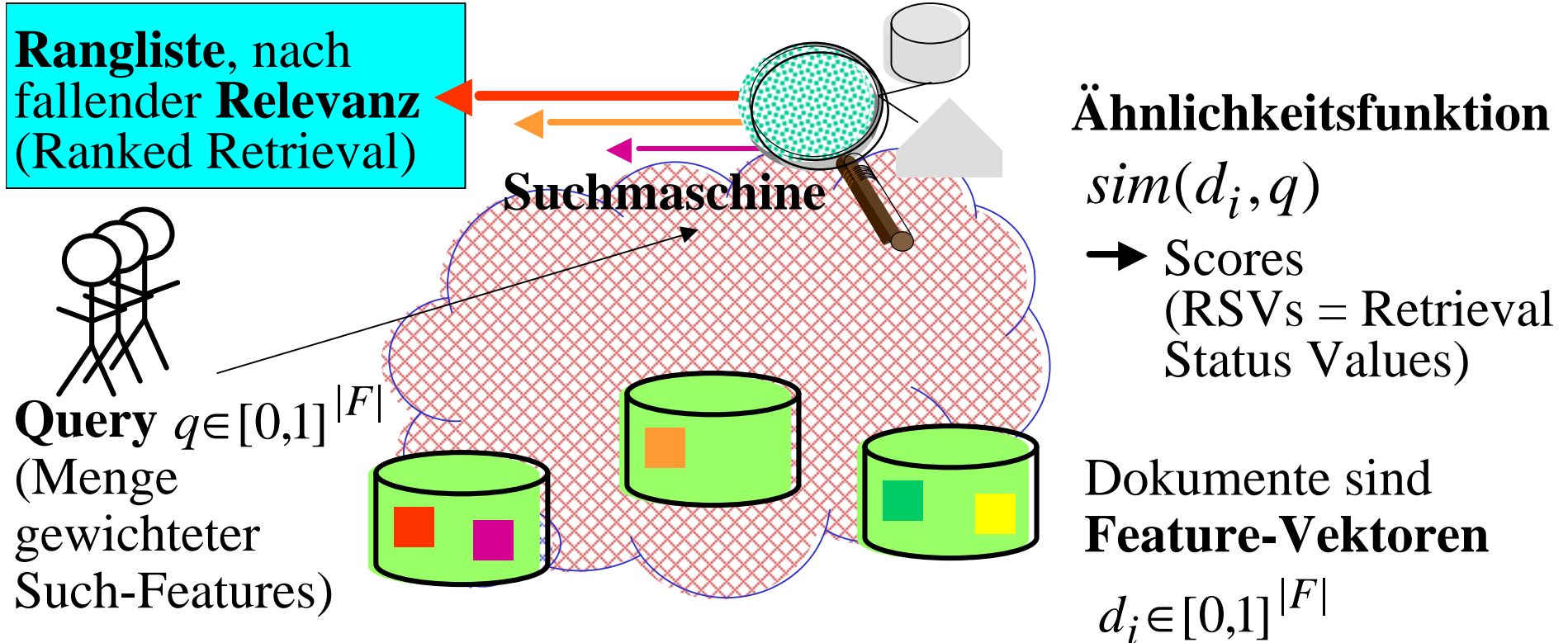


2 Vektorraummodell und IRS-Architektur

Grundprinzipien:

- **Featureraum:** Wörter in Dokumenten werden auf Terme reduziert.
- **Dokumentenmodell:** Jedes Dokument d_i wird als Vektor $\in [0,1]^{|F|}$ repräsentiert, wobei d_{ij} das Gewicht des j -ten Terms in d_i angibt.
- **Anfragemodell:** Anfragen sind Vektoren $q_i \in [0,1]^{|F|}$
- **Relevanz:** Suchresultatsranking basiert auf einer Ähnlichkeitsfunktion im Vektorraum $[0,1]^{|F|}$
- **Crawling:** Das Web wird entlang von Hyperlinks traversiert, um Dokumente zu analysieren und zu indexieren.
- **Indexierung:** Zu jedem Term wird eine Liste von Dokumenten-Ids (z.B. URLs) mit dem jeweiligen Gewicht in einem „invertierten File“ (Suchbaum oder Hash-File) angelegt.
- **Anfrageverarbeitung:** Anfragen werden zerlegt in Index-Lookups für Einzelterme, um Trefferkandidatenlisten zu bestimmen.

Vektorraummodell für Relevanz-Ranking



Verwendete Ähnlichkeitsfunktionen sind z.B.:

$$sim(d_i, q) := \sum_{j=1}^{|F|} d_{ij} q_j$$

(Skalarprodukt)

$$sim(d_i, q) := \frac{\sum_{j=1}^{|F|} d_{ij} q_j}{\sqrt{\sum_{j=1}^{|F|} d_{ij}^2 \sum_{j=1}^{|F|} q_j^2}}$$

(Cosinus-Maß)

Termgewichtung in Dokumenten

Betrachtet werden die folgenden Werte
(für N Dokumente und M Terme):

- tf_{ij} : Häufigkeit (term frequency) des Terms t_i in Dokument d_j
- df_i : Anzahl der Dokumente mit Term t_i (doc. frequency)
- idf_i : N / df_i (inverse document frequency)
- cf_i : Häufigkeit von t_i in allen Dokumenten (corpus frequency)
(ggf. mit separater Berücksichtigung von Termen in title u.ä.)

Grundprinzip:

Das Gewicht w_{ij} von Term t_i in Dokument d_j sollte mit tf_{ij} und mit idf_i monoton wachsen.

→ erster Ansatz: $w_{ij} = tf_{ij} * idf_i$ (**tf-idf-Formel**)

Ggf. sollten die Gewichte w_{ij} wie folgt zu ω_{ij} normiert werden:

$$\omega_{ij} := w_{ij} / \sqrt{\sum_k w_{kj}^2}$$

Variationen der Termgewichtung mit tf und idf

Empirische Resultate zeigen, daß in der Regel die tf- und idf-Werte normalisiert und/oder gedämpft sein sollten.

Normalisierung tf-Werte: $tf_{ij} := \frac{tf_{ij}}{\max_k tf_{kj}}$

Gedämpfte tf-Werte: $tf_{ij} := (1 + \log tf_{ij})$

Gedämpfte idf-Werte: $idf_i := \log \frac{N}{df_i}$

Häufige Variante:

$$w_{ij} := \frac{tf_{ij}}{\max_k tf_{kj}} \log \frac{N}{df_i}$$
$$\mathbf{w}_{ij} := w_{ij} / \sqrt{\sum_k w_{kj}^2}$$

**tf*idf-
Formel**

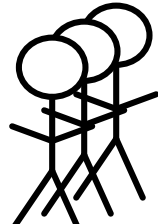
Termgewichtung in Anfragen

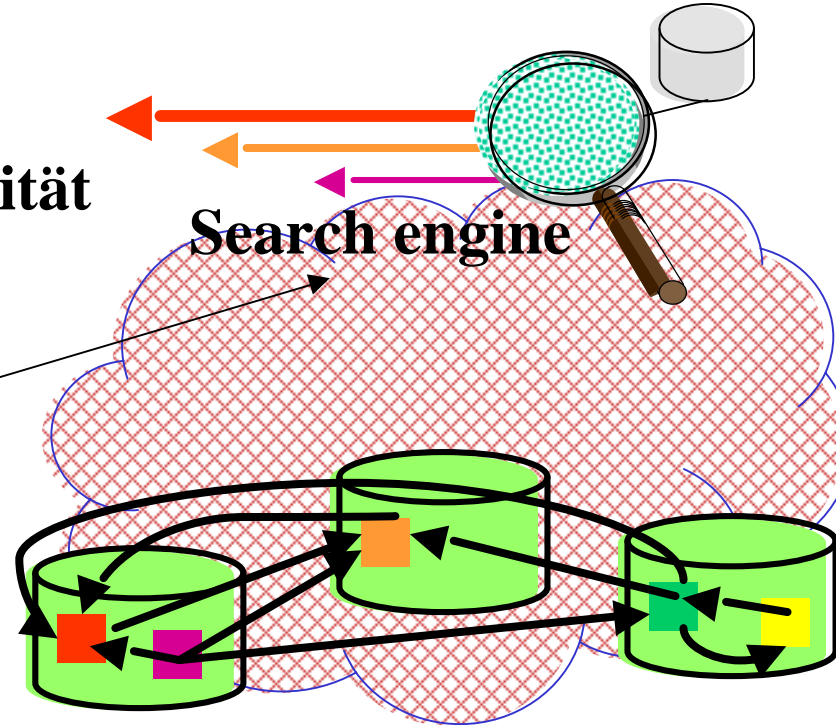
Je nach Anfrage-Interface und Benutzerkategorie kommen z.B. die folgenden Gewichtungen der Terme in einer Anfrage q_j in Frage:

- $w_{ij} = (0.5 + \frac{0.5 tf_{ij}}{\max_k tf_{lj}}) \times \log \frac{N}{df_i}$
- $w_{ij} = 1$ falls t_i in q_j vorkommt, 0 sonst
- $w_{ij} = 1 / k$
falls q_j k (konjunktiv verknüpfte) Terme enthält
und t_i der an l -ter Stelle in q_j aufgeführt ist

Linkanalyse für Autoritäts-Ranking

Rangliste nach
fallender
Relevanz & Autorität


Query $q \in [0,1]^{|F|}$
(Menge
gewichteter
Such-Features)



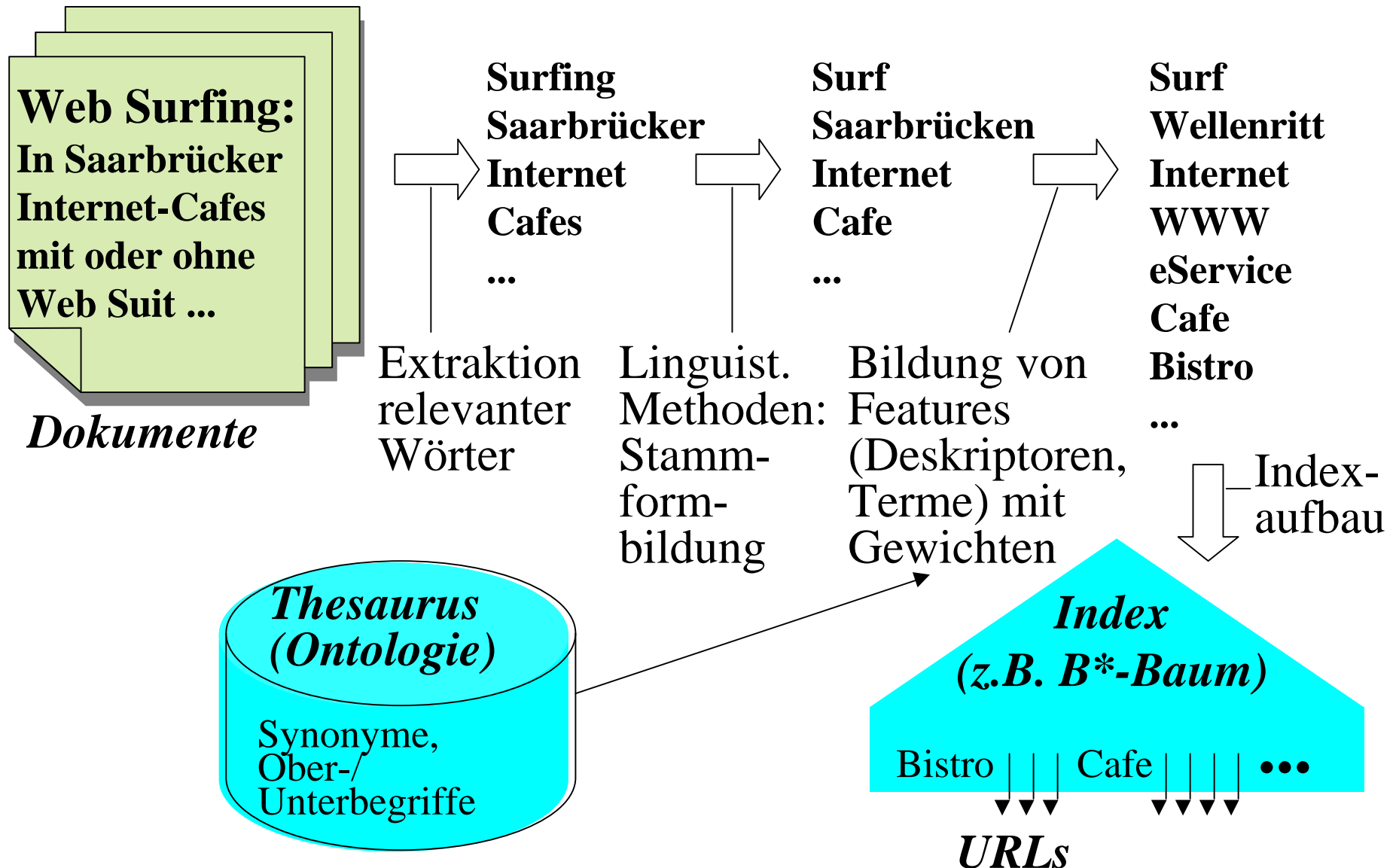
+ Analysiere In-Degree und Out-Degree von Webseiten:

Autoritätsrang (d_i) :=

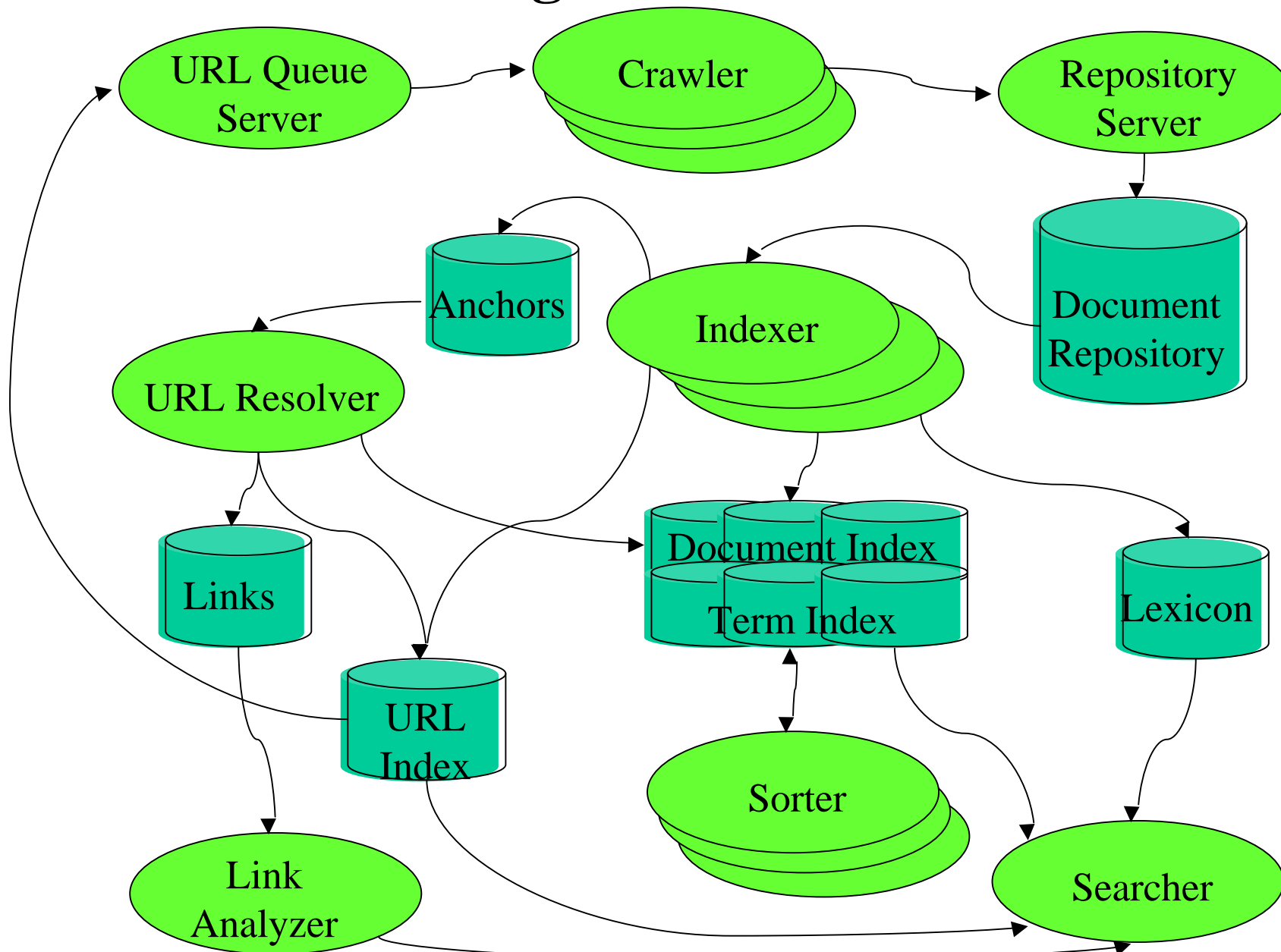
Stationäre Besuchswahrscheinlichkeit [d_i]
bei Random Walk auf dem Web

Gesamtrang = (ad hoc) gewichtete Summe von
Relevanz und Autorität

Inhaltserschließung von Dokumenten und Indexierung



Crawling und Indexaufbau



Datenstrukturen einer Web-Suchmaschine

- **Document Repository:**
alle traversierten HTML-Dokumente in komprimierter Form
- **Lexicon:** alle vorkommenden Stammformen jeweils mit TermId
- **DocumentIndex:** sortiertes/indexiertes File mit Einträgen der Form
(DocId, Terms(TermId, #Hits, tf-Value, Hits(PosFontTagEncoding)))
- **TermIndex:** sort./index. invertiertes File mit Einträgen der Form
(TermId, #Docs, idf-Value, Docs(DocId, #Hits, Hits(...)))
- **Anchors:** alle Hyperlinks in der Form
(SourceDocId, TargetDocId, AnchorText)
- **URLIndex:** sortiertes/indexiertes File mit Einträgen der Form
(URL, DocId, PtrToRepository) z.B. sortiert nach Checksum(URL)

Komponenten einer Web-Suchmaschine

- **URL Queue Server:** verwaltet Prioritätsliste (noch) zu traversierender Links (z.B. basierend auf In-degree)
- **Crawler (Robot, Spider):** holt Dokumente unter Beachtung von Nebenbedingungen (Filetyp, Robot Exclusion Protocol, usw.)
- **RepositoryServer:** verwaltet DocumentRepository
- **Indexer (incl. Parser, Stemmer):** analysiert Dokumente und erzeugt Einträge in Lexicon, Anchors und DocumentIndex
- **URLResolver:** übersetzt URLs in DocIds (SourceDocId, TargetDocId, AnchorText)
- **Sorter:** erzeugt TermIndex aus DocumentIndex
- **LinkAnalyzer:** berechnet Autoritäts-Ranking aufgrund von Links
- **Searcher:** wertet Anfragen durch Lexicon-, Index- und ggf. Thesaurus-Lookups aus und berechnet Resultats-Ranking aufgrund von tf (evtl. separat für title u.ä.), idf und Autoritätsrang; holt Kandidatendokumente und prüft ggf. weitere Bedingungen

Exkurs: Effizientes Sortieren auf Sekundärspeicher

Annahmen: Dateigröße S Blöcke, Hauptspeichergröße M Blöcke

- 1) Bilden von sortierten Teilsequenzen (Runs):
Lade M Blöcke in Hauptspeicher, sortiere intern und schreibe Run in temporäres File.
- 2) Mischen der Runs:
Mische M Runs zu einem längeren Run und schreibe in temp. File.
- 3) Iteriere 2), bis ein einziger Resultat-Run entsteht.

Platten-I/O-Kosten:

- $2S$ I/Os, um S/M Runs der Größe M zu bilden.
- 2) $2S$ I/Os, um S/M^2 Runs der Größe M^2 zu bilden.
- 3) insgesamt $i-1$ Mischphasen bis $S/M^i \leq 1$, also $\log_M S \leq i$

Gesamtkosten: $\text{ceil}(\log_M S - 1)2S + 2S$ Platten-I/Os
(bzw. $\leq 4S$ bei $M \geq \sqrt{S}$)

Dimensionen typischer Web-Suchmaschinen

- > 200 Mio. Dokumente
 - > 1 Terabyte Rohdaten
- > 10 Mio. Terme
 - > 1 Terabyte Index
- > 50 Mio. Anfragen pro Werktag
 - < 1 Sek. mittlere Antwortzeit
- < 30 Tage Indexaktualität
 - > 200 Webseiten pro Sek. Crawling (> 1 MByte/s)

Typische (High-End-) Serverkonfiguration:

20 SMPs mit jeweils

- 10 CPUs
- 12 GBytes RAM
- 300 GBytes RAID-Plattenplatz

Vereinfachter Query Processor

- 1) Entferne Stoppwörter aus der Anfrage und
reduziere alle anderen Wörter auf Terme (in Stammform)
- 2) Transformiere Anfrage q in die Form
$$\left. \begin{array}{l} [(t_{11} \text{ AND } t_{12} \text{ AND } \dots \text{ AND } t_{1k_1}) \text{ OR } \dots \text{ OR} \\ (t_{r1} \text{ AND } t_{r2} \text{ AND } \dots \text{ AND } t_{rk_r})] \end{array} \right\} \begin{array}{l} q_1 \dots \\ q_r \end{array}$$
$$\text{AND NOT } t_{(r+1)} \text{ AND NOT } \dots \text{ AND NOT } t_{(r+s)}$$
- 3) for $i=1$ to r do
 - Finde DocId-Listen für t_{i1} bis t_{ik_i} und
 - berechne Vereinigung V_i dieser DocIds
 - Setze den RSV von $d \in V_i$ auf $\text{sim}(d, q_i)$
 - Berechne Vereinigung V der DocId-Mengen V_1, \dots, V_r
- 4) Sortiere alle $d \in V$ absteigend nach ihren RSVs
- 5) Hole Dokument $d \in V$ in Sortierfolge
 - ggf. mit einem heuristischen Abbruchkriterium -
 - und eliminiere Dokumente d , in denen einer oder mehrere
 - der Terme $t_{(r+1)}$ bis $t_{(r+s)}$ vorkommen

Beispiel für Vektorraummodell

- d1: Trainer Daum hat nie Drogen genommen.
Er hat weder Kokain noch andere Drogen genommen.
- d2: Manager Hoeness beschuldigt Daum des Drogenkonsums.
Der frühere Bundestrainer Beckenbauer bedauert Daum.
- d3: Fussballverein Leverkusen entlässt Trainer Daum.
Völlner übernimmt Training.
- d4: Daums Haarprobe weist Konsum von Drogen nach.
Der deutsche Fussball distanziert sich vom designierten
Bundestrainer Daum. Völlner bleibt bis auf weiteres Trainer
der deutschen Fussballnationalmannschaft.

q: Fussball Trainer Drogen
(Welcher Fussballtrainer nimmt Drogen?)