Universität Stuttgart
Institut für Intelligente Systeme
Studienprojekt MUMS

Ausarbeitung zum Seminar:

Information Retrieval im Internet:
einige Fallbeispiele

Ingo Hildebrandt

Betreuer: Dipl.-Inf. Mark Giereth

Inhaltsverzeichnis

Information Retrieval im Internet: 1
Inhaltsverzeichnis 2
Klassisches Information Retrieval; Grundbegriffe 3
  Die Geschichte des Information Retrieval 3
  Retrieval vs. Browsing 4
  Dokumentrepräsentation 5
    Die logische Sicht der Dokumente / Index-Terme 7
    Systeminterne Datenrepräsentation 9
    Index-Term-Relevanz 9
  Der Retrieval Prozess 10
  Definition Information-Retrieval-Modell 11
  Das Boolesche Modell 12
  Das Vektor-Modell 14
Information Retrieval im Internet 17
  Die Besonderheit der Daten 17
  Suchmaschinen / Metasuchmaschinen 18
    Zentralisierte Architektur 18
      Crawler / Crawling 19
    Beispiel Suchmaschine: Google 20
      Konzeptueller Ablauf / Funktionsweise der einzelnen Einheiten 21
      Google-Frontend 22
  Webverzeichnisse 25
  Fazit / Ausblick 27
Relevanz für MUMS 28
  Browsing 28
  Lokale „Suchmaschine“ / Suchmaschinen-Eintrag 28
  Metainformationen / Metadata 29
  Webkataloge / Webverzeichnisse 29
Anhang 29
  Abbildungsverzeichnis 29
  Literaturhinweise 30
  Interessante Links 30

Klassisches Information Retrieval; Grundbegriffe

Der Begriff des Information Retrieval ist zwar mittlerweile im Deutschen ein durchaus geläufiger Begriff, allerdings ist die wirkliche Übersetzung ins Deutsche doch sehr aufschlussreich. Man kann es etwa mit „Wiederauffinden von Informationen“, „Informationrückgewinnung“ oder „Informationsgewinnung“ übersetzen. Wenn man also heute über Information Retrieval spricht, meint man meistens den Versuch, (mehr oder minder) gezielt Informationen wiederzufinden bzw. zu gewinnen. Darüber hinaus ist ein zentrales Thema im Information Retrieval die Repräsentation, aber auch die Speicherung und Organisation von Daten.

Das Hauptziel des Information Retrieval ist es also Informationen zu finden, mit dem Unterton, dass natürlich nur solche Daten gefunden werden sollen, die für den Benutzer möglicherweise hilfreich bzw. relevant sein könnten. Aus diesem (Benutzer-)Ziel, auch als „User Task“ bezeichnet, ergeben sich einige Fragen:

Was sucht der Benutzer eines Information Retrieval Systems eigentlich, und wie kann das Information Retrieval System ihn dabei unterstützen diese Informationen finden?

Wie ist es überhaupt möglich die Anfrage(n) eines Benutzers zu interpretieren und zu verarbeiten?

Wie können Daten so organisiert werden, dass sie möglichst schnell wiedergefunden werden können?

Um nur einige zu nennen. Im Folgenden soll diese Fragen genauer untersucht und zumindest teilweise beantwortet werden.

Bevor wir nun aber tiefer in die Materie einsteigen, will ich den Lesern zunächst etwas über die Geschichte der Information Retrieval Systeme erzählen und dann mit einigen Grundbegriffen und Grundüberlegungen bekannt machen, ohne die ein tieferes Verständnis nicht möglich ist.

Die Geschichte des Information Retrieval

Seit jeher war es dem Menschen ein Bedürfnis sein Wissen in irgendeiner Form weiterzugeben – und das in möglichst dauerhafter Form. War es früher noch üblich die Geschichte durch erzählen von eben dieser weiterzugeben, gab es spätestens mit Johannes Gutenbergs Erfindung des Buchdrucks eine potente Möglichkeit Wissen in dauerhafter Form weiterzugeben, nämlich in Form eines gedruckten Buches. Seit jener Zeit im 15. Jahrhundert hat die Menschheit doch so einiges zustande bekommen und das sogenannte „Wissen der Welt“ wuchs und wuchs. So war es bald nötig das vorhandene Wissen irgendwie zu ordnen, da es nahezu unmöglich war etwas wiederzufinden. Es mag schon in dieser Zeit gewesen sein, dass es eins der ersten Information Retrieval „Systeme“ gab, nämlich z.B. den Bibliothekar, der zumindest wusste wo welches Buch zu finden war. Im Laufe der Jahrhunderte wuchs die Masse von vorhandenen Informationen immer weiter an, und mit dem technologischen Fortschritt ergab sich auch die Möglichkeit, neue Wege der Informationsverwaltung zu beschreiten. Umso verwunderlicher ist es, dass bis spät ins 20. Jahrhunderts die meisten Bibliotheken immer noch mit der alten Informationsverwaltung arbeiteten.

Namhaft waren dies die Karteikästen, in denen Kärtchen waren, auf denen z.B. der Autor und der Titel des Buches verzeichnet waren, sowie wo diese Buch denn zu finden war. Außerdem gab es im Buch selbst ein erstes Information Retrieval Werkzeug, nämlich den Index (also wo welches Wort im Buch selbst zu finden war). Mit dem breiten Einzug der Computer wurde dann damit begonnen die Verwaltung des Wissens zu digitalisieren. Diese Information Retrieval System der ersten Generation waren aber schlichte digitalisierte Karteikästen, also prinzipiell eine 1:1-Abbildung der alten in die neue Welt. Seither ist viel Forschungsaufwand investiert worden und die zweite Generation der Information Retrieval Systeme (namentlich Bibliothekssysteme zur Verwaltung riesiger digitaler Bibliotheken) bot bereits erweiterte Funktionen, wie z.B. die Suche nach Themen, Überschriften oder Schlüsselwörtern. Seit der neuerlichen Internet-Revolution Anfang der 1990er und der damit verbundenen Einführung des **W**orld **W**ide **W**eb (WWW) sind immer bessere und schnellere Information Retrieval System entwickelt worden. (Für einen Überblick über wichtige Entwicklungen und Persönlichkeiten: siehe [1] im Anhang).

Diese sind auch nötig, um die immer größer werdende Flut von Information überhaupt noch einigermaßen überschauen zu können. Einige dieser „neuen“ Information Retrieval System und deren Grundlage sowie essentielle Veränderung seit Einführung des WWW sollen im zweiten Teil dieses Dokuments betrachtet werden. Aber eins nach dem anderen. Zuerst wollen wir nun die Grundbegriffe und Modelle des „klassischen“ Information Retrieval betrachten, um eine gute Grundlage für die Betrachtung des Internet Information Retrieval zu bekommen.

Retrieval vs. Browsing

Es kann grundlegend unterschieden werden, ob das Ziel eines Benutzers eines Systems ist seine Informationen per Browsing oder Retrieval zu bekommen.

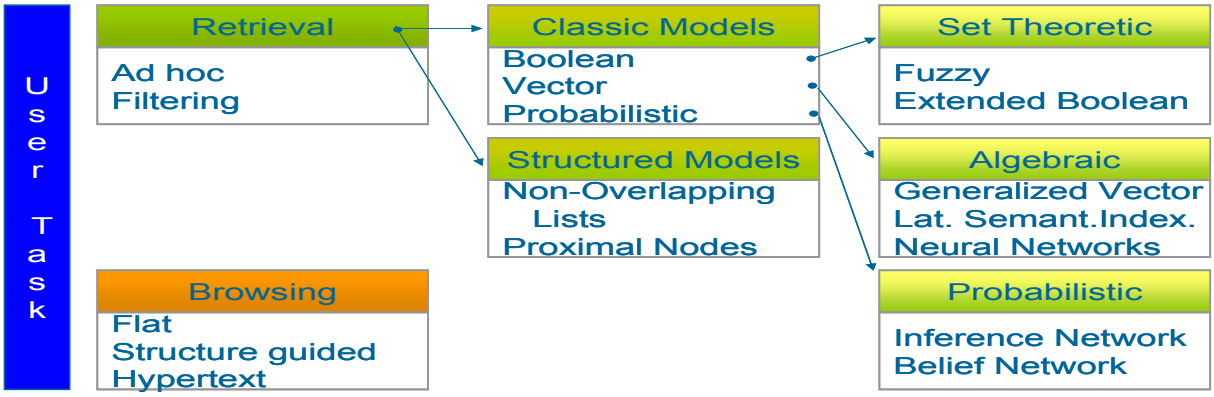


Abbildung 1: Browsing – Retrieval (Taxonomie der IR-Modelle)

Browsing bedeutet in diesem Zusammenhang also, dass er sich durch eine Sammlung von Dokumenten manövriert, um so die von ihm gesuchten Informationen zu finden. Diese Art des Information Retrieval sei nur kurz erwähnt, da das später betrachtete Information Retrieval im Internet prinzipiell immer auch mit Browsing zu tun hat.

Wie in Abbildung 1 zu sehen ist, kann man das Retrieval in zwei verschiedene Arten unterteilen. Zum einen in „Ad hoc“ zum anderen in „Filtering“.

Zu „Ad hoc“ ist zu sagen, dass diese Variante einen so genannten Pull-Service darstellt, dies bedeutet, dass der Benutzer im übertragenen Sinne am Information Retrieval System „zieht“, um einen spontan („ad hoc“) auftretenden Informationsbedarf zu befriedigen, was soviel bedeutet wie dass der Benutzer aktiv nach den gewünschten Informationen sucht. Die Hauptaufgabe des Information Retrieval Systems ist es also in ihrer Wissensbasis nach den gewünschten Informationen zu suchen und diese dem Benutzer zur Verfügung zu stellen.

Zu „Filtering“ ist zu sagen, dass diese Variante einen so genannten Push-Service darstellt. Dies bedeutet, dass der Benutzer über einen längeren Zeitraum aktuelle Informationen benötigt bzw. haben will, und das System „schiebt“ („push“) ihm diese über das Information Retrieval System zu. Die Aufgabe des Information Retrieval Systems ist es in diesem Falle also, die gewünschten Informationen aus dem Strom der verfügbaren Informationen entsprechend der Benutzervorgabe herauszufiltern („filtering“). Ein Beispiel für diese Variante des Information Retrieval wäre z.B. ein Börsen- oder Nachrichtenticker.

Wir wollen uns aber in diesem Dokument nur mit dem „ad hoc“-Modus beschäftigen.

Dokumentrepräsentation

Es wird jedem bewusst sein, dass wir Informationen in vielerlei Arten austauschen können. Die häufigste Art bei uns Menschen ist das Sprechen, also die Übermittlung von Informationen durch das Medium der Sprache. Da diese die Grundlage vieler anderer Formen des Kommunikationsaustauschs bildet, wollen wir die Repräsentation von Informationen durch Sprache als erstes unter die Lupe nehmen. Wenn wir mit dem Medium der Sprache kommunizieren, ist dies nur möglich, weil wir (normalerweise) wissen worüber der Andere spricht. Das ist der Fall, weil wir, wenn wir uns z.B. im deutschen Sprachraum bewegen, über dasselbe Vokabular verfügen, abgesehen vielleicht von Regionalsprachen (wie z.B. schwäbisch), die von „Auswärtigen“ schwer bzw. überhaupt nicht zu verstehen sind. Daran erkennt man eines: eine sinnvolle Kommunikation ist nur dann möglich, wenn beide Seiten wissen „worüber gesprochen“ wird.

Da der Benutzer eines Information Retrieval Systems in der Regel die natürliche Sprache zur Kommunikation benutzt und das System diese in aller Regel nicht versteht, ist es nötig beide Kommunikationsformen auf eine sinnvolle und gegenseitig verständliche Repräsentation abzubilden. In Abbildung 1 soll dieser Sachverhalt verdeutlicht werden:

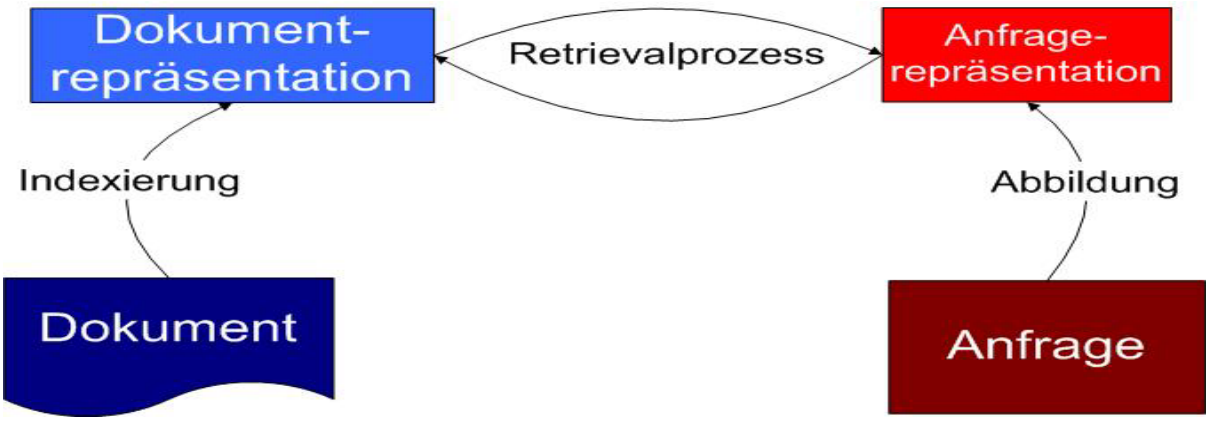


Abbildung 2: Dokumentrepräsentation

Dies geschieht in aller Regel dadurch, dass die Anfrage des Benutzers auf eine vom Information Retrieval System zu verstehende Repräsentationsform abgebildet werden muss.

Die am häufigsten verwendete Methode ist hierbei diejenige, die Anfrage, die z.B. ein natürlich sprachlicher Satz ist, in eine Ansammlung von so genannten Schlüsselwörtern (auch als Index-Terme bekannt) umzuwandeln, die dann wiederum während des Retrieval-Prozesses mit den Index-Termen, die im Information Retrieval System vorhanden sind, verglichen werden können. Auf die genauen Methoden der Umwandlung von der Ausgangsform des Dokuments in die gewünschte Repräsentationsform, wie z.B. die Indexierung („Indexing“) wollen wir später genauer eingehen.

Ein weiteres Hauptproblem ist spracheigener Natur. Stellen wir uns einmal vor, dass ein Benutzer sich an ein Information Retrieval System (wie auch immer dieses aussehen möge) begibt und nur ein Wort als Suchanfrage stellt, nämlich „go“. Nun ist es die Aufgabe des Information Retrieval Systems geeignete Informationen dazu zu suchen, diese zu finden und dem Benutzer zurückzuliefern. Bei genauerer Betrachtung drängt sich dabei nur die Frage auf, über welches „go“ der Benutzer überhaupt Informationen haben will? Prinzipiell könnte er das englische Verb **go** meinen, aber genauso gut könnte es sein, dass er sich Informationen über das alte japanische Brettspiel **go** verschaffen will. Bei solch einem Phänomen spricht man von einer sprachübergreifenden Mehrfachbedeutung (oder im linguistischen Fach-Terminus: von einer sprachübergreifenden *Polysemie*). Ein damit verwandtes Problem ist das der sprachinternen *Polysemie*. Stellen wir uns vor derselbe Benutzer tippt dieses Mal „Jaguar“ ein. Nun stellt sich wiederum die Frage über was der Benutzer Informationen erhalten will. Über das Tier? Über die Automarke? Über ein altes Atari-Videospiel? etc. Wie man an diesen zwei einfachen Beispielen erkennen kann, gibt es beim Information Retrieval viele Aspekte, die es zu beachten gilt, eben weil es um die Gewinnung von Informationen geht und diese Informationen nicht immer eindeutig sind. Im Gegensatz dazu beschäftigt sich das Feld des **Data Retrieval** (Datengewinnung) mit dem schlichten Auffinden von Schlüsselwörtern in den vorhandenen Dokumenten, was einer binären Relation entspricht. Also ist ein Dokument beim Data Retrieval entweder relevant (erfüllt Kriterien der binären Relation) oder nicht, was den normalen Data Retrieval Systemen (z.B. relationale Datenbank) auch völlig genügt. Im Gegensatz dazu ist eine binäre Entscheidung für ein Information Retrieval System einfach nicht ausreichend, da beim teilweise komplexen Inhalt der Dokumente durch eine strikte Ja/Nein-Entscheidung zu viele möglicherweise relevanten Dokumente einfach überhaupt nicht erfasst werden. Diese Unterscheidung zwischen Data Retrieval und Information Retrieval ist für uns insoweit wichtig, dass wir bei der späteren Analyse geeigneter Vorgehensweisen diejenigen Aussortieren können, die „nur“ für das Data Retrieval geeignet wären, nicht aber für das Information Retrieval. Außerdem bringt uns diese Betrachtung eine weiter wichtige Erkenntnis: Das Information Retrieval System muss nicht nur schlicht entscheiden können, ob ein Schlüsselwort bzw. mehrere gewisse Schlüsselwörter (Index-Terme) in einem Dokument enthalten sind oder nicht, sondern eben, ob diese im Kontext der gestellten Benutzeranfrage relevant sind oder nicht. Also kann man durchaus davon sprechen, dass die **Relevanz** (von Dokumenten) im Mittelpunkt des Information Retrieval steht.

	Data Retrieval	**Information Retrieval**
Übereinstimmung	exakte Übereinstimmung	teilweise Übereinstimmung, beste Übereinstimmung
Anfragesprache	Künstlich	Natürlich
Anfragespezifikation	Vollständig	nicht vollständig
Suche nach	Übereinstimmung	**Relevanz**

Die logische Sicht der Dokumente / Index-Terme

Dokumente, also in unserer Betrachtung Texte, können auf verschiedene Weise betrachtet bzw. repräsentiert werden. Sie können z.B. durch eine Menge von Schlüsselwörtern („Index-Terme“) dargestellt werden. Rein prinzipiell wäre es bei heutigen Computersystem nahezu problemlos möglich Dokumente in ihrem vollen Umfang zu speichern, was der logischen Sicht: Volltext („full text logical view“) entsprechen würde. Aber mit immer größer werdenden Ansammlungen von Dokumenten kann es trotzdem nötig sein, die Repräsentation zu verbessern, z.B. um die Suchgeschwindigkeit deutlich zu erhöhen – denn es erscheint logisch, dass eine Menge von Schlüsselwörtern (z.B. 10 Stück) schneller zu durchsuchen ist als ein ganzes Dokument (z.B. 10 A4-Seiten Text). Dies kann durch verschiedene Operationen erreicht werden (siehe Abbildung 2).

Diese Operationen nennt man Textoperationen oder auch Texttransformationen. Diese verringern die Komplexität der Dokumentendarstellung beträchtlich und erlauben es die Dokumentrepräsentation von der logischen Sicht „Volltext“ auf die logische Sicht „Schlüsselwörter“ zu reduzieren.

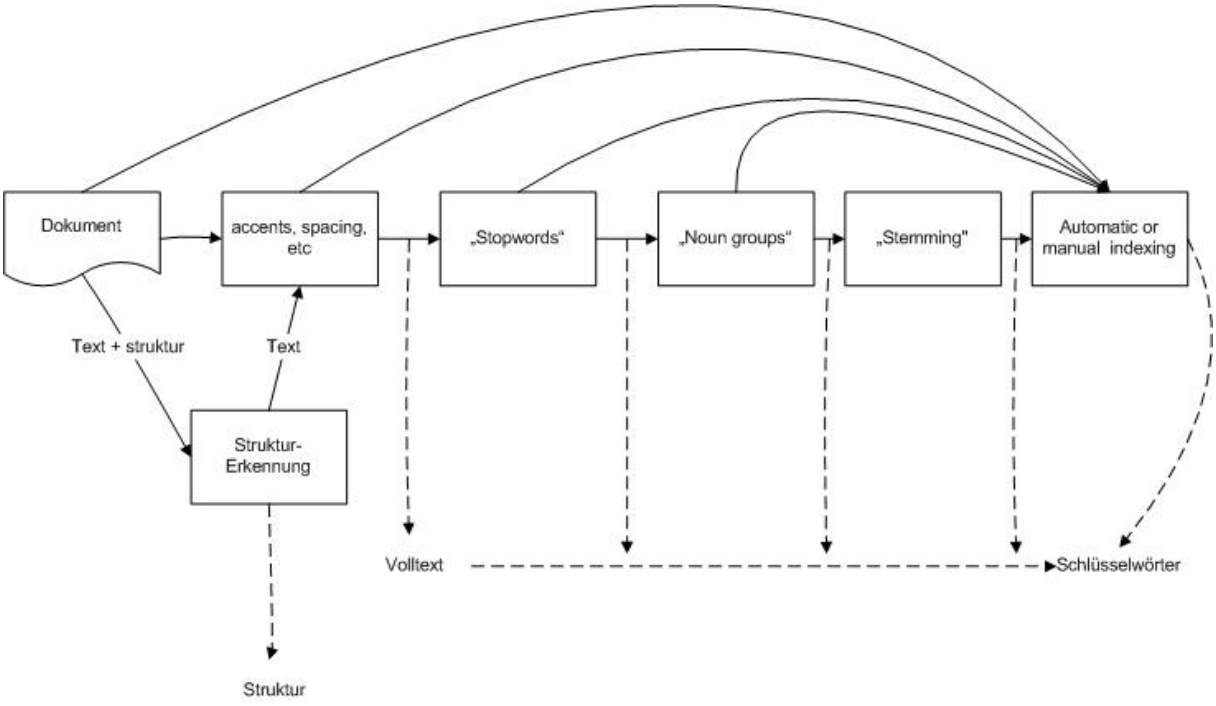


Abbildung 3: Textoperationen – vom Volltext zu einer Menge von Schlüsselwörtern

Wir wollen uns diese Textoperationen nicht bis in kleinste Detail anschauen, jedoch soll der Ablauf kurz erklärt werden. Dazu betrachten Sie die obige Abbildung 2 von links nach rechts:

1. Zuerst wird das Dokument dahingehen untersucht, ob es nur fortlaufender Text („flach“) ist, also Text ohne Struktur.

1.a)Wenn ja, werden alle Zeichen wie Apostroph oder doppelte Leerzeichen entfernt.

1.b)Wenn nein, dann wird die Struktur sozusagen extrahiert und der Ablauf geht bei 1.a) los.

2. Danach werden die sogenannten Stoppwörter („**Stopwords**“) entfernt. Diese sind sehr häufig auftretende Wörter wie z.B. Artikel und Verbindungswörter und können daher entfernt werden.
3. Danach folgt die Identifizierung von Nomen (Nouns) bzw. Nomen-Gruppen („**Noun Groups**“), wobei eine Nomen-Gruppe meist zwei Hauptwörter sind, die entweder in einem gewissen Abstand zueinander im Text stehen dürfen (…the **power** of the **internet**…. – also hier würden power und internet eine Nomen-Gruppe bilden, sofern der Abstand auf <= 3 gesetzt wurde, da diese ja nur zwei Worte zwischen sich stehen haben) bzw. die zusammengesetzte Begriffe darstellen (so genannte Komposita wie z.B. „Weißes Haus“).
4. Danach wird das sogennante „stemming“ durchgeführt. Dies bedeutet, dass versucht wird alle Wörter auf ihren Wortstamm zurückzuführen. Dies verhindert, dass später keine Überstimmung zwischen Anfrage und Dokument gefunden wird, nur weil in der Anfrage z.B. Computer steht und im Dokument Computers – also zur Eliminierung des Plurals und der Zeitform.
5. Danach kann dann die Auswahl der Schlüsselwörter erfolgen, was heute normalerweise automatisch erfolgt (vielleicht abgesehen von einigen Domänen, wo die Auswahl von Schlüsselwörtern immer noch durch Experten erfolgt).

Zu erwähnen ist dabei, dass es natürlich je nach im Dokument verwendeter Sprache unterschiedlich sinnvoll ist einzelne, der oben genannten, Schritte durchzuführen. Es macht nämlich in einigen Sprachen, z.B. in der spanischen Sprache, keinen Sinn Lautzeichen (siehe 2.)) zu entfernen, da es viele Worte gibt, die mit bzw. ohne Lautzeichen verschiedene Bedeutungen haben. Außerdem ist die Stammbildung („stemming“) in verschiedenen Sprachen ebenfalls kein wirklich probates Mittel.

Wenn man nun also die oben genannten Schritte durchgeführt hat, erhält man eine gewisse Anzahl von Schlüsselwörtern, die weitläufig nicht nur als „Keywords“ sondern vor allem auch als **Index-Terme** bekannt sind. Wir wollen von nun an diesen Terminus benutzen. Nun stellt sich die Frage, wie eben diese Index-Terme dargestellt werden. – Normalerweise werden die Index-Terme in Index-Term-Vektoren dargestellt, wobei man sich solch einen Vektor als geordnete Menge von Werten für das Wortvorkommen des jeweiligen Wortes vorstellen kann. Die Werte sind dabei konjunktiv, also mit dem logischen „und“ verknüpft.

binär

	Term 1 *Öl*	Term 2 *Preis*	Term 3 *Alaska*
Dokumentvektor: (1,1,0)	1	1	0
Anfragevektor: (1,1,0)	1	1	0

Frequenz

Dokumentvektor: (4,8,0)	4	8	0
Anfragevektor: (3,6,0)	3	6	0

Abbildung 4: Beispiel Vektordarstellung

Das Beispiel in Abbildung 4 zeigt uns z.B., dass das Wort „Öl“ im Dokument vorkommt und in der Anfrage ebenfalls (binär) und dass das Wort im Dokument insgesamt viermal genannt ist, im Anfragevektor dreimal (Frequenz).

Diese Vektorenrepräsentation wird, wenn wir zu den Information Retrieval Modellen kommen, noch eine wichtige Rolle spielen.

Systeminterne Datenrepräsentation

Nachdem wir nun das Prinzip der Index-Terme und deren Gewinnung kennen, können wir uns einer kurzen Betrachtung der internen Datenrepräsentation zuwenden. Die darunterliegenden Datenstrukturen wollen wir außer Acht lassen. Wenn das Information Retrieval System nun also die Index-Terme zur Repräsentation des Dokuments gewonnen hat, werden diese normalerweise im Index gespeichert.
Dieser Index kann prinzipiell in zwei verschiedenen Formen vorliegen, entweder als (Vorwärts-)Index oder als (Rückwarts-)Index. Im (Vorwärts-)Index stehen also ganz einfach ausgedrückt die Dokumente, repräsentiert durch ihre Index-Terme, d.h. man müsste jedes Dokument anschauen, um zu erfahren, ob ein bestimmter Index-Term enthalten ist oder nicht – dies wäre natürlich relativ ungeschickt. Im (Rückwarts-)Index („inverted index“) dagegen stehen die Index-Terme, mit einer Reihe von angehängten Informationen, z.B. in welchem Dokument kommt dieser Index-Term vor, und u.U. sogar noch wesentlich detailliertere Informationen, wie z.B. in welchem Satz oder in welchem Absatz dieser Index-Term vorkommt. Da vor allem der (Rückwarts-)Index bei den Suchmaschinen im Internet die entscheidende interne Repräsentationsform darstellt, wollen wir uns zur Verdeutlichung ein kurzes Beispiel anschauen.

Eintrag:	**Preis**
Dok.-Nr.	**1, 35, 42, 67**
# Dok.	**4**
# insg.	**7**
Wort-Nr.	**(1: 4, 28), (35: 99), (42: 13, 17, 55), (67: 432)**
Satz-Nr.	**(1: 1, 3), (35: 15), (42: 9, 9, 15), (67: 58)**
Absatz-Nr.	**(1: 1, 1), (35: 2), (42: 2, 3), (67: 7)**
Font	**(1.4: 28), (1.28: 10), (35.99: 12), (42.13: 72), (42.17: 12), (42.55: 12), (67.432: 20)**

Abbildung 5: Beispiel RückwartsIndex (Inverted Index)

Ein weiterer wichtiger Punkt ist die

Index-Term-Relevanz

Zuerst wollen wir uns die formale Definition anschauen. Sie ist die Basis zum Verständnis des Relevanz-Begriffs und somit eine essentielle Grundlage für das Verständnis der später eingeführten Ranking-Funktionen.

$w_{i,j}$	quantifiziert Relevanz eines Index-Terms für Beschreibung eines Dokumentinhalts
Definition	
t	Gesamtanzahl Index-Terme im System
k_i	eindeutiger Index-Term
$K = \{k_1, ..., k_t\}$	Menge aller Index-Terme
$w_{i,j} > 0$	Gewicht für Index-Term k_i in Dokument d_j, $w_{i,j} > 0$: Term k_i nicht in Dokument d_j
d_j	Dokument d_j
$\overrightarrow{d_j} = (w_{1,j}..., wt_{,j})$	Index-Term-Vektor für Dokument d_j
$g_i(\overrightarrow{d_j}) = w_{i,j}$	Funktion: liefert Gewicht für Index-Term k_i in t-dimensionalem Vektor

Abbildung 6: Definition Index-Term-Relevanz

Der Retrieval Prozess

Hier soll der eigentliche Retrieval-Prozess einmal schematisch anhand einer Beispielarchitektur (siehe Abb. 3) erläutert werden:

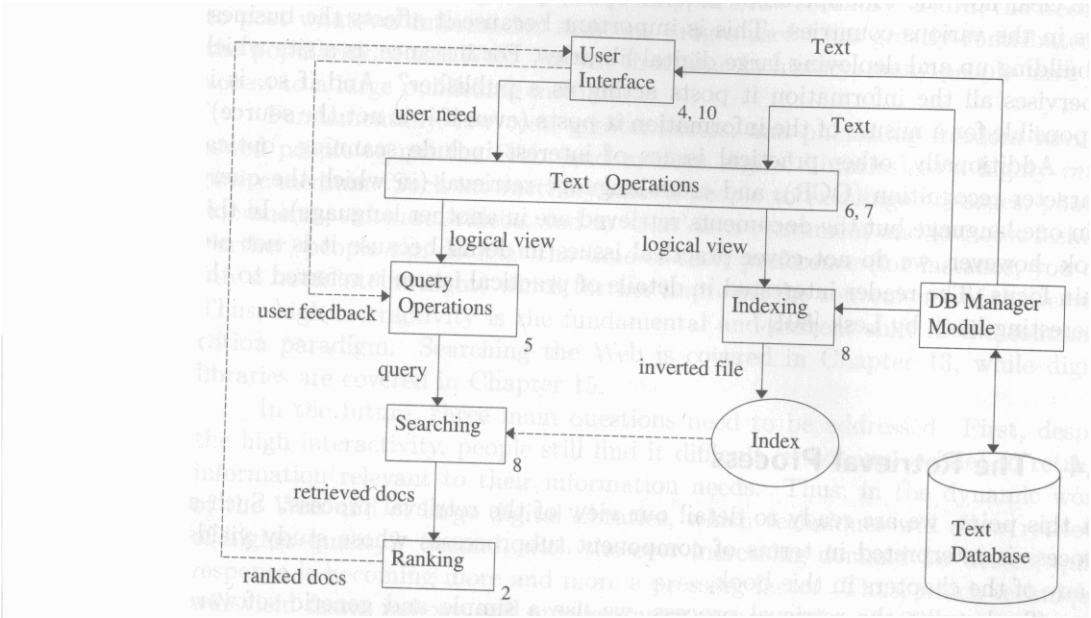


Abbildung 7: Der Retrieval Prozess

Die Abbildung 7 wurde dem diesem Seminar zugrunde liegenden Buch, Modern Information Retrieval [2], entnommen. Deshalb sind auch die Ziffern in der Abbildung zu vernachlässigen (diese beziehen sich auf die Kapitel im Buch, in denen die Vorgänge jeweils näher erklärt werden).

Bevor der eigentliche Retrieval-Prozess gestartet werden kann, muss zuerst eine Text Datenbank definiert werden. Das übernimmt normalerweise der Datenbank-Manager, der folgendes spezifiziert: (a) die zu verwendenden Dokumente

(b) die Operationen, die auf dem Text ausgeführt werden dürfen
(c) das Textmodell (also konkret z.B. die Textstruktur und welche Elemente überhaupt gesucht werden können).

Danach erzeugt der Datenbank Manager einen Index des Textes, diese kompakte Datenstruktur ermöglicht es erst eine große Datenmenge schnell zu durchsuchen. Der Index ist also die Gesamtheit der Index-Terme, mit deren Gewinnung wird uns ja bereits beschäftigt haben.

Danach stellt der Benutzer seine Anfrage, welche dann geparst und transformiert wird. Danach werden die Anfrageoperationen (query operations) ausgeführt, um die Anfrage in eine geeignete systeminterne Anfrage (query) zu transformieren. Diese wird ausgeführt, und es werden Dokumente gefunden. Bevor der Benutzer diese jedoch angezeigt bekommt, werden sie entsprechend ihrer **Relevanz** in Bezug auf die Anfrage bewertet und sortiert. In dieser Beispiel-Architektur kann der Benutzer jetzt anhand der Ergebnisse nochmals seine Anfrage verfeinern. Dadurch bekommt er normalerweise bessere Ergebnisse.

Nachdem wir nun den groben Ablauf eines Retrieval-Prozesses kennen, können wir uns der näheren Betrachtung zuwenden.

Definition Information-Retrieval-Modell

Die klassischen Information-Retrieval-Modelle basieren alle auf der Annahme, dass Dokumente durch Index-Terme repräsentiert werden. Wie man diese Index-Terme gewinnt haben wir schon im Abschnitt über die logische Sicht der Dokumente behandelt.
Wie in der folgenden Abbildung 7 zu sehen ist, ist das Information-Retrieval-Modell ein Quadrupel, welches aus einer Menge von logischen Sichten auf Dokumente (D), einer Menge von logischen Sichten auf Anfragen (Queries/Q), einem Modellierungsrahmen (Framework/F) und einer Ranking-Funktion R(q, d) besteht.

Ein IR-Modell ist ein Quadrupel $[D,Q,F,R(q_i,d_j)]$

D	Menge logischer Sichten auf Dokumente (Repräsentationen)
Q	Menge logischer Sichten auf Informationswünsche (Anfragen/Queries)
F	Modellierungsrahmen (Framework) für Dokumentrepräsentationen, Anfragen und deren Beziehungen
$R(q_i,d_j)$	Ranking-Funktion ordnet Anfrage $q_i \in Q$ und Dokument $d_j \in D$ einen Wert zu; definiert Reihenfolge der Dokumente bezüglich Query q_i

Abbildung 8: Definition IR-Modell

Dieses Modell stellt die allgemeine Grundlage dar. Der Modellierungsrahmen und die Ranking-Funktion sind hierbei die wichtigsten Elemente. Um nun tatsächlich Dokumente vergleichen zu können, müssen die definierten Elemente, wie z.B. die Ranking-Funktion

genau spezifiziert werden. Das wollen wir nun tun und uns exemplarisch zwei der klassischen Information Retrieval Modelle anschauen. Zuerst wollen wir das Boolesche Modell betrachten, da man dieses Modell als Grundlage für das zweite Modell, das Vektor-Modell, ansehen kann. Danach wollen wir dann eben das zweite Modell, das Vektor-Modell („vector space modell (vsm)“), betrachten.

Das Boolesche Modell

Benannt nach George Boole (1815 – 1864), englischer Mathematiker und Logiker:

Abbildung 9: George Boole

Die Basis für das Boolesche Modell sind die Mathematik bzw. die Logik. Namhaft die Gruppentheorie und, wie der Name schon richtig vermuten lässt, die Boolesche Algebra. Man könnte sagen, dass das Boolesche Modell das einfachste der klassischen Modelle ist. Die Anfragen werden in Form von booleschen Ausdrücken formuliert. Dabei werden gültige Ausdrücke aus Index-Termen in Verbindung mit den booleschen Operatoren: „und“, „oder“ und „nicht“ gebildet. Ein kurzes Beispiel soll diese Syntax verdeutlichen:

Beispiel-Anfrage:

Öl AND (Preis OR (NOT(Alaska)))

$$= q_i = k_a \wedge (k_b \vee \neg k_c)$$

Der Hauptvorteil ist dabei, dass die Anfragen alle einer präzisen Syntax unterliegen und somit auch die Semantik der Anfragen in der Regel relativ schnell ersichtlich ist. Allerdings stellt dieser Sachverhalt auch gleichzeitig einer der Hauptnachteile dar, da nur Anfragen zulässig sind, die genau dieser strikten Syntax folgen. Was im Beispiel noch so einfach anmutet, kann bei umfangreicheren Anfragen schnell zum Problem werden, da es nicht immer leicht ist, Anfragen, die in natürlicher Sprache formuliert sind, in einen booleschen Ausdruck umzuwandeln. Außerdem ist die Formulierung in booleschen Ausdrücken wohl mathematisch geschulten Menschen intuitiv klar, allerdings kann nicht davon ausgegangen werden, dass ein „Normal-Benutzer“ eines Information Retrieval Systems mathematisch geschult ist.

Nun wollen wir uns die formal korrekte Definition des Booleschen Modells näher anschauen.

$w_{i,j} \in \{0,1\}$

q Anfrage (query): Boolesche Ausdrücke (Terme verknüpft durch 'nicht', 'und', 'oder')

$\overrightarrow{q_{dnf}}$ disjunktive Normalform (DNF) der Anfrage q

$\overrightarrow{q_{cc}}$ konjunktive Komponente in $\overrightarrow{q_{dnf}}$

Ähnlichkeit von Dokument d_j und Anfrage q:

$$sim(d_j, q) = \begin{cases} 1 \; falls & \exists \overrightarrow{q_{cc}} \mid (\overrightarrow{q_{cc}} \in \overrightarrow{q_{dnf}}) \wedge (\forall k_i, g_i(\overrightarrow{d_j}) = g_i(\overrightarrow{q_{cc}})) \\ 0 & sonst \end{cases}$$

Abbildung 10: Definition boolesches Modell

Der Modellierungsrahmen (Framework/F) besteht aus einer Gruppe von Dokumenten und den Standard Mengenoperationen, die aus der Mathematik bekannt sein sollten.
Da es sich beim booleschen Modell um ein binäres Modell handelt, kennt die Gewichtungsfunktion des Modells nur genau zwei Werte. Entweder ein Dokument ist in Bezug auf die Suchanfrage relevant, hat also den Wert 1 (true/relevant/die Index-Terme sind enthalten), oder es ist irrelevant, hat also den Wert 0 (false/irrelevant/die Index-Terme sind nicht enthalten), was das System doch sehr einschränkt. Daraus folgt, dass es keinerlei Bewertungsskala der Ergebnisse geben kann, da die Dokumente, die der Anfrage genügen, eben alle den Wert 1 besitzen, also relevant sind, und alle Dokumente, die der Anfrage nicht genügen, eben alle den Wert 0 besitzen, also als irrelevant angesehen werden. Das heisst im Klartext, dass das boolesche Modell eher nicht um ein richtiges Information-Retrieval-Modell, sondern wohl doch eher um ein Data-Retrieval-Modell handelt. Da es beim Data-Retrieval genügt alle Dokumente als Ergebnis zu bekommen, die der Anfrage genügen.

Wir wollen uns nun die obige Beispiel-Anfrage genauer anschauen:

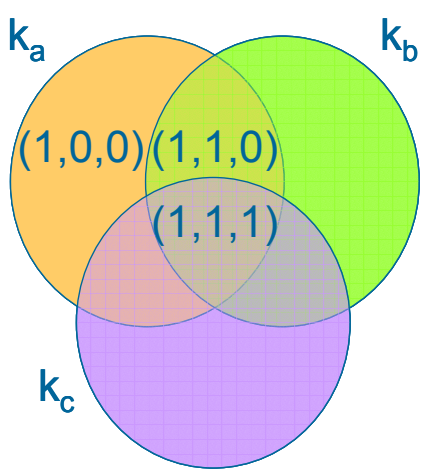
Öl ∧ (Preis ∨ (¬(Alaska)))

$$q_i = k_a \wedge (k_b \vee \neg k_c)$$

Öl	Preis	Alaska			
k_a	k_b	k_c	$\neg k_c$	$k_b \vee \neg t_c$	$k_a \wedge (k_b \vee \neg k_c)$
0	0	0	T	T	F
0	0	1	F	F	F
0	1	0	T	T	F
0	1	1	F	T	F
1	0	0	T	T	T
1	0	1	F	F	F
1	1	0	T	T	T
1	1	1	F	T	T

Abbildung 11: Beispiel-Anfrage

Wie bereits erwähnt, setzt sich diese Anfrage aus Index-Termen (in diesem Fall sind dies: Öl, Preis, Alaska) und booleschen Operatoren (in diesem Fall: „oder“ und „nicht“) zusammen. Wenn man die Anfrage als logische Formel betrachtet und die möglichen Belegungen durchgeht, wird klar, dass es drei verschiedene Kombinationen gibt. Diese bilden die konjunktiven Komponenten der Disjunktiven Normalform (DNF). Die Anfrage q_i kann also in disjunktiver Normalform (DNF) geschrieben werden als q_i = (1, 1, 1) „oder“ (1, 1, 0) „oder“ (1, 0, 0). Diese konjunktiven Vektoren (z.B. (1, 1, 1)) können also als jeweils binär gewichtete Vektoren angesehen werden.



$$q_i = k_a \wedge (k_b \vee \neg k_c)$$

Disjunktive Normalform (DNF)
(Disjunktion konjunktiver Vektoren):

$$\vec{q}_{dnf} = (1,1,1) \vee (1,1,0) \vee (1,0,0)$$

Abbildung 12: Anfrage in Disjunktiver Normalform (DNF)

Also würde z.B. das Dokument d_j = (0, 1, 0) in Bezug auf die Anfrage q_i als nicht relevant angesehen, obwohl ja zumindest k_b (also der Index-Term „Preis“) enthalten ist. Genauso würde das Dokument d_k = (1, 0, 1) zurückgewiesen werden, also ist auch zu sehen, dass es egal ist, ob das Dokument nun 1 oder 2 der 3 Teile enthält – es wird genauso als irrelevant angesehen, als wenn überhaupt kein Index-Term enthalten wäre.

Darum wäre es sinnvoll, das boolesche Modell dahingehen zu erweitern, dass in irgendeiner Form eine nicht binäre Bewertung stattfindet (also nicht nur die Werte 0 und 1 selbst zulässig sind, sondern auch alle Wert zwischen 0 und 1), um so eine (Rang-)Ordnung (Ranking) bzw. Sortierung der Ergebnisse zu ermöglichen. Diese Erweiterung führt zum Vektor-Modell, welches das zweite klassische Information-Retrieval-Modell darstellt. Deshalb werden wir dieses Modell als nächstes betrachten.

Das Vektor-Modell

Zuerst wollen wir uns auch hier die Definition anschauen.

$w_{i,j}$	Gewicht für Paar (k_i, d_j), positiv, nicht binär
$w_{i,q}$	Gewicht für Paar $[k_i, q]$, ≥ 0
t	Gesamtanzahl der Index-Terme im System
$\vec{q} = (w_{1,q}, w_{2,q}, \ldots, w_{t,q})$	Anfrage-Vektor
$\vec{d_j} = (w_{1,j}, w_{2,j}, \ldots, w_{t,j})$	Dokument-Vektor

Abbildung 13: Definition Vektormodell

Wie man leicht erkennt, besitzt das Vektor-Modell („vector space model“) zwei Gewichte. Dies ist der Fall, weil beim Vektor-Modell sowohl die Dokumente, als auch die Anfrage(n) durch (gewichtete) Vektoren repräsentiert werden. Diese Vektoren werden aus gewichteten Index-Termen gebildet. Außerdem gibt es jetzt nicht mehr nur zwei Gewichte, 0 und 1 wie beim booleschen Modell, sondern generell positive und nicht binäre.

Als erstes wollen wir die Index-Term-Gewichtung betrachten. Beim Vektor-Modell wird nämlich nicht nur berücksichtigt, ob ein Index-Term im Dokument enthalten ist, sondern es wird betrachtet wie oft der Index-Term enthalten ist (Termfrequenz – tf) und inwieweit dieser Index-Term auch in anderen Dokumenten der bekannten Dokumente vorkommt (inverse Dokument Frequenz – idf). Deshalb wird diese Berechnung der Index-Term-Gewichtung auch Berechnung nach der (tf * idf)-Formel genannt.
Wir wollen uns nun also zuerst die Formeln dazu betrachten und dann ein Beispiel durchrechnen.

N	Gesamtzahl der Dokumente im System
k_i	Indexterm
n_i	Anzahl der Dokumente, die Term k_i enthalten
$freq_{i,j}$	Anzahl der Erwähnungen von Term t_i in Dokument d_j
$max_l\ freq_{l,m}$	maximale Frequenz eines Terms in Dok. d_j
$f_{i,j}$	normalisierte Frequenz von Term k_i in Dokument d_j

$$f_{i,j} = \frac{freq_{i,j}}{\max_l freq_{l,j}} \qquad idf_i = \log \frac{N}{n_i} \qquad w_{i,j} = f_{i,j} \cdot \log \frac{N}{n_i}$$

Abbildung 14: Index-Term-Gewichtung (tf*idf)

Gesamtzahl der Dokumente im System: N=2048

Index-Terme:			
	„Öl“	in	128 Dokumenten
	„Preis“	in	16 Dokumenten
	„Alaska“	in	1024 Dokumenten

Beispiel-Dokument:		Öl	P.	A.
Termfreq		4	8	10
Normal. Termfreq	$f_{i,j} = \frac{freq_{i,j}}{\max_l freq_{l,j}} = (\frac{4}{10}, \frac{8}{10}, \frac{10}{10}) =$	0.4	0.8	1.0
Inverse Dokfreq	$idf_i = \log \frac{N}{n_i} = (_2\log \frac{2048}{128}, _2\log \frac{2048}{16}, _2\log \frac{2048}{1024}) = (_2\log 16, _2\log 128, _2\log 2) =$	4	7	1
	$w_{i,j} = f_{i,j} \cdot \log \frac{N}{n_i} = (0.4 \times 4, 0.8 \times 7, 1.0 \times 1) =$	1.6	**5.6**	1.0

Abbildung 15: Beispiel Berechnung der Term-Gewichte

Zur Erklärung des Beispiels ist zu sagen, dass was intuitiv vermutet wurde, nämlich dass der Index-Term der im Beispiel-Dokument häufig vorkommt (8 mal), aber in der Sammlung der bekannten Dokumente kaum vorkommt (16 aus 2048), die höchste Gewichtung zur Beschreibung des Dokuments erhalten sollte.

Wenn nun also die Index-Terme gewichtet sind, fehlt nur noch die Ranking-Funktion, welche die gefundenen Dokumente nach Relevanz ordnen soll.

Diese Formel (Cosinus-Formel) sieht folgendermaßen aus:

$$sim(d_j, q) = \frac{\vec{d_j} \bullet \vec{q}}{|\vec{d_j}| \times |\vec{q}|} = \frac{\sum_{i=1}^{t} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} \times \sqrt{\sum_{i=1}^{t} w_{i,j}^2}}$$

Abbildung 16: Ähnlichkeitsformel Vektor-Modell

Diese Formel betrachtet den Winkel zwischen dem Dokumentvektor und dem Anfragevektor. Zeigen die Vektoren genau in dieselbe Richtung, dann ist der Winkel zwischen den beiden Vektoren α = 0, was den Cosinus-Wert 1 (100% relevant) ergibt [siehe auch folgendes Beispiel in Abbildung 17]. Zeigen die beiden Vektoren nicht in die selbe Richtung, dann wird sich der Cosinus-Wert mit immer größer werdendem Winkel immer weiter der 0 (0% relevant) annähern – und so kann dann also eine Reihenfolge der Ergebnisse festgelegt werden.

			Term 1 *Öl*	Term 2 *Preis*	Term 3 *Alaska*
Dokumentvektor:	(4,8,0)		4	8	0
Vektor Query$_1$:	(1,2,0)		1	2	0
Vektor Query$_2$:	(3,6,0)		3	6	0

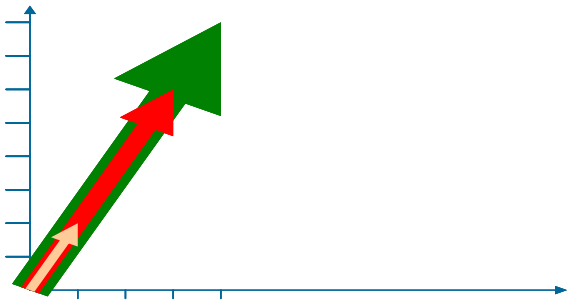


Abbildung 17: Beispiel Vektoren mit Winkel α=0

In diesem Fall wäre also der Winkel zwischen dem Dokumentenvektor und den beiden Anfragevektoren jeweils α=0, also wäre das Beispiel-Dokument für beide Anfragen jeweils 100% relevant – hätten also vermutlich jeweils eine obere Platzierung sicher.

Da wir nun die Grundlagen des klassischen Information-Retrieval kennen gelernt haben, können wir uns nun dem Information-Retrieval im Internet zuwenden.

Information Retrieval im Internet

Seit der Internetrevolution Anfang der 90er Jahre wächst das Internet mit unaufhaltsamer Geschwindigkeit. Die folgende Abbildung 18 soll diesen Sachverhalt verdeutlichen:

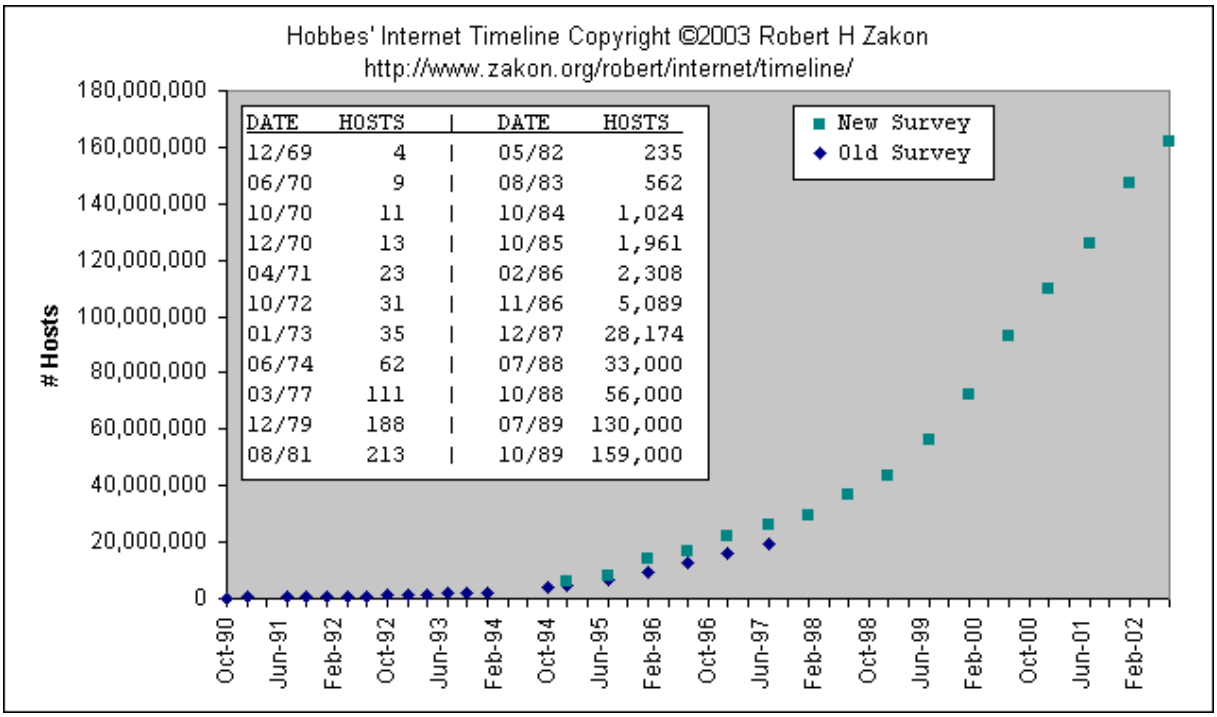


Abbildung 18: Internet-Wachstum

Mit der immer weiter wachsenden Anzahl von Servern und den darauf gelagerten Informationen in Form von Webseiten oder anderen Inhalten wird es immer schwierig einzelne Dokumente, geschweige denn Informationen zu finden. Die schiere Menge an Daten und deren Eigenschaften mach(t)en die Entwicklung von neuen Möglichkeiten der Informationsgewinnung nötig. Diese Thematik soll in den folgenden Abschnitten behandelt werden.

Die Besonderheit der Daten

Ein Hauptmerkmal der Daten ergibt sich direkt aus der Architektur des Internets. Das Internet an sich besteht aus einer riesigen Menge von Servern und Clients, die sich auf der ganzen Welt befinden, und einer sehr großen Menge von Verbindungen zwischen all diesen Computern (z.B. Transatlantikleitung, um Europa mit Amerika zu verbinden). Auch wenn mir einige Experten vielleicht widersprechen würden, kann man (unter gewissen Voraussetzungen) das Internet an sich als riesiges Verteiltes System betrachten – dabei sind eben nicht nur die Computer an sich über die ganze Welt verstreut, sondern eben mit ihnen auch die Daten, die sich auf deren Speichermedien befinden. Mit der Vielzahl an Computern einhergehend ist auch eine nahezu unendlich große Anzahl von möglichen Rechnerkonfigurationen und eine nicht so große Anzahl an installierten Betriebssystemen. Einhergehend mit verschiedenen Betriebssystemen sind aber auch viele verschiedene Datenspeicherungsformen. Da wir hier von einer Größenordnung von mehrere hundert Terrabyte (1 Terrabyte = 1024 Gigabyte) reden kann sich jeder leicht vorstellen, dass es eine nicht gerade triviale Aufgabe ist, genau diejenige Information zu finden, die man gerade sucht. Eben deshalb sind die „Werkzeuge“, namentlich die Suchmaschine und das

Webverzeichnis, die Tools die immer mehr an Bedeutung gewinnen. Wir wollen uns später in diesem Abschnitt beispielhaft die wohl zurzeit bekannteste Suchmaschine anschauen, nämlich Google [4]. Allerdings wollen wir uns zuerst die beiden Standardsuchmaschinen-Modelle anschauen, und erst dann auf die komplexe Google-Architektur eingehen.

Suchmaschinen / Metasuchmaschinen

Eine Suchmaschine ist ein intelligenter Werkzeug (auch als „Agent“ bekannt), das auf eine Benutzeranfrage alle dem System bekannten Webseiten durchforscht und alle relevanten Ergebnisse zurückliefert. Die Hauptvorteile einer Suchmaschine sind vor allem die schnelle Bearbeitung und die große Abdeckung. Es werden in sehr kurzer Zeit (gute Suchmaschinen wie Google bearbeiten eine Suchanfrage normalerweise unter einer Sekunde) sehr viele Webseiten durchforstet, was für den Benutzer durch Browsen unmöglich wäre. Außerdem ist zu erwähnen, dass es dem Benutzer auch kaum möglich wäre alle diese Seiten überhaupt von hand zu finden, da er wohl kaum Milliarden von Seiten kennt.

Eine Metasuchmaschine (z.B. www.metacrawler.com) ist ein Werkzeug, welches sich anderer Suchmaschinen bedient, um eine Benutzeranfrage zu bearbeiten. Der Benutzer gibt also seine Anfrage in ein Webinterface ein, und die Metasuchmaschine leitet diese Benutzeranfrage, in der jeweils gültigen Syntax, an „richtige“ Suchmaschinen weiter. Die Metasuchmaschine analysiert dann die Ergebnisse der verschiedenen eingesetzten Suchmaschinen, eliminiert Duplikate und sortiert die Ergebnisse nach eigenen Rangkriterien. Der Hauptvorteil einer Metasuchmaschine ist also, dass sie eine größere Abdeckung erreicht also eine einzelne Suchmaschine, und so unter Umständen ein wesentlich besseres Ergebnis liefern kann. Ein weiter Vorteil für den Benutzer ist es, dass er nur einmal eine Anfrage stellen muss, und sich so nur eine Anfrage-Syntax merken muss und außerdem nochmals Zeit einspart, weil er nicht dieselbe Anfrage in verschiedene Suchmaschinen eingeben muss.

Nun wollen wir uns exemplarisch die „Zentralisierte Architektur“ von Suchmaschinen betrachten und dann zur Betrachtung der Suchmaschine Google übergehen.

Zentralisierte Architektur

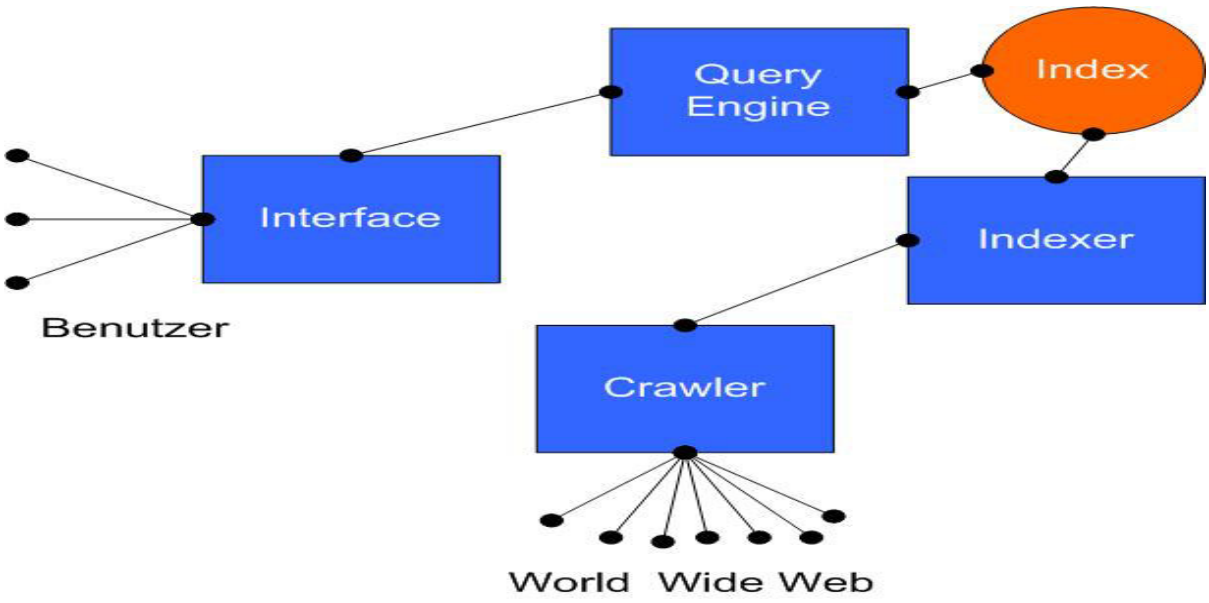


Abbildung 19: Zentralisierte Architektur

Wie in Abbildung 19 zu sehen ist, besteht eine Suchmaschine der zentralen Architektur schematisch betrachtet aus fünf Hauptkomponenten. Dem Benutzer-Interface, das dem Benutzer in geeigneter Form eine Suchoberfläche anbietet (z.B. durch eine graphische Repräsentation – Suchformular), der Query-Engine, welche die Anfrage des Benutzers verarbeitet, dem Index, dem Indexer der diesen Index aufbaut und einem oder mehreren Crawler(n), der/die das World Wide Web traversieren. Die Funktionen des Indexers haben wir ja bereits analysiert, und wir wissen auch wie so ein Index aussehen kann. Deshalb beschäftigen wir uns jetzt einmal mit dem Crawler.

Crawler / Crawling

Ein Crawler (auch: Robot, Spider, Wanderer, Walker, Knowbot) wird von seinem Betreiber, als meist von den Betreibern einer Suchmaschine, auf eine beliebige Startadresse im World Wide Web „losgelassen“. Grob beschrieben nimmt er diese erste Webseite und analysiert sie. Bei dieser Analyse erkennt der Crawler etwaige Links zu Unterseiten bzw. zu externen Seiten im World Wide Web und überprüft, ob diese ihm bereits bekannt sind. Wenn nicht, dann werden diese Seiten auf seine „Noch nicht erledigt“-Stapel („to-do-heap“) gelegt. Wenn die Seite fertig analysiert ist, wird sie dem Indexer übergeben, der sie automatisch indexiert. Dann wird die nächste Seite behandelt, und so geht das dann immer weiter und weiter, bis der Crawler entweder seinen ganzen Stapel abgearbeitet hat bzw. bis eine gewisse Anzahl von Seiten bearbeitet wurde bzw. bis eine gewisse Zeit verstrichen ist, dies hängt aber vom einzelnen Suchmaschinenbetreiber und damit Crawler-Betreiber ab. In der folgenden Abbildung ist die konzeptuelle Vorgehensweise des Crawlers „Scooter“ der Suchmaschine AltaVista dargestellt:

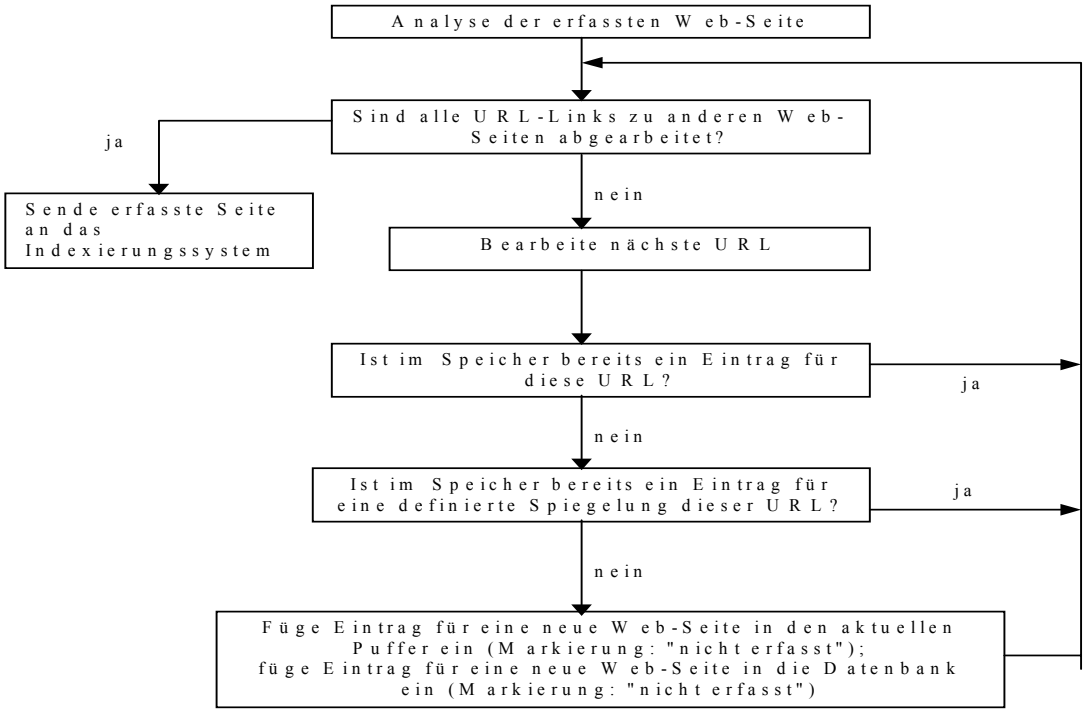


Abbildung 20: Crawler „Scooter“ der Suchmaschine AltaVista

Da dieses Crawling doch sehr hohe Belastungen der Server mit sich bringen kann, verwenden die verschiedenen Suchmaschinen doch gewisse Richtlinien, die z.B. eine maximale Anzahl von Verbindungen zu einem Webserver festlegen. Wir wollen uns nun aber einmal die beliebteste und wohl weltweit größte Suchmaschine „Google“ anschauen.

Beispiel Suchmaschine: Google

Die Suchmaschine Google ist heute wohl die größte, beliebteste und größte Suchmaschine im Internet. Zum Namen: abgeleitet von „googol“ was soviel bedeutet wie eine „1“ gefolgt von 100 „0“-en [10000000000000000000000000000000……00000].

Zuerst wollen wir einige Eckdaten betrachten:

1995 – Die „Erfinder“ von Google, Sergey Brin und Larry Page[1], treffen sich an der Universität Stanford und gegen Ende 1995 arbeiten sie zusammen.
1996-1997 – Sergey und Larry entwickeln „BackRub“, den legitimen Vorgänger von Google.
1998 – Anfang 1998 arbeiten die beiden weiter an der Verbesserung ihrer Suchmaschine. Am 7. September 1998 gründen sie mit geliehenem Geld von Yahoo!-Mitbegründer und Freunden die Google Inc. Sie ziehen mit ihren Server aus ihrem Unischlafsaal in die Garage eines Freundes in Menlo Park, Kalifornien. Google beantwortet **10 000 Anfragen pro Tag**.
1999 – Google zieht in ihr neues Hauptquartier nach Palo Alto, Kalifornien. Google beantwortet mittlerweile **500 000 Anfragen pro Tag**. Gegen Ende 1999 zieht Google erneut um, und zwar in ihr neues Hauptquartier in Mountain View, Kalifornien.
2000 – Das Magazin Yahoo! Internet Life ernennt Google zur Suchmaschine des Jahres. Google ist mittlerweile die größte Suchmaschine mit einer Milliarde indizierter Seiten. Google beantwortet zum jetzigen Zeitpunkt bis zu **18 000 000 Anfragen pro Tag**. Gegen Dezember 2000 beantwortet Google **60 000 000 Anfragen pro Tag**.
2001 – Google beantwortet über **100 000 000 Anfragen pro Tag** und hat einen Index von 3 Milliarden Seiten.
2002 – Google gewinnt mehrere Awards bei der 2001 Search Engine Watch Preisverleihung
…
…
Heute – Google kann dank des größten kommerziellen Linux Clusters der Welt, der aus über **10 000 Server**n besteht, mehr als 200 Millionen Anfragen pro Tag bewältigen.

Nachdem wir nun mit der Geschichte und der technischen Ausstattung von Google vertraut sind, können wir uns der Google-Architektur zuwenden. Diese ist in Abbildung 22, auf der folgenden Seite, dargestellt. Was man schon nach kurzem Hinsehen deutlich erkennen kann ist, dass es sich dabei um eine höchstkomplizierte Architektur handelt, die wir nun Stück für Stück analysieren und verstehen wollen.

[1] Bilder der beiden Erfinder sind im Anhang zu finden.

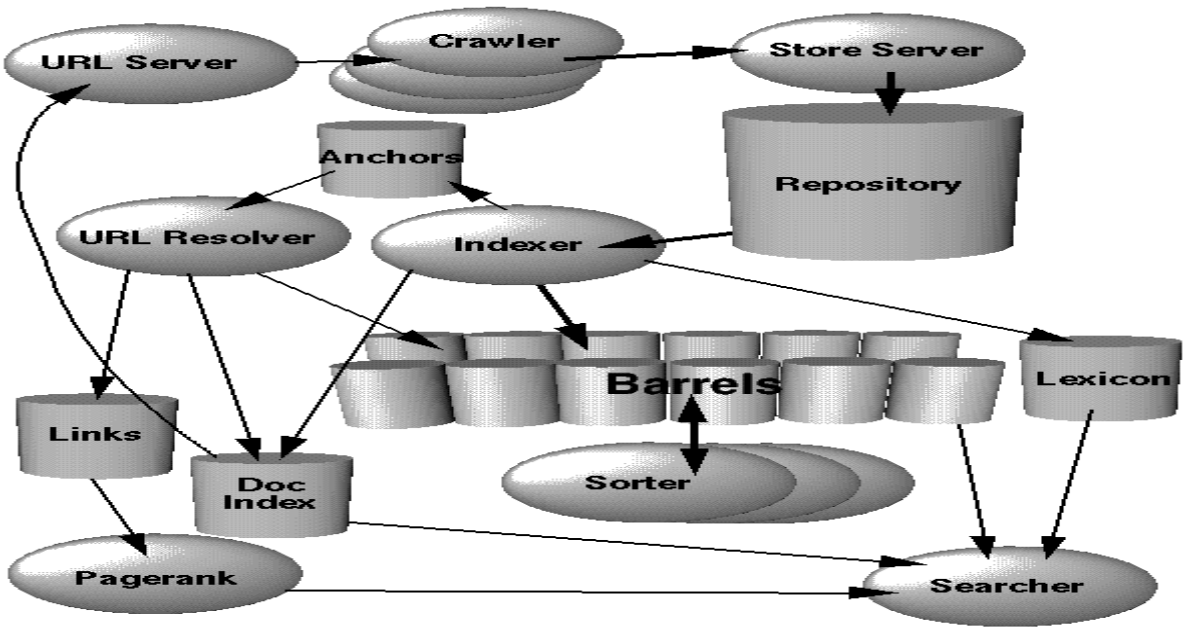


Abbildung 21: Google-Architektur [5]

Konzeptueller Ablauf / Funktionsweise der einzelnen Einheiten

***URL* Server:** Der *URL*-Server sendet eine Liste der zu bearbeitenden Webseiten an die Crawler.

Crawler: Die Crawler durchwandern die Internetseiten, die sie vom *URL*-Server bekommen haben, und laden diese herunter und senden sie an den Store Server. Google setzt normalerweise drei Crawler gleichzeitig ein.

Store Server: Der Store Server ist für die Speicherung der von den Crawlern heruntergeladenen Seiten verantwortlich. Er komprimiert diese in einem Repository und weißt jeder Webseite eine eindeutige Identifikationsnummer (docID) zu.

Repository: Das Repository enthält den kompletten HTML-Code jeder bereits vom Crawler besuchten Webseite. Um die Datenmenge einigermaßen überschaubar zu halten, werden alle diese Seiten komprimiert (mit zlib: Kompressionsfaktor etwa 3:1). Im Repository werden die Dokumente nacheinander mit der docID, der Länge und ihrer *URL* gespeichert.

Indexfunktion: Diese Funktion wird vom Indexer und vom Sorter ausgeführt.

Indexer: Der Indexer von Google erfüllt eine Vielzahl von Aufgaben.

1. Lesen des Repository und Dekomprimierung:
   Jedes Dokument wird in eine Menge von Wortvorkommenshäufigkeiten („hits“) konvertiert. Jeder „hit“ protokolliert das Wort, die Position innerhalb des Dokuments und eine Näherung der Fontgröße und die Groß- bzw. Kleinschreibung. Der Indexer verteilt alle „hits“ auf eine Menge von Fässern („barrels“) und erzeugt einen teilweise sortierten Vorwärts-Index („forward index“).
2. Parsen aller Links aus jeder Webseite:
   In einem Anchor File werden die Informationen zu einem Link (Linktext, Ausgangs- und Endpunkt des Links) gespeichert.

***URL* Resolver:** Der *URL* Resolver liest das Anchor File und konvertiert die relativen *URL*s in absolute *URL*s und weist gleichzeitig eine docID zu. Er generiert eine Datenbank von Links aus Paaren von docIDs.

Sorter: Der Sorter nimmt die nach docID sortierten „barrels" und sortiert diese nach der wordID für den Rückwärts-Index („inverted index"). Der Sorter erzeugt eine Liste von wordIDs und Offsets im invertierten Index.

Dump Lexikon: Dieses Programm nimmt die vom Sorter erzeugte Liste und das vom Indexer erzeugte Lexikon und generiert ein neues Lexikon für den Searcher.

Searcher: Der Searcher läuft auf einem Webserver und benutzt das Lexikon zusammen mit dem invertierten Index und den PageRanks zur Beantwortung von Anfragen.

Nachdem wir nun das Backend von Google kennen, wollen wir uns nun einmal das Frontend betrachten, also die Benutzerschnittstelle, und einige Beispielanfragen an das Google-System stellen.

Google-Frontend

Um Google benutzen zu können benötigt man eine Webbrowser und eine Internetverbindung. Der schematische Ablauf einer Anfrage an Google ist in der folgenden Abbildung 22 zu sehen:

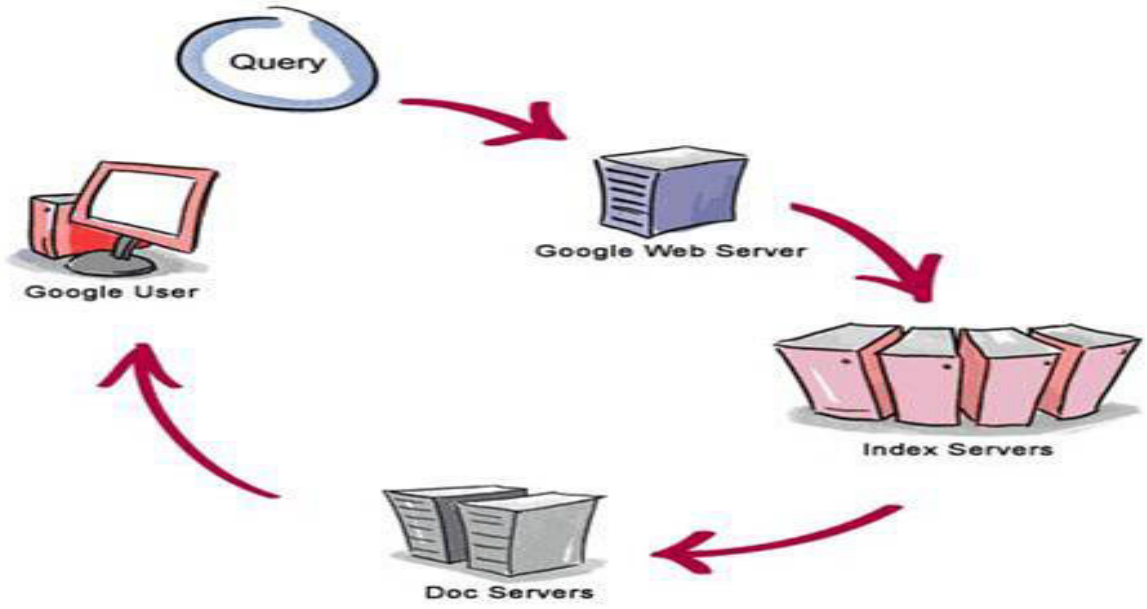


Abbildung 22: „Das Leben einer Google-Anfrage" [6]

1. Der Webserver sendet die Anfrage an die Index-Server, diese schauen im Index nach und erkennen welche Seite die Wörter enthält.
2. Die Anfrage „wandert" zu den Dokument-Servern, welche die Dokumente wirklich raussuchen. „Snippets" werden generiert – zur Beschreibung des Ergebnisses. Snippets sind kurze Ausschnitte der Webseite, in denen das oder die Suchwörter fett markiert werden.
3. Die Suchergebnisse werden zurückgeliefert – normalerweise im Bruchteil einer Sekunde.
Nun wollen wir selbst einmal einige Beispiel-Anfragen an Google stellen.

Um Anfragen an Google stellen zu können benötigt man, wie bereits erwähnt, einen Webbrowser und einen Internetzugang. Google ist unter http://www.google.de zu erreichen und die graphische Benutzeroberfläche sieht folgendermaßen aus:

Abbildung 23: Google Deutschland Frontend

Besonders zu erwähnen ist dabei noch die Schaltfläche „Auf gut Glück!“. Diese Schaltfläche führt den Benutzer direkt auf die Seite, die von Google in Bezug auf die Anfrage als bestes Ergebnis eingestuft wurde. Nun wollen wir einmal nach „mums“ suchen, natürlich mit der Erwartung möglichst etwas zu unserem Studienprojekt MUMS zu finden:

Abbildung 24: Beispiel-Anfrage 1

Wie in Abbildung 24 zu erkennen ist, liefert diese doch sehr allgemeine Anfrage jedoch 679 000 möglicherweise relevante Seiten. Um weniger Ergebnisse zu erhalten versuchen wir nun unsere Suchanfrage weiter zu spezifizieren, und schränken die Suche insofern ein, dass wir nur Seiten auf Deutsch durchsuchen wollen (siehe Abbildung 25).

Abbildung 25: Beispiel-Anfrage 2

Auf diese modifizierte Anfrage erhalten wir immer noch 2270 relevante Seiten. Allerdings wäre eine Seite unseres Studienprojekts bereits als drittes Ergebnis zu finden. Trotzdem wollen wir versuchen die Anfrage noch weiter einzugrenzen. Dazu benutzen wir die erweiterte Suche von Google, und fügen einen weiteren Suchbegriff hinzu (Studienprojekt). Außerdem begrenzen wir die Suche auf Seiten, die aus Deutschland kommen, also auf den Domainbereich „ .de “ (Abbildung 26).

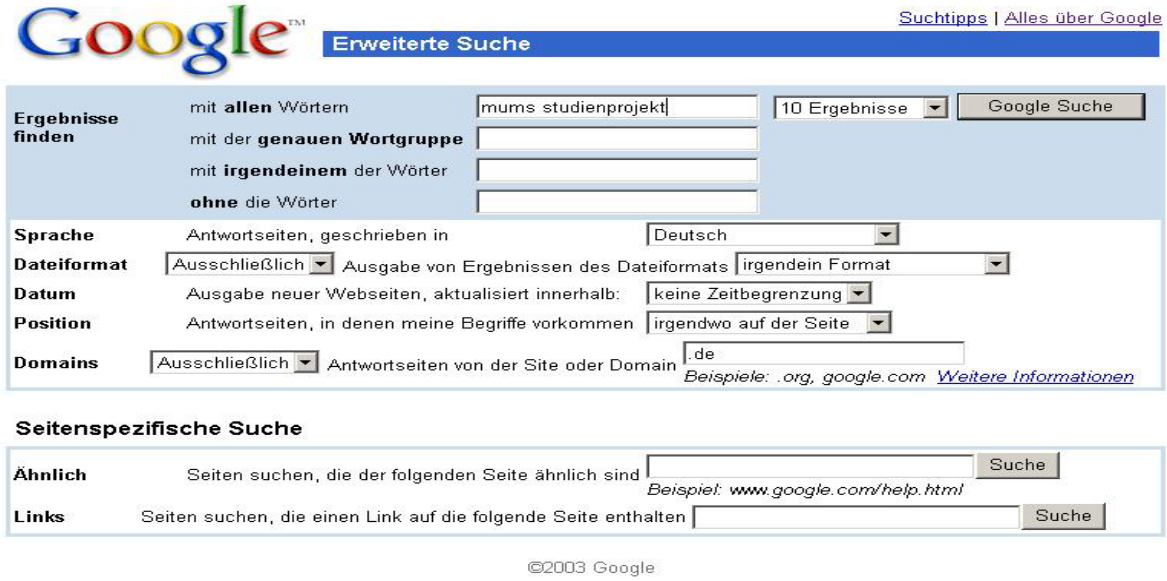


Abbildung 26: Beispiel-Anfrage 3 (erweitert)

Daraufhin erhalten wir das gewünschte Ergebnis (siehe Abbildung 27):

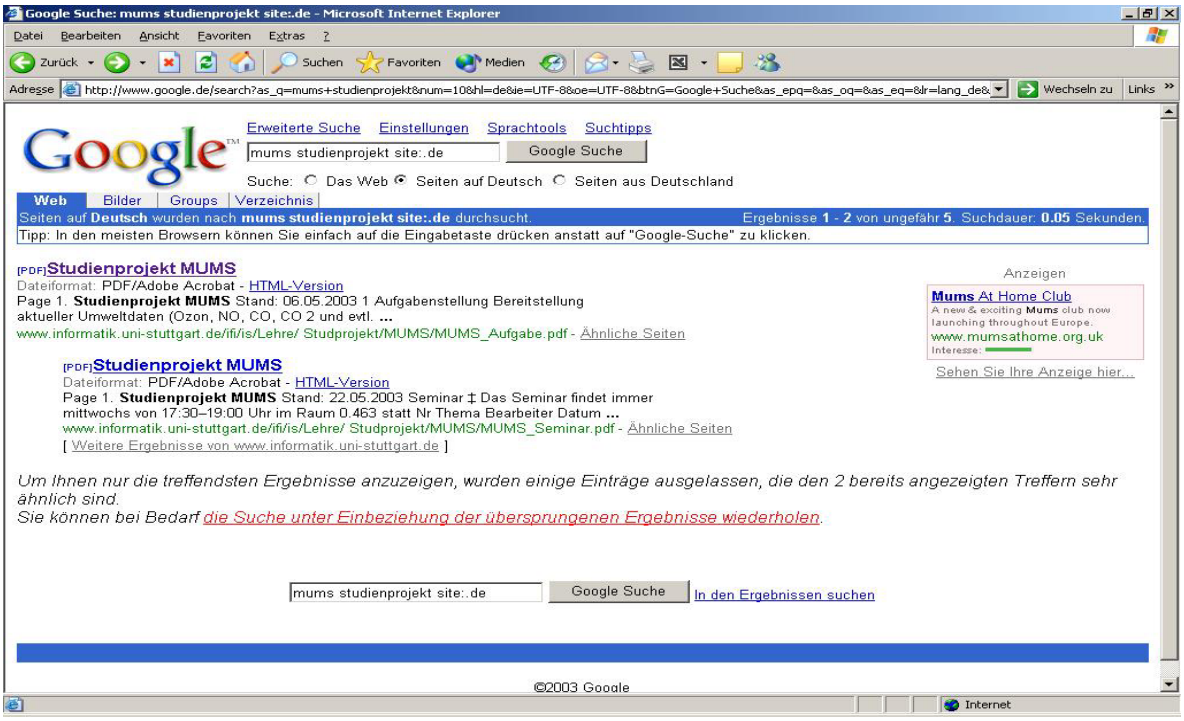


Abbildung 27: Ergebnis der Beispiel-Anfrage 3

Es werden wirklich nur noch die Seiten unseres Studienprojekts gefunden. Daraus können wir ableiten, dass ein „gutes“ bzw. besseres Suchergebnis dadurch erreicht werden kann, dass der Benutzer das System, durch die Angabe von sinnvollen Suchworten und etwaigen Zusatzparametern (z.B. Domain-Bereich), unterstützt.

Nun wollen wir aber das Prinzip der Webverzeichnisse näher betrachten:

Webverzeichnisse

Im heutigen Internet, ist es mittlerweile schwierig ein wirklich alleinstehendes Webverzeichnis zu finden. Die klassischen Webverzeichnisse, wie z.B. Yahoo! Oder Web.de, haben sich mittlerweile von reinen Webverzeichnissen zu „Portalen“ entwickelt, also zu Webseiten, die einen idealen Einstiegspunkt ins Internet darstellen und außerdem den Betreibern bares Geld einbringen (z.B. durch Werbeeinnahmen etc). Diese Portale bestehen zumeist aus der klassischen Komponente des Webverzeichnisses, aber sind dahingehen erweitert worden, dass allerlei andere Dinge hinzugefügt wurden, namentlich wären das in der Regel: eine Suchmaschine (entweder eine eigene oder eine „hinzugekaufte“, also externe), Kommunikationsdienste (wie z.B. E-Mail Service) und andere Webanwendungen, wie z.B. Shops oder Spiele.

In einem Webverzeichnis sind Webseiten, die nach deren Themen und Inhalten in entsprechende Kategorien unterteilt wurden, enthalten. Wenn man sich nun also Informationen z.B. über Computer Hardware verschaffen will, würde man in die Kategorie Computer gehen und dort in die entsprechenden Unterkategorie (z.B. Hardware). Der Hauptunterschied zur Suchmaschine besteht darin, dass die Seiten, die darin enthalten sind manuell, also von Menschen, hinzugefügt wurden. Daraus abgeleitet ergeben sich folgende Erkenntnisse:

1.) Webverzeichnisse decken einen wesentlich geringeren Teil des Internets ab als Suchmaschinen, da das Hinzufügen und das Einordnen von Webseiten (i.d.R.) von Menschen übernommen wird. Ein kleiner Vergleich der bekannten Seiten zwischen Web.de [~ 400 000 Seiten] und Google.de [~ 3 000 000 000 Seiten] verdeutlicht dies.
2.) Die Informationen, die auf den im Webverzeichnis enthaltenen Seiten zu finden sind, sind meist sehr nützlich, da heutige Webverzeichnisse nicht jede beliebige Seite aufnehmen, sondern diese doch vorher überprüft werden.
3.) Es kann sein, dass Seiten Informationen enthalten, so dass die Einordnung in mehrere Kategorien sinnvoll wäre, was aber unter Umständen nicht erfolgt. Es kann aber auch durchaus vorkommen, dass Seiten falsch bzw. mehrfach klassifiziert werden.
4.) Außerdem erscheint es nur logisch, dass wenn eine Klassifikation durch Menschen erfolgt, eben diese Klassifikation je nach Bearbeiter und dessen Tagesform sehr unterschiedlich sein kann.

Nun wollen wir uns kurz die Oberfläche von Yahoo! und Web.de anschauen:

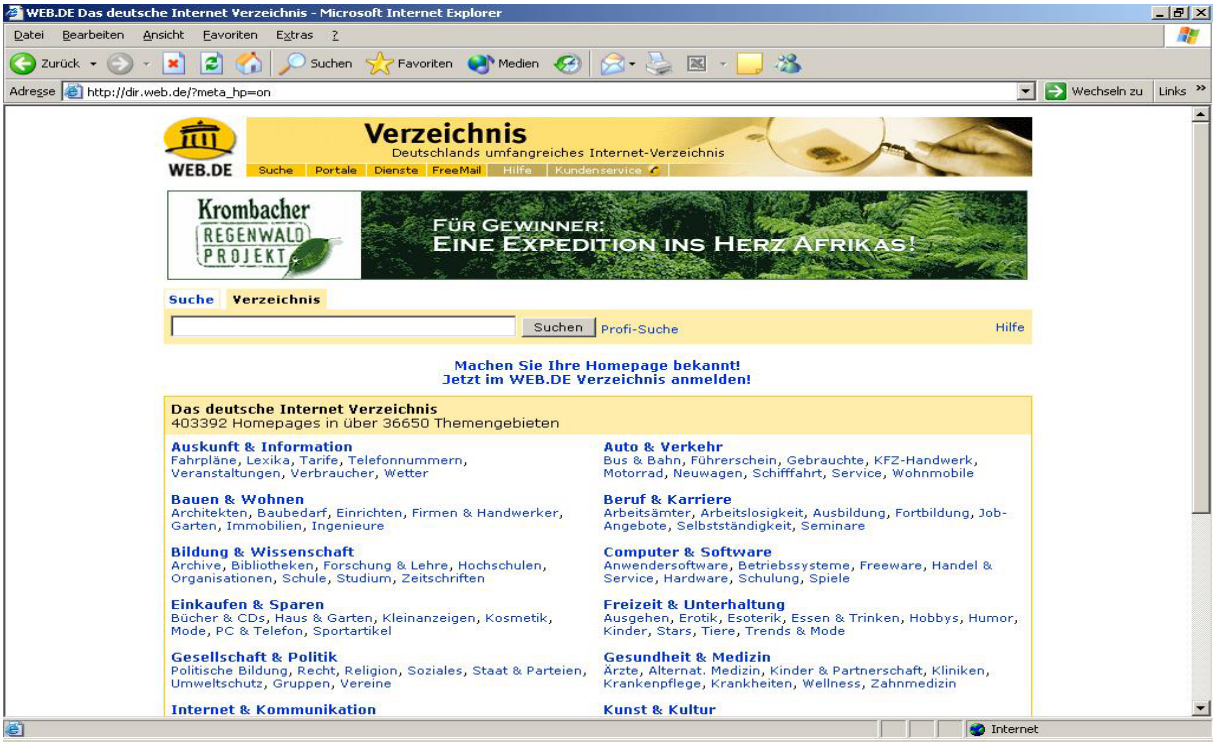


Abbildung 28: Oberfläche Webverzeichnis Web.de

Abbildung 29: Oberfläche Webverzeichnis Yahoo!

Fazit / Ausblick

Kennen Sie das alte Sprichwort: „Die Nadel im Heuhaufen suchen?“
Nach der Betrachtung der Suchmaschinen und Webverzeichnisse im Internet bleibt ein Fazit zu ziehen: Die Suchmaschinen liefern immer noch zuviel Heu mit der Nadel mit, und Webverzeichnisse besitzen meist überhaupt nicht die nötige Tiefe, um die Nadel überhaupt zu finden. Es bleibt also noch viel zu tun, und wir dürfen gespannt sein, was die Zukunft mit sich bringen wird. Einige Ansatzpunkte für Verbesserungen wären z.B. die Einführung einer intuitiveren Benutzeranbindung (z.B. direktes Anfrage per Spracheingabe), einer intelligenteren Suchmaschine (die z.B. (An)Fragen des Benutzers intelligent verstehen kann und so in der Lage ist, wirklich relevante Antworten zu geben) und vieles mehr.

Zum Abschluss wollen wir nun noch die Relevanz des Information Retrievals für unser Studienprojekt MUMS betrachten.

Relevanz für MUMS

Nach der ausführlichen Betrachtung einiger Aspekt des Information Retrievals bzw. des Information Retrievals im Internet, stellt sich nun die Frage was von alledem nun eigentlich wirklich relevant ist für das Studienprojekt MUMS?

Wir wollen einige Punkte betrachten, die mit dem Information Retrieval in Verbindung gebracht werden können und für das Studienprojekt MUMS interessant sein könnten – allerdings nur unter der Vorraussetzung, dass die fertige Software überhaupt dem Benutzer im Internet zugänglich gemacht werden soll.

Browsing

Um dem Benutzer des MUMS-Systems zu erleichtern den Retrieval-Prozess durchzuführen, also gezielt seine gewünschten Informationen zu finden, es ist nötig, dass die Benutzerbedürfnisse genau analysiert und erkannt werden.
Der Hauptgesichtspunkt ist also, dass die Seite(n) einer klaren Gliederung unterliegen sollten, damit sich der Benutzer intuitiv zurecht findet. Im Einzelnen bedeutet das, dass die Seiten nicht zu tief geschachtelt sein sollten und wenn möglich durch ein globales Navigationsinstrument ergänzt werden sollten (Sitemap).

Lokale „Suchmaschine“ / Suchmaschinen-Eintrag

Es ist zu überlegen, ob es auf den MUMS-Seiten eine lokal Suchmaschine geben sollte. Dadurch könnte ein Benutzer des Systems, falls er es eilig hat oder sich nicht auf den Seiten zurecht findet, durch Eingabe von geeigneten Suchwörtern (z.B. „aktueller Ozonwert Ludwigsburg“) direkt auf die Seite mit den aktuellen Ozonwerten an der Station Ludwigsburg gelangen. Wie aufwendig die Implementierung bzw. Integration solch einer lokalen Suchroutine ist, bleibt natürlich zu evaluieren.

Name	Title	First lines	First page	Meta	"beliefert von"
altavista	Ja	ja	Ja	ja	looksmart, overture
MSN	?	?	?	?	looksmart
Lycos	Ja	ja	Ja	ja	direct hit
Overture	pay	pay	Pay	pay	inktomi
Yahoo!*	?	?	?	?	Google (suche), Overture
alltheweb	Ja	ja	Ja	ja	fast search, overture
Google	Ja	ja	Ja	ja	open directory (verzeichnis)
excite	?	?	?	?	overture, inktomi, fast, looksmart, ask jeeves
looksmart*	Ja	?	?	?	?
Ask jeeves	Ja	?	?	?	google, teoma
dmoz*	Ja	?	?	ja	google
netscape	Ja	?	?	ja	dmoz, google
aol	Ja	?	?	ja	google
inktomi	Ja	?	?	ja	?
fast	Ja	ja	Ja	ja	overture
teoma	Ja	?	?	?	google, ask jeeves
*Webverzeichnisse					

Ein weiterer relevanter Aspekt ist es, MUMS in eine bzw. mehrere Suchmaschine(n) einzutragen und zwar mit geeigneten Stichwörtern (siehe hierzu auch den nächsten Punkt Metadaten). Außerdem wäre es sinnvoll die Seite(n) mit anderen artverwandten Seiten zu verlinken, um auch so ein höheres Ranking in den verschiedenen Suchmaschinen zu erreichen. Welche Eintragungen in welche Suchmaschine besonders wichtig sind, muss noch genauer evaluiert werden. Als Grundlage der Evaluation können aber meine bereits getätigten Recherchen verwendet werden:

Metainformationen / Metadata

Es ist wünschenswert, die MUMS-Seiten mit geeigneten Metadaten zu „spicken", um so eine bessere Klassifizierung und Bewertung durch Suchmaschinen zu erreichen. Welche Metainformationen für unsere Domäne sinnvoll sind bleibt zu evaluieren. Ein geeignetes Format dazu wäre z.B. die Verwendung des „Dublin Core"-Formats innerhalb des Ressource Description Framework (RDF) [3].

Webkataloge / Webverzeichnisse

Durchaus sinnvoll scheint es auch, die MUMS-Seite(n) in ein oder mehrere Webverzeichnis(se) einzutragen. In welche Webverzeichnisse wir MUMS eintragen sollten, muss noch genauer evaluiert werden, allerdings können auch hier meine bisher recherchierten Erkenntnisse verwandt werden (siehe weiter oben bei Suchmaschinen).

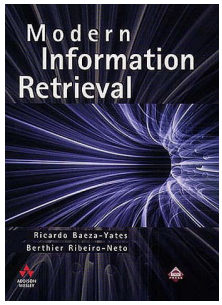
Anhang

Abbildungsverzeichnis

Abbildung 1: Browsing – Retrieval (Taxonomie der IR-Modelle) 4
Abbildung 2: Dokumentrepräsentation 5
Abbildung 3: Textoperationen – vom Volltext zu einer Menge von Schlüsselwörtern 7
Abbildung 4: Beispiel Vektordarstellung 8
Abbildung 5: Beispiel RückwartsIndex (Inverted Index) 9
Abbildung 6: Definition Index-Term-Relevanz 10
Abbildung 7: Der Retrieval Prozess 10
Abbildung 8: Definition IR-Modell 11
Abbildung 9: George Boole 12
Abbildung 10: Definition boolesches Modell 13
Abbildung 11: Beispiel-Anfrage 13
Abbildung 12: Anfrage in Disjunktiver Normalform (DNF) 14
Abbildung 13: Definition Vektormodell 14
Abbildung 14: Index-Term-Gewichtung (tf*idf) 15
Abbildung 15: Beispiel Berechnung der Term-Gewichte 15
Abbildung 16: Ähnlichkeitsformel Vektor-Modell 16
Abbildung 17: Beispiel Vektoren mit Winkel α=0 16
Abbildung 18: Internet-Wachstum 17
Abbildung 19: Zentralisierte Architektur 18
Abbildung 20: Crawler „Scooter" der Suchmaschine AltaVista 19
Abbildung 21: Google-Architektur [5] 21
Abbildung 22: „Das Leben einer Google-Anfrage" [6] 22

Abbildung 23: Google Deutschland Frontend 23
Abbildung 24: Beispiel-Anfrage 1 23
Abbildung 25: Beispiel-Anfrage 2 24
Abbildung 26: Beispiel-Anfrage 3 (erweitert) 24
Abbildung 27: Ergebnis der Beispiel-Anfrage 3 25
Abbildung 28: Oberfläche Webverzeichnis Web.de 26
Abbildung 29: Oberfläche Webverzeichnis Yahoo! 27

Literaturhinweise



- [2] Buch: Modern Information Retrieval
  Autoren: Ricardo Baeza-Yates, Berthier Ribeiro-Neto
  Verlag: Addison Wesley Publishing Company
  Ausgabe: Mai 1999
  ISBN: 020139829X
  Auch zu finden in der Fakultätsbibliothek

Interessante Links

- [4] http://www.google.de | http://www.google.com wohl bekannteste und beliebteste Suchmaschine im Internet
- http://www.google.com/corporate/history.html Geschichte von Google
- http://www.yahoo.com eines der ältesten und besten Webverzeichnisse
- http://www.dmoz.org Open Directory Project
- [3] http://www.w3.org/RDF/ Resource Description Framework
- [5] http://www-db.stanford.edu/~backrub/google.html Seite der Google „Erfinder" mit Erklärung des PageRank-Algorithmus etc.
- http://webworkshop.net/pagerank_calculator.php3 Google PageRank-Calculator
- [6] http://www.google.com/press/query.html "Life of a query"
- http://www.web.de

[1]  Erfinder von Google: Lawrence „Larry" Page, Sergey Brin (links)