Netzbasierte Informationssysteme
Suchmaschinenverfahren

Prof. Dr.-Ing. Robert Tolksdorf
Freie Universität Berlin
Institut für Informatik
Netzbasierte Informationssysteme
mailto: tolk@inf.fu-berlin.de
http://www.robert-tolksdorf.de

Inhalt

- Pagerank
- HITS
- Metasuchmaschinen

Pagerank

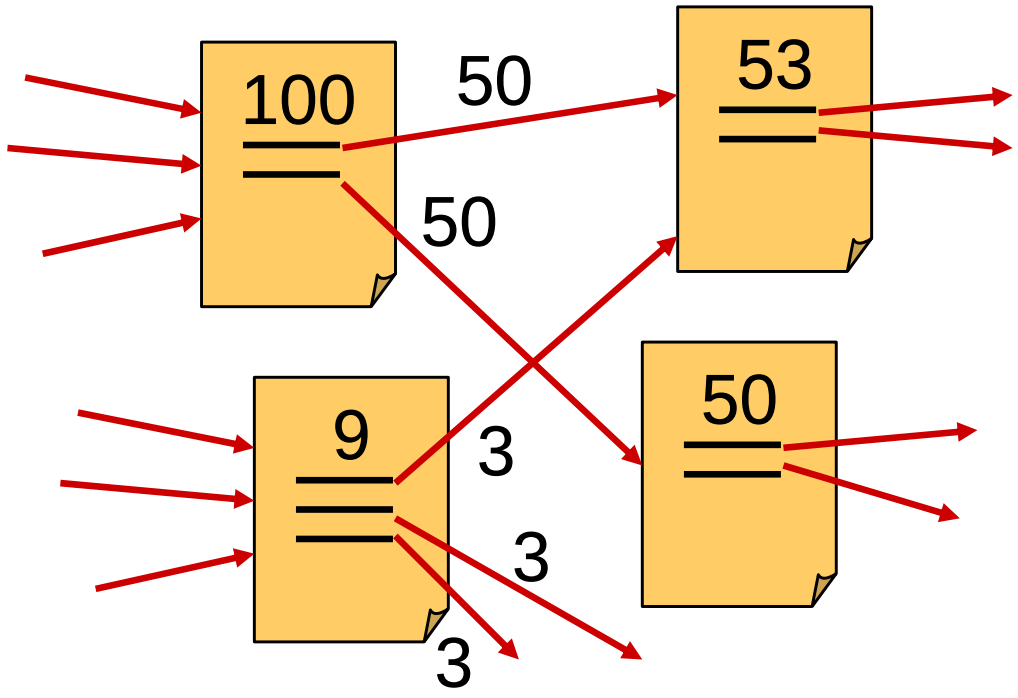
Page, L., Brin, S., Motwani, R., Winograd, T.L.:
The PageRank Citation Ranking: Bringing Order to the Web
http://newdbpubs.stanford.edu:8090/pub/1999-66

PageRank

- PageRank Verfahren: Bewertung aller Web-Seiten nach ihrer relativen Popularität
- Kerntechnologie von Google
- Viele Verweise auf eine Seite legen nahe, dass die Seite wichtig ist
- Setzen eines Links ist Einschätzung der Wichtigkeit der referenzierten Seite (ähnlich einem Zitat)
  - Sie ist aber eigentlich nur populär!
- Links von wichtigen Seiten erhöhen Wichtigkeit
  - Eine Seite hat hohen PageRank, wenn die PageRanks der Seiten, die auf sie verweisen, hoch sind
- Seiten mit hohem Pagerank werden in Ergebnismengen zuerst gelistet

Verteilung der Ranks

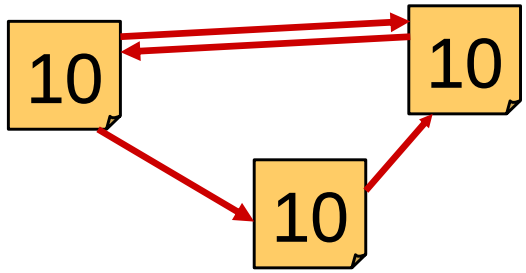
- Jede Seite “verteilt” ihren Pagerank auf die von ihre referenzierten Seiten



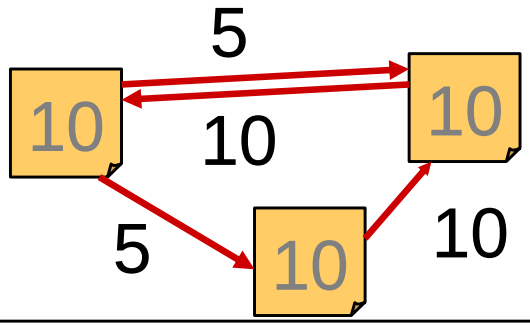
- Wird iterativ errechnet
- Notwendig: Komplette Linkstruktur des Web

Beispieliterationen

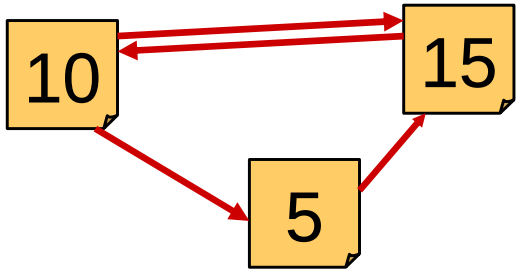
a) Start



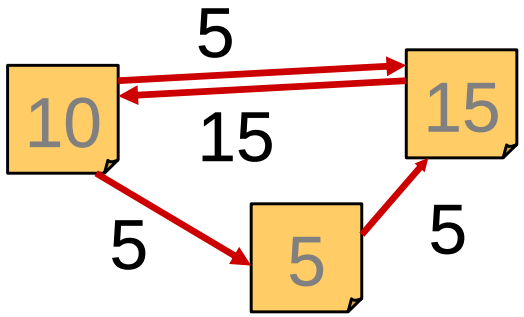
b) Ranks verteilen



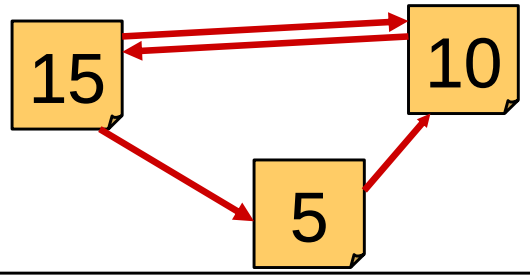
c) Zwischenstand



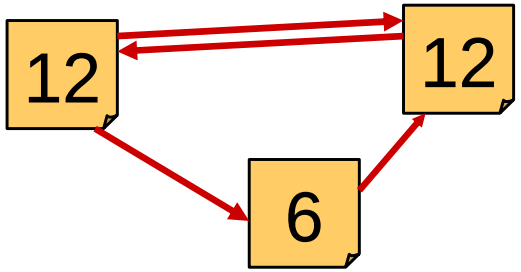
d) Ranks verteilen



e) Zwischenstand



...x) Konvergierter Stand



PageRank

- Wir suchen den Pagerank für eine Web-Seite u
- F_u: Menge der Seiten, auf die u verweist
- B_u: Menge der Seiten, die auf u verweisen
- $N_u = |F_u|$: Anzahl der Links von u (out-degree)
- c: Faktor zur Normalisierung
- Vereinfachter PageRank R(u) der Seiten u:

$$R(u) = c * \sum_{v \in B_u} \frac{R(v)}{N_v}$$

Anteil am Rank einer Seite die auf u verweist

Summe aller Anteile

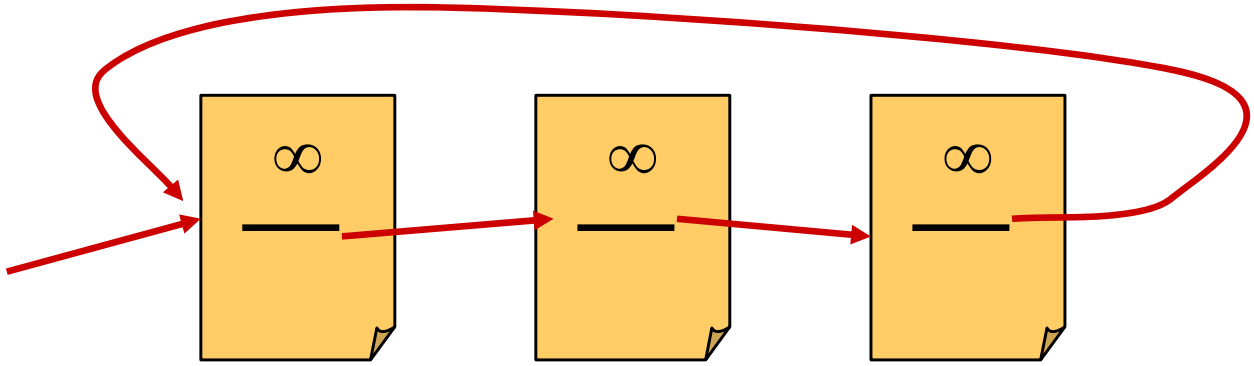
- c normalisiert die Summe aller PageRanks zu 1

Iteration konvergiert

- Sei A eine Matrix der Größe n*n mit Web-Seiten als Reihen und Spalten
- Sei $A_{u,v}=1/N_u$ falls Link von u nach v existiert, 0 sonst
- Sei R ein Vektor über Web-Seiten mit dem PageRank als Wert
- PageRank Formel: R=cAR
- R ist Eigenvektor und c Eigenwert von c
- R kann berechnet werden durch wiederholte Anwendung von A auf einen Startvektor

Vollständiger PageRank

- Problem Schleifen:



- Ist ein „Ranksink“, weil aus der „Schleife“ heraus keine Ranks abgegeben werden
- Lösung: Ausgleichen durch „Ranksource“, die ein Vektor E(u) aus Webseiten mit Rank ist
- Dieser Vektor wird zu den PageRanks hinzugerechnet

$$R'(u) = c * \sum_{v \in B_u} \frac{R'(v)}{N_v} + c * E(u)$$

so dass c maximiert ist und $||R'||_1 = 1$

Vollständiger PageRank

- Intuition: Einfache Version modelliert Zufalls-Surfer und die Wahrscheinlichkeit, dass Seiten von ihm durch zufälliges Verfolgen von Links besucht werden
- In einer Schleife springt der Surfer zu einer beliebigen andern Seite. E modelliert die Verteilung dieser zufälligen Auswahl

- PageRank ist die Verteilung der Wahrscheinlichkeit eine bestimmte Seite durch zufällige Navigation zu erreichen
- Bestimmung aus globaler Sicht

Algorithmus

- Basierend auf Vektorensicht:

$R_0 = S$
loop:
$R_{i+1} = AR_i$
$d = ||R_i||_1 - ||R_{i+1}||_1$
$R_{i+1} = R_{i+1} + dE$
$\delta = ||R_{i+1} - R_i||_1$
while $\delta > \varepsilon$

- Die Vektoren sind aber *sehr* groß (=Anzahl der besuchten Seiten des Web)
- Verschiedene Verfahren zur effizienten Errechnung vorhanden

HITS

Nach: Jon M. Kleinberg: Authoritative Sources in a Hyperlinked Environment. Journal of the ACM 46(5): 604-632 (1999)
http://www.cs.cornell.edu/home/kleinber/auth.pdf

Idee

- Netzstruktur in einem Hypertext ist selber Information über den Inhalt
- Idee:
  Algorithmen entwickeln, um mit Informationen über Graphstruktur besser relevante Informationen zu finden
- HITS - Hyperlink-Induced Topic Search

Arten von Anfragen

- Spezifische Anfragen:
  “Does Netscape support the JDK 1.1 codesigning API?”
  Wenig spezifische Antwortseiten, schwer zu finden

- Breite/Vage Anfragen:
  “Find information about the Java programming language.”
  Viele Antwortseiten, wenige relevant

- Ähnlichkeitsanfragen: “Find pages ‘similar’ to java.sun.com.”

Autoritative Quellen

- Autoritative Antworten als Relevanzfilter
- Wie findet man autoritativ Seiten?
  - www.harvard.edu für "Harvard"?
  - Suche nach "Search engine" für Google?
  - Suche nach "automobile manufacturers" für Honda?

- Fachliche Autorität ist nicht ausschließlich endogene Eigenschaft sonder in großem Maß exogen

- Schreiben Hyperlinks dem Ziel eine fachliche Autorität zu?

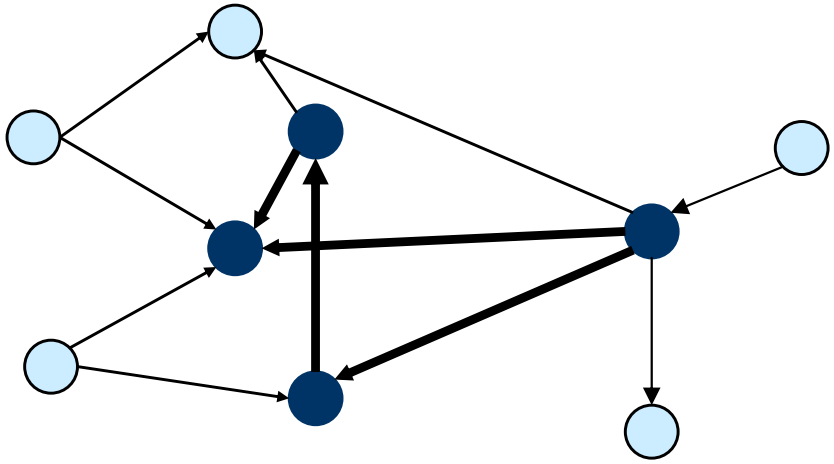
Art der Links

- Nicht alle Links verfolgen gleiche Absicht
  - Hinweis auf wichtige Seiten
  - Navigationsunterstützung
  - Anzeigen

- Popularität vs. Autorität
  - Viele Seiten die einen Begriff enthalten und oft verlinkt werden sind nicht autoritativ
  - Seiten die allgemeinen Inhalt haben und oft verlinkt werden sind nicht für alle Themen autoritativ (eg. www.yahoo.com)

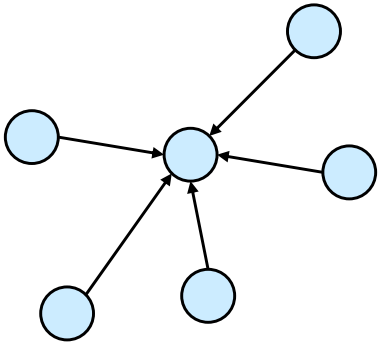
Modell

- Web ist $G=(V,E)$
  $(p,q) \in E$ ist Link von p nach q
- *out-degree* einer Seite: Anzahl abgehender Links
- *in-degree* einer Seite: Anzahl eingehender Links
- Mit $W \subseteq V$ meint $G[W]$ einen Subgraphen bei dem Knoten die Seiten aus W sind und Kanten alle Kanten zwischen Seiten aus W

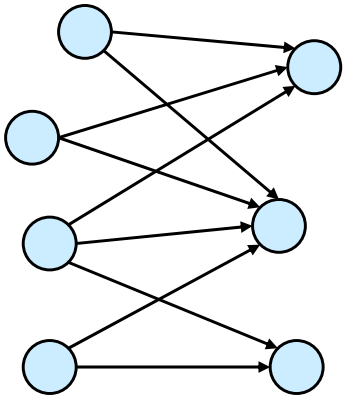


Autoritäten und Hubs

- Autoritative Quellen in G_σ sollten sich dadurch auszeichnen, dass die Mengen der Seiten, die auf sie zeigen überlappen
- *Hubs* verweisen auf mehrere Autoritäten



grosser in-degree



Hubs Autoritäten

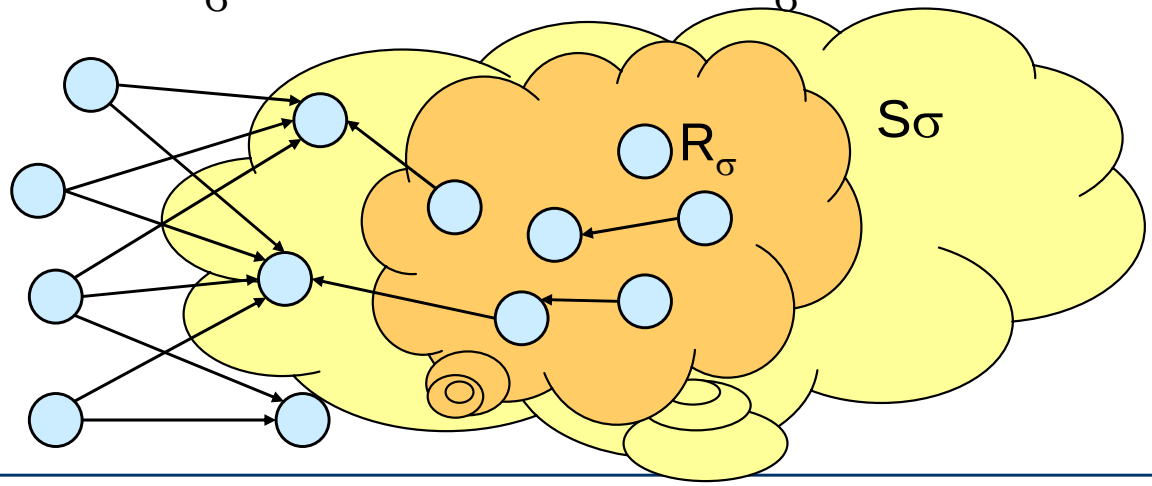
- Guter Hub zeigt auf gute Autoritäten
- Gute Autorität wird von vielen guten Hubs referenziert
- PageRank ermittelt nur populäre Autoritäten

Analyseziel

- Anfrage σ
- Ziel: Analyse der Linkstruktur in einem Teil des Web um autoritative Quellen zu finden
- Qσ (alle Seiten, die σ enthalten)?
  - Zu groß
  - Autoritative Quellen eventuell nicht enthalten
- Gesucht: S_σ so dass:
  - Sσ vergleichsweise klein ist (i)
    -> Aufwand begrenzt
  - Sσ viele relevante Seiten enthält (ii)
    -> gute Autoritäten auffindbar
  - Sσ die meisten oder viele Autoritäten enthält (iii)

Root Set / Base Set

- Root-set R_σ aus ersten t Antworten einer herkömmlichen Suchmaschine auf Anfrage σ (t=200)
  - Erfüllt (i) und (ii): R_σ ist Untermenge von Q_σ
  - Erfüllt nicht (iii), weil Q_σ (iii) schon nicht erfüllt
  - R_σ ist nicht sehr eng verlinkt
- Wenn es aber eine Autorität für die Anfrage gibt, ist es aber wahrscheinlicher, dass auf diese von R_σ aus verwiesen wird
- -> Root-Set R_σ zum Base-Set S_σ erweitern



Errechnung des Base Set

Subgraph(σ,E,t,d)

σ: a query string, E: a text-based search engine,

t, d: natural numbers, Let R_σ denote the top t results of E on σ.

Set $S_\sigma := R_\sigma$

For each page p $\in$ Rσ

Let Γ^+(p) denote the set of all pages p points to.

Let Γ^-(p) denote the set of all pages pointing to p.

Add all pages in Γ^+(p) to S_σ.

If $|\Gamma^-(p)| \leq d$, then

Add all pages in Γ^-(p) to S_σ.

Else

Add an arbitrary set of d pages from Γ^-(p) to S_σ.

End

Return S_σ

Heuristik

- $G[S_\sigma]$ = Subgraph(σ,Altavista,200,50)
  - Erfüllt in der Regel (i), (ii) und (iii)
  - Größe ca. 1000-5000 Seiten
  - Eine Referenz in 200 Seiten "findet" Autorität

- Zwei Arten von Links in $G[S_\sigma]$:
  - Transvers: Zwischen Seiten aus unterschiedlichen IP-Domains
  - Intrinsic: zwischen Seiten aus gleicher Domain
- Heuristik: Intrinsic Links dienen hauptsächlich der Navigation

- Also nur noch G_σ betrachten ($G[S_\sigma]$ ohne intrinsic)

Auffinden von Autoritäten

- Sind Seiten mit höchsten in-degree in G_σ Autoritäten?
- Nein, universell populäre Seiten sind auch in G_σ populär
- Bei Anfrage "Java" auch in G_σ mit hohem in-degree:
  - www.gamelan.com
  - java.sun.com,
  - Reklame für Karibikreisen
  - Home page von Amazon Books

Hubs und Autoritäten identifizieren

- Für jede Seite p:
  - $x^{<p>}$: Authority weight
  - $y^{<p>}$: Hub weight
  - Normalisiert: $\sum_{p \in S\sigma}(x^{<p>})^2 = 1$ und $\sum_{p \in S\sigma}(y^{<p>})^2 = 1$
- Hohes $x^{<p>}$: p ist gute Autorität
- Hohes $y^{<p>}$: p ist guter Hub
- Operatoren P und O verrechnen Gewichte:

$$\mathcal{P} : x^{<p>} \leftarrow \sum_{q:(q,p)\in E} y^{<q>}$$

$$\mathcal{O} : y^{<p>} \leftarrow \sum_{q:(q,p)\in E} x^{<q>}$$

Algorithmus: Gegenseitig verstärken bis Fixpunkt erreicht

Freie Universität Berlin

Iterate(G,k)

G: a collection of n linked pages

k: a natural number

Let z denote the vector (1, 1, 1, . . . , 1) 2 R^n.

Set x_0 := z.

Set y_0 := z.

For i = 1, 2, . . . , k

  Apply P operation to (x_{i-1}, y_{i-1}), obtaining new x-weights x'_i

  Apply O operation to (x_i, y_{i-1}), obtaining new y-weights y'_i

  Normalize x'_i, obtaining x_i.

  Normalize y'_i, obtaining y_i.

End

Return (x_k, y_k).

Angewandt zum Filtern

Filter(G,k,c)

G: a collection of n linked pages

k,c: natural numbers

(x_k, y_k) := Iterate(G, k).

Pages with the c largest coordinates in x_k as authorities.

Pages with the c largest coordinates in y_k as hubs.

- Mit steigendem k konvergieren $\{x_k\}$ und $\{y_k\}$ zu Fixpunkten x^* und y^* (Beweis siehe Papier)
- Konvergenz sehr schnell: Schon k=20 reicht, um gefundene beste Autoritäten und Hubs stabil zu halten

Ergebnisse

Freie Universität Berlin

- Anfrage "Java"

.328	http://www.gamelan.com/	*Gamelan*
.251	http://java.sun.com/	*JavaSoft Home Page*
.190	http://www.digitalfocus.com/digitalfocus/faq/howdoi.html	*The Java Developer: How Do I . . .*
.190	http://lightyear.ncsa.uiuc.edu/~srp/java/javabooks.html	*The Java Book Pages*
.183	http://sunsite.unc.edu/javafaq/javafaq.html	*comp.lang.java FAQ*

- Anfrage "Search Engines"

.346	http://www.yahoo.com/	*Yahoo!*
.291	http://www.excite.com/	*Excite*
.239	http://www.mckinley.com/	*Welcome to Magellan!*
.231	http://www.lycos.com/	*Lycos Home Page*
.231	http://www.altavista.digital.com/	*AltaVista: Main Page*

Metasuchmaschinen

Weiyi Meng and Clement Yu and King-Lup Liu: Building efficient and effective metasearch engines. ACM Computing Surveys, 34(1), pp. 48-89, March 2002.

E. Selberg, and O. Etzioni: The MetaCrawler Architecture for Resource Aggregation on the Web IEEE Expert, January/February, pp. 11-14. 1997 http://www.cs.washington.edu/homes/speed/papers/ieee/ieee-metacrawler.ps

A. Howe, and D. Dreilinger. SavvySearch: A MetaSearch Engine that Learns Which Search Engines to Query. AI Magazine, 18(2), 1997.

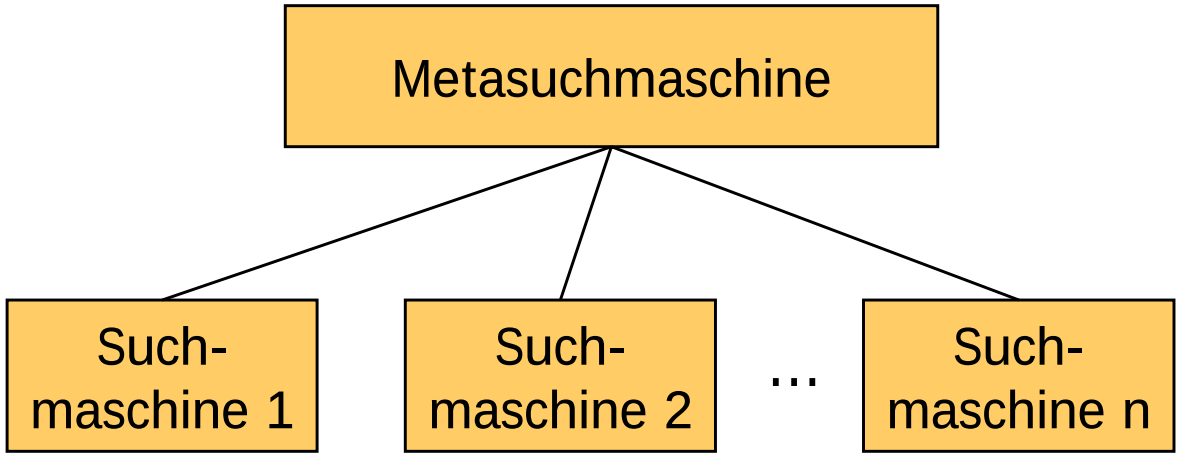
Suchmaschinen

- Effiziente Informationssuche ist problematisch
  - Allgemeine Suchmaschinen
  - Spezielle Suchmaschinen

- Allgemeine Suchmaschinen skalieren schlecht:
  - Wachstum der Verarbeitungskapazität vs. Wachstum der Informationen
  - Wartungsproblematik
  - Zugriffsbeschränkungen

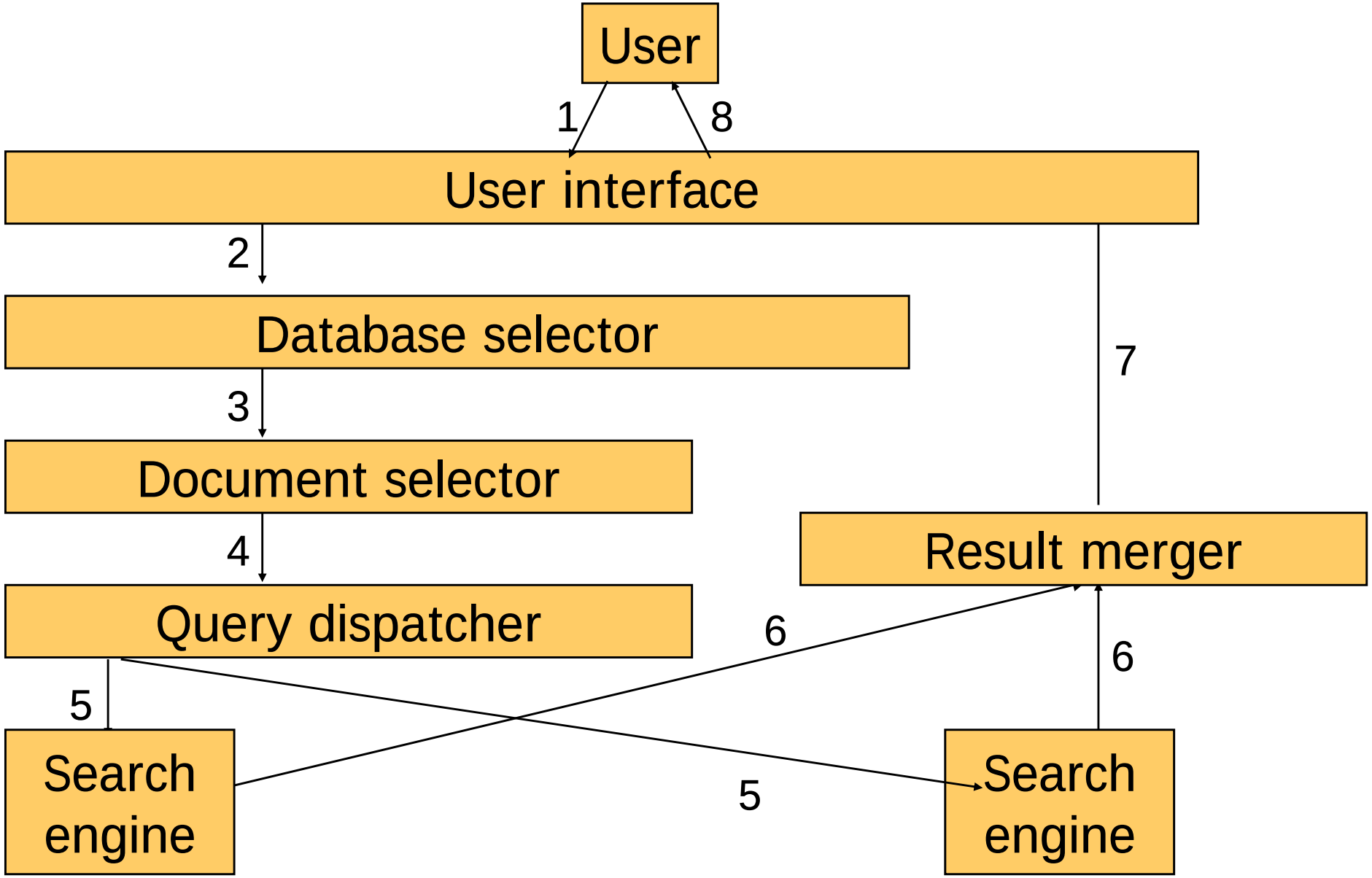
Definition Metasuchmaschine

- Metasuchmaschine ermöglicht einen Zugang zu mehreren Suchmaschinen



Vorteile

- Bessere Abdeckung des Suchraums Web:
  Suchmaschinen decken immer nur einen Teil des Web aus, Metasuchmaschinen deren Summe
- Bessere Skalierbarkeit
  Suche wird auf Suchmaschinen verteilt
- Bessere automatische Absuche mehrerer Suchmaschinen
  Keine manuellen Abfragen notwendig
- Effektivität der Suche verbessern
  Spezialsuchmaschinen gruppieren Dokumente thematisch


User
1
8
User interface
2
Database selector
3
Document selector
4
Query dispatcher
5
Search engine
6
5
Search engine
6
7
Result merger

Database selector

- Kleine Anzahl von Suchmaschinen:
  Anfrage kann an alle weitergeleitet werden
- Hohe Anzahl von Suchmaschinen:
  Großteil der Suchmaschinen ist irrelevant für Anfrage
- Besser Anfragen beschränken
  - Anfragen binden Ressourcen in Metasuchmaschine
  - Anfragen binden Ressourcen des Netzes
  - Anfragen binden Ressourcen bei den Suchmaschinen
  - Antworten binden Ressourcen in Metasuchmaschine zur Auswahl relevanter Dokumente
- Problem:
  Auswahl der relevanten Suchmaschinen für Anfragen (*database selection problem*)

Database selector

- Ist eine Datenbank nützlich für die Anfrage?
- Repräsentative Daten können zur Abschätzung dienen
  - Anlieferung durch kooperative Anbieter
  - Vorangegangene Suchergebnisse
  - Zufallsauswahl

Document selector

- Suchmaschinen unterscheiden sich in
  - der Anzahl zu einer Anfrage passender Dokument im Index
  - der Ermittlung der Ähnlichkeit von Anfrage und Dokument
  - ...
- Document selector soll dafür sorgen, dass
  - möglichst viele potentiell relevante Dokument abgefragt werden
  - möglichst wenige nicht relevante Dokumente abgefragt werden
- Problem:
  Anpassung der Masse in der Anfrage für
  - Anzahl der Dokumente in Antwort
  - Ähnlichkeit der Dokumente mit Anfrage

Query Dispatcher

- Anfragen an eine bestimmte Suchmaschine folgen
  - einer bestimmten Syntax
  - einem bestimmen Protokoll (GET/POST)
- Query Dispatcher muss
  - Anfrage für Suchmaschine syntaktisch anpassen
  - Protokoll der Suchmaschine sprechen
  - Relative Gewichtung von Anfragetermen anpassen
  - Größe der Anfragemenge anpassen

Result merger

- Antworten der Suchmaschinen sind
  - teilweise überlappend
  - jeweils relativ für die Suchmaschine geordnet
  - nach unterschiedlichen Ähnlichkeitsmaßen geordnet
- Result merger muss
  - Dubletten aussortieren
  - Globales Ranking der Ergebnisse nach Ähnlichkeit zur Suchanfrage erstellen
- Problem: Unterschiedliche globale und lokale Ähnlichkeit zur Anfrage
  - Teilweise beeinflusst durch Suchmaschineneigenschaft wie document frequency
  - Neue Precision/Recall Optimierung

Unterschiede in Suchmaschinen

- Indexing
  - Indexierung des Volltext vs. Teile von Texten
  - Stopwords, ja/nein und welche
  - Stemming, ja/nein und wie
  - ...
- Gewichte von Dokumententermen
  - tf oder idf
  - Gewichtung durch Vorkommen im Titel, durch unterschiedliche Auszeichnung
  - ...
- Gewichte von Anfragetermen
  - tf in Anfrage berücksichtigt/nicht berücksichtigt
  - ...

Unterschiede in Suchmaschinen

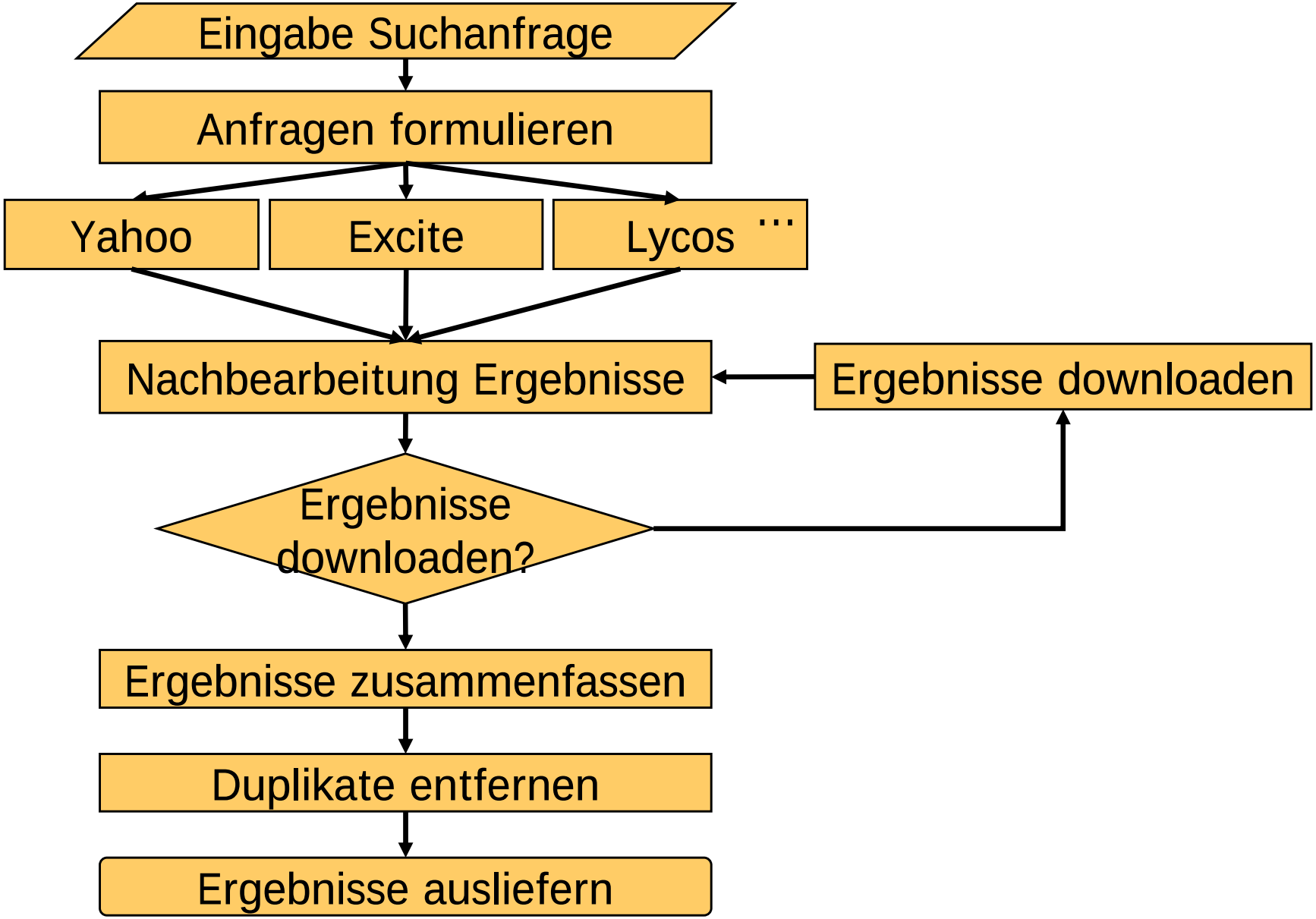
- Ähnlichkeitsmaß
  - Euklidisch, Cosinus etc.
- Dokumentenbestand
  - Themenspezifisch
  - Abdeckungsgrad
  - Version der Dokumente, Wartungszyklus
- Ergebnisdarstellung
  - technisches Format
  - Inhalte der Darstellung (z.B. Angaben über Ähnlichkeitsmaße?)

Beispiel: Metacrawler

Metacrawler

- University of Washington, Seattle 1995
- Beobachtungen
  - Unterschiedliche Ergebnisse auf gleiche Anfragen von unterschiedlichen Suchmaschinen
  - Unterschiedliche Aktualität der Antworten
  - Unterschiedliche Ordnung der Ergebnisse
- Metacrawler fragt mehrere Suchmaschinen an und integriert Ergebnisse

Ablauf


Eingabe Suchanfrage
Anfragen formulieren
Yahoo
Excite
Lycos
...
Nachbearbeitung Ergebnisse
Ergebnisse downloaden
Ergebnisse downloaden?
Ergebnisse zusammenfassen
Duplikate entfernen
Ergebnisse ausliefern

Anfragestellung

- Metacrawler unterstützt
  - Konjunktion, Disjunktion, Negation von Termen
  - Phrasen
  - Einschänkung auf Domäne
- Anpassung an
  - Suchmaschinensyntax ("a AND b", "+a +b",...)
  - Suchmaschineninteraktion (Menüs, Buttons zur Anfrageart)

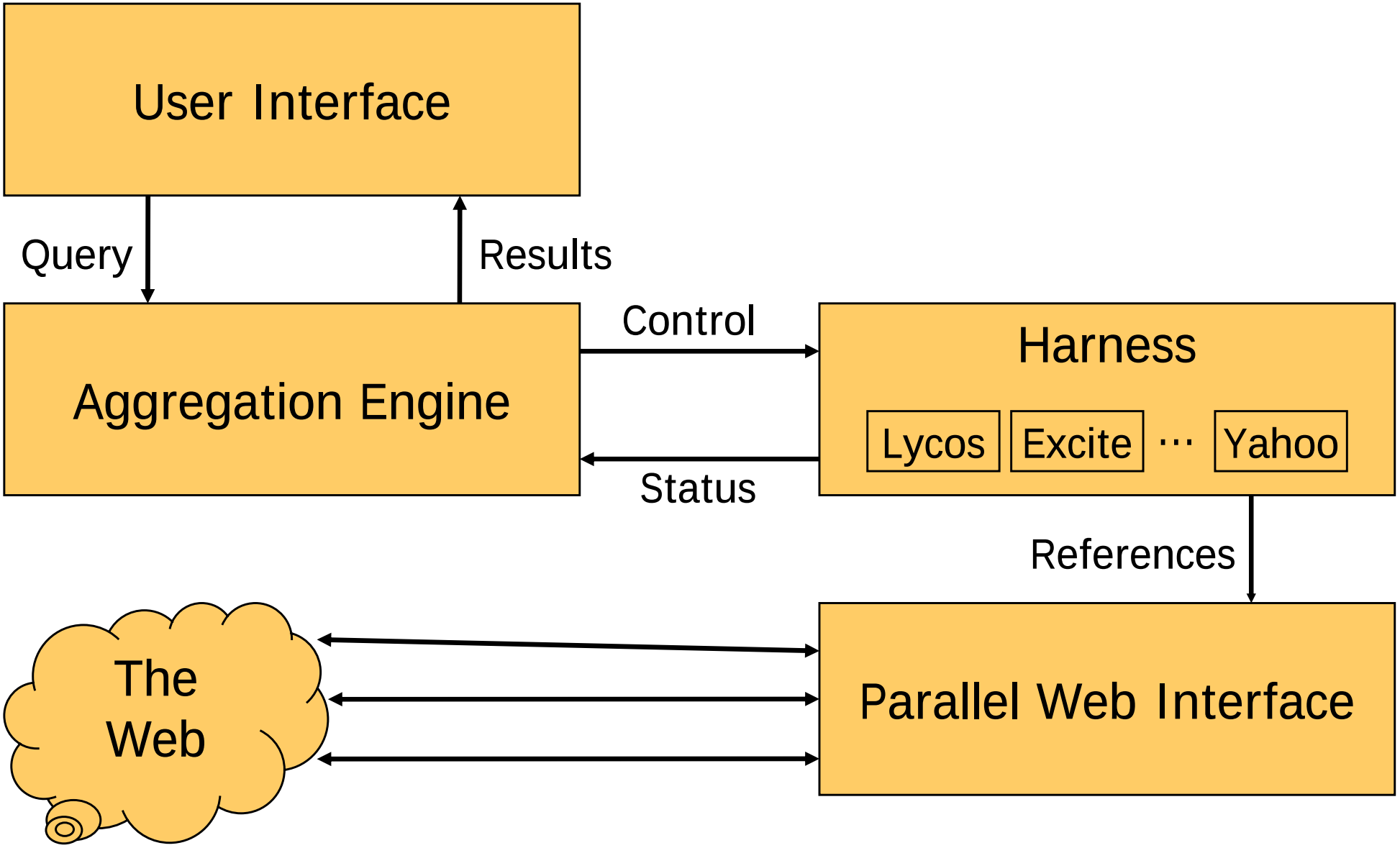
Nachbearbeitung

- Anfragemächtigkeit von MetaCrawler teilweise größer als die der Suchmaschinen
- Direkte Analyse der Ergebnisdokumente notwendig
- Download der Dokumente in Folge
- Nebeneffekt: Test ob Dokument existiert

Aufbereitung

- Doppelte Referenzen entfernen
  - grunge.cs.tu-berlin.de/~tolk (= grunge.cs.tu-berlin.de/) = flp.cs.tu-berlin.de/~tolk = www.cs.tu-berlin.de/~tolk = www.robert-tolksdorf.de = www.inf.fu-berlin.de/~tolk
  - www.robert-tolksdorf.de = www.robert-tolksdorf.de/index.html (oft…)
  - http://www.ag-nbi.de/lehre/0607/V_NBI/ = http://nbi.inf.fu-berlin.de/lehre/0607/V_NBI/ = http://www.inf.fu-berlin.de/inst/ag-nbi/lehre/0607/V_NBI/
  - Metainformationen zur Unterscheidung nutzen
- Ranking:
  - Confidence score
  - Originalranking normalisiert auf 0…1000
  - Addition der Summe der Scores der Duplikate

Architektur



Beispiel: SavvySearch

SavvySearch

- 1995, Colorado State University
- Heute search.com
- Beobachtung
  - Metasuchmaschinen überlassen Nutzer Database Selection
  - MetaCrawler überprüft Ergebnisse
  - Individuell größter Nutzen: Alle Suchmaschinen anfragen
  - Widerspricht global größtem Nutzen
- SavvySearch lernt von vorherigen Anfragen, welche Suchmaschinen anzufragen sind und wie

Ziel

- Ziel
  - Wahrscheinlichkeit relevanter Ergebnisse erhöhen
  - Netz- und Serverbelastung reduzieren
  - Database Selection als Schlüssel dafür
- Server messen
  - Langzeit-Performance zu Anfragen
  - Kurzzeit-Performance zu Erreichbarkeit

- Wie viele Suchmaschinen anfragen?
- In welcher Reihenfolge?

Wie viele Anfragen gleichzeitig?

- Anzahl der nebenläufigen Anfragen auf Basis von
  - Schätzung der Netzlast
    - Bisherige Laufzeiten in Tabelle
  - Messung der CPU-Last bei SavvySearch
    - uptime-Messungen
  - Basis 2 Anfragen, durch gute Maße bis 6

Welche Suchmaschinen / Metaindex

- Beziehungen zwischen Anfragetermen und Suchmaschinen als Langfrist-Messung
  - No Results: Keine Ergebnisse für Term geliefert
  - Keine Ergebnisse: Suchmaschine nicht geeignet
  - Visits: Vom Nutzer danach besuchte Links (Erfordert Ergebnisse mit Link-Forwarding wie bei http://www.search.com/click?sl,marin.1.channel.296-142.1.0.w3c.0,http://click.alltheweb.com/go2/2/atw/1cDC4FC2F7/MSxILHdlYg/http/www.w3.org/)
  - Viele Visits: Suchmaschine sehr geeignet
- Verstärkung relativ zur Anzahl der Terme
  - Keine Antwort von S auf „artificial intelligence conferences“ -> $M_{artificial,s}$=-0.33 $M_{intelligence,s}$=-0.33 $M_{conferences,s}$=-0.33

Metaindex

- Kurzfrist-Messung: Erreichbarkeit von Suchmaschinen
  - Anzahl der Ergebnisse der letzten 5 Anfragen
  - Antwortzeit bei den letzten 5 Anfragen

Ordnung der Suchmaschinen

- Anfragerang $Q_{q,s}$ für Anfrage q bei Suchmaschine s

$$Q_{q,s} = \sum_{t \in q} \frac{M_{t,s} * I_t}{\sqrt{T_s}}$$

  - tfidf Verrechnung
  - $M_{t,s}$: Gelerntes Gewicht für Term t
  - I_t: Inverse Dokumentenfrequenz von t im Metaindex
  - T_s: Summe aller Gewichte für s, generelle Nützlichkeit von s

- Rang der Suchmaschine
  - $R_{q,s} = Q_{q,s} - (P_{s,h} + P_{s,r})$
  - $P_{s,h}$: Strafe für wenig Ergebnisse
  - $P_{s,r}$: Strafe für schlechte Erreichbarkeit

Messungen

- Ergebnisse
  - Ausgangsbasis: Kleiner Metaindex aus Daten von 2 Tagen
  - 28 Tage lernen
  - Visits Maß im Mittel von .36 auf .42 im Mittel gesteigert
  - No Results Maß im Mittel von .142 auf .135 gesenkt
  - Wenig Nutzung notwendig um Maße zu ändern
    - 10-malige Nutzung von Termen halbierte No Results
    - Häufigster genutzter Term (5000) hatte
      - Visits von .76
      - No Results bei 0.08

- Durch einfaches Maß und einfaches Lernen ist deutliche Verbesserung der Performance möglich