Diplomarbeit

Michael Maretzke

Evaluation einer Suchmaschine und Integration in eine existierende Internet-Shoppingapplikation

Erstprüfer:	Prof. Dr. Lore Kern-Bausch
Abgabesemester:	Sommersemester 2000
Aufgabenstellende Firma:	Conrad.com AG Klaus-Conrad-Straße 1 92240 Hirschau

4. Suchmaschinen im Allgemeinen

Bevor im nächsten Kapitel mögliche Lösungen für die beschriebene Problematik der Conrad.com AG behandelt werden, soll dieses Kapitel Informationen allgemeiner Natur rund um Suchmaschinen und deren verwendeten Hintergrundtechnologien bieten. Hierbei wird der Fokus nicht auf unendliche Tiefe gelegt, sondern nur ein breiter Überblick über die einzelnen Technologien gegeben. Für tiefergreifendere Informationen möge der geneigten Leser im Literaturverzeichnis nachschlagen.

4.1. Überblick über Suchmaschinen

Nach einem kurzen Ausflug in die Geschichte der Suchmaschinen und den möglichen Einsatzbereichen folgen technische Möglichkeiten der Suche auf einem Computer. Dort werden Vor- und Nachteile der einzelnen Varianten aufgezeigt. Anschließend kommt ein Überblick über Arten von Suchmaschinen.

4.1.1 Geschichte und Einsatzbereiche

Die Geburtstunde der modernen Suchmaschinen ist nahe an der Geburtstunde der eigentlichen elektronischen Datenverarbeitung und der Möglichkeit Massendaten zu speichern. Mit dem Anwachsen von Datenmengen kam auch immer öfters das Problem auf, Daten schnell und zuverlässig zu finden. Anwachsende Quellcodedateien und auch Textdateien verlangten bald nach effektiven Volltextsuchmöglichkeiten. Von der Suche innerhalb einer Datei erfolgte ein Übergang von der Suche innerhalb von Dateien hin zur Suche nach Dateien und die Untersuchung deren Inhalts. Hier wurde bald die dateisystembasierte Suche durch indexunterstützte Suchmethoden ersetzt. Langsam nähern wir uns den Konzepten heutiger Suchmaschinen an. Heutzutage basieren die meisten Suchmaschinen auf großen Indexdatenbanken, die nach bestimmten Kriterien speziell für die performante Suche optimiert sind.
Suchmaschinen werden heutzutage vor allem dort eingesetzt, wo dynamisch dezentrale Inhalte verwaltet werden und schnell aufgefunden werden sollen. Dies ist beispielsweise im Internet- oder Intranetumfeld der Fall. Suchmaschinen, die dort eingesetzt werden, gleichen fortlaufend die tatsächlich gespeicherten Dokumente mit den indexierten Dokumenten ab. Änderungen an Dokumenten oder neue Dokumente werden so schnell und sauber erfasst und der Datenbestand der Suchmaschine fortlaufend aktualisiert. Mit anwachsender Menge an Dokumenten steigt auch das Alter des Indexdatenbestandes der Suchmaschine. Die Suchmaschinen haben dann Probleme die anwachsende Datenmenge zu indexieren. Dieses Problem tritt ganz extrem im Internet und in großen Intranet-Netzen auf.
Andere Einsatzgebiete von Suchmaschinen sind sogenannte OPACs (Online Public Access Catalogues). Das sind z.B. Datenbestände von Bibliotheken oder Verzeichnisse von Magazinen. Hier zeichnet sich die Anwendung durch extrem viele Suchabfragen auf einem statischen Datenbestand aus. Suchmaschinen, die auf derart statischen Datenbeständen agieren sind anders konzipiert, als Suchmaschinen, die mit der Schnelllebigkeit des Internets umgehen müssen. [SUF99a]

4.1.2 Grundlegende Suchkonzepte auf einem Computer

Mit dem Fortschreiten der Geschichte wurden natürlich auch unterschiedliche Konzepte der Suche auf einem Computersystem durchlaufen.

Nach der Volltextsuche innerhalb einer Datei wurde die Suche auf das Dateisystem des Computers ausgedehnt. Speicherressourcen waren plötzlich nicht mehr so knapp und Dateien und darin gespeicherte Daten konnten leichter abhanden geraten. Die Suche im Dateisystem zeichnet sich im Wesentlichen dadurch aus, dass keinerlei Vorbereitungen für die Suche an sich notwendig sind. Die Suchmaschine muss keine Indexdatei oder –datenbank aktualisieren bzw. aufbauen. Die Daten sind immer aktuell, weil eben keine Indexdatei existiert. Neben diesen Vorteilen fallen aber vor allem die starke Belastung für das Gesamtsystem auf. Für jede Suchabfrage müssen alle Dateien einzeln angesprochen und evtl. auch noch durchsucht werden. Logischerweise wird die Suche auf dem Dateisystem je nach Rechner und Zahl der zu durchsuchenden Dateien immer langsamer. Außerdem hat die Suche im Dateisystem auch noch das Problem, dass Mehrbenutzertauglichkeit nur in gewissem Umfang gegeben ist. Wollen beispielsweise zwei Suchprozesse die gleiche Datei durchsuchen, so wird einer der beiden Prozesse blockiert.
Nach der Suche in vielen Dateien auf Dateisystemebene hat sich die indexbasierte Suche etabliert. Hier wird die Suche in einer einzigen Datei mit der Suche im Dateisystem kombiniert. Eine Indexsoftware speichert in einer (Index-)Datei Verweise auf die zu durchsuchenden Dateien ab. Die Indexsoftware sammelt Informationen über und in den Dateien auf und schreibt diese kombiniert in eine Indexdatei. Die Suchsoftware durchsucht die Indexdatei hinsichtlich der Relevanz des eingegebenen Suchtextes mit dem Index. Bei Treffern verweist die Suchsoftware auf die eigentliche (Quell-)Datei. Vorteilhaft wirkt sich auf die Systembelastung aus, dass lediglich eine Datei durchsucht werden muss. Auch wird die Suche schneller sein, als bei der reinen dateisystembasierten Suche. Schlechter hingegen ist die Erhöhung der Komplexität der Suchsoftware. Jetzt reichen nicht allein die Hilfsmittel des Betriebssystem zur Dateisystemverwaltung für die Suche. Die Funktionalität des Suchens und Indexierens wird in zwei getrennte Prozesse gespaltet. Auch der Ressourcenverbrauch für die Indexdatei kann ziemlich hohe Ausmaße erreichen. Je nach Detailierungs- und Kompressionsgrad kann die Indexdatei gleich groß, oder sogar noch größer werden, als die Summe der eigentlich zu durchsuchenden Dateien. Bei hohen Kompressionen kann der Datenbestand in der Indexdatei aber wieder bis auf ca. 4-6% der ursprünglichen Größe reduziert werden. Problematisch an der Indexdatei ist die Tatsache, dass der Index immer veraltet ist. Die Indexsoftware muss den Datenbestand periodisch nach Änderungen und Neuerungen durchsuchen und den Index aktualisieren. Bei relativ kleinen Datenbeständen kann die Aktualisierung nahe Realtime erfolgen, aber bei großen Datenbeständen (wie im Internet) kann die Überarbeitung teilweise bis zu mehreren Wochen dauern. Diese Problematik wächst mit der Zahl der Dateien und der Häufigkeit der Änderungen an diesen Dateien.
Problematisch an der Speicherung von Daten in der Indexdatei ist nun noch der Verlust der Semantik der Daten. In der Indexdatei werden die Daten unstrukturiert gespeichert. Um die Strukturierung der Indexinformationen nicht zu verlieren, wurde die Indexdatei im Laufe der Zeit in eine Datenbank verlegt. In der Datenbank muss der Indexdatenberg auf einzelne Datenattribute verteilt werden, um überhaupt sinnvoll abgespeichert zu werden. In der strukturierten Indexdatenbank kann bei der Suche jetzt zwischen der Berufsbezeichnung „Müller“ und dem Nachnamen „Müller“ unterschieden werden. Bei der Suche in der Indexdatei geht diese semantische Unterscheidung verloren. Ebenso

von Vorteil ist, dass eine Datenbank speziell auf Datensuch– und –einfügeoperationen optimiert ist. Die Datenbank an sich stellt also die derzeit bestmögliche Performance bei Suchoperationen zur Verfügung. Aus diesem Grund operieren die meisten Suchmaschinen heutzutage auf Datenbankbasis. [SUF99a]

Suche im Dateisystem	
die Funktionen des Betriebssystem zur Verwaltung von Dateien werden ausgenutzt;	
Vorteile	Nachteile
Keine Vorbereitungen notwendig (keine Indexdatei notwendig)	**jede einzelne Datei muss einzeln adressiert werden**
Daten sind immer aktuell	**starke Belastung für das Gesamtsystem**
	Suche wird je nach Rechner und Zahl der Dateien langsamer
	mehrere Sucher greifen auf eine Datei zu (zwangsweise Sequentialisierung)

Tabelle 4-1 Vor- und Nachteile der Suche im Dateisystem

Suche in einer Indexdatei	
Kombination der Suche in einer einzigen Datei und der Suche im Dateisystem; Indexsoftware sammelt alle Informationen über und in den Dateien und schreibt alles in eine (Index-)Datei; die Suchsoftware durchsucht die Indexdatei und verweist auf die eigentliche (Quell-)Datei	
Vorteile	Nachteile
Höhere Suchgeschwindigkeit (nur eine Datei durchsuchen)	**Suchsoftware wird komplexer (Suchsoftware und Indexsoftware nötig)**
Geringere Systembelastung	**große Indexdatei (enthält alle Informationen der indexierten Dateien)**
	Indexsoftware muss periodisch gestartet werden
	Index ist immer veraltet (Problem wächst mit der Zahl der Dateien und der Häufigkeit der Änderungen)

Tabelle 4-2 Vor- und Nachteile der Suche in einer unstrukturierten Indexdatei

Suche in einer Indexdatenbank	
Strukturierung der Indexinformationen; kein Verlust der Semantik der einzelnen Indexinformationen; optimierte Suche möglich; kommerzielle, spezialisierte Produkte einsetzbar; die Vor- und Nachteile sind denen der Suche in einer Indexdatei ähnlich	
Vorteile	Nachteile
Semantik der einzelnen Informationen bleibt erhalten (Name „Müller“ != Beruf „Müller“)	**eigene spezialisierte Software für Indexverwaltung (Datenbanksystem) nötig**
Spezialisiertes System optimiert für Datenverwaltungsaufgaben (Suche, Einfügen, Ändern)	**ebenfalls Indexsoftware nötig**

Tabelle 4-3 Vor- und Nachteile der Suche in einer strukturierten Indexdatenbank

4.1.3 Arten von Suchmaschinen

Hinter Suchmaschinen verstecken sich vielerlei Konzepte. Am einfachsten ist die Unterscheidung in Volltextsuchmaschinen, Kataloge und Metasuchmaschinen.

Metasuchmaschinen sind Kombinationen mehrerer Suchsysteme. Sie übergeben den eingegebenen Suchstring an andere Suchmaschinensysteme und verarbeiten die Gesamttreffermenge nach eigenen Maßgaben. Die Suchergebnisse der einzelnen angefragten Systeme werden vor der eigentlichen Präsentation hinsichtlich eigener Relevanzmaßgaben sortiert und nochmals ausgewertet. Der Vorteil für den Sucher ist die parallele Anfrage an mehrere Suchsysteme und die kanalisierte Ergebnisbündelung.

Neben den Metasuchmaschinen existieren Kataloge. Ein ganz bekanntes Katalogsystem ist Yahoo. Kataloge unterscheiden sich im Wesentlichen von Volltext- und Metasuchmaschinen dadurch, dass die zugrundeliegenden Indexdaten manuell erstellt werden. Die Kategorisierung der Daten wird manuell von Redakteuren erledigt.

Eingängigster Vorteil der manuellen Verarbeitung der zu indexierenden Daten ist die hohe Qualität des Suchergebnisses. Die zu indexierenden Daten werden von Redakteuren gesichtet und gewertet. Dabei werden die Daten kategorisiert und im Index platziert. Für den Aufbau, die Suche und die Datenvorhaltung werden keine so großen Rechenressourcen benötigt wie bei der Volltextsuchmaschine. Ebenso entfällt die Netzbelastung durch die Agenten und Roboter, die den Volltextsuchindex aufbauen.

Neben den obigen Varianten an Suchmaschinen gibt es noch die eigentlichen Volltextsuchmaschinen. Volltextsuchmaschinen unterscheiden sich im Wesentlichen durch die Art der Informationssammlung und –kategorisierung. Was bei Katalogen hauptsächlich manuell passiert, wird bei Volltextsuchmaschinen automatisiert. Agenten, Roboter oder Crawler durchsuchen ständig die Datenbasis (z.B. das Internet oder das Intranet) und melden neue oder aktualisierte Daten an die Suchmaschine. Diese kategorisiert die Neudaten und aktualisiert den Index. Die Agenten durchstreifen ständig den Datenbestand und erzeugen dadurch eine nicht unerhebliche Netzlast. Im Internet sollen laut neuerer Untersuchungen bis zu 7% des Gesamtdatenverkehrs durch Agenten und Roboter entstehen. Trotz dieser ständigen Aktualisierungen der Agenten ist der Index ab einer bestimmten Größe des Datenbestandes immer veraltet. Die Aktualität des Index hängt ebenso stark von der Änderungsfrequenz am Datenbestand ab. Für den Aufbau, die Pflege und die Suche im Indexdatenbestand sind erhebliche Rechnerressourcen erforderlich. Zum einen braucht der automatisch aufgebaute Index viel Plattenspeicher und zum anderen müssen die Agenten im Netz gesteuert und die Daten indexiert werden. Dafür sind leistungsfähige Rechner von Nöten. Zudem muss der Rechner noch die ganzen Anfragen der Sucher verarbeiten. Volltextsuchmaschinen durchsuchen den Datenbestand oft nur stur nach der eingegebenen Zeichenkette. Meistens werden hier noch boole'sche Verknüfungsoperationen ("UND" / "ODER") berücksichtigt. Bei der Suche auf einem festgelegten Datenbestand werden mittlerweile auch schon fortgeschrittenere Suchmethodiken angewendet. Dabei kommen z.B. die phonetische Suche (Suche nach bestimmten Sprachmustern) oder die Mustersuche (Suche nach bestimmten Wort-/ Buchstabenmustern) zum Einsatz. Diese Methodiken sind jedoch durch extrem große Indexdateien nur für einen eingeschränkten Datenbestand einsetzbar.

4.1.4 Generelle Aufgaben von Suchmaschinen

Die Aufgaben der Suchmaschinen im Allgemeinen kann man in vier Phasen unterteilen: Akquise der Daten, Indexierung der Daten, Aktualisierung des Index und die Anfragenbearbeitung. Gemäß dieser Aufteilung existieren für die verschiedenen Arten von Suchmaschinen auch verschiedene Softwarekomponenten für die Realisierung dieser Aufgaben.

	Metasuchmaschine	Katalog	Volltextsuchmaschine
Akquise der Daten	**-**	**Manuell / Redakteur und Kunde**	**Maschinell / Agent**
Indexierung der Daten	**-**	**Manuell / Redakteur**	**Maschinell / Datenbank**
Aktualisierung des Index	**-**	**Manuell / Redakteur**	**Maschinell / Datenbank**
Anfragenbearbeitung	**Maschinell / Suchmaschine**	**Maschinell / Suchmaschine**	**Maschinell / Suchmaschine**

Tabelle 4-4 Arten von Suchmaschinen und deren Aufgaben

4.2. Überblick über die Aufgaben und Einsatzgebiete von Agenten

Agenten, oder auch Roboter, Crawler, Spider, ... genannt, sind für die Datenakquise aus dem Datenbestand zuständig. Sie wandern ständig über den gesamten bekannten Datenbestand und melden Neudaten und Änderungen an bestehenden Daten an die Suchmaschine zurück. Dort wird der Index den Meldungen entsprechend aktualisiert. Die Agenten folgen Stück für Stück den Verknüpfungen (in HTML-Dokumenten: Links) in einem Dokument. Als Startpunkt verwenden Agenten entweder bereits vorhandene Daten oder stützen sich auf neugemeldete Seiten. Treffen diese Agenten auf Verknüpfungen in Dokumenten entscheiden die Agenten für jeden Link, wie mit diesem weiterverfahren werden soll. Die weitere Vorgehensweise hängt von der jeweiligen Strategie der Suchmaschine bzw. des Agenten ab. Manche Agenten folgen beispielsweise maximal 3 Linkebenen auf einem Server und senden den jeweiligen Dokumenteninhalt an die Suchmaschine. Andere verfolgen nur eine Linkebene und verzweigen dann auf andere Server. Agenten im Internet verfolgen zumeist nur explizit gekennzeichnete Links, dem HTML-Element <A>. Mit Frames*, Imagemaps* oder anderen Verweismöglichkeiten haben die meisten Agenten Probleme und ignorieren derart gestaltete Verweise. Durch die Arbeitsweise der Agenten entsteht eine nicht zu verachtende Netzlast. Die Agenten fordern ständig Dokumente über das Netz an und verarbeiten die empfangenen Dokumenten ihrer Strategie entsprechend. Die Zeit zwischen den Besuchen des Datenbestandes variiert zwischen einigen Tagen und mehreren Wochen. Auch hier existieren verschiedene Konzepte und Strategien. Manche Suchmaschinen steuern die Besuchsfrequenz des Roboters danach, wie häufig die zu indexierenden Dateien geändert werden. Im Internet kann sich ein Serverbetreiber auch davor schützen, von einem Agenten besucht zu werden oder dem Agenten spezielle Informationen mitgeben. Dadurch können Dokumente geschützt werden oder aber auch der Server entlastet werden. Derzeit funktioniert diese Art Steuerung über Meta-Tags innerhalb der HTML-Dokumente oder eine Datei *robots.txt*, die an einer festgelegten Stelle auf dem Server hinterlegt wird. Viele Agenten werten diese Informationen aus. [SUF99b]

4.3. Überblick über verschiedene Methoden und Techniken von Suchmaschinen

Suchmaschinen basieren je nach Anwendungsgebiet auf verschiedensten Technologien. Die einzelnen Suchmaschinen verwenden zur optimalen Erfüllung ihrer Aufgaben die jeweils optimalste Technologie. Andere Suchmaschinen wiederum verwendeten in frühen Versionen einfache Methoden und verfeinerten die dahinterliegenden Konzepte. Mittlerweile verwenden sie komplexe mathematische Modelle, um die Suche zu optimieren. In diesem Abschnitt soll ein Überblick über die Methoden und Techniken verschiedener Suchmethoden gegeben werden, ohne dabei jedoch zu tief in die dahinterliegenden mathematischen Konzepte und Formeln einzugehen.

4.3.1 Boolean Retrieval Methoden

[OAR99, HIL95, CAW99a] Beim Boolean Retrieval werden vor der Indexierung der Dokumente Indexbegriffe definiert. Die einzelnen Dokumente werden hinsichtlich der Existenz dieser Begriffe im Inhalt untersucht und eine Indextabelle erstellt. In der

Indextabelle wird die Relevanz der einzelnen Dokumente hinsichtlich der Indexbegriffe vermerkt. Das Dokument und die darin enthaltene Information wird also durch eine Menge von Schlüsselworten oder Indexbegriffen beschrieben (z.B. Geld, Macht, Bauern, ...). Es erfolgt eine Abstraktion des Inhaltes des Dokumentes auf einige wenige Begriffe. Dabei ist generell ein Informationsverlust zu erwarten. Der Grad des Verlustes ist stark von der Wahl der Indexbegriffe abhängig. Werden Begriffe für die Indexierung gewählt, die wenig Information repräsentieren und dabei auch noch häufig vorkommen (z.B. "die" oder "macht"), so leidet die Qualität des Indexes darunter. Anfragen, die derlei Begriffe enthalten, produzieren eine extrem große Treffermenge [SUF99c].

Dokumentenname / Inhalt	Indexiertes Wort und gleichzeitig Suchwort					
	Geld	Macht	Bauern	Kuchen	Schrank	Schränke
d_1 = "Geld allein macht glücklich"	**X**	**X**				
d_2 = "Bauernmöbel und Schränke"			**X**			**X**
d_3 = "Kuchen backen für Singles"				**X**		
d_4 = "Die Macht der Könige"		**X**				
d_5 = "Gebäck im Kühlschrank"					**X**	
d_6 = "Macht Kuchen dick ?"		**X**		**X**		

Tabelle 4-5 Beispiel einer Indextabelle

Die Indextabelle hat den Vorteil, dass sich die Menge der zu speichernden Informationen pro Dokument stark verringert und somit das zu speichernde Datenvolumen drastisch einschränkt.
Als Suchabfrage wird ein boole'scher Ausdruck verwendet, der Indexbegriffe mit boole'schen Operatoren verknüpft.
Dazu ein kleines Beispiel anhand Tabelle 4-5.

Anfrage	Ergebnismenge
"Geld AND (Macht OR Erfolg) AND Reichtum"	**leer**
"Geld OR Macht"	**d_1, d_4, d_6**
"Kuchen AND (Schrank OR Macht)"	**d_6**

Tabelle 4-6 Beispielabfragen auf der obigen Indextabelle

Das boole'sche Suchverfahren ist eine Methode nach dem "Exact-Match" Verfahren. Dokumente, die exakt der Suchabfrage entsprechen werden der Treffermenge zugeordnet. Problematisch an dieser Vorgehensweise ist, dass ein Dokument, das bei einer Abfrage wie "A OR B OR ... OR Z" nur eine Bedingung erfüllt genauso in der Treffermenge ist, wie ein Dokument, das alle Bedingungen erfüllt. Andererseits ist ein Dokument, das bei einer Abfrage wie "A AND B AND ... AND Z" nur eine Bedingung nicht erfüllt, genauso wenig in der Treffermenge, wie ein Dokument, das keine der Bedingungen erfüllt [SAL82a].
Der Benutzer erhält also immer eine exakte Antwort auf seine Anfrage. Dokumente, die sich in der Ergebnismenge befinden, sind alle gleichwertig. Ein Ranking nach der Relevanz bezüglich der Anfrage wird nicht vorgenommen. Die Größe der Antwortmenge ist vom Benutzer kaum kontrollierbar. Sie ist abhängig von den festgelegten Indexbegriffen und den Begriffen in der Abfrage. Meistens erhält der Anfragende entweder extrem viele Ergebnisse oder überhaupt keine.
Ein zentrales Problem der boole'schen Suche ist der Anspruch an den Benutzer seinen komplexen und unscharfen Informationsbedarf in eine formale Frageformulierung unter Verwendung logischer Operatoren zu übersetzen. Diese Umsetzung wird immer näherungsweise und grob passieren. Bei diesem Prozess geht je nach Erfahrung des

Benutzers Information verloren. Die eigentlich nicht korrekte Anfrage des Benutzers wird von der boole'schen Suche allerdings exakt ausgewertet. Spätestens hier stellt sich allgemein die Frage, ob die Methodik der boole'schen Suche dem Problem "Informationen finden" angemessen ist [KNO94a].
Trotzdem ist die boole'sche Suche das am weitesten verbreitete Verfahren und kommt bei großen Suchdiensten fast ausschließlich zur Verwendung. Die Suche mit der boole'schen Suchoperatorenverknüpfung ist eine der ersten optimierten Suchmethoden überhaupt und ist einfach und schnell zu implementieren. Die Suchabfrage ist aufgrund des erstellten Indexes performant.

4.3.2 Vector Space Retrieval Methoden

[KNO94b, CAW99b] Das Vector Space Retrieval Modell ist das älteste - und trotzdem ein aktuelles - Modell zur strukturierten Suche nach Informationen. Das Vector Space Verfahren ist ein mathematisch einfaches und gut handhabbares Modell mit starker Analogie zum 3-D Raum. Ausgangspunkt des Verfahrens ist ein Vektorraum, der für jeden Indexbegriff eine eigene Koordinatenachse – also Dimension – besitzt. Bei N Indexbegriffen wird ein N-dimensionaler Vektorraum aufgespannt. Jedes einzelne zu indexierende Dokument wird über N Koordinaten im N-dimensionalen Vektorraum platziert.

Dazu ein kleines Beispiel im 3-D Raum.
Für N=3 mit den Indexbegriffen t_1 = automatisch, t_2 = manuell und t_3 = Indexierung besitzt das Dokument "Alles über automatische Indexierung" die Darstellung (1 / 0 / 1). In diesem Beispiel wurden keine differenzierten Gewichte benutzt, sondern "1" zeigt die Anwesenheit und "0" die Abwesenheit eines Indexbegriffes an [KNO94b].

Eine an das System gestellte Abfrage wird wie ein Dokument als Punkt im Vektorraum dargestellt. Berechnet wird nun die Distanz des Punktes im Raum, der die Abfrage darstellt und den umliegenden Punkten, die Dokumente darstellen. Die Distanz zwischen den Punkten wird als Maß für die Ähnlichkeit zwischen Abfrage und Dokument angesehen. Die Ausgabe der Ergebnisse an den Benutzer erfolgt nun nach steigendem Abstand bzw. fallender Ähnlichkeit. Der Benutzer erhält also alle Dokumente der Ergebnismenge nach Relevanz gewichtet. Bei der Gewichtung der Dokumente wird implizit die Annahme getroffen, dass die Ähnlichkeit zwischen Abfrage und Dokument ein Indikator für die Relevanz bezüglich der Abfrage darstellt. Das Vector Space Modell ist ein heuristisches Modell, das nach dem "Partial Match" Verfahren arbeitet. Durch die Ähnlichkeit zum 3-D Raum ist es ein anschauliches Verfahren. Im Unterschied zum boole'schen Verfahren müssen die Anfragen an das System nicht strukturiert sein. Dadurch ist es jedoch nicht mehr möglich, satzähnliche Konstrukte oder synonyme Begriffe durch boole'sche Operatoren* zu verknüpfen.
Wie auch beim Boolean Retrieval Modell ist es von zentraler Bedeutung die Indexbegriffe zu definieren. Bei der Entscheidung, welche Indexbegriffe den Vektorraum aufspannen sollen, kann die Berechnung eines "term discrimination value" hilfreich sein. Der "term discrimination value" misst für jeden Indexbegriff, ob dieser dem Ziel dienlich ist, oder ob seine Verwendung negative Folgen auf die Verteilung der Dokumente hätte [KNO94b].

Beim Aufstellen der Indexbegriffe sollte beachtet werden, dass die Verwendung zu häufiger Begriffe als Indexes die Dokumente im Vektorraum zusammenrücken lässt, denn sehr viele Dokumente gleichen sich dann in der Hinsicht, dass sie diesen Begriff enthalten. Analoges, wenn auch mit umgekehrtem Sinn, gilt für den Fall zu seltener Begriffe. Ziel der Indexbegriffwahl ist es, die Dokumente so zu repräsentieren, dass sie möglichst gut unterscheidbar, also möglichst verschieden sind. Neben der Optimierung der Indexbegriffen, die den Vektorraum aufspannen, kann das Verfahren noch durch weitere Optionen verbessert werden. Zunächst kann durch eine Gewichtung der Indexbegriffe eine Verbesserung der Retrievalergebnisse erreicht werden. Durch die Gewichtung der einzelnen Begriffe wird die Abstandsmessung der Punkte im N-dimensionalen Vektorraum beeinflusst. Durch geeignete Gewichtung der Begriffe kann die Abstandsmessung und damit die Ähnlichkeitsberechnung optimiert werden. Zur Gewichtung der Indexbegriffe kann beispielsweise das "Document Frequency" [KNO94c] Verfahren herangezogen werden. Das Document Frequency Verfahren (auch bekannt als "Inverse Dokumenthäufigkeit") ist eine einfache, plausible und effektive Formel zur Termgewichtung, die in verschiedenen Varianten vorkommt.
Beispielhaft soll hier auf die einfachste Form eingegangen werden.
Betrachtet man ein bestimmtes Dokument *d*, so kann man für einen darin vorkommenden Indexbegriff *i* folgende Gewichtung definieren:

$$inverse\ Dokumentenhäufigkeit(i,d) = \frac{Auftretenshäufigkeit\ von\ i\ in\ d}{Dokumenthäufigkeit\ von\ d}$$

Das Gewicht eines Indexbegriffes ist hoch, wenn es nur wenige Dokumente gibt, in denen er auftaucht und wenn er gleichzeitig im fraglichen Dokument häufiger vorkommt.

Dazu ein kleines Beispiel:
Betrachtet werden 3 Dokumente d_1, d_2 und d_3 aus einer Kollektion von 10.000 Dokumenten.

d_1 = "Anwendung von VectorSpace Modellen zum Information Retrieval"
d_2 = "Zusammenhang zwischen Information, Daten und Information Retrieval"
d_3 = "VectorSpace Modelle im Vergleich"

Damit mit der obigen Formel gearbeitet werden kann, wird die Anzahl der Dokumente benötigt, in denen der jeweilige Indexbegriff gefunden wurde.

Indexbegriff	Anzahl Dokumente
Anwendung	**3.000**
Information	**2.000**
Modell	**500**
Retrieval	**200**
VectorSpace	**1.200**
Zusammenhang	**1.800**
Daten	**700**

Tabelle 4-7 beispielhafte Indexierung von Dokumenten

Aus der Tabelle 4-7 kann für die einzelnen Dokumente folgende Indexierung erstellt werden:

d_1	d_2	d_3
Anwendung (1/3000); **VectorSpace (1/1200);** **Modellen (1/500);** **Information (1/2000);** **Retrieval (1/200);**	**Zusammenhang (1/1800);** **Information (2/2000);** **Daten (1/700);** **Retrieval (1/200);**	**VectorSpace (1/1200);** **Modelle (1/500);**

Tabelle 4-8 beispielhafte Indexierung für drei Beispieldokumente

Für die Anfrage "VectorSpace und Information Retrieval" ergibt sich dann folgende Gewichtung für die 3 Beispieldokumente:

	Gewicht	übereinstimmende Indexbegriffe
d_1	**1/1200 + 1/2000 + 1/200 = 77/1500 =** 0,051	**VectorSpace, Information, Retrieval**
d_2	**2/2000 + 1/200 = 3/500 =** 0,006	**Information, Retrieval**
d_3	**1/1200 =** 0,0008	**VectorSpace**

Tabelle 4-9 beispielhafte Gewichtung der drei Beispieldokumente

Eine weitere Optimierung im Zusammenhang des Vector Space Retrieval Modells ist die mögliche Clusterung von Dokumenten. Dabei wird aus einer Gruppe von Dokumenten ein virtueller Repräsentant berechnet, der letztlich diese Dokumente bei der Ähnlichkeitsberechnung vertritt. Wird der virtuelle Vertreter der Gruppe als relevant angesehen, so werden für weitere Berechnungen die geclusterten Dokumente verwendet.

4.3.3 Probabilistic Retrieval Methoden

[KNO94d, CAW99c] Wie der Name dieser Kategorie von Techniken schon sagt, basiert diese Technologiefamilie auf Wahrscheinlichkeiten. Im Gegensatz zu Vector Space Modellen, die an die Anschauung appellierende, heuristische Verfahren sind, versuchen probabilistische Modelle normativ* zu arbeiten. Grundlage der Methode ist der mathematische Beweis, dass dasjenige Ranking optimal ist, das die Dokumente nach der Wahrscheinlichkeit ihrer Relevanz für die Frage anordnet. Diese Wahrscheinlichkeit gilt es – unter Berücksichtigung vereinfachender Annahmen – abzuschätzen. Grundlage für die Wahrscheinlichkeitsabschätzung ist die Verteilung der Indexbegriffe auf die Dokumente.
Durch die vereinfachenden Annahmen bezüglich der Wahrscheinlichkeit der Relevanz der Dokumente werden die Modelle erst rechnerisch oder vom Datenaufkommen her handhabbar. Die Annahmen sind meist sehr realitätsfremd, haben aber den Vorteil, dass sie letztlich eine optimale Lösung liefern und explizit offengelegt sind – also nachvollziehbar sind.
Jedem probabilistischem Verfahren liegt ein spezifisches Modell zu Grunde, das die Daten und Wahrscheinlichkeiten definiert, die benötigt werden, um die gesuchte Wahrscheinlichkeit (nämlich die, dass ein Dokument, das gewisse Begriffe enthält, auf eine Frage, die bestimmte andere Begriffe enthält, relevant ist) zu errechnen. Wahrscheinlichkeiten, die vom Modell als nötig definiert werden, müssen abgeschätzt werden.

Dazu ein Beispiel:
Gibt es in einer Datenbank mit N Dokumenten r relevante Dokumente auf eine Frage, so errechnet sich die Wahrscheinlichkeit, dass ein beliebiges Dokument der Datenbank "nicht relevant" ist als

$$p(nicht\ relevant) = \frac{1}{N - r}$$

Da *r* in der Praxis aber nicht bekannt ist und weil *N* normalerweise sehr viel größer ist als *r*, kann man statt des korrekten Wertes ohne großen Fehler auch die einfache Abschätzung

$$p(nicht\ relevant) = \frac{1}{N}$$

verwenden.

Von der Qualität der Schätzungen hängt in hohem Maße die Brauchbarkeit des Verfahrens ab. Um das spezifische Modell zu verfeinern, können vereinfachende Annahmen eliminiert werden. Das Modell nähert sich mit Reduktion der vereinfachenden Annahmen der Realität an. Mit der Erhöhung der Komplexität des Modells steigt aber auch die Zahl der zu schätzenden Parameter. Da bei jeder Einzelschätzung immer ein Fehler akzeptiert wird, summieren sich die Einzelfehler bei mehreren Einzelschätzungen zu einem größeren Gesamtfehler. Aus diesem Grund sind einfachere Modelle mit wenigen Abschätzungen den komplexeren, realitätsnäheren überlegen [KNO94d].

Anfragen an probabilistische Suchmethoden sind genauso unstrukturiert, wie Anfragen an das Vector Space Modell.
Als Beispiele für probabilistische Verfahren werden hier kurz drei Verfahren skizziert.

4.3.3.1. Grad der Wortübereinstimmung

[CIS, ROB76] Bei diesem Verfahren wird die Wahrscheinlichkeit *p*, dass ein Dokument auf die Anfrage *f* passt, an der Anzahl der übereinstimmenden Wörter gemessen.

f = "Informationen Datenbanken finden"

Dokument	Grad der Übereinstimmung
"Moderne Datenbanken"	**1/3**
"Speichern von Informationen in Datenbanken"	**2/3**
"Suchen und Finden in Datenbanken"	**2/3**
"Informationen über's Glücklichsein finden"	**2/3**
"Die Kunst, Informationen in Datenbanken zu finden"	**3/3**

Tabelle 4-10 beispielhafte Anwendung des Verfahrens "Grad der Wortübereinstimmung"

4.3.3.2. Anwendung verschiedener Verfahren zur Gewichtung von Indexbegriffen

[CIS, ROB76] Ein anderer Ansatz ist es jeden einzelnen Indexbegriff – oder auch Kombinationen von Indexbegriffen – zu gewichten. Das Wahrscheinlichkeitsmaß für die Relevanz eines Dokumentes auf eine Anfrage ist dann die Summe aller Gewichte. Damit wird das Ranking dahingehend optimiert, dass Dokumente, die eine gleiche

Anzahl an Wörtern mit der Anfrage gemeinsam haben bezüglich der Relevanz der jeweiligen Indexbegriffe bewertet werden.
Dazu drei Beispiele:

- "Collection Frequency" Indexbegriffe, die nur in wenigen Dokumenten auftauchen sind eher nützlich, als solche, die in vielen Dokumenten vorkommen.
- "Term Frequency" Je häufiger ein Indexbegriff in einem Dokument vorkommt, desto wichtiger erscheint dieser Begriff für das gesamte Dokument zu sein.
- "Document Length" Ein Indexbegriff, der gleich oft in einem kurzen Dokument vorkommt, wie in einem langen Dokument, ist für das kurze Dokument charakteristischer, als er das für das lange Dokument ist.

Für ein Dokument/Indexbegriff-Paar werden alle obigen Gewichte berechnet und anschließend kombiniert. Daraus ergibt sich ein Gesamtgewicht, das für das Maß der Relevanz der einzelnen Dokumente hinsichtlich der Anfrage steht.

4.3.3.3. "Relevance Feedback"

[CAW99d, KNO94d, POO99a] Die Wahrscheinlichkeit p, dass ein Dokument auf eine Frage mit dem Frageterm f relevant ist, wird in diesem Verfahren berechnet als

$$p(relevant) = \frac{r\,(1-u)}{u\,(1-r)}$$

r ist dabei die Wahrscheinlichkeit, dass f im Dokument vorkommt, unter der Vorraussetzung, dass es relevant ist.
u ist analog die Wahrscheinlichkeit, dass f im Dokument vorkommt und es nicht relevant ist.

4.3.4 Extended Boolean Retrieval Methoden

[CAW99e, SAL82b] Die Extended Boolean Retrieval Modelle ergänzen die herkömmliche boole'schen Suchmodell durch Verwendung eines Rankingmechanismus, um die Ähnlichkeit zwischen Dokument und Anfrage zu berechnen. Den Erweiterten boole'schen Suchmodellen gemeinsam ist, dass sie durch Austausch der logischen Operatoren AND und OR versuchen die Relevanz der Treffer zu erhöhen. Für die Ersetzung der Operatoren verwenden die Methoden die unterschiedlichste Ansätze. Durch die Verwendung dieser komplexen Operatoren wird die einfache boole'sche Suche durch Rankingmechanismen ergänzt. Dadurch wird das einfache Modell mit seinen Unzulänglichkeiten hinsichtlich der Suchergebnisse wesentlich verbessert.
[LEE95a] Extended Boolean Retrieval Modelle können generell durch ein Quadrupel $<T, D, Q, F>$ charakterisiert werden.

- T ist eine Menge von Indexbegriffen, die Anfragen und Dokumente repräsentieren
- D ist eine Menge von Dokumenten. Jedes Dokument $d \in D$ wird dargestellt durch $\{(t_1, w_1), \ldots, (t_n, w_n)\}$, wobei w_i die Gewichtung des Begriffes t_i im Dokument d ist und w_i jeden Wert zwischen 0 und 1 annehmen kann $(0 \le w_i \le 1)$
- Q ist eine Menge von Abfragen, die vom System erkannt werden können. Jede Abfrage $q \in Q$ ist ein gültiger boole'scher Ausdruck, zusammengesetzt aus Indexbegriffen und den boole'schen Operatoren AND, OR und NOT.

- *F* ist eine Rankingfunktion

$$F : D \times Q \rightarrow [0,1]$$

die einem jeden Paar (*d*,*q*) eine Zahl im geschlossenen Intervall [0,1] zuweist. Diese Zahl ist ein Maß für die Ähnlichkeit zwischen Dokument *d* und der Abfrage *q* und wird der "Dokumentenwert" für das Dokument *d* hinsichtlich der Abfrage *q* genannt. Die Funktion *F* ist wie folgt definiert:
1. Für jeden Begriff t_i in der Abfrage *q* definiert die Funktion $F(d, t_i)$ das Gewicht des Begriffes t_i im Dokument *d*, z.B. w_i
2. Boole'sche Operatoren* wie AND, OR und NOT werden bewertet, indem die zugehörigen mathematischen Formeln benutzt werden. Diese Berechnungsformeln der Operatoren sind ein wichtiger Faktor für die Qualität des Ranking.

In der Funktion *F()* werden beim extended boole'schen Suchen die Operatoren definiert. In der einschlägigen Literatur wurden bereits viele Operatoren analysiert und bewertet. Die einzelnen Operatoren sind unter Namen wie "Fuzzy-Set", "Waller-Kraft", "Paice", "p-norm", "infinite-one", "network Boolean", "INQUERY Boolean", ... bekannt [LEE95b]. Die einzelnen Operatoren sind durch unterschiedlichste Problematiken ausgehend vom zugrundeliegenden mathematischen Modell geprägt. Zur Übersicht hier einige ausgewählte Operatoren.

Operator	AND	OR
Fuzzy-Set	$MIN(w_1, w_2)$	$MAX(w_1, w_2)$
INQUERY Boolean	$w_1 \cdot w_2$	$w_1 + w_2 - w_1 w_2$
infinite-one	$\beta \cdot MIN(w_1,..., w_n) + \frac{(1-\beta)(w_1 + ... + w_n)}{n}$	$\beta \cdot MAX(w_1,..., w_n) + \frac{(1-\beta)(w_1 + ... + w_n)}{n}$

Tabelle 4-11 verschiedene Operatoren des Extended Boole'schen Modells

Beim letzten Operator gilt $0 \leq \beta \leq 1$.

Die Berechnungsvorschriften für die unterschiedlichen Operatoren können beliebig komplex werden. Eine Abhandlung über die Qualität der einzelnen Operatoren im Vergleich, deren Vor- und Nachteile kann in [LEE95b] nachgelesen werden. Problematisch an der Vorgehensweise, bekannte Operatoren mit neuer Semantik zu überladen, ist die Auffassung des Menschen. Manche der neueren Operatoren liefern komplett von der Vorstellung des Menschen abweichende Resultate, die zwar mathematisch korrekt, aber nur sehr schwer nachvollziehbar bzw. steuerbar sind. Durch die Verwendung dieser komplexeren Operatoren wird aus dem Boole'schen Suchmodell ein "Partial-Match" Verfahren.

4.3.5 Expertensystembasierte Retrieval Methoden

[POO99b] Während obige Suchmethoden immer davon ausgehen, dass der Index über die Dokumente neu erstellt werden kann, stellen die "expertsystem based" Methoden den Investitionsschutz bereits erstellter Indexe in den Vordergrund.
Expertensystembasierte Suchmodelle werden darum hauptsächlich in OPAC's oder stark themenbezogenen Suchsystemen verwendet.
Bei diesen Systemen werden Forschungsergebnisse im Bereich der neuronalen Netze und Expertensysteme umgesetzt. Ziel der Verfahren ist es die vom Benutzer eingegebene Suchanfrage unter Verwendung von Agentenwissen bzw.

Expertensystemen in eine boole'sche Anfrage zu wandeln. Die Umformung soll dabei so geschehen, dass der Inhalt der Benutzeranfrage möglichst ohne semantische Verluste umgesetzt wird. Bei der Umsetzung wird ein Expertensystem eingesetzt, das das jeweils fachbezogene Wissen eines menschlichen Experten und dessen Entscheidungsgrundlagen bzw. Lösungsverfahren codiert. Es wurden bereits mehrere Systeme entwickelt, die auf Basis von Produktionsregeln und semantischen Netzen arbeiten [POO99b].

- "Gauch's Query Reformulation" System verwendet Produktionsregeln, um die Qualität der Anfrage zu verbessern. Die boole'schen Operatoren werden verändert, oder verwandte Begriffe zur Abfrage hinzugefügt, oder verschiedene Begriffe durch breiter oder enger gefasste Begriffe aus einem Wörterbuch ersetzt. [GUA99]
- "PLEXUS" ist ein expertenbezogenes System, das Wörter aus dem Bereich "Garten" auf semantische Primitive abbildet. Es beinhaltet einige Suchstrategien, die als Produktionsregeln implementiert sind und baut ein temporäres Benutzerprofil über dessen Wissen zum Thema "Garten" auf. Basierend auf diesem Profil erhält der Benutzer seinem Kenntnisstand entsprechende Hilfestellung. [VIC87]
- "Rule Based Retrieval Information by Computer" kurz RUBIC, benutzt Produktionsregeln, um eine ganze Hierarchie an Suchstrategien bzw. Vorgehensweisen zu definieren. Die jeweilige Wissensbasis wird durch eine Ansammlung von Konzepten abgebildet. Jedes Konzept beinhaltet eine Beschreibung, die Beziehung zu anderen Konzepten und die Regeln, die das Textmuster beschreiben, die vorhanden sein müssen, um das gespeicherte Suchkonzept anzuwenden. Dieses System verlangt vom Benutzer, dass er systemkonforme Anfragen stellt. [TON87]
- Drabenstott und ihre Kollegen entwickelten eine prototypische Implementierung eines Online-Kataloges, der Such- oder Entscheidungsbäume verwendet, um abzubilden, wie erfahrene Bibliothekare eine Suchstrategie auswählen und formulieren davon abhängig einen Suchausdruck. Der Entscheidungsbaum wird durch ein Flussdiagramm dargestellt. [DRA96a, DRA96b, DRA96c]

4.3.6 Verwendung von fortgeschrittenen mathematischen Modellen

[AUT00] In diesem Kapitel werden die vom Autonomy Knowledge Server verwendeten Technologien und die Hintergründe näher erläutern.
Autonomy's Vorgehensweise unterscheidet sich fundamental von den vorig genannten. Autonomy will den Computer in die Lage bringen, Bedeutungen aus Texten zu extrahieren und anhand dieser Informationen die Einteilung in Kategorien zu verbessern. Der Computer soll Zusammenhänge und den Kontext "verstehen", von den Worten auf Ideen abstrahieren und die Kerngedanken zwischen den Zeilen aufnehmen. Autonomy's Unternehmensziel: "Oracle der unstrukturierten Daten" werden. Die verwendete Technologie adressiert vornehmlich die Verwaltung unstrukturierter Massendaten. [ATW00a, WIR00a] Autonomy will die Explosion unstrukturierter Information in Form von Texten handhabbar machen. Um dieses Ziel zu erreichen werden neuronale Netzwerke* und fortgeschrittene Mustervergleichsalgorithmen (non-linear adaptive digital signal processing) kombiniert verwendet. Diese Algorithmen

fassen Dokumente digital zusammen und legen die Merkmale fest, die dem Text eine digital erfassbare Bedeutung geben. Bei Autonomy ist die Dynamic Reasoning Engine* (DRE*) für genau diese Aufgaben zuständig. Sie ist Grundlage aller Produkte von Autonomy.
Für eine Standardtextsuche gibt der Benutzer einfach eine boole'sche Abfrage, oder natürliche Sprache ein. Diese Abfrage wird an die DRE* zur Analyse übermittelt. Die DRE bearbeitet die Anfrage und gibt eine nach Relevanz geordnete Liste von Dokumenten zurück. Neben der "normalen" Suche unterstützt die DRE aber auch die Suche nach Konzepten und Ideen. Dadurch unterscheidet sich die Technologie zentral von den sonst üblichen Methoden. Der Benutzer umschreibt seine Idee, verwendet seine natürliche Sprache um die zu suchende Informationen zu definieren. Diese umfangreichere Abfrage wird von der DRE als Dokument interpretiert und digital erfasst. Anschließend sucht die DRE Dokumente mit dem höchsten Grade an Relevanz. Diese werden – geordnet nach der Relevanz – als Treffermenge zurückgegeben. Durch die Konzentration auf die Mustersuche ist die verwendete Technologie komplett sprachunabhängig. Wörter werden vom Computer als abstrakte Symbole mit Bedeutung angesehen. Die Bedeutung wird aus dem Vorkommen im Kontext abgeleitet, nicht aus einer einfachen Grammatikdefinition. Dialekte oder Abänderungen in der Sprache stören die verwendete Technologie nicht.
[ATW00b] Die verwendete Technologie grenzt sich stark von herkömmlichen Suchmethoden ab. Die reine Schlüsselwortsuche nach der boole'schen Methode kann Dokumente nur anhand der darin auftauchenden Wörter identifizieren, aber keine Aussage über die Relevanz zur Suche treffen. Genauso wenig kann eine Aussage getroffen werden, ob das Dokument thematisch etwas mit dem gesuchten Wort zu tun hat. Eine Suche nach Konzepten oder Ideen, die in natürlicher Sprache die gesuchte Information charakterisieren, ist nicht möglich, da in der boole'schen Suche mehr Suchwörter in der Eingabe zumeist mehr Dokumente als Treffer liefern. Zudem verwendet die boole'sche Suche einfache Relevanzkriterien, die jedoch oft fehlerhafte Resultate ergeben. Ein Romantext beispielsweise

" ... er ging auf der linken Seite der Straße die Straße hinunter ... die Straße war nass ... seine Schritte hallten auf der Straße ... er hörte Schritte auf der Straße ... er rannte die Straße hinab ... der Gegner erwischte ihn und ermordete ihn kaltblütig ..."

Im Text kommt das Wort "Straße" häufig vor, der Text handelt jedoch von einem Mord. Bei einer boole'schen Suche nach dem Wort "Straße" würde dieser Text sicherlich als relevant eingestuft. Bei der Suche nach "Mord" wohl eher nicht.
Ein anderer Ansatz Ideen und Konzepte aus Texten zu extrahieren ist die Verwendung von Parsern*, um natürliche Sprache zu "verstehen". Die unstrukturierte Information wird um Grammatikregeln und Lexikons ergänzt – wird also strukturiert. Mit diesen Hilfsmitteln wird versucht textuelle Information zu verstehen. Problematisch an diesem Ansatz ist die schlechte Performanz von Parsern mit komplexen Regeln. Zudem kann die Wichtigkeit von Ideen nicht bewertet werden. Der Parser trifft in seinem Entscheidungsbaum binäre Entscheidungen (wahr / falsch). Trifft der Parser aufgrund von Fehlern im Regelwerk oder im Parser einmal eine falsche Entscheidung, so ist der ganze Entscheidungsprozess gefährdet. Als größtes Manko ist die extreme

Sprachabhängigkeit zu sehen. Für jede zu parsende Sprache muss ein eigenes Regelwerk und ein Lexikon generiert werden.

[ATW00, WIR00a] Nach diesen Bemerkungen allgemeiner Natur soll die Technologie von Autonomy mehr hinterleuchtet werden. Autonomy's Technologieansatz basiert auf relativ alten Annahmen und bereits anerkannten Gesetzmäßigkeiten. Claude Shannon's Prinzip der Informationstheorie, Thomas Bayes' Annahmen über Wahrscheinlichkeitsmodelle bei mehreren Unbekannten und Kenntnissen im Feld der neuronalen Netze ermöglichen die schnelle Musteridentifikation in Texten und anderen Quellen. Die Kombination dieser drei Technologien ermöglicht es aus einem Text die Schlüsselkonzepte zu extrahieren, weil Autonomy "versteht", wie die Häufigkeit und Beziehung zwischen Begriffen mit deren Bedeutung korreliert.

Shannon's Informationstheorie ist die mathematische Grundlage aller digitaler Kommunikationssysteme. Laut Shannon ist "Information ein quantifizierbarer Wert in der Kommunikation". In seinem Werk "The Mathematical Theory of Communication" geht Shannon genauer auf die Aspekte der Kommunikation und die Möglichkeit Information zu berechnen ein. Prinzipiell verfolgt Shannon jedoch die Theorie, dass je geringer das Auftreten einer Kommunikationseinheit (Wort, Satz) ist, desto informativer ist diese. Ideen, die seltener im Umfeld einer Kommunikation sind, tendieren daher eher dazu, relevanter für die Bedeutung der Kommunikation zu sein. Unter Berücksichtigung dieser Aussagen ist es möglich die wichtigsten oder informativsten Konzepte in einem Dokument zu finden. Weiter soll an dieser Stelle nicht auf die Theorien von Claude Shannon eingegangen werden, da diese mathematisch fundiertes Wissen voraussetzen und trotz der Einfachheit der obigen Aussage doch nicht ganz einfach zu verstehen sind.

[ROU00] Thomas Bayes Theorem ist eine zentrale Säule moderner statistischer Wahrscheinlichkeitsmodelle. Seine Arbeiten konzentrierten sich auf die Berechnung von Wahrscheinlichkeitsbeziehungen zwischen mehreren Variablen und die Abhängigkeiten der Variablen untereinander. Bayes Theorem setzt Ereignisse der Gegenwart unter dem Wissen vergangener Ereignisse mit der in Zukunft zu erwartenden Ereignisse in Beziehung. Erst nach seinem Tod entdeckte ein Freund in seinen Aufzeichnungen die Arbeit "An Essay Towards Solving a Problem in the Doctrine of Chances". Dort skizziert Bayes ein Modell für die Vorhersage von Ereignissen unter Unsicherheitsbedingungen.

Die mathematische Formel

$$P(t \mid y) = \frac{P(y \mid t)P(t)}{P(y)}$$

drückt die Annahmen von Thomas Bayes exakt aus. Um die Formel auch von pragmatischer Sicht zu verstehen soll sie an folgendem Beispiel erläutert werden.

Ein Koch arbeitet in einem belebten Straßencafe. Kellner rufen dem Koch Bestellungen zu. Dies geschieht vor einer lauten Geräuschkulisse aus Straßenlärm, klappernden Tellern und redenden Gästen. Was ein Gast tatsächlich bestellt hat ist in der obigen Gleichung *t*. Die verstümmelte Bestellung, die der Koch hört ist *y*. Ein bayes'scher Entscheidungsprozess würde dem Koch ermöglichen so viele Bestellungen wie möglich korrekt abzuarbeiten. Er hat noch ein hilfreiches Hintergrundwissen: Wenige Kunden bestellen Lachsschnitten, wohingegen Steak mit Pommes Frites ein richtiger Renner ist. Der Koch kann diese Art von Information verwenden, um die vergangene Wahrscheinlichkeit zu berechnen, dass ein geliefertes Essen bestellt wurde: *P(t)* oder die Wahrscheinlichkeit von *t*. Zusätzlich weiß der Koch noch, dass ein Kellner Gerichte mit französischem Namen immer falsch ausspricht, ein Bus alle 10 Minuten vorbeifährt und brutzelnde Bratpfannen den Unterschied zwischen z.B. Hammel und Semmel verwaschen. Nimmt er nun alle diese Faktoren zusammen, kann er ein komplexes Entscheidungsmodell erstellen, wie Bestellungen durcheinanderkommen können: *P(y|t)* – die Wahrscheinlichkeit von *y* unter der Vorraussetzung *t*. Durch Bayes' Theorem kann der Koch die Wahrscheinlichkeit *P(t|y)* berechnen, dass unter Berücksichtigung aller äußeren Einflussfaktoren das, was er verstanden hat, auch dem entspricht, was der Gast bestellt hat. Der Koch jedoch verwendet in der Realität nicht mathematisch komplexe Modelle, sondern baut auf seine Erfahrung.
Bayes hatte damals, als er seine Theorien aufgestellt hat das Problem, dass er durch seine bescheidenen Mittel – Stift, Notizblock und die Zeit, die er für die Berechnungen benötigte – stark eingeschränkt wurde. Durchgesetzt haben sich seine Theorien erst, als Computer Millionen von Wahrscheinlichkeiten in einem Augenblick berechnen konnten.

Die Technologie wird für Aufgaben aus der Chaostheorie und Bedeutungsfindung eingesetzt. Dort beispielsweise in der Mustererkennung, Signalverarbeitung, Verbesserung der MP3-Qualität, Bilddatenbanken, Fingerabdruck-, Gesichts- und Handschrifterkennung, ... Die Computer können aus der Erfahrung der Vergangenheit lernen und dadurch Rückschlüsse auf zu bewertende Situationen ziehen.

4.4. Ein exemplarisches Beispiel: http://www.webcrawler.com

[PIN94] Einige der erwähnten Konzepte und Vorgehensweisen sollen an dieser Stelle durch die Internet-Suchmaschine „WebCrawler“ beispielhaft verdeutlicht werden. Der Autor und Betreiber dieser Suchmaschine geht mit den realisierten Konzepten weniger restriktiv um, als andere Suchmaschinenbetreiber. Die Vorgehensweise einer Suchmaschine und die Realisierung der Qualität der zurückgelieferten Abfragen sind streng gehütete Betriebsgeheimnisse von so bekannten Suchmaschinen wie „FireBall“, „HotBot“ oder anderen. Um so angenehmer ist es, dass Brian Pinkerton umfangreiche Informationen über die Struktur des WebCrawlers öffentlich zur Verfügung stellt. Leider beschreibt das Dokument eine ältere Version des WebCrawlers. Anschließend wird kurz die Problematik beschrieben, die der WebCrawler zu lösen versucht und darauf folgend die Suchmaschine genauer beschrieben.

4.4.1 Allgemeiner Überblick

Benutzer des Internets finden Informationen, indem sie Hypertextlinks* folgen. Mit steigender Größe der Datenbasis muss der Benutzer immer mehr Links folgen, um sein Ziel zu erreichen. Der WebCrawler hilft dem Benutzer dadurch, dass er automatisch durch das Internet wandert und Seiten indexiert. Der WebCrawler bietet einen öffentlichen Zugang zum Index und eine Realtimesuche für autorisierte Benutzer. Ein WebRobot wartet den Index durch laufende Aktualisierung. Der WebCrawler bildet durch eine inhaltsbasierte Indexierung einen hochqualitativen Index.

4.4.2 Architektur* WebCrawler

Der WebCrawler folgt grundsätzlich dem gleichen Designansatz, wie das Internet – ein dezentralisiertes Design.

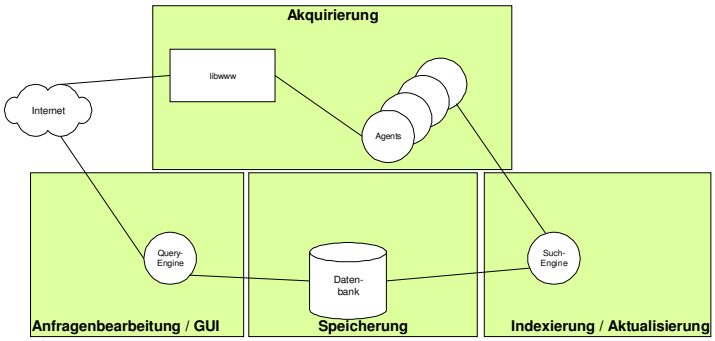

Abbildung 4-1 Architektur* des WebCrawlers

Die ganze Architektur* orientiert sich an den Aufgaben, die grundsätzlich von einer Suchmaschine zu erfüllen sind. Die Aufgaben sind folgende:

- Akquirierung der Dokumente
- Indexierung der Dokumente und Aktualisierung des Indexes
- Speicherung der Informationen
- Anfragenbearbeitung und grafische Benutzerschnittstelle

Für die einzelnen Aufgaben sind jeweils verschiedene Kernkomponenten zuständig. Die SuchEngine steuert die WebCrawler-Aktivitäten und ist zuständig dafür, welche neuen Dokumente untersucht werden sollen. Die SuchEngine stößt auch die Akquirierung der Dokumente durch die Agents an. In der SuchEngine wird die weitere Vorgehensweise für jedes einzelne Dokument entschieden. Die Datenbank speichert Dokumenten-Metadaten, die Verbindungen zwischen den Dokumenten und dem Volltextindex darstellen. Die Agents sind dafür verantwortlich, die von der Suchmaschine beauftragten Dokumente aus dem Netz zu holen. Die QueryEngine bietet den Abfrageservice im Netz an.

4.4.3 SuchEngine

Die SuchEngine ist für die Indexierung der Dokumente und die Aktualisierung des Indexes zuständig. Die SuchEngine verfolgt dabei folgende Vorgehensweise. Ausgehend von einer bestehenden Dokumentenmenge werden die nach extern führenden Links* extrahiert und diese Links verfolgt, die zu einem neuen Dokument führen. Neue Dokumente werden zum Index hinzugefügt und anschließend wiederholt

sich der gesamte Prozess. Die Suchmaschine entscheidet nicht nur, welche Dokumente besucht werden sollen, sondern auch die Arten von Dokumenten. Nicht indexierbare Dokumente – z.B. Bilder oder nicht unterstützte Dateiformate - werden ignoriert und erst gar nicht abgerufen. Die SuchEngine unterstützt generell zwei Arbeitsweisen. Den Indexing Modus und die Realtime Suche.

4.4.3.1. Indexing Mode

Ziel der Indexierung ist es einen möglichst umfassenden Webindex aufzubauen. Die verfolgte Strategie ist ein "breadth-first"*-Algorithmus, um sicherzustellen, dass jeder Server mindestens ein Dokument im Index hat. Dabei wird jedes Mal, wenn ein "neuer" Server gefunden wurde dieser Server* in die Liste der zu besuchenden Server gestellt. Bevor weitere Dokumente geholt werden, wird von dem neuen Server ein Dokument geholt und indexiert. Wurden alle bekannten Server besucht, durchsucht der WebCrawler sequentiell die Liste der zu besuchenden Server nach neuen Servern und die Prozedur wiederholt sich.

4.4.3.2. Realtime Suche

Bei der Realtime Suche will der WebCrawler den Benutzer dahingehend bei der Informationsfindung unterstützen, indem er roboterartig das Internet nach relevanten Informationen durchsucht. Das geschieht dann nicht im eigenen Index, sondern "online" im Internet. Zunächst wird eine Abfrage gegen den eigenen Index gestartet, um eine Anfangsdokumentenliste oder auch nur ein Dokument zu finden. Aus der Liste werden die relevantesten markiert und die unerforschten verfolgt. Werden neue Dokumente gefunden, so werden diese zum Index dazugefügt und die Abfrage wird wiederholt. Dieser Prozess wird solange wiederholt, bis der WebCrawler für den Benutzer genügend Dokumente gefunden hat, oder ein Zeitlimit erreicht wurde. Problematisch ist, dass der WebCrawler jedem Link folgt. Dabei kann eventuell ein irreführender Pfad entstehen, da als Maß für die Relevanz die Ähnlichkeit zwischen dem "Anchor-Text"* und der Abfrage angenommen wird. Anchor-Texte* beschreiben aber äußerst kurz, wenn überhaupt, wohin sie führen und werden dann gegen einen Volltextindex abgeglichen.

4.4.4 Agents

Die Agents sind Prozesse* mit der festgelegten Aufgabe, die Dokumente aus dem Internet zu holen, die von der SuchEngine zur weiteren Bearbeitung angefordert werden. Zur Realisierung dieser Aufgabe verwenden die Agents die "libwww"* vom W3C-Konsortium als Basistechnologie. Der WebCrawler verwendet maximal 15 solcher Agenten.

4.4.5 Datenbank

Die Datenbank ist der Informationsspeicher schlechthin. Sie enthält den Volltextindex und eine Repräsentation des Internet als Graph. Der Index wird nach dem Vector Space Modell* gespeichert. Um Anfragen an den Index zu beschleunigen wird der Index invertiert gespeichert. Indexiert werden der Dokumententitel und auch der Dokumenteninhalt. Bei der Indexierung unterteilt ein lexikalischer Analyzer* das Dokument in einen Wortstrom, der Tokens* aus dem Titel und dem Dokumenteninhalt

enthält. Dieser Strom wird durch eine sogenannte "Stop-List"* gefiltert. Die Stop-List beinhaltet gewöhnliche und gebräuchliche Wörter, die keinen Informationsgehalt haben. Die übrigen "wichtigen" Wörter werden nach der inversen Dokumentenhäufigkeit – wie in 4.3.2 bereits beschrieben – gewichtet. Neben dem Volltextindex werden in der Datenbank auch Metainformationen über die indexierten Dokumente gespeichert. Zu den einzelnen Dokumenten werden Informationen über Links, Server und URL gespeichert. Die URL wird nicht textuell gespeichert, sondern aufgespaltet in Objekte. Jedes URL Objekt wird letztlich als btree* gespeichert um den Zugriff auf unbesuchte Server zu vereinfachen.

4.4.6 QueryEngine

Die QueryEngine stellt die Schnittstelle zum Index im WWW dar. Der Benutzer kann eine Anfrage eintippen und überlässt der QueryEngine die weitere Bearbeitung. Das Abfragemodell entspricht der Vorgehensweise beim Vector Space Modell auf einer Volltextdatenbank. Als Relevanzkriterium wird ein einfaches, leistungsstarkes Modell verwendet. Die Relevanz *r* wird nach der folgenden Formel ermittelt

$$r = \frac{(alle\ Wörter\ in\ der\ Abfrage) + (Wortgewicht\ im\ Dokument) + (Gewichtung\ in\ der\ Abfrage)}{Anzahl\ der\ Wörter\ in\ der\ Abfrage}$$

Die Bestimmung der Relevanz erfolgt nach einem einfachen Modell. Wie aber die Erfahrung zeigt, sind einfache Modelle im Endergebnis meist mindestens genauso gut, wie komplexere Modelle.

4.4.7 Fazit

Zentraler Ansatzpunkt des WebCrawlers ist die inhaltsbasierte Indexierung. Es stellt sich heraus, dass dieser Ansatz der einzig gangbare ist, um einen guten Index zu erhalten. Viele andere Suchmaschinen verfolgten nur die Indexierung der Titel von Dokumenten. Problematisch daran ist, dass 20% aller Seiten entweder keinen Titel haben, oder nur einen nichtssagenden. Die Qualität des Indexes leidet an dieser Stelle natürlich erheblich. Andererseits ist es erstaunlich, dass der verfolgte "breadth-first"* Ansatz einen brauchbaren, qualitativ hochwertigen Index ergibt. Es werden ja eigentlich pro Server im Internet nur eine Seite gespeichert und die Durchdringung bei Gelegenheit intensiviert. Ebenfalls aufgefallen ist, dass die Realtime Suche eine extrem ressourcenfressende Angelegenheit ist. Für jede Anfrage wird das Internet mit Anfragen an die einzelnen Server überhäuft. Das ist auch der Grund dafür, dass die Realtime Suche nicht mehr angeboten wird.

5. Literaturverzeichnis

[AUT00] Autonomy, Unternehmensgeschichte, http://www.autonomy.com/company/background.htm, 2000

[ADC99a] Autonomy, Produktdokumentation, Knowledge Server User Manual zur Version 1.9.1, 1999

[ADC99b] Autonomy, Produktdokumentation, ODBCFetch User Manual zur Version 1.9.1, 1999

[ATW00a] Autonomy, Technology White Paper, http://www.autonomy.com/tech/wp.html, 2000

[ATW00b] Autonomy, Technology White Paper, http://www.autonomy.com/tech/wp.html, Seite 6-9, 2000

[ATW00c] Autonomy, Informationen über die Autonomy Produkte auf der Website, http://www.autonomy.com, 2000

[BME00] Bundesverband Materialwirtschaft, Einkauf und Logistik – BME, Spezifikationen und Informationen über BMECat, http://www.bmecat.de/bmecat/, 2000

[CAW99a] Cawsey, Alison: Natural Language Processing Course, http://www.cee.hw.ac.uk/~alison/nl/lectures/l22sh/l22sh.html, dort: Boolean Retrieval, Heriot-Watt-Universität Edinburgh, 1999

[CAW99b] Cawsey, Alison: Natural Language Processing Course, http://www.cee.hw.ac.uk/~alison/nl/lectures/l22sh/l22sh.html, dort: Vector Space Retrieval, Heriot-Watt-Universität Edinburgh, 1999

[CAW99c] Cawsey, Alison: Natural Language Processing Course, http://www.cee.hw.ac.uk/~alison/nl/lectures/l22sh/l22sh.html, dort: Probabalistic Retrieval, Heriot-Watt-Universität Edinburgh, 1999

[CAW99d] Cawsey, Alison: Natural Language Processing Course, http://www.cee.hw.ac.uk/~alison/nl/lectures/l22sh/l22sh.html, dort: Relevance Feedback, Heriot-Watt-Universität Edinburgh, 1999

[CAW99e] Cawsey, Alison: Natural Language Processing Course, http://www.cee.hw.ac.uk/~alison/nl/lectures/l22sh/l22sh.html, dort: Ranked Retrieval, Heriot-Watt-Universität Edinburgh, 1999

[CCM00] Conrad.com AG, Unternehmensprofil, Hirschau, 2000

[CIF00] CoMedia – Hennig, M.: Internet Schnittstellen – Schnittstellend es Conrad Internetauftritts zu den Core-Systemen der Conrad-Holding, Hirschau, 2000

[CIN97] Conrad Electronic GmbH, Internetauftritt, http://www.conrad.de, dort: "Kontakte", "Conrad Historie", 1997

[CIS] Centre for Interactive Systems Research: The Probabilistic Retrieval Model, http://web.cs.city.ac.uk/research/cisr/okapi/prm.html

[CNM99] Conrad Neue Medien GmbH, Unternehmensprofil, Hirschau, 1999

[CON98] Conrad Electronic GmbH, Broschüre: 75 Jahre Conrad Electronic, Hirschau, 1998

[CT99a] Jörg, M.: Doppelgänger gesucht – Ein Programm für kontextsensitive phonetische Textumwandlung, in Magazin c't Ausgabe 25/99, S.252, 1999

[CT99b] Jörg, M.: Sourcen zum Artikel Doppelgänger gesucht unter ftp://ftp.heise.de/pub/ct/listings/phonet.zip

[DRA96a] Drabenstott, K.M.: Enhancing a new design for subject access to online catalogs, Library Hi Tech 14, Seite 87-109, 1996

[DRA96b] Drabenstott, K.M.; Weller, M.S.: Failure analysis of subject searches in a test of a new design for subject access to online catalogs, J. Amer. Soc. Inform. Sci. 47, Seite 519-537, 1996

[DRA96c] Drabenstott, K.M.; Weller, M.S.: The exact-display approach for online catalog subject searching, Inform. Proc. Manag. 32, Seite 719-745, 1996

[DUM44] Dumas, Alexandre: Les Trois Mousquetaires, 1844

[EFE99] EFEU-Elektronik GmbH, Joachim Katz, Dokument: Artikel finden im Internet - Leitfaden für ein kundenfreundliches Suchsystem im Internet-Auftritt der Firmen Conrad Electronic und Völkner, Berlin, 1999

[GUA99] Guach, S.: Search improvement via automatic query reformulation, ACM Trans. Inform. Syst. 9, 1991

[HIL95] Hildreth, Charles R.: Online Catalog Design Models: Are We Moving in the Right Direction ?, http://www.ou.edu/faculty/H/Charles.R.Hildreth/clrthree.html, Oklahoma, 1995

[KNO94a] Knorz, Gerhard: Automatische Indexierung, http://www.iud.fh-darmstadt.de/iud/wwwmeth/publ/skript/autind94/paper1.htm, dort: 3.1. Boole'sches Retrieval versus Rankingverfahren, Universität Potsdam, 1994

[KNO94b] Knorz, Gerhard: Automatische Indexierung, http://www.iud.fh-darmstadt.de/iud/wwwmeth/publ/skript/autind94/paper1.htm, dort: 3.3.1. Das Vektormodell, Universität Potsdam, 1994

[KNO94c] Knorz, Gerhard: Automatische Indexierung, http://www.iud.fh-darmstadt.de/iud/wwwmeth/publ/skript/autind94/paper1.htm, dort: 3.4. Ein einfaches Verfahren der Termgewichtung, Universität Potsdam, 1994

[KNO94d] Knorz, Gerhard: Automatische Indexierung, http://www.iud.fh-darmstadt.de/iud/wwwmeth/publ/skript/autind94/paper1.htm, dort: 3.3.2. Probabilistische Verfahren, Universität Potsdam, 1994

[KNU97] Knuth, D.E.: The art of Computer Programming – Sorting and Searching 2nd ed., ISBN 0-201-89685-0, Seite 394-395, 1997

[LEE95a] Lee, J.H.: Analyzing the Effectiveness of Extended Boolean Models in Information Retrieval, Seite 3f, http://ncstrl.cs.cornell.edu:80/Dienst/UI/1.0/Display/ncstrl.cornell/TR95-1501, Cornell Universität, März 1995

[LEE95b] Lee, J.H.: Analyzing the Effectiveness of Extended Boolean Models in Information Retrieval, Seite 4ff, http://ncstrl.cs.cornell.edu:80/Dienst/UI/1.0/Display/ncstrl.cornell/TR95-1501, Cornell Universität, März 1995

[OAR99] Oard, Douglas W.: The Boolean Retrieval Model, http://raven.umd.edu/courses/708a/fall99/notes/708f992/ppframe.htm, Maryland, 1999

[ODC99a] Oracle Corporation, Oracle8 Documentation – Oracle 8i Concepts Release 8.1.5, dort: 26 Parallel Execution, Redwood City, 1999

[ODC99b] Oracle Corporation, Oracle8 Documentation – Oracle 8i Parallel Server Concepts and Administration Release 8.1.5, Redwood City, 1999

[PFP00] PixelFactory GmbH, Rene Herzer, Dokument: CONRAD ONLINE AG Pflichtenheft Version 1.0 vom 28.02.2000, Offenbach, 2000

[PIN94] Pinkerton, B.: Finding what people want – Experiences with the WebCrawler, http://www.thinkpink.com/bp/WWW94.html, 1994

[PIP99] PixelPark AG, Susan Plaumann, Dokument: Projekt-Architektur, Berlin, 2000

[POO99a] Poo, D.C.C.; Toh, T.K.; Khoo, C.S.G.: Design and implementation of the E-referencer, Data & Knowledge Engineering 32, Seite 199-201, 2000

[POO99b] Poo, D.C.C.; Toh, T.K.; Khoo, C.S.G.: Design and implementation of the E-referencer, Data & Knowledge Engineering 32, Seite 202, 2000

[PSW99] PixelPark AG, Bernd Schmeil, Dokument: CONRAD Schlagwortsuche, Berlin, 1999

[PVT99] PixelPark AG, Bernd Schmeil, Dokument: CONRAD Volltextsuche, Berlin, 1999

[RAN96a] Rankins, R.; Garbus, J.R.; Solomon, D.; McEwan, B.W.: Sybase SQLServer unleashed, ISBN 0-672-30909-2, SAMS Publishing Indianapolis, 1996

[RAN96b] Rankins, R.; Garbus, J.R.; Solomon, D.; McEwan, B.W.: Sybase SQLServer unleashed, ISBN 0-672-30909-2, Seite 546ff, SAMS Publishing Indianapolis, 1996

[ROB76] Robertson, S.E.; Sparck, J.K.: Perevance weighting of search terms, Journal of the American Societey for Information Science, 27, Seite 129-146, 1976

[ROU00] Rougier, J.: Introduction to Dynamic Linear Modelling, Seite 3ff, 2000

[SAG99a] Software AG – Oppel, K.: Tamino White Paper, Oktober 1999

[SAL82a] Salton, G.; Fox, E.A.; Wu, H.: Extended Boolean Information Retrieval, Seite 2ff, http://ncstrl.cs.cornell.edu:80/Dienst/UI/1.0/Display/ncstrl.cornell/TR82-511, Cornell Universität, August 1982

[SAL82b] Salton, G.; Fox, E.A.; Wu, H.: Extended Boolean Information Retrieval, http://ncstrl.cs.cornell.edu:80/Dienst/UI/1.0/Display/ncstrl.cornell/TR82-511, Cornell Universität, August 1982

[SDC99] Software AG: Dokumentation zum Tamino Informationsserver, 1999

[SUF99a] Karzauninkat, Stefan: Die Suchfibel – Wie findet man Informationen im Internet, http://www.suchfibel.de/5technik/systeme.htm, Leipzig, 1999

[SUF99b] Karzauninkat, Stefan: Die Suchfibel – Wie findet man Informationen im Internet, http://www.suchfibel.de/5technik/sammeln.htm, Leipzig, 1999

[SUF99c] Karzauninkat, Stefan: Die Suchfibel – Wie findet man Informationen im Internet, http://www.suchfibel.de/5technik/struktur.htm, Leipzig, 1999

[SUN97] SUN Microsystems: The Sun Enterprise Cluster Architecture – Technical White Paper, http://www.sun.com/clusters/wp.html, 1997

[TON87] Tong, R.M.; Applebaum, L.A.; Askmann, V.N.; Cunningham, J.F.: Conceptual information retrieval using RUBIC in Yu, C.T.; Van Rijsbergen, C.J.: Proceedings of the Tenth Annual International ACM SIGIR Converence on Research and Development in Formation Retrieval, Seite 247-253, 1987

[VIC87] Vickery, A.; Brooks, H.M.: PLEXU – The expert system for referral, Inform. Proc. Manag. 23, Seite 99-117, 1987

[WIR00a] Silberman, Steve: The Quest for Meaning, Magazin Wired, http://www.wired.com/wired/archive/8.02/autonomy.html, Februar 2000