

openHPI: **Wie funktioniert eine Suchmaschine?**
Crawling

Dr. Ralf Krestel
Hasso-Plattner-Institut
Universität Potsdam

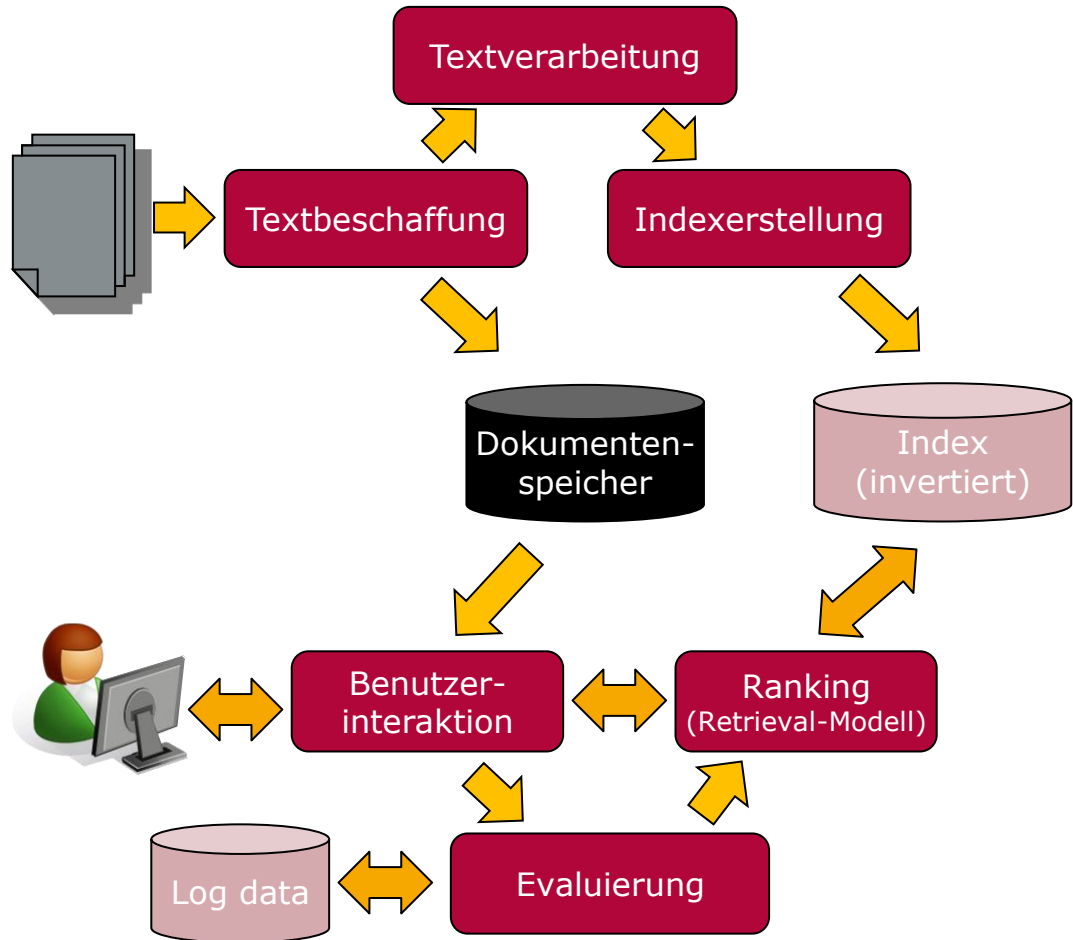
■ Grundlegende Bausteine

□ Indexierung

- Textbeschaffung
- Textverarbeitung
- Indexerstellung

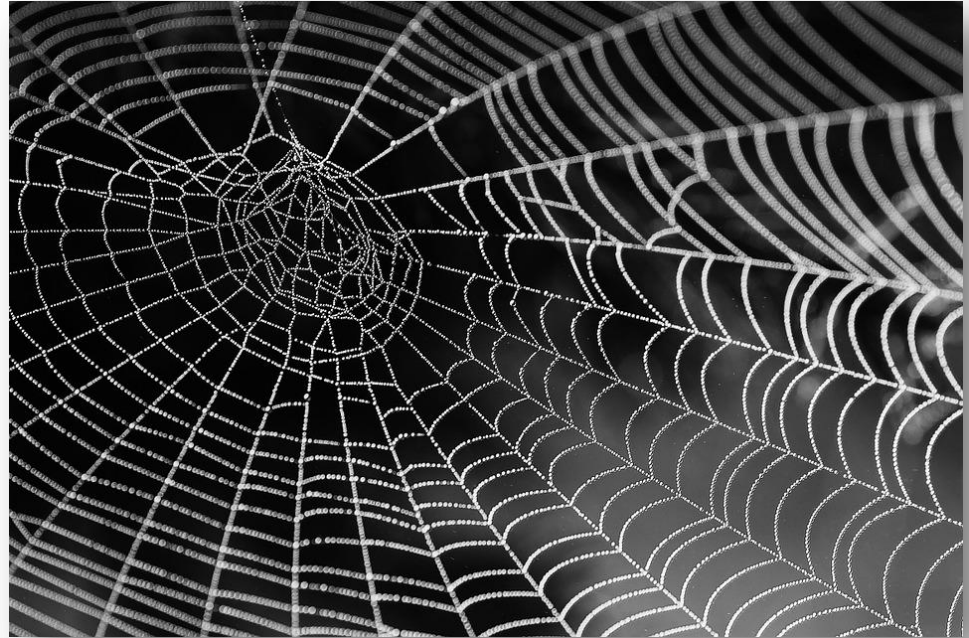
□ Anfrage

- Benutzerinteraktion
- Ranking
- Evaluierung



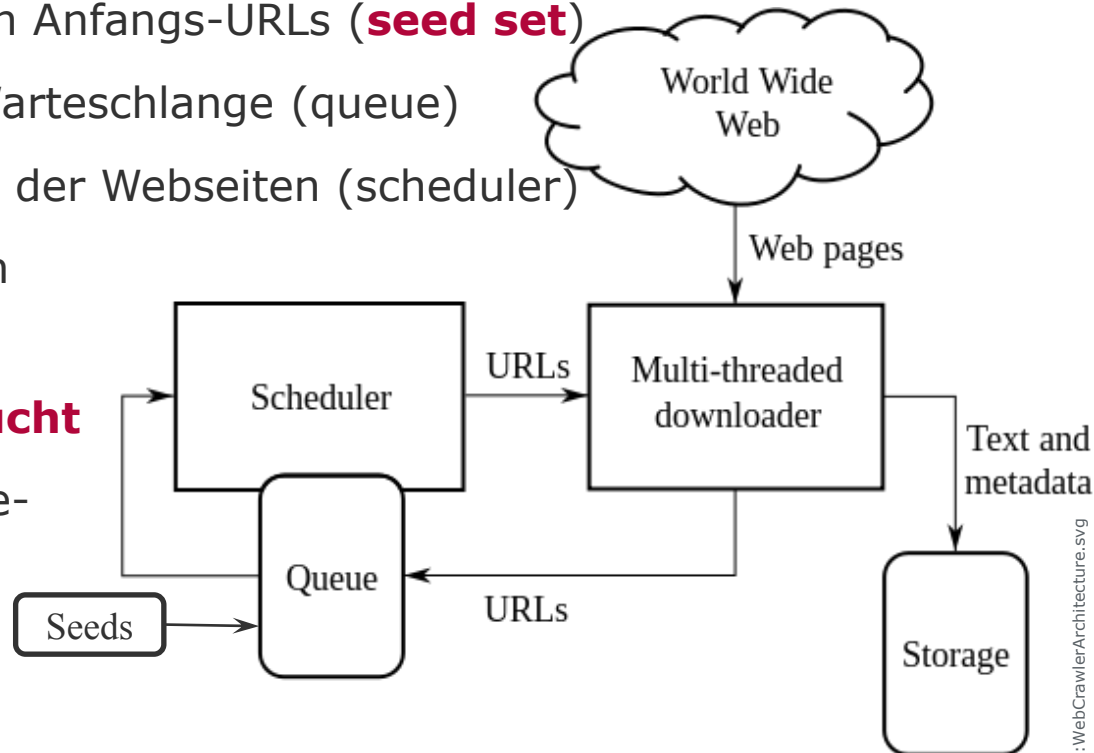
- Jedes Dokument beantwortet mindestens eine Anfrage:
 - „Wo war dieses Dokument noch mal...?“
- Qualitativ minderwertige Dokumente
 - fluten den Index
 - verlangsamen die Anfragebearbeitung
- Aber...
 - haben zumindest irgendeinen, wenn auch kleinen, Wert
 - haben den Erfolg kommerzieller Suchmaschinen nicht verhindert
- Daher: **Alles (so viel wie möglich) indexieren**
- Wichtiger:
 - Dokumente auf dem **aktuellen** Stand halten
 - Ältere Dokumente **archivieren**

- **Web Crawler (Spider)** finden Webseiten zum indexieren
- Theoretisch einfach: meiste Seiten wollen gefunden werden
- Problem: Es gibt kein globales Verzeichnis aller Webseiten
- Crawler folgt Links um neue Seiten zu entdecken
- Herausforderungen:
 - Web **wächst** ständig (Nicht zählbar viele Seiten (Was ist eine Seite?))
 - Suchmaschine muss Seiten **speichern** → riesiger Speicherbedarf
 - Manche Seiten sind **versteckt** (Hidden Web)
 - Seiten **ändern** sich ständig



<http://maxpixel.freepresspicture.com/Network-Dewdrop-Spider-Web-With-Water-Beads-921039>

- Ausgehend von einer Menge von Anfangs-URLs (**seed set**)
 - **Hinzufügen** der URLs zur Warteschlange (queue)
 - Crawler **plant** herunterladen der Webseiten (scheduler)
 - Herunterladen der Webseiten
 - Heruntergeladene Webseiten werden nach **Links durchsucht**
 - Neue URLs werden zur Warteschlange **hinzugefügt**
 - Crawlergrenze (**Frontier**)



- Crawler warten sehr oft auf Antwort von Servern (DNS, Verbindung, Download)
 - Daher multi-threaded!
- Um keinen Webserver lahmzulegen
 - Eine Warteschlange pro Server, langsames Crawlen (**politeness policy**)

- Wer nicht will, dass seine Seite indexiert wird, kann dies durch Regeln in einer Datei namens `robots.txt` im Wurzelverzeichnis seiner Webseite festlegen

```
User-agent: *  
Disallow: /private/  
Disallow: /confidential/  
Disallow: /other/  
Allow: /other/public/  
  
User-agent: FavoredCrawler  
Disallow:  
  
Sitemap: http://mysite.com/sitemap.xml.gz
```

Alle Crawler

Spezial-Crawler

Zeige Hidden-Web

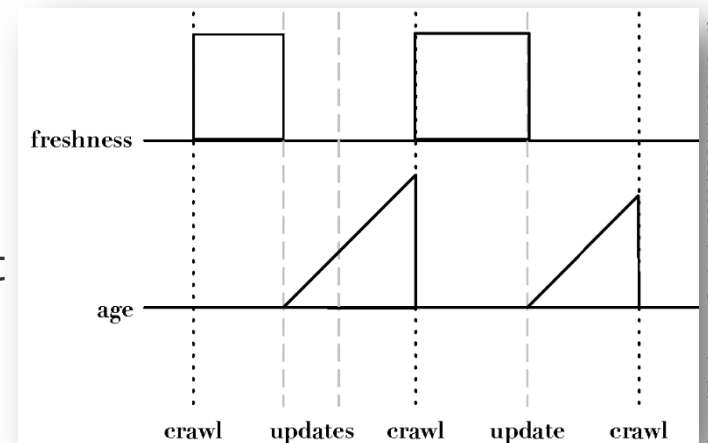
- **Sitemaps** beinhalten URLs und Daten über diese URLs (Änderungszeitpunkt, Änderungsfrequenz)
- Werden von Webserveradministratoren erstellt
- Enthält Informationen über Seiten, die ein Crawler sonst **nicht finden** würde
- Gibt **Hinweise**, wann ein Crawler wieder vorbeischauchen sollte

```
<?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>
```

Zwischen 0.0 und 1.0 (default 0.5)

always
hourly
daily
weekly
monthly
yearly
never

- Webseiten können sich **ändern**
 - Crawler muss immer wieder vorbei schauen und ggf. Seite herunterladen
- **Unmöglich** alle Seiten ständig zu **überprüfen**
 - Wichtige Seiten und sich schnell ändernde Seiten bevorzugen
 - Unterschiedliche Arten ändern sich unterschiedlich schnell
 - Homepage vs. Online-Nachrichten
 - *HTTP Head Request* kann „Last-Modified“ Informationen liefern
- **Wann crawlen?** (Cho und Garcia-Molina 2000)
- „Freshness“: Anteil der Seiten im Index die aktuell sind (fresh, up-to-date, vs. stale)
 - Führt zu schlechten Ergebnissen: Seite ändert sich ständig → nie frisch → nie gecrawlt
- Besser „Age“
 - Crawlen lohnt sich immer



- **Fokussiertes** (Thematisches) Crawl versucht nur Seiten herunterzuladen, die ein bestimmtes Thema abdecken
 - Wird von vertikalen Suchmaschinen benutzt
- Versprechen: **Höhere Genauigkeit**
 - Benutzer wählt eine bestimmte vertikale Suchmaschine und schränkt damit Menge der relevante Seiten ein
 - Index enthält **keine irrelevanten** Seiten
- Basiert auf der Annahme, dass Seiten eines **Themas** eher auf Seiten des gleichen Themas **verlinken**
 - Ankertext hilft
 - Beliebte Seiten eines Themas dienen als Ausgangspunkte (seeds)
- Crawler benutzt einen **Textklassifizierer**, um festzustellen, ob Seite über ein Thema ist

- **Mehrere Computer** zum Crawlten benutzen
 1. Crawler kann **näher** an der zu crawlenden Seite sein
 - Weniger Latenz, hoher Durchsatz
 2. Jeder einzelne Crawler muss **weniger** Seiten verwalten
 - Im Index
 - In der Warteschlange
 - Im Scheduler
 3. **Reduziert** die benötigte Rechenleistung
 - Parsen, Netzwerkbandbreite
- URLs werden mit Hilfe einer Hash-Funktion auf Rechner verteilt
 - Und gebündelt verschickt
 - Hash-Funktion über host-Teil der URL
 - Führt zu Ungleichgewicht, da hosts unterschiedlich viele Seiten haben
 - Nicht schlimm

- Wird für Desktop- und Enterprise-Suche verwendet
- Unterschiede zum Web Crawl:
 - Viel einfacher Dokumente zu finden
 - Auf Änderungen muss schneller reagiert werden
 - Erwartung der Nutzer sind anders als im Webkontext
 - Dateisystem kann Crawler bei Änderungen benachrichtigen (push vs. pull)
 - Viele unterschiedliche Dateiformate
 - Umwandeln nötig
 - Datenschutz und Privatsphäre sehr wichtig
 - Ressourcenschonender Prozess (CPU, RAM, Festplatte)
 - Vorhalten von Kopien der Dokumente nicht notwendig