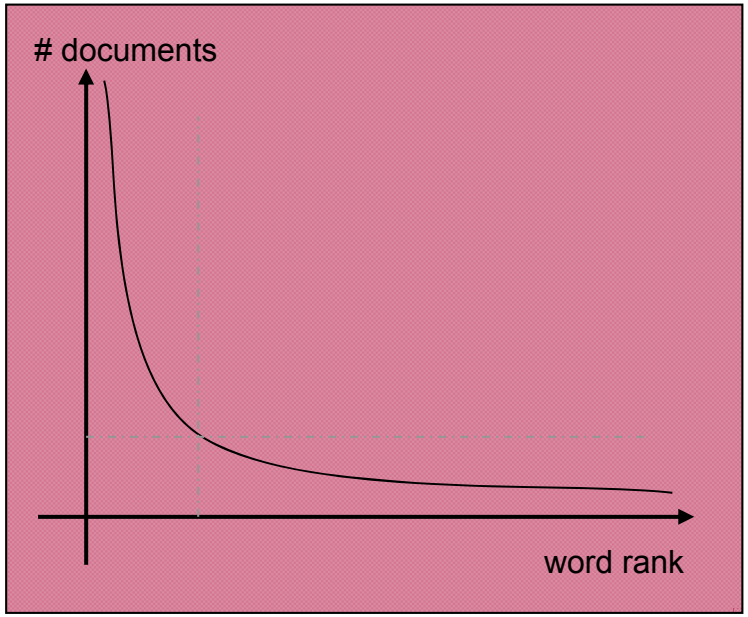


Wort	Häufigkeit in Sample
Krebs	150 (out of 300)
aids	114 (out of 300)
Herz	98 (out of 300)
…	
Basketball	2 (out of 300)

Quelle: Folie aus [IGS01]

Zufälliges Sampling – Probleme

- *df*(w) zwischen 1 und Anzahl der Dokumente
- Es wird nicht Document-frequency ermittelt, sondern Sample-frequency.
  - Absolute Zahlen sind nicht aussagekräftig.
  - Große Quellen haben ähnliche *content summary* wie kleine Quellen.
  - Zahlen sind nur relativ zu interpretieren (als ranking).
- Viele Anfragen ohne oder nur mit kleinem Ergebnis (Zipf's law)
- Viele, seltene Worte fehlen in der *content summary*.



Zipf's law

Deshalb jetzt verbesserte Lösung

Quelle: Folie aus [IGS01]

Zufälliges Sampling – Verbesserung

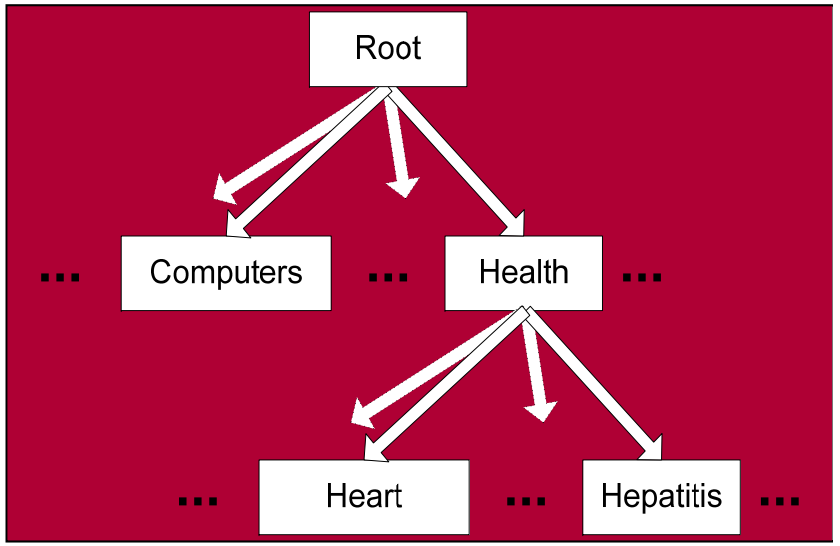
Algorithmus: Überblick

1. Trainiere Dokument-Klassifikatoren
   - Finde repräsentative Wörter für jede Kategorie.
2. Verwende Klassifikationsregeln um ein themenspezifisches Sample aus Quelle zu erhalten.
3. Schätze *df*(w) aller entdeckten Wörter.

Quelle: Folie aus [IGS01]

Fokussiertes Sampling: Trainingsphase

- Start mit vordefinierter Themen-Hierarchie und bereits klassifizierten Dokumenten
  - Bsp: Yahoo, dmoz Open Directory, Google ...
- Trainiere Dokument-Klassifikatoren für jeden Knoten der Hierarchie.
  - Mittels der bekannten Dokumente
- Extrahiere automatisch Regeln aus den Klassifikatoren:
  - ibm AND computers → Computers
  - lung AND cancer → Health
  - ...
  - angina → Heart
  - hepatitis AND liver → Hepatitis
  - ...



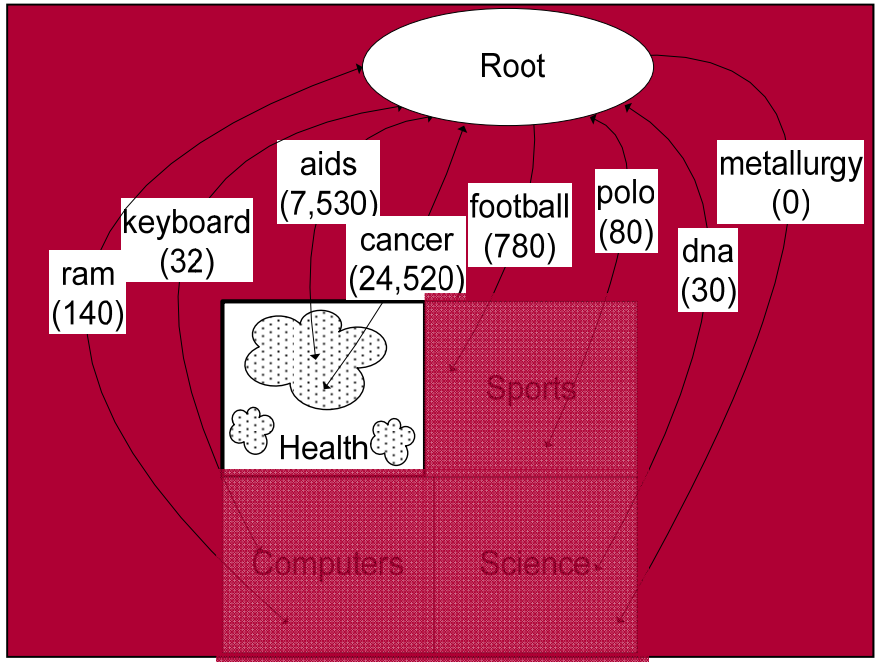
Root

Health

Quelle: Folie aus [IGS01]

Fokussiertes Sampling

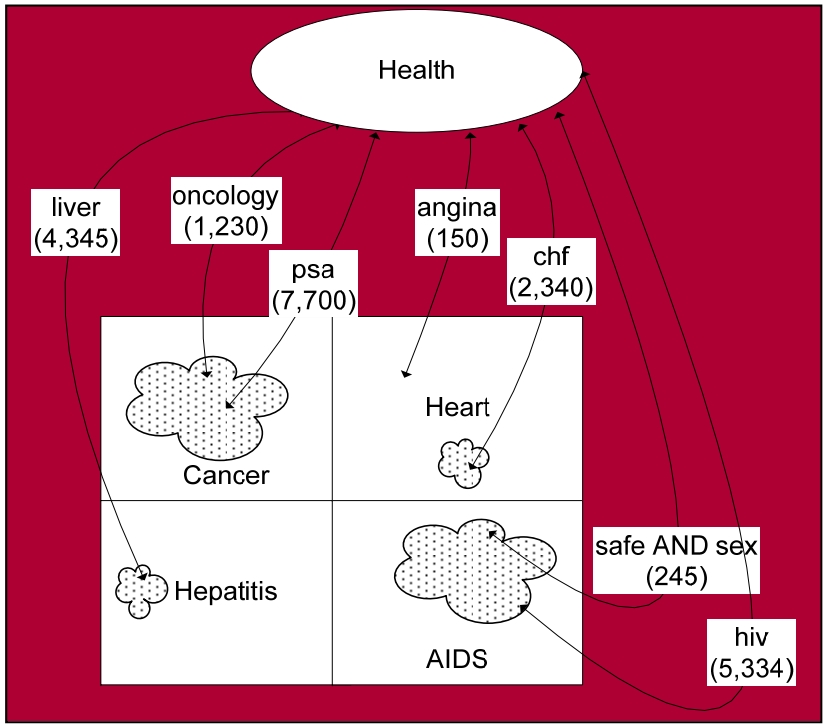
- Transformiere jede Regel in eine Boolesche Anfrage.
- Für jede Anfrage:
  - Schicke Anfrage an Quelle
  - Merke Anzahl der Ergebnisse
    - Parsing
  - Hole top-*k* Dokumente ein.
- Am Ende einer Runde:
  - Analysiere Ergebnisse für jede Kategorie (zählen).
  - Wähle Kategorie zum fokussieren in nächster Runde.



Quelle: Folie aus [IGS01]

Fokussiertes Sampling

- Fokus nun auf Subkategorie
- Neue Regelmenge, deshalb neue Anfragemenge
- Vorteile
  - Weniger Anfragen
  - Fokussierte Anfragen



Quelle: Folie aus [IGS01]

GetContentSummary(Category C, Database D)

α: $\langle SampleDF, ActualDF, Classif \rangle = \langle \emptyset, \emptyset, \emptyset \rangle$

if C is a leaf node

then return $\langle SampleDF, ActualDF, \{C\} \rangle$

Probe database D with the query probes derived from the classifier for the subcategories of C

β:

$newdocs = \emptyset$

foreach query probe q

$newdocs = newdocs \cup \{top\text{-}k\ documents\ returned\ for\ q\}$

if q consists of a single word w

then $ActualDF(w) =$ #matches returned for q

foreach word w in $newdocs$

$SampleDF(w) =$ #documents in $newdocs$ that contain w

Calculate *Coverage* and *Specificity* from the number of matches for the probes

foreach subcategory C_i of C

if $(Specificity(C_i) > \tau_s$ **AND** $Coverage(C_i) > \tau_c)$

then

γ:

$\langle SampleDF', ActualDF', Classif' \rangle =$ GetContentSummary(C_i, D)

Merge $\langle SampleDF', ActualDF' \rangle$ into $\langle SampleDF, ActualDF \rangle$

$Classif = Classif \cup Classif'$

return $\langle SampleDF, ActualDF, Classif \rangle$

Aufruf für jede Kategorie und Subkategorie

Anfragen entsprechend der Regeln des Klassifikators

Sammle Dokumente ein

Bei Ein-Wort Anfragen erlernen wir die tatsächliche *df*(w)

Zähle sample-frequency für jedes Wort

Maße zur Berechnung des Grades der Zugehörigkeit zu einer Kategorie

Falls hinreichend zu einer Subkategorie zugehörig

Wiederhole für Subkategorie

Vereinige gesammelte Metadaten

Quelle: [IG02]

Sample-frequency vs. Document-frequency

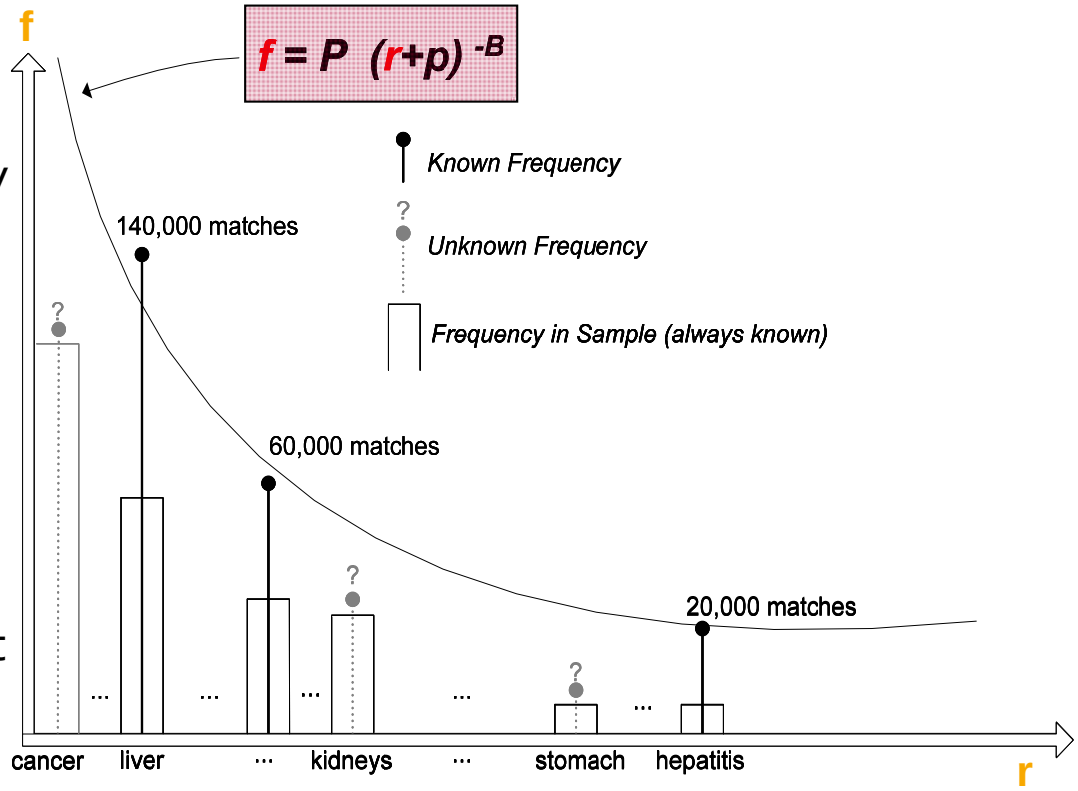
- Motivation:
  - Sample-frequencies sind nur relativ.
  - Quelle mit ähnlichem Inhalt aber unterschiedlicher Größe haben gleiche content summary.
- Sample Frequencies
  - "Leber" erscheint in 200 von 300 Dokumenten im Sample.
  - "Niere" erscheint in 100 von 300 Dokumenten im Sample.
  - "Hepatitis" erscheint in 30 von 300 Dokumenten im Sample.
- Document-frequencies
  - Anfrage "Leber" ergibt 140,000 Matches.
  - Anfrage "Hepatitis" ergibt 20,000 Matches.
  - "Niere": keine Ahnung - war kein Trainingswort…
  - "Darm" und "Krebs" waren zwar Trainingsworte, aber nur gemeinsam.

Zur Abschätzung der (besseren) Document-frequencies werden Infos der Ein-Wort Anfragen verwendet.

Quelle: Folie aus [IGS01]

Abschätzen der Document-frequencies

- Bekannt aus Algorithmus
  - Ranking *r* der Worte nach Sample-frequencies
  - Document-frequency *f* der Worte aus Ein-Wort Anfragen
- Mandelbrot's Formel verfeinert Zipfs Formel:
  - $f = P\ (r+p)^{-B}$
  - P, p und B sind Parameter der Quelle
  - Niedriger *rank* ergibt hohe *frequency*
- Dann: Kurvenanpassung
  - z.B.: $P = 8*10^5$, p =.25, B = 1.15



http://www.math.yale.edu/mandelbrot/web_pdfs/9_E7rankSizePlots.pdf

Quelle: Folie aus [IGS01]

Abschätzen der Document-frequencies

- Algorithmus also
  1. Sortiere Wörter absteigend nach Sample-frequency
  2. Ermittle P, p und B durch Fokus auf Wörter mit bekannter Document-frequency. (Kurvenanpassung)
  3. Berechne $df(w_i) = P\ (r_i+p)^{-B}$ für alle anderen Wörter.

Quelle: Folie aus [IGS01]

Vorteile des Fokussierten Sampling

- Wenige Anfragen (Fokus auf Thema)
- Vielversprechende Anfragen
- Klassifikation „*along the way*“
  - Nützlich für Auswahl relevanter Quellen
- Schätzung Document-frequency statt nur Sample-frequency.

Quelle: Folie aus [IGS01]

Overview

- Crawling the Web
- The Deep Web
- Document Conversion
- Document Storage
- Duplicate Detection
- Removing Noise

Conversion

- Text is stored in hundreds of incompatible file formats
  - Raw text, RTF, HTML, XML, Microsoft Word, ODF, PDF
- Other types of files also important
  - PowerPoint, Excel
  - Old, obsolete file formats
- Typically use a conversion tool
  - Converts the document content into a tagged text format
    - such as HTML or XML
    - => Readable in browser
    - => Retains some of the important formatting information
  - But often barely readable

These slides as seen by GDS

Dies ist eine Version von Y:\Lehre\Sommer09\VL Search Engines\SearchEngines_03_Crawling.pptx von Ihrem persönlichen Cache.
Die Datei hat sich eventuell seit dieser Zeit geändert. Klicken Sie hier, um die aktuelle Datei anzuzeigen.
Da diese Datei in Ihrem Computer gespeichert ist, funktionieren keine öffentlichen Links auf diese Datei.

Google steht unter Umständen nicht in Verbindung mit den Autoren dieser Seite und ist nicht verantwortlich für den Inhalt dieser Seite. Diese Seite ist eventuell urheberrechtlich geschützt.

Search Engines
Chapter 3 – Crawls & Feeds
30.4.2009
Felix Naumann
1
What to crawl
Every document answers at least one question:
„Now where was that document again?“
Poor quality documents
Swamp index
Slow down query answering
But
Have at least some value
Have not hindered success of commercial search engines
Thus: Crawl everything

More important:
Keep documents up to date
Keep archive of older material
Felix Naumann | Search Engines | SoSe 2009
2
2
Overview
Crawling the Web
58
Conversion
Text is stored in hundreds of incompatible file formats
e.g., raw text, RTF, HTML, XML, Microsoft Word, ODF, PDF
Other types of files also important
e.g., PowerPoint, Excel
Typically use a conversion tool
converts the document content into a tagged text format such as HTML or XML
retains some of the important formatting information
Felix Naumann | Search Engines | SoSe 2009
59
59
Character Encoding
A character encoding is a mapping between bits and glyphs
i.e., getting from bits in a file to characters on a screen
Can be a major source of incompatibility
ASCII is basic character encoding scheme for English
encodes 128 letters, numbers, special characters, and control characters in 7 bits, extended with an extra bit for storage in bytes
Felix Naumann | Search Engines | SoSe 2009
60
60
Character Encoding
Other languages can have many more glyphs
e.g., Chinese has more than 40,000 characters, with over 3,000 in common use
Many languages have multiple encoding schemes
e.g., CJK (Chinese-Japanese-Korean) family of East Asian languages, Hindi, Arabic
must specify encoding
can't have multiple languages in one file
Unicode developed to address encoding problems
Felix Naumann | Search Engines | SoSe 2009
61

Character Encoding

- A character encoding is a mapping between bits and glyphs
  - i.e., getting from bits in a file to characters on a screen
  - Can be a major source of incompatibility
- ASCII is basic character encoding scheme for English
  - encodes 128 letters, numbers, special characters, and control characters in 7 bits, extended with an extra bit for storage in bytes
- Other languages can have many more glyphs
  - e.g., Chinese has more than 40,000 characters, with over 3,000 in common use
- Many languages have multiple encoding schemes
  - e.g., CJK (Chinese-Japanese-Korean) family of East Asian languages, Hindi, Arabic
  - Must specify encoding (“code pages”)
  - Can’t have multiple languages in one file
- Unicode developed to address encoding problems

ASCII

- American Standard Code for Information Interchange

Code	…0	…1	…2	…3	…4	…5	…6	…7	…8	…9	…A	…B	…C	…D	…E	…F
0…	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1…	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2…	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3…	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4…	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5…	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6…	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7…	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

http://de.wikipedia.org/wiki/American_Standard_Code_for_Information_Interchange

Unicode

- Single mapping from numbers to glyphs that attempts to include all glyphs in common use in all known languages
- Unicode is a mapping between numbers and glyphs
  - Does not uniquely specify bits to glyph mapping!
  - e.g., UTF-8, UTF-16, UTF-32
- Proliferation of encodings comes from a need for compatibility and to save space
  - UTF-8 uses one byte for English (ASCII), as many as 4 bytes for some traditional Chinese characters
  - variable length encoding, more difficult to do string operations
  - UTF-32 uses 4 bytes for every character
- Many applications use UTF-32 for internal text encoding (fast random lookup) and UTF-8 for disk storage (less space)

Unicode

Decimal	Hexadecimal	Encoding			
0–127	0–7F	0xxxxxxx			
128–2047	80–7FF	110xxxxx	10xxxxxx		
2048–55295	800–D7FF	1110xxxx	10xxxxxx	10xxxxxx	
55296–57343	D800–DFFF	Undefined			
57344–65535	E000–FFFF	1110xxxx	10xxxxxx	10xxxxxx	
65536–1114111	10000–10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

- e.g., Greek letter pi (π) is Unicode symbol number 960
- In binary, 00000011 11000000 (3C0 in hexadecimal)
- Final encoding is **110**01111 **10**000000 (CF80 in hexadecimal)

Overview

- Crawling the Web
- The Deep Web
- Document Conversion
- Document Storage
- Duplicate Detection
- Removing Noise

Storing the Documents

- Many reasons to store converted document text
  - Saves crawling time when page is not updated
    - When index needs to be re-generated
    - Compare with HEAD response
  - Provides efficient access to text for
    - Snippet generation
    - Information extraction
    - etc.
- Disadvantages
  - Storage space
  - Longer crawl-time
- Database systems can provide document storage for some applications. But:
  - Too large volume, specialized on small data items
  - Too expensive disks
  - => Web search engines use customized document storage systems

Storing the Documents

- Requirements for document storage system:
  - Random access
    - But only simple lookups: Request the content of a document based on its URL
    - Hash function based on URL is typical
      - Hash-number identifies file and/or server
    - Then: Secondary index to locate page in file.
  - Compression and large files
    - reducing storage requirements and efficient access
  - Update
    - handling large volumes of new and modified documents
    - adding new anchor text

Large Files

- Store many documents in large files, rather than each document in a file
  - Avoids overhead in opening and closing files
  - Reduces seek time relative to read time
    - Seektime 10ms
    - Readtime 70kB/ms
- Compound document formats
  - Used to store multiple documents in a file
  - e.g., TREC Web

TREC Web Format


```
<DOC>
<DOCNO>WTX001-B01-10</DOCNO>
<DOCHDR>
http://www.example.com/test.html 204.244.59.33 19970101013145 text/html 440
HTTP/1.0 200 OK
Date: Wed, 01 Jan 1997 01:21:13 GMT
Server: Apache/1.0.3
Content-type: text/html
Content-length: 270
Last-modified: Mon, 25 Nov 1996 05:31:24 GMT
</DOCHDR>
<HTML>
<TITLE>Tropical Fish Store</TITLE>
Coming soon!
</HTML>
</DOC>
<DOC>
<DOCNO>WTX001-B01-109</DOCNO>
<DOCHDR>
http://www.example.com/fish.html 204.244.59.33 19970101013149 text/html 440
HTTP/1.0 200 OK
Date: Wed, 01 Jan 1997 01:21:19 GMT
Server: Apache/1.0.3
Content-type: text/html
Content-length: 270
Last-modified: Mon, 25 Nov 1996 05:31:24 GMT
</DOCHDR>
<HTML>
<TITLE>Fish Information</TITLE>
This page will soon contain interesting
information about tropical fish.
</HTML>
</DOC>
```

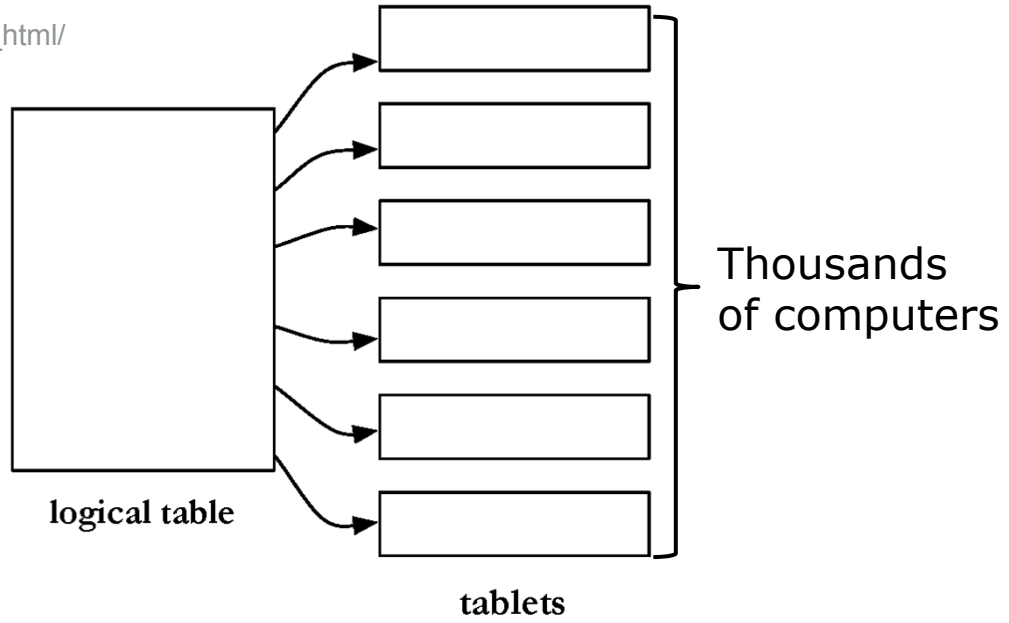
Compression

- Text is highly redundant (or predictable)
  - Entropy: Native English speakers guess next letter with 69% accuracy (Shannon 1951: http://www.princeton.edu/~wbialek/rome/refs/shannon_51.pdf)
- Compression techniques exploit this redundancy to make files smaller without losing any of the content.
  - Less disk space
  - Faster read time
- Compression of indexes covered later
- Popular algorithms can compress HTML and XML text by 80%
  - e.g., DEFLATE (zip, gzip) and LZW (UNIX compress, PDF)
  - No random access => May compress large files in blocks to make access faster

BigTable

- Google's document storage system
  - Customized for storing, finding, and updating web pages
  - Handles large collection sizes using inexpensive computers
  - Big = Petabytes, but only single table
- „Bigtable: A Distributed Storage System for Structured Data"
  - Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach
    Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber

http://www.usenix.org/events/osdi06/tech/chang/chang_html/

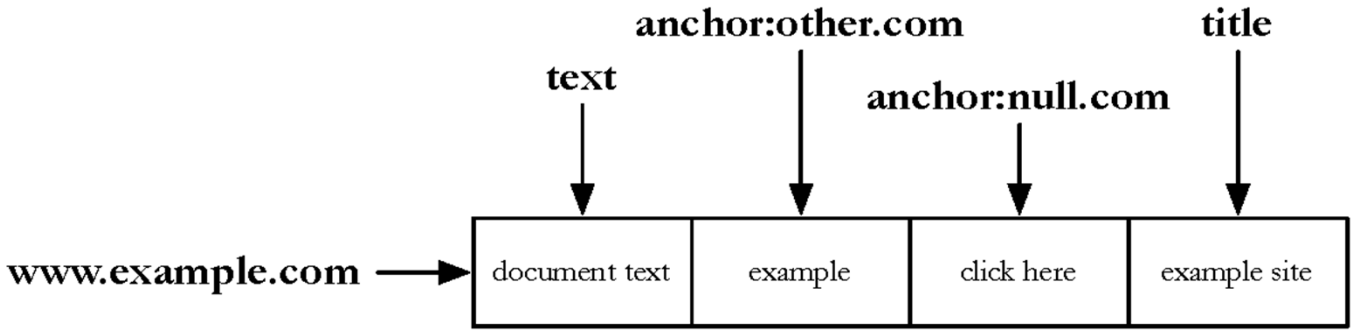


BigTable

- No query language
  - => no complex queries to optimize
- Only row-level transactions
  - => No complex locking mechanisms
- Tablets are stored in a replicated file system that is accessible by all BigTable servers
- Any changes to a BigTable tablet are recorded to a transaction log, which is also stored in a shared file system
- Data in immutable files
  - Data in file is never changed – cannot be corrupt, only incomplete
- If any tablet server crashes, another server can immediately read the tablet data and transaction log from the file system and take over.
- Table updates in RAM
  - Peridodic merge with disk files

BigTable

- Logically organized into rows
- A row stores data for a single web page
- Many attribute-value pairs



- Combination of a row key, a column key, and a timestamp point to a single *cell* in the row
  - Timestamp: Extra dimension for versioning

BigTable

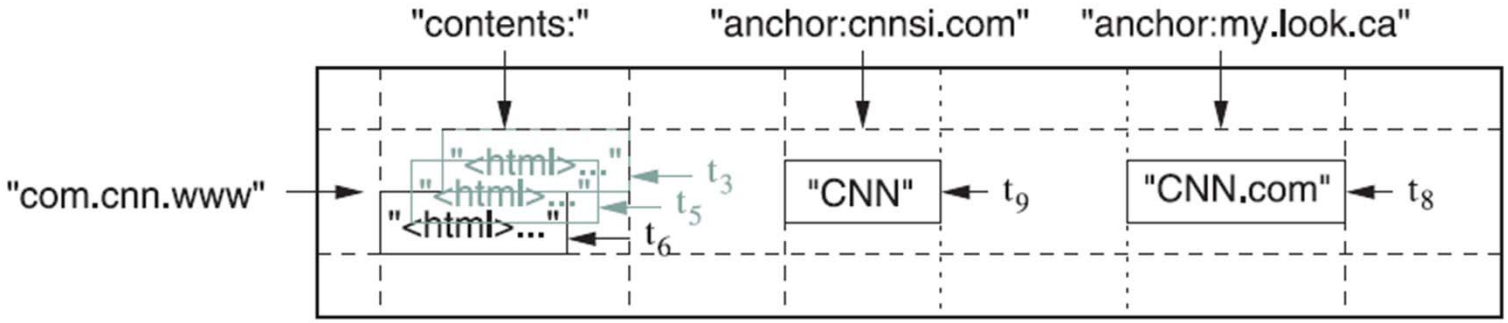


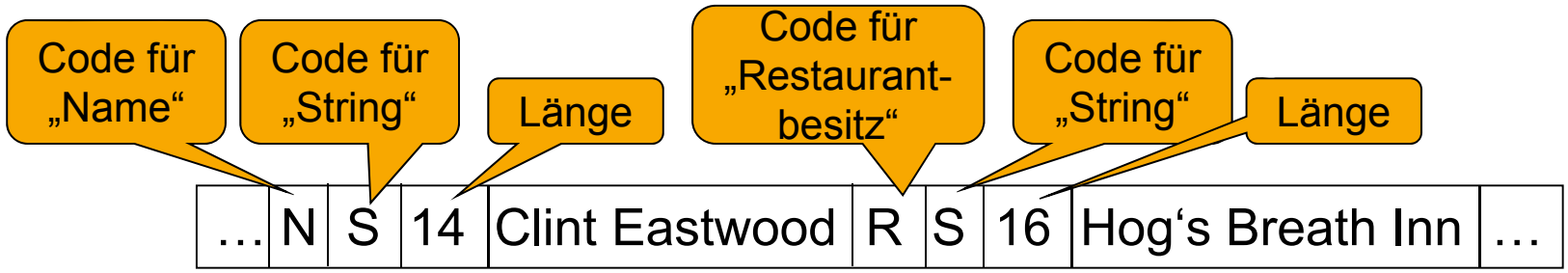
Fig. 1. A slice of an example table that stores Web pages. The row name is a reversed URL. The contents column family contains the page contents, and the anchor column family contains the text of any anchors that reference the page. CNN's home page is referenced by both the Sports Illustrated and the MY-look home pages, so the row contains columns named anchor:cnnsi.com and anchor:my.look.ca. Each anchor cell has one version; the contents column has three versions, at timestamps t_3, t_5, and t_6.

BigTable

- BigTable can have a huge number of columns per row
  - All rows have the same column families
    - Families: Columns of same type
    - Not too many families, all known in advance
  - Not all rows have the same columns
  - Important for reducing disk reads to access document data
- Rows are partitioned into tablets based on their row keys (i.e., URLs)
  - Simplifies determining which server is appropriate

From DBS II
Datensätze variabler Länge

- Es ist unbekannt welche und wieviele Felder der Datensatz haben wird.
- *Tagged fields* (getaggte Felder)
  - Feldname (Attributname)
  - Feldtyp
  - Feldlänge
  - Feldwert
- Nützlich bei
  - Informationsintegration: Es ist noch unbekannt welche Felder von Quellen hinzukommen.
  - Dünn besetzte Datensätze: Tausende Attribute, nur wenige haben Werte



Overview

- Crawling the Web
- The Deep Web
- Document Conversion
- Document Storage
- Duplicate Detection
- Removing Noise

Detecting Duplicates

- Duplicate and near-duplicate documents occur in many situations
  - Multiple links point to same page
  - Copies, versions, plagiarism, spam, mirror sites
  - 30% of the web pages in a large crawl are exact or near duplicates of pages in the other 70%.
    - Fetterly et al. 2003 (http://portal.acm.org/citation.cfm?doid=775152.775246)
- Duplicates consume significant resources during crawling, indexing, and search.
  - Little value to most users

Duplicate Detection

- *Exact* duplicate detection is relatively easy
- *Checksum* techniques
  - A checksum is a value that is computed based on the content of the document
    - e.g., sum of the bytes in the document file

T	r	o	p	i	c	a	l		f	i	s	h	*Sum*
54	72	6F	70	69	63	61	6C	20	66	69	73	68	508

  - Possible for files with different text to have same checksum
- Functions such as a *cyclic redundancy check* (CRC), have been developed that consider the positions of the bytes
  - Otherwise "acfhiiloprs T" would be duplicate

Near-Duplicate Detection

- More challenging task
  - Are web pages with same text context but different advertising or format near-duplicates?
  - Do small revisions or updates constitute a new page?
- A near-duplicate document is defined using a threshold value for some similarity measure between pairs of documents
  - e.g., document *D1* is a near-duplicate of document *D2* if more than 90% of the words in the documents are the same

Near-Duplicate Detection

- *Search*:
  - Find near-duplicates of a document *D*
  - *O(n)* comparisons required
- *Discovery*:
  - Find all pairs of near-duplicate documents in the collection
  - $O(n^2)$ comparisons
    - Really?
- IR techniques are effective for search scenario
  - Similarity based on words
  - Too computationally complex
- For discovery, other techniques used to generate compact representations: Fingerprints

Fingerprints (shingling)

1. The document is *parsed* into words.
   - Non-word content, such as punctuation, HTML tags, and additional whitespace, is removed
2. The words are *grouped* into contiguous *n*-grams, for some *n*.
   - Usually overlapping sequences of words, although some techniques use non-overlapping sequences.
3. Some of the *n*-grams are *selected* to represent the document.
   - Distinguishing feature of different algorithms
4. The selected *n*-grams are *hashed* to improve retrieval efficiency and further reduce the size of the representation.
5. The hash values are *stored*, typically in an inverted index.
6. Documents are *compared* using overlap of fingerprints
   - Absolute number of shared n-grams
   - Relative number of shared n-grams (Jacard distance)

Fingerprint Example

- Example text
  - *Tropical fish include fish found in tropical environments around the world, including both freshwater and salt water species.*
- 3-grams
  - tropical fish include, fish include fish, include fish found, fish found in, found in tropical, in tropical environments, tropical environments around, environments around the, around the world, the world including, world including both, including both freshwater, both freshwater and, freshwater and salt, and salt water, salt water species
- Hash values (hypothetical)
  - 938 664 463 822 492 798 78 969 143 236 913 908 694 553 870 779
- 3-gram selection: 0 mod 4 (hash values whose mod 4 value is 0)
  - 664 492 236 908

Simhash

- Similarity comparisons using word-based representations more effective at finding near-duplicates
  - Problem is efficiency
- Simhash combines the advantages of the word-based similarity measures with the efficiency of fingerprints based on hashing
  - Similar documents have similar hash values!
- Property of simhash: Similarity of two pages as measured by the cosine correlation measure is proportional to the number of bits that are the same in the simhash fingerprints.

Simhash

1. Process the document into a set of features with associated weights.
   - Assume simple case: Features are words weighted by their frequency
2. Generate a hash value with *b* bits (the desired size of the fingerprint) for each word.
   - The hash value should be unique for each word.
   - *b* typically several hundred
3. In *b*-dimensional vector *V*, update the components of the vector:
   - For each feature, add the weight to every component for which the corresponding bit in the word's hash value is 1.
   - Subtract the weight if the value is 0.
4. After all words have been processed, generate a *b*-bit fingerprint by setting the *i*-th bit to 1 if the *i*-th component of *V* is positive, or 0 otherwise.

Simhash Example

- Original text
  - Tropical fish include fish found in tropical environments around the world, including both freshwater and salt water species.
1. Words with weights (here: tf)
   - tropical 2 fish 2 include 1 found 1 environments 1 around 1 world 1 including 1 both 1 freshwater 1 salt 1 water 1 species 1
2. 8-bit hash values (in practice hundreds of bits)
   - tropical 01100001 fish 10101011 include 11100110
     found 00011110 environments 00101101 around 10001011
     world 00101010 including 11000000 both 10101110
     freshwater 00111111 salt 10110101 water 00100101
     species 11101110
3. Vector *V* formed by summing weights (weight for 1s; negative weight for 0s)
   - 1 -5 9 -9 3 1 3 3
4. 8-bit fingerprint formed from *V* (1 for positive, 0 for negative)
   - 1 0 1 0 1 1 1 1
- Duplicate if certain (high) percentage of fingerprint bits agree

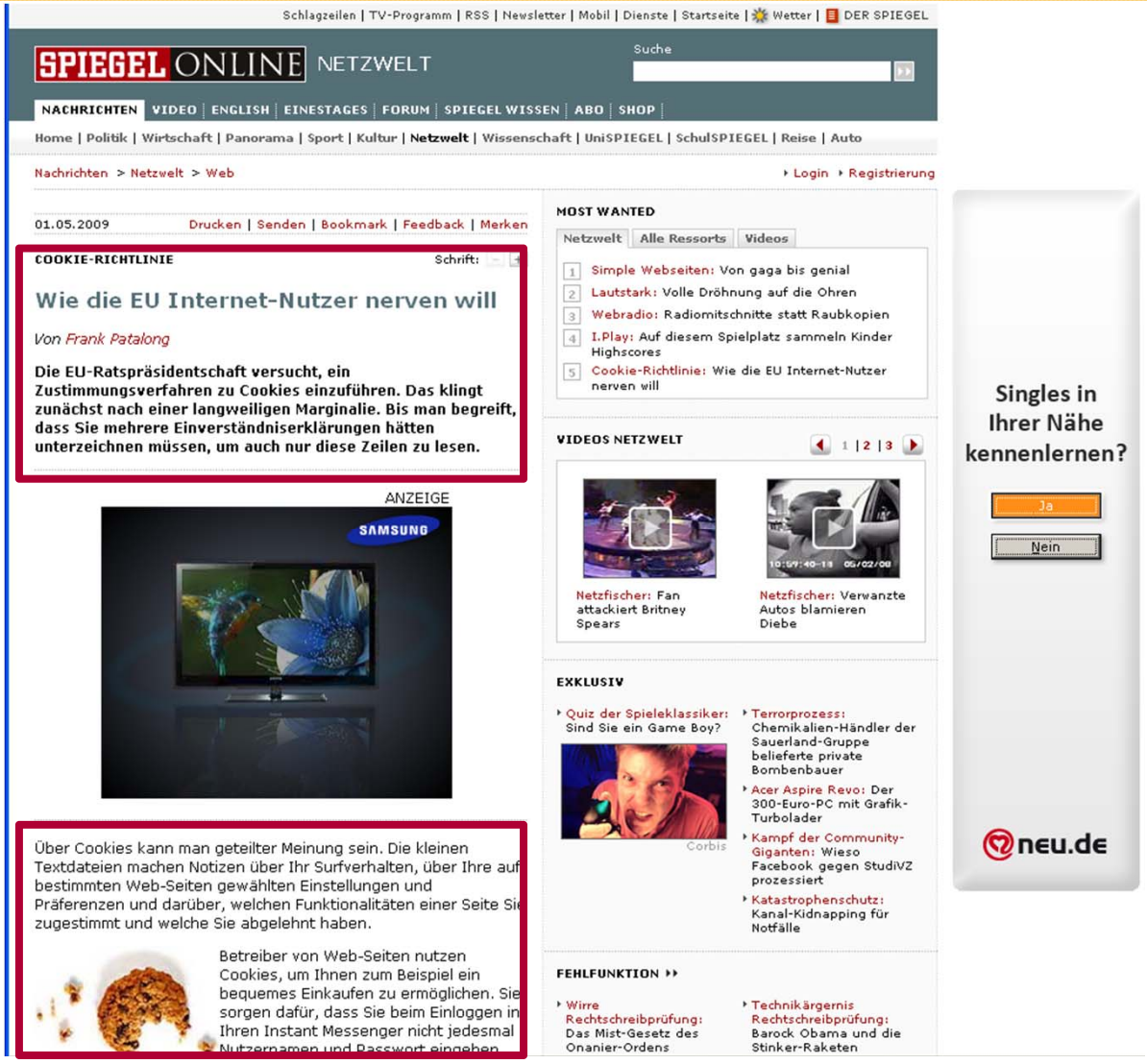
Overview

- Crawling the Web
- The Deep Web
- Document Conversion
- Document Storage
- Duplicate Detection
- Removing Noise

Removing Noise

- Many web pages contain text, links, and pictures that are not directly related to the main content of the page.
- This additional material is mostly *noise* that could negatively affect the ranking of the page.
  - Presence of large numbers of irrelevant words
- Techniques have been developed to detect the content blocks in a web page.
  - Non-content material is either ignored or reduced in importance in the indexing process

Noise example



Noise example

denn Cookies, das haben die EU-Parlamentarier durch die Diskussionen um Google, vor allem aber um den britischen Datenkraken Phorm gelernt, können dazu dienen, das Konsumverhalten von EU-Bürgern auszuspionieren. Es gäbe wohl wichtigere Datenschutzthemen, die eine Hinterfragung lohnen würden - von Ursula von der Leyens abstrusen Internet-Sperr-Plänen über Wolfgang Schäubles Überwachungsphantasien bis hin zur Bürgerdaten-Tombola im internationalen Sicherheitsbehörden-Club und zur leidigen Vorratsdatenspeicherung. Sei's drum: Die EU-Parlamentarier finden es okay, wenn unsere Vorstrafen zwecks Terrorrabwehr an US-Behörden vermailt werden und unsere ganze Kommunikation auf Monate dokumentiert wird. Hauptsache, niemand bekommt ohne ausdrückliches Einverständnis einen Cookie.

Die klammheimlichen Verordnungspläne gehen zahlreichen Medien-, Werber- und Händlerverbänden dagegen ganz schön auf den Keks.

Bürger, duck' Dich: Man hält Dich für nicht mündig

MEHR ÜBER...

Cookies Europäische Union Ratspräsidentschaft ePrivacy Datenschutz Phorm

zu SPIEGEL WISSEN

Kein Wunder, denn in der Praxis würde die Ergänzung zur ePrivacy-Richtlinie bedeuten, dass Sie auf jeder besuchten Web-Seite mit ganzen Kaskaden von Pop-up-Verträgen belästigt würden.

Jedesmal, wenn Sie einen Artikel aufriefen, eine Seite weiterblätterten, der Ad-Server, der die Werbung einspielt, eine Anzeige wechselte; wenn Sie sich bei Amazon einen Artikel ansähen, wenn Sie versuchen würden, die Artikel-Merkfunktion bei Ebay zu nutzen; wenn Sie Benutzereinstellungen in Ihrem Facebook-, MySpace- oder StudiVZ-Account vornähmen; wenn Sie sich in Ihr E-Mail-Konto einloggten, bei jedem Schritt in Ihrem Online-Banking würde Ihnen ein Pop-up serviert, in dem Sie erst einmal bestätigen müssten, dass Sie wirklich wollen, was Sie da gerade tun und damit einverstanden sind, dass Ihnen auch ermöglicht wird, was Sie wollen: umzublättern, Artikel aufzurufen, einzukaufen, Post zu lesen, Social Networks zu nutzen. Herr, wirf Hirn vom Himmel!

Europa, die Nervzone des Webs?

Man muss sich das einmal vorstellen. Mit der bequemen Nutzung des Webs wäre es vorbei, sobald man sich auf Web-Seiten bewegt, die in der Europäischen Union hinterlegt sind - denn natürlich würde all das für außereuropäische Seiten nicht gelten. SPIEGEL ONLINE unterschiede sich vom "Sydney Morning Herald online" nicht nur in Sprache, Design und geografischem Fokus, sondern auch dadurch, dass es seine Nutzer gesetzlich verordnet dauernerven müsste - denn EU-Direktiven sind nach dem Subsidiaritätsprinzip in Landesgesetze umzusetzen.

Dabei kommt viel Sinnvolles heraus, leider zu oft aber auch geballter Schwachsinn. Das Problem dabei: Oft erfahren wir zu spät davon, weil EU-Themen in der Öffentlichkeit selten diskutiert werden. Vieles, was dort in der Mache ist, erscheint als bürokratisch und sehr weit weg - bis es uns in Form gültiger Gesetze serviert wird mit dem Argument, EU-Richtlinien müssten halt umgesetzt werden.

Im Fall Cookies steht zu befürchten, dass viele der Leute, die möglicherweise schon am 5. Mai darüber abzustimmen haben, nicht den geringsten Schimmer haben, worüber sie da eigentlich abstimmen. Es einfach nicht begreifen, sich nicht vorstellen können. Das Thema nicht erfassen, weil sie es nicht kennen. Das Internet nicht nutzen, sondern nutzen lassen, aber durchaus darüber entscheiden. In informierten Kreisen nennt man solche einflussreichen Menschen auch "Internet-Ausdrucker". Man kann nur hoffen, dass von denen wenige beteiligt sind; dass auch unsere deutschen Abgeordneten die Wichtigkeit des vermeintlich technischen Themas begreifen.

Denn man kann leider nicht davon ausgehen, dass das Thema vor der Abstimmung wirklich vertieft, erklärt und beredet würde. Es ist eine kleine, scheinbar marginale Änderung von vielen in einer Richtlinie, die nur einige wenige wirklich lesen werden. Erheblich größer wird die Zahl der Betroffenen ausfallen: Aktuellen Schätzungen zufolge gibt es davon allein in Deutschland rund 55 Millionen. Gemeint sind alle Internet-Nutzer, vor allem aber die, die dort versuchen, ihr Geld zu verdienen.

Verordneter Wettbewerbsnachteil

Ingo Notthoff, Sprecher des Bundesverbands Digitale Wirtschaft BVDW, entdeckt in den wohlmeinenden Plänen einen "Rückschlag für Internet-Nutzer und die Online-Wirtschaft". Er ist damit nicht allein: Wenn eine EU-Richtlinie die Nutzung europäischer Web-Seiten verkomplizieren würde, liefe das auf eine verordnete Wettbewerbsverzerrung hinaus. "Die Einführung eines Opt-in-Systems würde die Nutzung des Internets für den einzelnen Nutzer durch zahlreiche Zustimmungserfordernisse verlangsamen und den Zugang zu Online-Services und -Inhalten verkomplizieren", heißt es auch in einem Brief der Verlegerverbände an die deutschen Europa-Abgeordneten. "Eine einfache und verbraucherfreundliche Nutzung von Web-Seiten wäre dann nicht mehr möglich."

IHRE MEINUNG IST GEFRAGT

Diskutieren Sie über diesen Artikel

Es ist sogar noch schlimmer: Die Regelung ist schlicht kontraproduktiv. Sie schützt die Daten der Nutzer nicht, sondern nervt sie nur unnötig. Wenn ich mit meiner Browsereinstellung, die alle Cookies und Verlaufsdaten nach jeder Sitzung löscht, solch eine EU-kompatible Seite besuche, werde ich jedes Mal aufs Neue mit diversen "Sind Sie einverstanden, dass ..."-Fenstern genervt. Mein Kollege, der auf Datenschutz pfeift und alle Cookies und Verlaufsdaten bis zum Jüngsten Gericht speichert, wird damit nur ein einziges Mal konfrontiert: Sein einmal gegebenes Einverständnis bleibt archiviert.

Was bitte schön soll an dieser Regelung im Sinne des Verbrauchers sein? Wie soll ihn das schützen? Wer sich selbst schützt, indem er Cookies killt, statt sich damit EU-konform einverstanden zu erklären, wird mit endlosen Pop-up-Kaskaden bestraft. Wer seinen Datenschutz vernachlässigt, wird auch mit den EU-Cookie-Pop-ups nicht behelligt.

Erde an Brüssel: jemand zu Hause?

Diskutieren Sie über diesen Artikel! FORUM

DIE NEUESTEN BEITRÄGE ZUM ARTIKEL

1. heute 13.45 Uhr von Fackus Glaube macht selig
2. heute 13.25 Uhr von SeaBug ohne Wertung
3. heute 13.11 Uhr von StonyBrook Sinn/Unsinn von Cookies
4. heute 12.27 Uhr von Fackus na denn
5. heute 12.26 Uhr von Boïndil Cookies

Alle Beiträge zum Artikel

MELDUNGEN AUS ANDEREN RESSORTS

WISSENSCHAFT

Impfstoff gegen Schweinegrippe: Roulettespiel mit dem Seuchenvirus
Forscher arbeiten an einem Impfstoff gegen das Schweinegrippe-Virus - fürchten aber, dass H1N1 mutiert. Kann ein jetzt entwickeltes Serum überhaupt vor einer veränderten Version des Erregers schützen?
Von Heike Le Ker mehr... [Forum]

- **Klimaschutz:** Australien verschiebt Emissionshandel
- **Drogen- und Suchtbericht:** Deutschlands Jugend hat ein Alkoholproblem
- **Arktisches Monopoly:** Heimliche Herren des Eismeer-Schatzes
- **mehr Wissenschaft**

KULTUR

Taschenbuch-Bestseller: Buch schreiben, Münze reiben
Infotainment heißt das Genre, Moderator Ranga Yogeshwar gehört im TV zu seinen erfolgreichsten Vertretern. Jetzt hat der studierte Physiker ein Buch vorgelegt. Ein bisschen Spaß, ein bisschen Know-how: Fertig ist der Wissens-Bestseller. mehr...

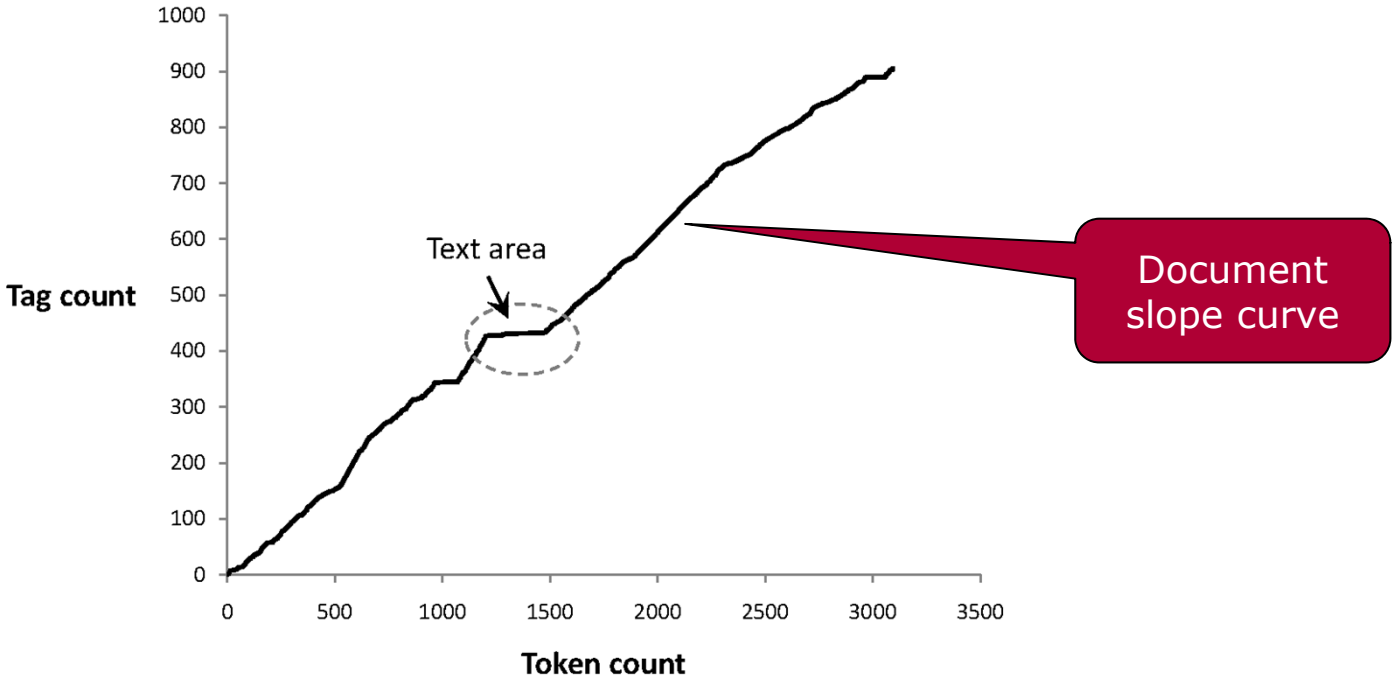
- **Soulstar Raphael Saadiq:** Verehrer, Verführer, Veredler
- **Karmitz-Ausstellung:** "Öl, Ei, Salz und rühren"
- **Box Office:** Wolverine krallt sich die Kinokasse
- **mehr Kultur**

© SPIEGEL ONLINE 2009
Alle Rechte vorbehalten
Vervielfältigung nur mit Genehmigung der SPIEGELnet GmbH

TOP

Finding Content Blocks

- Intuition: less HTML tags in main content
- Cumulative distribution of tags in the example web page



  - Main text content of the page corresponds to the "plateau" in the middle of the distribution

Finding Content Blocks

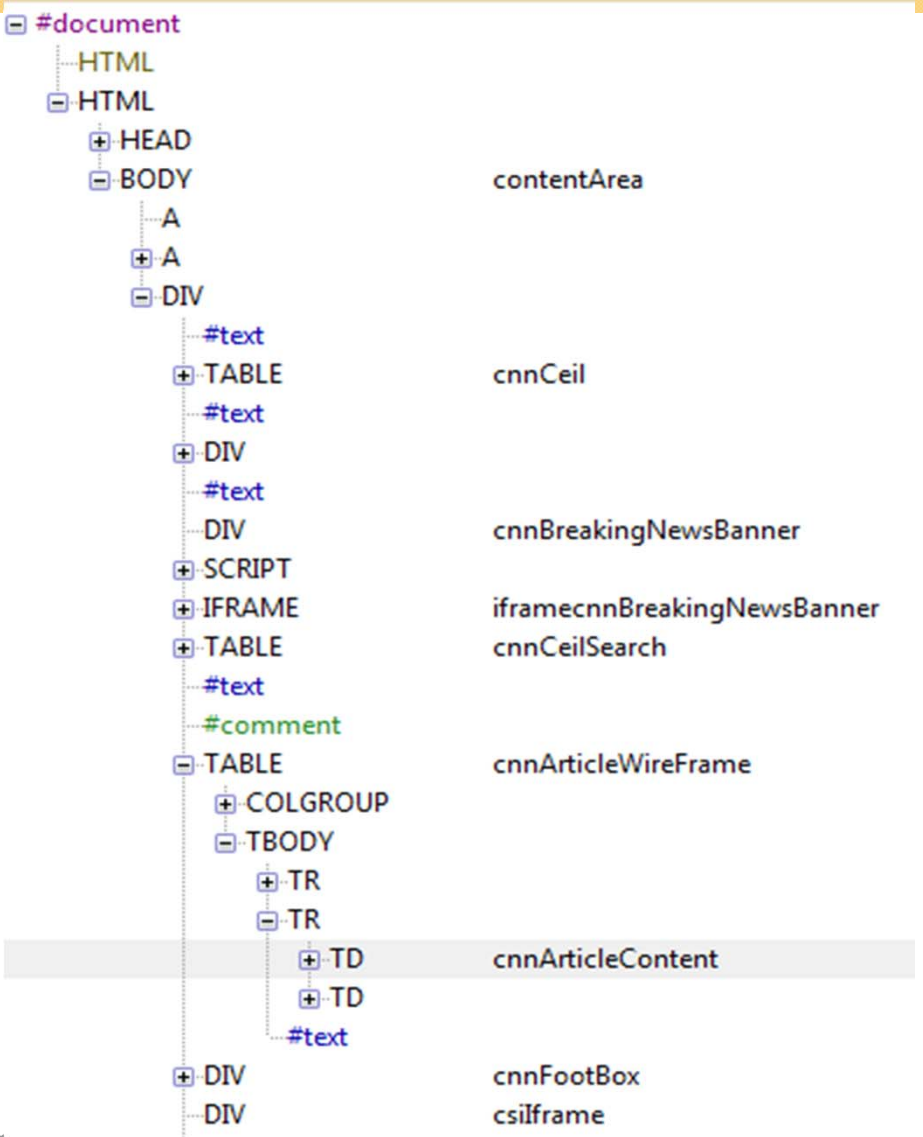
- Represent a web page as a sequence of bits, where $b_n = 1$ indicates that the nth token is a tag
  - Corresponds to document slope curve
  - Ignore text formatting tags
- Optimization problem where we find values of i and j to
  - Maximize the number of tags below i
  - Maximize the number of tags above j
  - Maximize the number of non-tag tokens between i and j
  - i.e., maximize

$$\sum_{n=0}^{i-1} b_n + \sum_{n=i}^{j} (1 - b_n) + \sum_{n=j+1}^{N-1} b_n$$

  - Compute by scanning i-s and j-s

Finding Content Blocks

- Other approaches use DOM structure and visual (layout) features
- Effective for multiple content sections in a document
  - As opposed to the previous tag distribution approach



Deep Web Literature

- Wichtigste Literatur
  - [IGS01] Probe, Count, and Classify. P.G. Ipeirotis, L. Gravano, and M. Shami. SIGMOD 2001
  - [BC04] A. Bergholz and B. Chidlovskii. Learning Query Languages of Web Interfaces, SAC04

- Weiteres
  - [Be01] The Deep Web: Surfacing Hidden Value Michael K. Bergman, Whitepaper at http://www.completeplanet.com/Tutorials/DeepWeb/index.asp
  - [To01] Foliensatz von Dawne Tortorella (BellCow) nach [Be01]
  - [IG02] Distributed Search of the Hidden Web: Hierarchical Data Sampling and Selection. P.G. Ipeirotis and L. Gravano in VLDB 2002.