

Information Engineering 1 - HS20
Pascal Brunner - brunnpa7

Inhaltsverzeichnis

| | | |
|----------|--|-----------|
| 1 | Einführung | 3 |
| 1.1 | Information Retrieval - Definition | 3 |
| 1.1.1 | Begriffsklärung | 3 |
| 1.1.2 | Was ist das Ziel eines Information-Retrieval Systems sein? | 4 |
| 1.2 | Das Retrievalproblem | 4 |
| 1.2.1 | Konsequenzen des Retrievalproblems | 4 |
| 1.2.2 | IR Prozess | 5 |
| 1.2.3 | Verbalisierung / Codierung | 5 |
| 1.3 | Das Suchparadox | 5 |
| 1.3.1 | Vom Informationsbedürfnis zur Information | 6 |
| 1.4 | Information Retrieval Paradigmen | 6 |
| 1.4.1 | Ad-hoc Suche PULL | 6 |
| 1.4.2 | Bringdienste PUSH | 6 |
| 1.4.3 | Browsing | 7 |
| 1.5 | Datenbank vs. IR | 7 |
| 1.6 | Masse für Retrieval Effektivität | 7 |
| 1.6.1 | Probability Ranking Principle | 8 |
| 2 | Indexierung und Vergleich | 9 |
| 2.1 | Ausgangslage | 9 |
| 2.1.1 | Was ist so schwierig? | 9 |
| 2.2 | Indexierung | 9 |
| 2.2.1 | Worthäufigkeit | 9 |
| 2.2.2 | Anfrage und Dokument vergleichen | 10 |
| 2.2.3 | Verschlagwortung | 10 |
| 2.2.4 | Volltextindexierung | 10 |
| 2.3 | Bag of Words Modell | 11 |
| 2.4 | Rangierregeln | 11 |
| 2.4.1 | Rangierprinzipien | 11 |
| 2.4.2 | Wortstatistiken | 12 |
| 2.4.3 | Vektorraummodell | 12 |
| 2.4.4 | Probability Ranking Principle | 14 |
| 2.5 | Probabilistisches Retrieval | 15 |
| 2.5.1 | Werkzeuge | 15 |
| 2.6 | Boole'sches Retrieval | 15 |
| 2.6.1 | Anhaftende Probleme | 15 |
| 2.6.2 | Paradox | 15 |
| 3 | Systeme und Architektur | 16 |
| 3.1 | Invertierte Liste | 16 |
| 3.1.1 | Inhalt einer invertierten Liste | 16 |
| 3.1.2 | Granularität einer invertierten Liste | 16 |

| | | |
|----------|--|-----------|
| 3.1.3 | Boolesches Retrieval mittels invertierter Liste | 17 |
| 3.1.4 | Lokalisierungsmerkmale in Listen | 17 |
| 3.2 | MiniRetrieve | 18 |
| 4 | Evaluation | 19 |
| 4.1 | Warum sollte man Information Retrieval Systeme evaluieren? | 19 |
| 4.2 | Zutaten einer Evaluation | 19 |
| 4.2.1 | 1. Aufgabenstellung | 19 |
| 4.2.2 | 2. Ausgestaltung der Evaluation | 20 |
| 4.2.3 | 3. Testdaten | 20 |
| 4.3 | Das Cranfield Paradigma | 20 |
| 4.3.1 | Aufgabenstellung | 20 |
| 4.3.2 | Ausgestaltung (Design) | 21 |
| 4.3.3 | Ausbeute / Präzision | 21 |
| 4.3.4 | Average Precision | 21 |
| 5 | Kategorisierung | 23 |
| 5.1 | Dokumentkategorisierung | 23 |
| 5.1.1 | Kategorisierung Aufgaben | 23 |
| 5.2 | Verfahren | 24 |
| 5.2.1 | Rocchio Model | 24 |
| 5.2.2 | kNN | 24 |
| 5.2.3 | Bayes Klassifizierung | 24 |
| 5.2.4 | Collaborative Filtering | 24 |
| 5.2.5 | Thesaurus | 24 |
| 5.2.6 | Herausforderungen bzgl Klassifikationen | 24 |
| 5.3 | Clustering | 24 |
| 6 | Web-Suche | 25 |
| 6.1 | Suche im Web | 25 |
| 6.1.1 | Anatomie von Suchmaschinen | 25 |
| 6.2 | Indexierung Webseiten | 26 |
| 6.2.1 | Anfrageverarbeitung | 26 |
| 6.3 | Suchmaschinen Schlussfolgerung | 26 |
| 6.4 | Suche und Hyperlink | 27 |
| 6.4.1 | Spreading Activation (SA) | 27 |
| 6.4.2 | Indegree | 27 |
| 6.4.3 | PageRank | 27 |
| 6.4.4 | Kleinberg Model | 27 |

Kapitel 1

Einführung

1.1 Information Retrieval - Definition

- Das akademische Fachgebiet, welches Methoden untersucht, um grosse Mengen an unstrukturierter und strukturierter information zu organisieren und bedürfnisgerecht aufzufinden
- Zugriff erfolgt im Allgemeinen in Form einer Anfrage (drückt Informationsbedürfnis mehr oder weniger treffend aus)
- Resultat im Allgemeinen in Form einer Rangliste von Dokumenten, (die gesuchte Information potentiell enthält)

⇒ Information Retrieval wird oft mit Retrieval auf unstrukturiertem Volltext in der Form von natürlichsprachigen Dokumenten gleichgesetzt. Es werden fortgeschrittene Indexierungs- und Gewichtungsmethoden verwendet.

1.1.1 Begriffsklärung

Was ist der Unterschied zwischen Daten, Informationen, Wissen?

Daten

- Daten sind Codes
- Daten sind Symbole ohne keinen Kontext
- Fakten in einer codierten Form
- Daten tragen per se keine Bedeutung
- Daten sind in einem bestimmten Format (bspw. RTF, XML, JPEG)
- Daten können falsch oder korrekt sein (bspw. falsche Datenerfassung)

Informationen

- Daten werden interpretiert → Information
- Daten plus die Bedeutung, die ihnen beigemessen wird
- Information ist relevant oder irrelevant
- Information ist immer an ein Informationsbedürfnis gebunden
- Information benötigt man, um Aufgaben zu erledigen
- wird mit Hilfe von Daten dargestellt
- ist bzgl. einer Aufgabe mehr oder weniger relevant

- ist bzgl. einer Aufgabe mehr oder weniger vollständig

Wissen

- Das Resultat der Verarbeitung von Information (Schlüsse, Erkenntnisse)
- Wissen ist vernetzte Information (bspw. Geschäftsprozess)
- Häufig müssen interne und externe Informationen vernetzt werden
- Beispiele
 - Bestellung eines Kunden abwickeln
 - Flugzeug warten
 - Marketing-Strategie entwickeln

1.1.2 Was ist das Ziel eines Information-Retrieval Systems sein?

- Retrievalproblem: *Das Auffinden von möglichst viel relevanter Information bei gleichzeitigem Minimieren der ebenfalls gelieferten irrelevanten Informationen*
- Wir wollen nicht nur Informationen wieder finden, sondern vor allem neue Informationen finden. Die Information wird indirekt geliefert, in Form von relevanten Dokumenten

1.2 Das Retrievalproblem

- Sprache ist nicht *eindeutig* (Synonym, Homonyme, Metaphern, Schreibfehler etc.)
- Informationsbedürfnisse werden ungenügend verbalisiert
- Informationsbedürfnisse werden ungenügend formuliert
- Die Dokumente und Informationen im System sind unstrukturiert oder inhomogen
- Irreführender Inhalt
- Autorität, Quelle, Aktualität, Urheberrecht, Einsammeln der Dokumente
- Widersprüchliche Ziele → Ausbeute vs. Präzision

Typische Annahmen für Volltextsuche:

- Benutzer sucht relevante Elemente
- Benutzer kennt das gesuchte Elemente nicht
- Anzahl der relevante Elemente ist unbekannt

Der Begriff **Relevanz** ist wichtig für Information Engineering.

Das Verständnis einer Anfrage und eines Dokuments hängt immer auch vom konkreten Benutzer ab.

⇒ Das gelieferte Resultat ist nicht für jeden Benutzer gleich relevant.

1.2.1 Konsequenzen des Retrievalproblems

Der 'unscharfe' Begriff der Relevanz und die vielfältige Darstellungsformen der Information, welche eine Übereinstimmung von Anfragen und Dokument erschweren, führen zu einer wahrscheinlichkeitsbasierten Lösung:

- Es werden diejenigen Dokumente geliefert, für welche die Wahrscheinlichkeit, dass sie vom Benutzer als relevant zur Anfrage beurteilt werden, am höchsten sind
- Ein Retrievalresultat ist fast nie vollständig 'korrekt': Es fehlen relevante Dokumente, oder irrelevante Dokumente werden zusätzlich gefunden
- Merke: 'scharfe Kriterien' (ja / nein) sind ungeeignet, da der Benutzer die Anzahl und die Form der gesuchten Dokumente kennen müsste, um eine Anfrage zu formulieren, welche das gewünschte Resultat liefert → Paradox

1.2.2 IR Prozess

1. Man braucht eine gewisse Information
2. In einem ersten Schritt möchte man eine Verbalisierung vornehmen → Ich brauche einen gute CD-Brenner
3. Das ganze wird auf ein paar Stichworte hinuntergebrochen → CD-Brenner
4. Das System muss nun eine Rangliste zusammenstellen, welche aufgrund der Anfrage geführt werden → Retrieval

In diesem Prozess liegt die eigentliche Power von Information Engineering von Informationsbedürfniss-Formulierung zur Rangliste

5. Man erhält das Resultat in Form einer Rangliste → Die Relevanz folgt aus dem eigentlichen Informationsbedürfnis

Das System muss aus einer schlecht formulierten Informationsbedürfnis, ein möglichst gutes Ergebnis erhalten

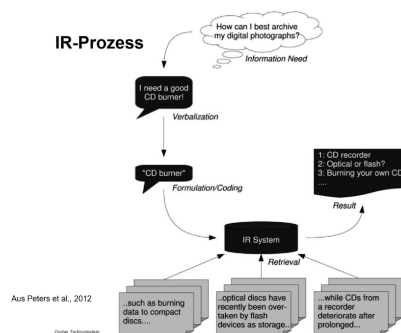


Abbildung 1.1: Ablauf eines Informationretrieval-Prozesses

1.2.3 Verbalisierung / Codierung

- Verbalisierung Verstehen des Problems → richtige Begriff, Vollständigkeit
Paradox: Muss Problem verstanden haben, um es richtig zu verbalisieren
- Codierung Verstehen des Systems → richtige Operatoren etc.
Paradox: muss die Resultate kennen, um Anfragen richtig zu codieren
- Das System kann a priori nur die explizite Information, welche in der 'codierten' Anfrage enthalten ist, auswerten.
- Das Resultat bevorzugt also Entscheide, welche bestmöglich zu dieser 'codierten Anfrage' passen. Nicht explizit formulierte Präferenzen oder Hintergrundwissen des Benutzers können nicht in die Resultatfindung einfließen.
- ⇒ Das Resultat ist immer nur so gut wie die Anfrage

1.3 Das Suchparadox

- Google kann sich Vereinfachungen erlauben dank dem Suchparadox
- Es ist einfacher, in mehreren Milliarden Dokumenten zu suchen als in mehreren Tausend
- Wird auf kleinen Datenmenge gesucht, so ist eine Übereinstimmung zwischen Informationsbedürfnis und Dokumente schwerer nachzuweisen

1.3.1 Vom Informationsbedürfnis zur Information

- Der 'Bibliothekar' hilft dem Benutzer das Informationsbedürfnis zu verbalisieren und in eine Form zu bringen, um nach den gewünschten Information suchen zu können
- Hilft gegebenenfalls auch, das Informationsbedürfnis besser zu verstehen
- \Rightarrow IR-Systeme können ähnliche Funktionen übernehmen
- Informationelle, navigationale, transaktionale, örtlich gebundene Bedürfnisse

1.4 Information Retrieval Paradigmen

- Pull: Aufgrund eines Ad-hoc Informationsbedürfnisses Dokumente mit relevanten Informationen suchen
Google Suche
- Push: Dokumente mit relevanten Informationen aus einem Dokumentenstrom herausfiltern und weiterleiten
Bspw. Mail von Netflix wenn neue Serie auf den Markt kommt wo mir gefallen könnte
- Browse: Neue Dokumente kategorisieren und in eine Informationsstruktur einordnen

1.4.1 Ad-hoc Suche PULL

Beispiele:

- Suche des Bundesgerichtes
- Google Suche

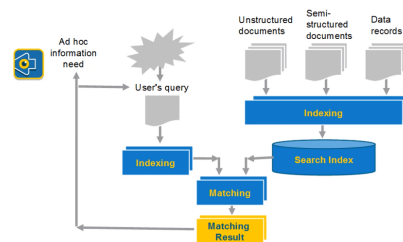


Abbildung 1.2: Ablauf einer Ad-hoc Suche

1.4.2 Bringdienste PUSH

- Das User-Profil dient für die Langlebigkeit
Instagram sammeln der Likes, Bewertung von Filmen auf Netflix

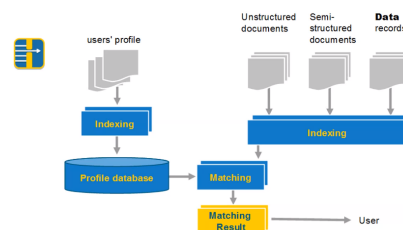


Abbildung 1.3: Ablauf eines Bringdienstes

1.4.3 Browsing

Beispiel

- Ehemals Yahoo → War das Telefonbuch des Internets mit den diversen Links

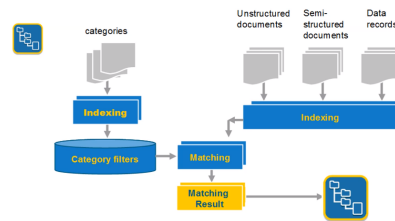


Abbildung 1.4: Ablauf des Browsers

1.5 Datenbank vs. IR

Datenbank = strukturiert, exact-match (passt oder nicht) IR = unstrukturiert, best-match (rangliste)

- DB liefern für strukturierte Daten mit kontrolliertem Vokabular perfekte Resultate
- Daten in DB sollten unabhängig sein von der Applikation. Redundanz wird vermieden
- Elemente sind entweder Teil der Resultatmenge oder nicht
- Boole'sche Kriterien zur Selektion
- Geeignet für die Suche in (hoch-)strukturierter Information mit kontrolliertem Vokabular

| | Data Retrieval | Information Retrieval |
|---------------------|----------------|---------------------------|
| Matching | Exact match | Partial match, best match |
| Inference | Deduction | Induction |
| Model | Deterministic | Probabilistic |
| Classification | Monothetic | Polythetic |
| Query language | Artificial | Natural |
| Query specification | Complete | Incomplete |
| Items wanted | Matching | Relevant |
| Error response | Sensitive | Insensitive |

Abbildung 1.5: Unterschied von IR vs DB

1.6 Masse für Retrieval Effektivität

Jedes IR-System liefert unterschiedliche Resultate

- Retrievalresultate sind fast nie perfekt. Es sind Masse nötig, um die Qualität eines Retrieval Resultats zu beurteilen
- Zwei allgemein anerkannte Masse für Retrieval Qualität sind **Ausbeute** und **Präzision**
- Diese Eigenschaften dieser Masse sind gut analysiert und verstanden
- Beide Masse modellieren die Annahme, dass möglichst viel relevante und möglichst wenig irrelevante Information gefunden werden

$$\text{Präzision} := \frac{\text{Anzahl relevante Dokumente im Resultat}}{\text{Anzahl Dokumente im Resultat}} \quad (1.1)$$

$$\text{Ausbeute} := \frac{\text{Anzahl Dokumente im Resultat}}{\text{Anzahl relevante Dokumente in der Kollektion}} \quad (1.2)$$

→ Ziel: Optimierung eines oder beider Kriterien

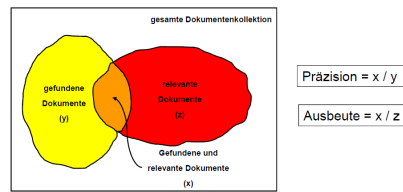


Abbildung 1.6: Visualisierung der Messgrößen

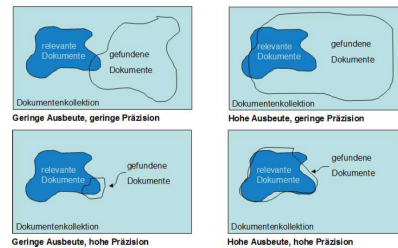


Abbildung 1.7: Szenarien der Messgrößen

1.6.1 Probability Ranking Principle

Eine Anordnung der gefundenen Dokumente in der Reihenfolge ihrer Wahrscheinlichkeit, dass sie zur Anfrage relevant sind unter Berücksichtigung sämtlicher zur Verfügung stehender Informationen und geeigneter Annahmen, ist optimal in mehrerer Hinsicht.

Falls das sogenannte Probability Ranking Principle befolgt wird, lassen sich die folgende Eigenschaften mathematisch beweisen

- Die Präzision an einem beliebigen 'cut-off-point' wird optimiert
- Die Ausbeute an einem beliebigen 'cut-off-point' wird optimiert
- Die Kosten der Auswertung (relevant = positiv, irrelevant = negativ) werden optimiert

⇒ Die Verwendung von wahrscheinlichkeitsbasierten Ranglisten ist theoretisch fundiert

Kapitel 2

Indexierung und Vergleich

2.1 Ausgangslage

- 1 vages Informationsbedürfnis
- 1'000'000 unstrukturierte Volltexte
- Gesucht → Resultat
- Vorgehen
 1. Anfrage und Dokumente vergleichbar machen
 2. Vergleich
- Datenbank vs. IR
 - Datenbank: in Tabelle Apfel sind nur Äpfel
 - IR: Es werden Apfel mit Birnen verglichen

2.1.1 Was ist so schwierig?

- Sprache ist nicht eindeutig
 - Synonyme, Homonyme, Umschreibungen, Metaphern etc.
- Bedeutung und Relevanz ändern mit der Zeit
 - nine eleven*: Notfallnummer oder Inbegriff einer Zeitwende? oder doch ein Porsche?
- Menschen lernen dazu oder vergessen
- Sprache Akronyme, Phrasen, Komposita, verschiedene Schreibweisen
- Unschärfe macht uns das Leben schwer (im Retrieval, automatische Sprachverbreitung)

2.2 Indexierung

2.2.1 Worthäufigkeit

- Worthäufigkeiten sind nicht gleich verteilt
- sehr viele Wörter sind sehr selten

Zipfsches Gesetz: Rang der Häufigkeit * Häufigkeit = konstant

2.2.2 Anfrage und Dokument vergleichen

Anfrage und Dokument können nicht direkt verglichen werden. Anfragen und Dokument müssen zuerst vergleichbar gemacht werden

2.2.3 Verschlagwortung

- manuell, automatisch oder beides passieren
- Deskriptoren stammen aus einem Vokabular (freie Indexierung) oder aus einer autorisierten Liste (kontrollierte Indexierung)
- Zu beachten gilt
 - Dokumente vokabular \neq Anfrage vokabular
 - Manuelle Indexierung sehr aufwendig und kostspielig

Automatische Verschlagwortung

Ist nicht so einfach wie man einfach glauben kann

2.2.4 Volltextindexierung

Sonderfall von automatischer Verschlagwortung mit freiem Vokabular **erschliessen**:

1. Buchstabenumwandlung
2. Wortextraktion
3. Stoppwortelimination

Tokenisierung

- Tokenisierung (Wortextraktion) ist der Prozess der die einzelnen Wörter aus dem Text extrahiert
- Im Allgemeine werden einige oder alle der folgen Schritte ausgeführt
 - Dokumentformate konvertieren
 - Zeichencodierung anpassen
 - Gross / Kleinschreibung normalisieren
 - Text entlang Trennzeichen in Tokens separieren (eigentliche Tokenisierung)
- Letzte Punkt ist heikel (bspw. Hans Peter ein oder zwei Wörter)

Stoppwortelimination

- Häufigsten Wörter werden aus dem Text eliminiert
- Diese Wörter sind nicht inhaltstragend
- Treffer auf diesem Wörter sind (meist!) nicht hilfreich, und verdecken echte, wertvolle Treffer
- Stoppwörter machen etwa 40 % des Textes aus

Erschliessen

- Wortzerlegung
- Wortnormalisierung

Stemming / Wortnormalisierung

- In Sprachen werden mehr oder weniger Wortformen verwendet, um Dinge wie Kasus, Numerus, Genus etc. anzuzeigen
- Dieses Phänomen ist je nach Sprache unterschiedlich stark ausgeprägt
 - Im Englischen gibt es relativ wenig verschiedene Formen für die meisten Wörter
 - Im Deutschen sieht das schon anders aus (bis zu 144 Formen für ein Verb)
- Information Retrieval ist kein linguistischer Schönheitswettbewerb
- Wir wollen Dokumente auffinden unabhängig von einzelnen Wortformen
- Aber: Die Dokumente sollten relevant sein, d.h. der gesuchte Sachverhalt sollte abgebildet werden
- Wenn möglich werden im IR einfache, regelbasierte Verfahren verwendet, um die Wörter zu normalisieren
- Dieser *Stemmer* produzieren teilweise unsinnige (abgeschnittene) Wortformen, oder produzieren falsche Treffer → Frage der Gewichtung

Komposita

- Im Deutschen können unendlich viele Komposita gebaut werden, indem man Wörter zusammengefügt
- Diese Komposita können nicht lexikalisch aufgezählt werden
- eigentlich sehr gute Suchbegriffe, aber: der gleiche Sachverhalt lässt sich immer auch als Nominalphrase umschreiben
- Es ist hilfreich solche Begriffe zu zerlegen (bis zu 30% Effektivität-Gewinnung)
 - jedoch sollte man bspw. *Frühstück* nicht trennen

2.3 Bag of Words Modell

2.4 Rangierregeln

2.4.1 Rangierprinzipien

- Versuch "common sense" Regeln aufzustellen
- Es gibt KEINE konkreten Systeme, die diese Rangierprinzipien so implementieren, aber:
- Hilft bei späterer konkreter Umsetzung
- Hilft dem Benutzer als mentales Modell

Prinzip 1

Je mehr Suchbegriffe (Terme) in einem Dokument vorkommen desto wahrscheinlicher ist das Dokument relevant

Prinzip 2

Je häufiger ein Suchbegriff in einem Dokument vorkommt, desto wahrscheinlicher ist das Dokument relevant → Merkmalshäufigkeit (Anzahl Vorkommen eines Merkmals in einem Dokument)

Prinzip 3

Dokumente, die seltene Suchbegriffe enthalten, sind mit höheren Wsk relevant als Dokumente, die häufige Suchbegriffe enthalten → Dokumentenhäufigkeit (Anzahl Dokumente, die einen Begriff enthalten)

Prinzip 4

Je mehr Hyperlinks auf ein Dokument zeigen, desto wahrscheinlicher ist es relevant

Prinzip 5

Je näher die Suchbegriffe beieinander liegen desto relevanter das Dokument

Prinzip 6

Je früher die Suchbegriffe in einem Dokument vorkommen, desto höher seine Relevanz

2.4.2 Wortstatistiken

Zur Umsetzung der Rangierregeln sind eine Reihe von Statistiken notwendig:

- tf - term frequency → Wie oft tritt ein Merkmal/Term in einem Dokument auf?
- df - document frequency → in wievielen Dokumenten tritt ein Merkmal/Term auf
- Dokumentlänge → ein Mass für die Länge des Dokuments bspw. Anzahl Tokens, Anzahl Merkmale, Bytelänge
- Position der Merkmale im Text, In-Links / Out-Links

Inverse document frequency

- $idf(\phi_k) = \log\left(\frac{1+N}{1+df(\phi_k)}\right)$
N: Anzahl Dokumente in der Kollektion
 $df(\phi_k)$: Anzahl Dokumente die Term k erhalten
- Mit $idf(\phi_k)$ kann die Ausprägung (Wichtigkeit; wie 'charakteristisch') eines Terms für ein bestimmtes Dokument besser bestimmt werden als durch simples Zählen der Häufigkeit
- $w(\phi_k, d_i) = tf(\phi_k, d_i) * idf(\phi_k)$
 $tf(\phi_k, d_i)$: Anzahl des Vorkommen des Term k in Dokument i
 $idf(\phi_k)$: inverse Dokumentenhäufigkeit des Terms k
- ⇒ Ein hohes Gewicht haben dabei Terme, die in wenigen Dokumenten innerhalb der Kollektion häufig auftreten

2.4.3 Vektorraummodell

- Die Gewichtung einzelner Terme reicht nicht. Wir müssen die Anfragen als Ganzes mit dem Dokument als Ganzes vergleichen
- Fasse informationsobjekte d_j und Anfrage q als Vektoren im Merkmalsraum auf, jedes Merkmal → eine Dimension
- Vektor $d_j = (\dots, w_{\phi_i}, \dots)$ (Gewicht w_{ϕ_i}) bspw. 1 wenn das entsprechende Merkmal im Dokument ist, 0 sonst
→ Anfrage als Vektor, wobei $\text{Sim}(\text{Dokument}, \text{Anfrage}) = \text{Kosinus}(\text{Winkel})$

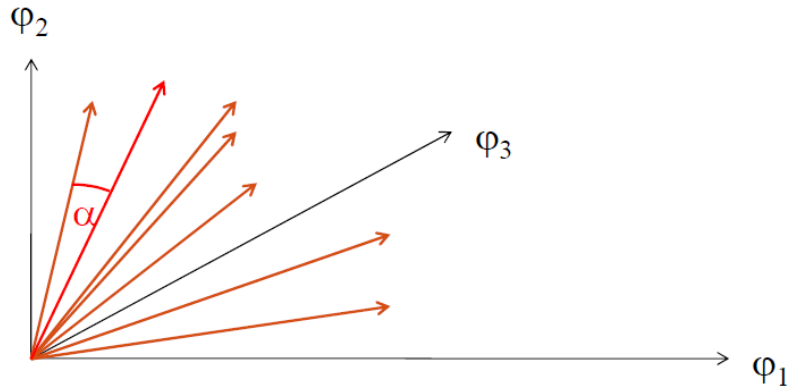


Abbildung 2.1: Dokumente als Vektoren im n-dimensionalen Raum

Problem: Merkmale spannen einen orthogonalen Raum auf. Merkmalabhängigkeiten passen nicht in das Modell. Das Modell liefert keine Antwort wie die einzelnen Merkmale zu gewichten sind

Bemerkung: Zwei Vektoren heißen zueinander orthogonal, wenn ihr Skalarprodukt null ist.

Längennormalisierung

- wenn Dokumente und Anfrage als Vektoren in einem n-dimensionalen Vektorraum dargestellt werden, hat sich folgende Gleichung als erfolgreich bewiesen

- $RSV = \frac{(q,d)}{\|q\|\|d\|} = \cos(\alpha) \rightarrow$ wobei RSV für *retrieval status value* steht

Winkel α ist der Winkel zwischen zwei Vektoren q und d

(q,d) ist das skalare Produkte

$\| \|$ kennzeichnet die Länge des Vektors

Gewichtungsformel

Es ergibt sich bei Kombination mit tf.idf-Gewichtung die folgende Gewichtungsformel (*tf-idf-cosinus*):

$$\begin{aligned}
 a_{i,j} &:= f f(\phi_i, d_j) * idf(\phi_i) \\
 b_i &:= f f(\phi_i, q) * idf(\phi_i) \\
 RSV(q, d_j) &:= \frac{\sum_{\phi_i \in \Phi(q) \cap \Phi(d_j)} a_{i,j} * b_i}{\sqrt{\sum_{\phi_i \in \Phi(d_j)} a_{i,j}^2} * \sqrt{\sum_{\phi_i \in \Phi(q)} b_i^2}}
 \end{aligned} \tag{2.1}$$

Relevanzrückkoppelung (englisch: relevance feedback)

Idee: der/die BenutzerIn identifiziert im Resultat relevante und irrelevante Dokumente

Dabei wird ein neuer Vektor konstruiert, der näher bei den relevanten Dokumenten und weiter weg von den irrelevanten Dokumenten liegt

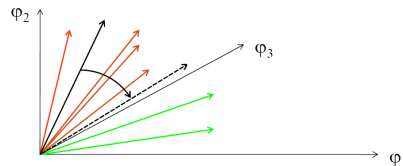


Abbildung 2.2: neuer Vektor im relevance Feedback Prozess

Dies kann man mit verschiedenen Spielarten realisieren:

- Originalanfrage wird neu gewichtet

- Originalanfrage wird um neue Begriffe aus den relevanten Dokumenten ergänzt
- Kombination aus 1 und 2
- Eine komplett neue Anfrage wird nur aus den relevanten Dokumenten konstruiert

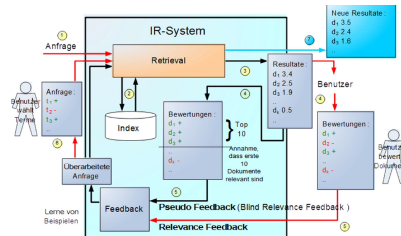


Abbildung 2.3: Schema von relevance feedback Prozess

2.4.4 Probability Ranking Principle

If a reference retrieval system's response to each request is a ranking of the documents in the collections in order of decreasing probability of usefulness to the user who submitted the request, where the probabilities are estimated as accurately as possible on the basis of whatever data has been made available to the system for this purpose, then the overall effectiveness of the system to its users will be the best that is obtainable on the basis of that data. (S.E. Robertson, 1977)

⇒ eigentlich geht es darum, dass man Systeme bauen, welche eine Rangliste zurückliefern in absteigender Wahrscheinlichkeit / Relevanz, dann ist das eine Optimale Strategie

- PRP ist eher eine Hypothese als ein Prinzip
- *will be the best*: Es gibt Rahmenbedingungen und Annahmen, z.b. Kosten für gefundene irrelevante Dokumente und Kosten für nicht gefundene relevante Dokumente, dann kann man beweisen. PRP minimiert die Erwartungskosten
- *Estimated as accurately as possible*: hier liegt das Problem → wie schätzen wir diese Wahrscheinlichkeit ab?

Beispiel:

- $q = \text{"Terrorismus, bekämpfen"} = \phi_1 \phi_2$
- $D1 = \text{"Gegenmassnahmen gegen Terrorismus"} = \phi_3 \phi_1$
- $D2 = \text{"Kampf gegen den Terror"} = \phi_2 \phi_1$
- $D3 = \text{"Sicherheit bei asymmetrischer Bedrohung und asymmetrische Sicherheit"} = \phi_5 \phi_6 \phi_7 \phi_6 \phi_5$
- $D4 = \text{"Terror bekämpfen"} = \phi_1 \phi_2$
- $D5 = \text{"Extremismus und Gewalt"} = \phi_8 \phi_9$
- $D6 = \text{"Terrorismus und innere Sicherheit"} = \phi_1 \phi_{10} \phi_5$
- Relevant erwiesen sich: $R(q) = \{D1, D2, D3, D6\}$

Abbildung 2.4: Beispiel eines PRP

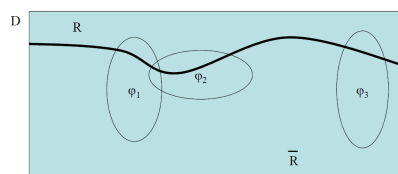


Abbildung 2.5: Beispiel visualisiert eines PRP

2.5 Probabilistisches Retrieval

- Probabilistisches Retrieval: Eine Gruppe von verwandten Modellen, die die wahrscheinlichkeitstheoretische Auffassung des Retrievalsproblems zugrunde legen
- Anfänge frühe 60er Jahre
- Die Grenzen des Probabilistisches Retrievals scheinen erreicht zu sein

2.5.1 Werkzeuge

- A, B stochastisch unabhängig, dann $P(A, B) = P(A) * P(B)$
- Bedingte Wahrscheinlichkeit: $P(A|B) = \frac{P(A, B)}{P(B)}$
- Bayes: $P(A|B) = \frac{P(A)P(B|A)}{P(B)}$
- DAS PRP fordert $RSV(q, d) = f(P(R|q, d)) + g(q)$ d.h. eine ordnungserhaltene Funktion auf der Wahrscheinlichkeit, dass ein Dokument d zur Anfrage q als relevant betrachtet wird
⇒ absolute Scores sind als für die Rangierung nicht wichtig

2.6 Boole'sches Retrieval

Dabei handelt es sich um eines der ältesten Methoden (bspw. auch in Datenbanken). Dabei wird die Frage als Boolesche Ausdruck bzw. durch boolescher Logik die Abfrage evaluiert

2.6.1 Anhaftende Probleme

- Das Resultat kann riesig sein, vor allem bei kurzen Anfragen (bspw. Bier) → Resultatmenge sind nicht nach Relevanz sortiert
- Kleine Anpassungen bei einer Anfrage kann zu sehr unterschiedlichen Resultatmengen führen → häufig zu gross oder zu klein

2.6.2 Paradox

Weil der Benutzer wenig Einfluss auf die Grösse der Resultatmenge hat (diese aber im Gegensatz zu wahrscheinlichkeitsbasierten Ranglisten wichtig ist), findet der Benutzer immer zuviel oder zuwenig → müsste das Resultat kennen um eine geeignete Anfrage zu stellen

Kapitel 3

Systeme und Architektur

3.1 Invertierte Liste

Eine invertierte Liste besteht aus einer geordneten Liste von Merkmalen mit mindestens Information über die Häufigkeit jedes Merkmals in der Dokumentenkollektion

| Merkmal # | Merkmal | Merkmal und Lokalisierungsinformation |
|-----------|----------|---------------------------------------|
| 1 | cold | 2: (1, 6), (4, 8) |
| 2 | days | 2: (3, 2), (6, 2) |
| 3 | hot | 2: (1, 3), (4, 4) |
| 4 | in | 2: (2, 3), (5, 4) |
| 5 | it | 2: (4, 3, 7), (5, 3) |
| 6 | like | 2: (4, 2, 6), (5, 2) |
| 7 | nine | 2: (3, 1), (6, 1) |
| 8 | old | 2: (3, 3), (6, 3) |
| 9 | pease | 2: (1, 1, 4), (2, 1) |
| 10 | porridge | 2: (1, 2, 5), (2, 2) |
| 11 | pot | 2: (2, 5), (6, 6) |
| 12 | some | 2: (4, 1, 5), (5, 1) |
| 13 | the | 2: (2, 4), (6, 5) |

Quelle: Fachinformatiker

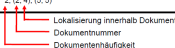


Abbildung 3.1: Abbildung einer beispielhaften invertierten Liste

3.1.1 Inhalt einer invertierten Liste

- invertierte Liste zeigt
 - Dokumenthäufigkeit (bspw. in wie vielen Dokumenten ein Merkmal auftritt)
 - in welchem Dokument das Merkmal auftritt
 - an welcher Stelle im Dokument das Merkmal auftritt (optional)
- invertierte Liste kann mehr oder weniger Informationen enthalten
 - Gewichtung der Merkmale
 - Kategorisierung der Merkmale

3.1.2 Granularität einer invertierten Liste

→ Ist die Genauigkeit, zu welcher eine invertierte Liste die Lokalisierung der Merkmale festlegt

- Grober Index
 - bspw. nur Blöcke von Texten in welchen mehrere Dokumente gespeichert sein können
- Mittlerer Index
 - bspw. die Lokalisierung sind als Dokumentennummern gespeichert

- Feiner Index

Index liefert ein Satz, Wort oder sogar Byte zurück

→ wir brauchen in etwa eine Grössenordnung des Index in Grösse des eigentlichen Dokumentes.

3.1.3 Boolesches Retrieval mittels invertierter Liste

Resultat wird aus der invertierten Liste generiert → durch and, or, oder not (wobei not, selten unterstützt ist)

3.1.4 Lokalisierungsmerkmale in Listen

Eine invertierte Liste ist eine extrem grosse Liste mit allen Merkmalen aus allen Dokumenten als Einträgen. Wenn Anfragen verarbeitet werden, müssen diese Merkmale aufgefunden werden:

- Speichere die Merkmale in einer alphabetischen Liste → Durchlaufe die geordnete Liste $\Rightarrow O(\log(n))$
- Speichere die Merkmale in einer Hash-Tabelle → Lokalisierung der Merkmale in einer Hash-Tabelle ist sehr schnell $\Rightarrow O(1)$

Hash-Datei

- Adressierung im Hash
- Die Hash-Funktion h erstellt Adressen, die zu den Merkmalen zeigen
- Idealerweise sollte die Hash-Funktion h die Adressen gleichmässig über den verfügbaren Speicherraum verteilen
- Idealerweise sollten zwei merkmale t_i und t_j die unterschiedlich sind ($t_i \neq t_j$) keine identischen Adressen produzieren, d.h. es sollte $h(t_i) \neq h(t_j)$ sein
- Kollisionen sind trotzdem unvermeidbar und müssen entsprechend behandelt werden
- Speicherplatzbedarf: Umso kleiner der Speicherplatz wird, desto wahrscheinlicher werden Kollisionen. Die Kollisionen müssen behandelt werden

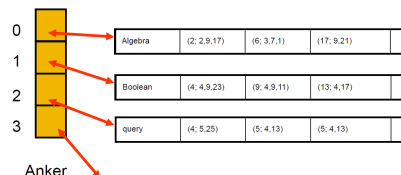


Abbildung 3.2: Abbildung einer Hash-Datei

Lösen von Kollisionen

- Lineare Methode: vermeiden den Adressraum der bereits besetzt ist
Suche den nächst freien Adressraum, wo die Kollision auftrat
Suchstrategie: Gehe zum nächsten freien Adressraum → verschiebe um eine fixe Anzahl von Adressen
Ziel besteht immer darin die Merkmale für welche die Kollision auftraten, so nahe als möglich bei den ursprünglichen Hash-Adressen zu speichern
- Verkettete Kollisionsdateien: Verknüpfe Dateien mit Zeigern
Zeiger auf einen anderen freien Adressraum im Speicherplatz
Stelle zusätzlichen Speicherplatz bereit für Kollisionsdateien

3.2 MiniRetrieve

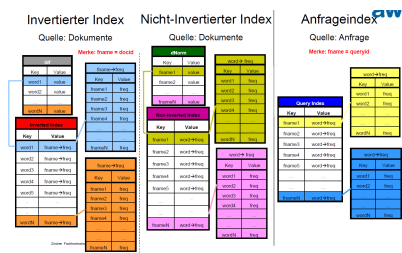


Abbildung 3.3: Abbildung der Architektur des Mini Retrieves

Kapitel 4

Evaluation

4.1 Warum sollte man Information Retrieval Systeme evaluieren?

IR Systeme unterscheiden sich grundsätzlich in ihren Grunddimensionen

- Was sind die Dokumente/Informationen, welche erschlossen werden?
- Was sind die Paradigmen des Zugriffs (Pull, Push, Katalog)?
- Was sind die Benutzerkategorien, welche angesprochen werden?
- Was sind die Bedürfniskategorien, welche befriedigt werden müssen?
- Was ist das Geschäftsmodell?

→ Daher muss evaluiert werden, ob die richtige Suchtechnologie am richtigen Ort eingesetzt wird

- Evaluation ist notwendig, um die Leistung des Systems zu bewerten
- Da Information kein 'greifbares Gut' ist, ist eine Kosten/Nutzen-Analyse nur schwer zu erstellen
- Es ist schwierig, den Anteil der einzelnen Komponenten (Faktoren) an einem positiven oder negativen Suchresultat zu ermitteln (Datenabdeckung, Dateneinteilung, Anfrageformulierung, ...)

Wie gut ist das System?

Wie gut könnte ein generisches System gleicher Bauart sein?

Wie gut wäre ein optimales System?

→ wird das richtige System eingesetzt?

Achtung: Es ist schwierig, aus vergangener Leistung auf zukünftige Leistung zu schliessen

4.2 Zutaten einer Evaluation

4.2.1 1. Aufgabenstellung

Eine seriöse Evaluation setzt eine klare Aufgabenstellung voraus:

- Was sind die Motivation, Ziele und Vorgehensweise der Evaluation?
- welche sind die an den Resultaten interessierten Parteien?
- Wird eine 'interne' oder 'externe' Evaluation angestrebt?
- Ist das Vorgehen 'investigativ' oder wird 'experimentell' gearbeitet?

- Wird das System als 'blackbox' oder als 'glass box' behandelt?
- Was ist der Richtwert (Benchmark)? und andere Massstäbe
- Wird erschöpfend oder indikativ analysiert?
- Werden qualitative oder quantitative Kriterien untersucht?

4.2.2 2. Ausgestaltung der Evaluation

Es folgt das 'Design' der Evaluation, welche der Aufgabenstellung gerecht werden muss:

- Was sind Leistungsfaktoren?
- Was sind die Leistungsmassstäbe?
- Was sind die Leistungsmasse?
- Was sind die Daten?
- Wie ist das Prozedere zu gestalten?

4.2.3 3. Testdaten

Für die Evaluation eines IR-Systems werden in der Regel Testdaten gebraucht.

- Daten über die Benutzer
- Daten über Informationsbedürfnisse
- Dokumentdaten
- Daten über die Relevanz der Dokumente in Hinsicht auf die Informationsbedürfnisse

4.3 Das Cranfield Paradigma

Ist das meistverbreitete Paradigma in akademischen Kreisen. Dabei gibt es drei Stufen der Evaluation der Retrieval-effektivität

- Ganzer Wissensbeschaffungsprozess
- Ganzes Retrieval-System
- Komponenten des Retrievalsystems

4.3.1 Aufgabenstellung

- zu evaluieren ist die Retrievaleffektivität des IR-Systems, mittels geeigneter Masse
- Es wird eine interne, direkte Evaluation angestrebt, d.h. die Effektivität soll direkt gemessen werden und nicht über ihren Anteil an einem grösseren Resultat, d.h. der IR-Lösung bewertet werden
- Die Evaluation soll experimentell erfolgen
- Das System wird als Blackbox behandelt
- erfolgt im Vergleich zu einem vorgegeben Richtresultat
- quantitative Evaluation

4.3.2 Ausgestaltung (Design)

Wird ein Labortest verwendet

4.3.3 Ausbeute / Präzision

- Zwei allgemein anerkannte Masse für Retrievalqualität sind Ausbeute und Präzision
- Diese Eigenschaften dieser Masse sind gut analysiert und verstanden
- beide Masse modellieren die Annahme, dass möglichst viel relevante und möglichst wenig irrelevante Information gefunden werden soll

Präzision: wenig irrelevante Information

Ausbeute: möglichst viele relevante Information

- Ziel: Optimierung eines oder beider Kriterien
- Die beiden Masse sind mengenbasiert
- Die Masse widersprechend sich
 - hohe Ausbeute \rightarrow niedrige Präzision
 - hohe Präzision \rightarrow niedrige Ausbeute

$$\begin{aligned}\text{Präzision} &:= \frac{\text{Anzahl relevante Dokumente im Resultat}}{\text{Anzahl Dokumente im Resultat}} \\ \text{Ausbeute} &:= \frac{\text{Anzahl relevante Dokumente im Resultat}}{\text{relevante Dokumente in der Kollektion}}\end{aligned}\tag{4.1}$$

Mathematische Definition

Ausbeute: $P_r(q) := \frac{|D_r^{rel}(q)|}{|D_r(q)|}$

Präzision: $\pi_r(q) := \frac{|D_r^{rel}(q)|}{|D_r(q)|}$

Interpolierte Ausbeute/Präzisionsfunktion: $\prod_q(p) := \max\{\pi_r(q) | p_r(q) \geq p\}$

Wobei:

- $D_r^{rel}(q) \rightarrow$ Relevante Dokumente in der Resultatmenge
- $D^{rel}(q) \rightarrow$ Menge aller relevanter Dokumente
- $D_r(q) \rightarrow$ Alle Dokumente der Resultatmenge

4.3.4 Average Precision

- Durchschnitt der Präzisionswerte an den Rängen aller relevanten Dokumente \rightarrow non-interpolated
- Durchschnitt der Präzisionswerte für spezielle Ausbeutungsgrade \rightarrow interpolated
- Beliebteste *Ein-Zahl-Mass*

Mean Average Precision

tbd - Kapitel 4 Slide 18

Einschränkungen unseres Vorgehens

- Nur eine Anfrage
- Präzisionsorientiertes Szenario
- Können Ausbeute nicht bestimmen
- Datenbasis nicht eingefroren
- Datenbasis nicht identisch
- Wie sind die Relevanzbeurteilungen zu betrachten?

Problem der Relevanzbewertung

Kapitel 5

Kategorisierung

5.1 Dokumentkategorisierung

Definition: *Bei der Kategorisierung werden Dokumente anhand ihres Inhaltes einer speziellen Kategorie zugewiesen*

- Eine Kategorie sollte innerhalb einer bestimmten Domäne einen wohl definierten Bereich darstellen
Es geht um den Umgang mit Bedeutung, obwohl wir in den meisten Fällen nur 'rohe Daten' zur Verfügung haben
- Kategorien sind unter anderem in folgenden Fällen einen Mehrwert
(Manuelle) Indexierung
Suche nach Information in Dokumentenarchiven oder News Feeds
Recommender (Netflix, Spotify etc)

5.1.1 Kategorisierung Aufgaben

Es gibt eine Menge unterschiedlicher Aufgaben

- Indexierung (Verschlagwortung)
Manuelle Methode → schwerfällig und kostenintensiv
Bei Automatischer Vergabe hat man ein kontrolliertes Vokabular
- Recommender (klassisch: Routing / Filtering)
Ein Userprofil bzw. Themenprofil wird regelmässig gegen die einkommenden Dokumente getestet, um zu bestimmen, wo letztere einzuordnen sind → verwandt: Push-Dienste
- Clustering
Gruppieren von Kollektionen in eine Menge von sich gegenseitig ausschliessende Kategorien - die aus den Daten gebildet werden
- Annotation
Gruppieren von Dokumenten mit weiterführender, dazugehöriger Information

Kategorisierung vs. Klassifikation

Klassifikation: Man ist nur bei einem zugehörig

Kategorisierung: hat mehrere 'Tags'
offizielle Unterscheidung ergänzen

5.2 Verfahren

5.2.1 Rocchio Model

tbd

5.2.2 kNN

tbd

5.2.3 Bayes Klassifizierung

- *gegeben:* Kategorie C_i mit einer angemessenen Anzahl von bereits zugeordneten Objekten (Trainingsdaten)
- *Methode:* Bilde statische Modelle aus diesen Kategorien. Benutze diese, um vorherzusagen, zu welcher Klasse ein neues Objekt gehört
- Wir kennen $P(t|C_i)$ für alle $t|C_i$, sind aber eigentlich interessiert an: $P(C_i|t)$ oder noch spezifischer in $P(C_i|D)$
- Aus alten Objekten kann man bestimmen nach welchen Merkmale zu suchen ist
- Annahme: Alle Dokumente sind voneinander unabhängig

5.2.4 Collaborative Filtering

Das collaborative filtering kann implizite und abstrakte Beziehungen zwischen zwei Filmen herstellen.

Beispielsweise: *Ist Jason Statham ein ähnlicher Schauspieler wie Vin Diesel?* → bei kollaborativen Filtering ist die Aussage *"Leute die A mögen, mögen auch B"*

Achtung: Ähnlichkeit bedeutet nicht notwendigerweise gleicher Geschmack

5.2.5 Thesaurus

tbd

- Aufpassen bei den Thesaurus muss man bspw. bei Synonymen, da es hier zu nicht korrekten Synonymen im Kontext kommen kann

5.2.6 Herausforderungen bzgl Klassifikationen

- Aufwand für Verwaltung, wie erfasst man Unschärfe
- Vollständigkeit
- Bedeutungen ändern sich → Eine Klassifikation muss sich ändern können, das hat grossen Einfluss auf die Verwaltung von Klassifikationen
- Menschen klassifizieren unterschiedlich → ebenfalls abhängig von der Zeit

5.3 Clustering

Definition: *Clustering ist ein Gruppierungsprozess bei dem eine Menge von Objekten in 'Cluster' von ähnlichen Objekten geordnet werden*

Kapitel 6

Web-Suche

6.1 Suche im Web

- Korpus: Das öffentliche zugreifbare Web besteht aus statischem und dynamischen Inhalt
 - Daten im Web sind unbeständig → 40% monatliche Änderung, unstrukturiert, teilweise schlechte Qualität und/oder heterogen
- Wird verwendet um sich zu navigieren
- Ziel: Finde qualitativ hochstehende Resultate (nicht zwingend Dokument), die für den Benutzer relevant sind
 - Statische Seiten
 - Resultate sind nicht ausbeuteorientiert
- Bedürfnisse
 - informationell
 - navigational
 - transaktional
 - graubereiche
- Soll nicht überschätzt werden

6.1.1 Anatomie von Suchmaschinen

- Spider (Crawler oder Roboter) - bildet den Korpus
 - Sammelt die Daten rekursiv
 - Für jede bekannte URL, hole die Webseiten, parse diese und extrahiere die neuen URLs
 - Zusätzliche Daten von direkten Einträgen und anderen Quellen
 - Verschiedene Suchmaschinen haben unterschiedliche Grundsätze - kleine Übereinstimmung unter den Korporen
- Der Indexer - verarbeitet die Daten (inverted files)
 - Verschiedene Grundsätze, welche Worte indexiert oder gestemmt werden, Phrasenunterstützung, Grossschreibung, UnicodeUnterstützung etc.
- Anfrageverarbeitung - akzeptiere Anfrage und liefere Resultate

Spidering

- Starte mit einer umfassenden Menge von URLs, von welchen die Suche (Spidering) gestartet wird (Seeds S_0)
- Speichere die Dokumente in D und die Hyperlinks in E . Dabei ist sowohl D als auch E ein eigenständiger Datenbehälter
- Während dem Crawling wird eine Liste Q von URLs intern unterhalten
- Wir extrahieren evtl. eine URL von Q oder fügen eine URL Q hinzu
Funktion: Dequeue() und Enqueue()
- Bezeichnungen
 u, v als eine URL
 $d(u)$ die zugehörige Webseiten

Anker

- extrahiere Ankertexte (zwischen $\langle a \rangle$ und $\langle /a \rangle$) für jeden Outlink)
- Normalerweise beschreibt der Ankertext das Dokument sehr gut, auf welches es zeigt
- Füge Ankertext hinzu, damit die Zielwebseite mit zusätzlichen Schlüsselwörter versehen wird
- Sehr effektiv, um relevante Information zu finden
- Google bietet teilweise Suchresultate, die nur aufgrund des Ankertexts zustande gekommen ist

6.2 Indexierung Webseiten

- Die exakte Indexierungsstrategie ist ein Geschäftsgeheimnis und variiert von Unternehmung zu Unternehmung
- Extremfall: Mann kann die Beschreibung jeder Seite auf die Kollektion der zugehörigen Ankertexte beschränken
- SPAM ist ein Problem, sowohl beim Spidern, als auch beim Indizieren

6.2.1 Anfrageverarbeitung

- Google verarbeitet ca. 5 Mrd. Anfragen pro Tag
- kleine Anzahl von häufigen Anfragen (80/20 Regeln) → aktuell bspw. Corona
- nächste Resultatseite vorberechnen
- Falls die Anfrage gesucht werden muss → cluster von PCs

6.3 Suchmaschinen Schlussfolgerung

- Spidering ist ein wichtiger Aspekt (was nicht im Index ist, kann nicht gefunden werden)
- Ranking ist der Schlüssel (eine gute Antwort in den Top 5/10)
- Verwendung von verschiedene Merkmalen → präzisionsorientiert
- Stemming / Stoppwortliste verbessert nicht immer das Resultat im gewünschten Sinne
- Geschwindigkeit und Wirtschaftlichkeit ist ein Schlüsselement

⇒ Je mehr Daten desto, einfacher ist die Suche

6.4 Suche und Hyperlink

6.4.1 Spreading Activation (SA)

- Das WWW wird als Hypertext verstanden
- tbd

6.4.2 Indegree

- Indegree ist die Anzahl der einkommenden Links zu einem gegebenen Dokument
- Indegree als Qualitätseinflussfaktor ist schwach
- tbd

6.4.3 PageRank

- Zu Beginn ist der Surfer auf einer zufälligen Webseiten
- PageRank einer Webseite = Wahrscheinlichkeit, dass ein Surfer sich zu einem beliebigen Zeitpunkt auf dieser Webseite befindet

Gedanken zu PageRank

- Berechnung rekursiv
- Kann offline berechnet werden
- Anfangs Page-Rank aller Knoten = 1
- Konvergiert gegen "Random-SurferWahrscheinlichkeiten"
- Durchaus anfällig auf Spam

6.4.4 Kleinberg Model

- HITS (Hypertext Induced Topic Selection) vorgeschlagen von Kleinberg
- Eine Webseite kann angesehen werden als
 - Ein Hub (Zeigt auf gute Quellen)
 - Eine Autorität (besitzt den Inhalt)
- Von einer Ursprungsmenge R , ziehen wir alle Nachbarn in Betracht
- Von dieser erweiterten Menge können wir die 'hubness' und autorität jedes einzelnen Knotens