Texttechnologie in der Sekundarstufe II

Claus-Peter Becke

7. September 2008

Inhaltsverzeichnis

1 Einführung **11**

I Maschinelle Verarbeitung natürlicher Sprache **15**

2 Textannotation **17**
2.1 Basis-Wissen . . . 19
2.1.1 HTML (statisch) . . . 19
2.1.2 HTML (interaktiv) . . . 26
2.1.3 Client-Server-Architekur . . . 31
2.2 XML-Grundlagen . . . 33
2.2.1 Das XML-Dokument und seine Grammatik als Schema oder Dokument-Typ-Definition . . . 33
2.2.2 Die Transformation durch XSLT-Dokumente . . . 36
2.3 Die TEI-TagSets 'Prosa' . . . 37
2.3.1 Der TEI-Header . . . 38
2.3.2 Das Body-Tag . . . 39

3 Suchmaschinen **43**
3.1 Einleitung: Information Retrieval . . . 43
3.2 Der Aufbau von Suchmaschinen . . . 44
3.2.1 Das Webcrawler-System . . . 44
3.2.2 Datenanalyse und Aufbau der Datenstruktur . . . 46
3.2.3 Relevanzgewichtung . . . 52
3.2.4 Der Suchvorgang . . . 54

4 Formale Sprachen **55**
4.1 Grundbegriffe . . . 55
4.1.1 Mengenlehre und Algebra . . . 55
4.1.2 Grundbegriffe der Theorie Formaler Sprachen . . . 61
4.2 Grammatiken . . . 61
4.3 Automaten . . . 65
4.3.1 Deterministische endliche Automaten (DEA) . . . 65
4.3.2 Nicht-deterministische endliche Automaten (NEA) . . . 69
4.4 Syntaxanalyse natuerlich-sprachlicher Saetze . . . 71

A Anhang **77**
A.1 Pagerank . . . 77
A.2 Formale Sprachen . . . 103

Abbildungsverzeichnis

2.1 Beispiel eines einfachen HTML-Codes 20
2.2 Darstellung dieses einfachen Codes im Browser 21
2.3 Die Menueleiste der Homepage der Domschule-Website 23
2.4 Ein Ausschnitt des HTML-Quellcodes der Homepage der Domschul-Website 24
2.5 Verknüpfungsziel des Verknüpfungstextes 'Schule' der Startseite der Domschule-Website 25
2.6 Ein HTML-Formular 27
2.7 Der zum Formular gehörige Seitenquelltext 28
2.8 Das ausgefüllte Formular 29
2.9 Die Ausgabe nach Verarbeitung der Formulardaten in antwort.php . . 30
2.10 Der PHP-Quellcode der Datei antwort.php 31
2.11 Server-Client-Archittekturen 32
2.12 gruss.xml 33
2.13 gruss.dtd 35
2.14 gruss.xsd 35
2.15 gruss.xsl 36
2.16 gruss.xml unter Verwendung einer Schema-Datei gruss.xsd 37
2.17 Der TEI-Header in der IDE des XML-Editors 'oXygen' 38
2.18 Das Tag <publicationStmt> im TEI-Header 39
2.19 Grundlegende Tags zur Annotation von Prosa-Texten 40
2.20 Erläuterungen zu den Tags der syntaktischen Annotation auf der TEI-Website http://www.tei-c.org 42
2.21 Syntaktische Annotationen 42

3.1 Das Webcrawler-System 45
3.2 Dokumentenanalyse 46
3.3 Parser 47
3.4 Datennormalisierung 48
3.5 Gewichtetes Vektorraummodell mit zwei Termen 53

4.1 Steuereinheit und Lesekopf eines einseitigen endlichen Automaten . . 66

A.1 Beweis der Aequivalenz linearer Grammatiken und endlicher Automaten 1 103
A.2 Beweis der Aequivalenz linearer Grammatiken und endlicher Automanten 2 104
A.3 Beweis der Aequivalenz linearer Grammatiken und endlicher Automaten 3 105

Tabellenverzeichnis

3.1 Einfaches Beispiel eines direkten Index ohne Codierung 51
3.2 Direkter Index im codierten Zustand 51
3.3 Invertierter Index 51

4.1 Phrasenstrukturregeln 61
4.2 Lexikalische Regeln 62
4.3 Ableitung fuer Beispielsatz (10) 63
4.4 Ableitung fuer Beispielsatz (22) 63
4.5 Zustandstafel zur Uebergangsfunktion im DEA aus Beispiel 4.3.1.1 . . 67
4.6 Zustandstafel zur Uebergangsfunktion im NEA aus Beispiel 4.3.2.1 . . 70
4.7 Zustandstafel zu Beispiel 4.4.1 72
4.8 Regeln der Grammatik zu Beispiel 4.4.1 73
4.9 Parallelen zwischen Grammatik-Regeln und Uebergangsfunktionen . . 73

Vorwort

Dieser Reader dient als einen Kurs begleitendes Material einer Einführung in die Texttechnologie in der gymnasialen Oberstufe. Dass es überhaupt möglich geworden ist, diesen Kurs als einen Projektkurs im 13. Jahrgang der Domschule in Schleswig im Schuljahr 2008/09 anzubieten, ist der Offenheit für neue Ideen des Schulleiters der Domschule, Herrn Oberstudiendirektor Georg Reußner, geschuldet, dem ich an dieser Stelle dafür ganz herzlich danken möchte. Der Kurs ist ein Experiment in mehrfacher Hinsicht. Zum Einen hat es meines Wissens außerhalb des Informatik-Unterrichts bis dahin noch kaum Ansätze dazu gegeben, die Theorie der Formalen Sprachen im gymnasialen Oberstufenunterricht zu behandeln. Zum Anderen sind manche der Untersuchungsobjekte dieses Kurses bisher kaum zum Unterrichtsgegenstand gemacht worden, weil sie erst seit wenigen Jahren einer breiteren Öffentlichkeit bekannt werden. Das gilt insbesondere für die Verwendung der XML-Sprachenfamilie, die seit etwa zehn Jahren die Arbeit im Bereich Webtechnologie zu dominieren begonnen hat, das gilt aber auch und vor allem für die Verwendung der Werkzeuge, die von der Text Encoding Initiative zur Verfügung gestellt werden, zumal letztgenannte Instrumente bis dahin vorwiegend in Universitäten und Forschungseinrichtungen zum Einsatz gekommen sind. Mit TagSets der Text Encoding Initiative auch in der Schule zu arbeiten, ist ein Wagnis mit offenem Ausgang. Sollte das Experiment gelingen, dann könnte dieser Kurs ein Pilotprojekt für die Einbindung der Texttechnologie in das sprachliche Profil in der reformierten Oberstufe sein.

Größere Textteile des Readers habe ich erstmals Studierenden eines dem Thema gewidmeten Didaktik-Kurses während des Sommersemesters 2008 im Germanistischen Seminar der Christian-Albrechts-Universität zu Kiel vorgelegt, die diese erste Fassung intensiv mit mir diskutiert haben. Ich danke Maike Lühr und Dennis Hübert für ihre engagierte Mitarbeit und ihre Anregungen zur Verbesserung des Textes.

Dieser Text ist in LaTeX gesetzt. Dieses Satzsystem benutze ich hier zum ersten Mal; dass ich überhaupt auf die Idee gekommen bin, dies zu tun, verdanke ich Herrn Stefan Wille und seiner (inzwischen von mir geteilten) Begeisterung für LaTeX. Herr Wille hat mir den Einstieg in die Arbeit mit LaTeX durch viele wertvolle Tipps erheblich erleichtert. Vielen Dank dafür.

Gintoft, im August 2008

Claus-Peter Becke

1 Einführung

Wenn Handbücher erscheinen, die in die verschiedenen Arbeitsfelder eines Themengebietes einführen wollen, darf dies als sicheres Indiz dafür gewertet werden, dass sich dieses Themenfeld zu etablieren begonnen hat. Zu diesen Themenfeldern darf inzwischen wohl auch die Texttechnologie gerechnet werden, und dies nicht nur deswegen, weil im Stauffenberg-Verlag im Jahr 2004 ein von Henning Lobin und Lothar Lemnitzer herausgegebenes Handbuch erschienen ist, dass mit den unterschiedlichen Forschungsfeldern innerhalb der Texttechnologie bekannt machen will. Zu diesen Feldern zählen Untersuchungen der Hypertext-Struktur auch mit Hinblick auf die Frage, inwieweit Hypertexte als Lernumgebungen geeignet sind. An solche Fragestellungen schlließen sich zwanglos Untersuchungen der Frage an, welche Möglichkeiten der Texterschließung sich durch Textauszeichnung bzw. -annotation eröffnen. Damit werden sogleich sehr grundsätzliche Fragestellungen ins Spiel gebracht. Wie verändert die Tatsache, dass Texte als semi-strukturierte Daten[1] anzusehen sind, unseren Umgang mit Texten? Nachdem Texte in vergangenen Jahrzehnten primär Gegenstand hermeneutischer Auslegung gewesen sind, also vor allem Interpretationsgegenstand innerhalb geisteswissenschaftlicher Fächer, bahnt sich insofern ein Wandel an, als Texteigenschaften messbar zu werden beginnen, wenn Texte elektronisch publiziert werden. Die Instrumente der Textannotation erlauben, Texte zunehmend differenziert auszuzeichnen, in welche Untersuchungsrichtung auch immer. Diese Veränderungen eröffnen Perspektiven, die von der Texttechnologie reflektiert werden. So weisen Lobin/Lemnitzer auf die Methoden quantitativer Textuntersuchung hin, verweisen auf die Grundlagen dieser Untersuchungen in Theorien Formaler Sprachen, werfen einen Blick auf linguistische Korpusanalysen und Parsing-Algorithmen, um last but not least den Bereich des Data-Mining, Information-Retrieval, der Informationserschließung in den Blick zu rücken.

Nun lässt sich zu Recht behaupten, dass all diese Untersuchungen und Themenbereiche nichts vollständig Neues sind. In der Theoretischen Informatik werden die Eigenschaften Formaler Sprachen seit langem mit beeindruckenden Ergebnissen untersucht. Die Computerlinguistik hat in den vergangenen zwei Jahrzehnten dramatische Fortschritte in der Entwicklung von Syntax- und Semantik-Theorien zu verzeichnen, die trotz aller noch bestehenden Probleme erhebliche Verbesserungen

[1] Im Gegensatz zu in Datenbanken gespeicherten Daten etwa, die als vollständig strukturiert angesprochen werden dürfen, weil die Daten in Matrizen zugänglich sind, so dass jedes einzelne Datum (z. B. das Baujahr eines Gebrauchtwagens in einer entsprechenden Datenbank), das innerhalb der Zelle einer Matrix abgelegt ist, eindeutig über Indizes identifizierbar und interpretierbar ist. Das gilt z. B. nicht von in hypertextuellen Umgebungen abgelegten Texten, die allenfalls durch HTML- oder XML-Tags ausgezeichnet sind. Da natürlich-sprachliche Texte, die in HTML oder XML ausgezeichnet sind, durch die verwendeten Tags nur schwach strukturiert sind, kann hinsichtlich dieser Daten von eindeutiger Identifizierbarkeit bzw. Interpretierbarkeit der Inhalte nur bedingt gesprochen werden.

im Bereich der maschinellen Sprachverarbeitung zu Tage gefördert haben. Die Webtechnologie entwickelt eine Dynamik, die nicht auf die Texttechnologie wartet.

Dennoch hat die Texttechnologie etwas zu bieten, was ihre Existenzberechtigung sichert, wie ich denke. Sie fördert eine integrative fächerübegreifende Fragerichtung und schafft so auch die Möglichkeit, in der Verbindung der Fragestellungen unterschiedlicher Fächer Ansatzpunkte für neue Einsichten zu eröffnen, die ohne diese integrative Perspektive so möglicherweise nicht zugänglich wären.

Darin liegen auch Chancen insbesondere für die Didaktik. Wir haben uns daran gewöhnt, Lerner unterschiedlichen Typen zuzuordnen. Von dem Einen sagen wir, dass er vor allem über Sprachbegabung verfüge, der Anderen attestieren wir eine ausgesprochen mathematisch-naturwissenschaftliche Begabung.

Wenn jemand leicht Sprachen lernt, dann besitzt er vermutlich sehr unterschiedliche, schwer messbare Fähigkeiten. Zunächst bedarf es eines guten bis sehr guten Gedächtnisses, um Vokabeln behalten zu können. Um sie leicht erlernen zu können, bedarf es einer schnellen Auffassungsgabe. Darüber hinaus sollen auch morphologische Syteme erworben werden, die Deklination und Konjugation prägen, das Gefühl für die syntaktischen Besonderheiten einer Fremdsprache will entwickelt werden. Schnelle Auffassungsgabe und gutes Gedächtnis garantieren neben Fleiß und der notwendigen Hartnäckigkeit Erfolg in dieser Phase propädeutischen Sprachenlernens. Parallel dazu gilt es gleichzeitig die Fähigkeit zu entwickeln, den Sinn fremdsprachlicher Texte zu erfassen. Die hier benötigten Fertigkeiten zu beziffern, fällt schon schwerer. Was ist es, das jemanden in die Lage versetzt, sich in einen Text „einzufühlen“, sich in dessen Sinn hineinzutasten, ohne dafür einen Algorithmus, salopp formuliert, ein Rezept, parat zu haben? Wie weit muss das Verständnis für den Menschen und sein Verhalten reichen, um fähig zu sein, Literatur zu verstehen? Wieviel Interesse für Geschichte muss man aufbringen und über wieviel historisches Wissen muss man verfügen, um Texte vergangener Epochen würdigen zu können?

Um als mathematisch-naturwissenschaftlich begabt zu gelten, ist kein Interesse für Geschichte und diesbezüglicher Lerneifer vorausgesetzt. Ein Lerner, der Begabungen in diesem Bereich mitbringt, erfasst schnell, wie sich Probleme algorithmisch lösen lassen. Dazu gehören Einfühlungsvermögen und Phantasie. Ein Problem selbständig zu lösen, erfordert ein hohes Maß an Kreativität. Auf der Grundstufe des Erwerbs entsprechender Qualifikationen ist zunächst Präzision gefragt. Nur wer Freude daran hat, Quantitäten exakt zu bestimmen, kann sich an der Lösung mathematischer Probleme erfreuen. Wer in diesem Feld gern und leicht lernt, verfügt über die Fähigkeit, die elementaren Regeln schnell zu verinnerlichen, nichts oder nicht viel davon zu vergessen, die Erweiterungen sinnvoll zu dem bereits erworbenen Wissen in Beziehung zu setzen, um so zunehmend komplexe Probleme verstehen und mit den Mitteln der angebotenen Algorithmen lösen zu lernen.

Beide Systeme von Fähigkeiten sind beeindruckende Zeugnisse dessen, wessen Menschen fähig sind, im positiven Sinn. Vielleicht liegen diese Fähigkeiten näher beieinander, als unsere möglicherweise auf Vorurteile gegründeten Ansichten wahrhaben wollen. Zumindest ist es eine lohnende und interessante Aufgabe, dabei mitzuwirken, diese Fähigkeiten entwickeln zu helfen. Die Texttechnologie eröffnet Chancen, dies auch in der Schule zu versuchen. Denn sie bietet die Möglichkeit, sollten Lerner gleichermaßen sprachlich und mathematisch-naturwissenschaftlich talentiert sein, alle

Fähigkeiten integriert weiterzuentwickeln, andererseits, sollte bei dem Einen oder der Anderen jeweils eine der Teilbegabungen überwiegen, durch entsprechende Herausforderungen der jeweils komplementären Fähigkeit zu deren Stärkung beizutragen.

In Anknüpfung an die oben erwähnte Schilderung der Arbeitsfelder der Texttechnologie setzt sich der Reader zum Ziel, Themen aus den Bereichen Textannotation, Informationserschließung und Formale Sprachen miteinander zu verknüpfen. Das einende Band dieses Angebots ist das Parsing-Problem, das in Textannotation, Informationserschließung und Formalen Sprachen gleichermaßen eine Rolle spielt.

Teil I

Maschinelle Verarbeitung natürlicher Sprache

2 Textannotation

Die Entwicklung des Projekts "**T**ext **E**ncoding **I**nitiative"(kurz: **TEI**) begann im November 1987 mit einer Planungskonferenz im Vassar College[1], auf der die ersten Richtlinien festgelegt wurden. Die erste Version der Richtlinien erschien im Juli 1990 unter dem Kürzel TEI P1[2]. Der Grund dafür, die TEI zu begründen, ist in der rasanten Entwicklung der EDV zu suchen. Der Siegeszug des Computers als eines Arbeitsmittels, das unter anderem auch Textverarbeitung enorm vereinfacht, führt zu einer Diversifizierung der Betriebssysteme und Anwendungsprogramme. Die einen arbeiten auf Unix-Maschinen, die anderen auf DOS-Rechnern und setzen die für die unterschiedlichen Betriebssysteme entwickelten Textverarbeitungsprogramme ein. Leider sind die auf unterschiedlichen Systemen geschriebenen Texte untereinander nicht kompatibel. Das erzeugt Redundanz, da ein in Universität A unter Unix entstandenes Produkt nicht mit den Maschinen der Universität B weiter verarbeitet werden kann, da Universität B DOS-Rechner verwendet. So erscheint es als verlockend, nach Wegen Ausschau zu halten, die plattformunabhängige Verarbeitungen ermöglichen.

Ein Ausweg aus dem Dilemma zeichnet sich in den Standards ab, die mit SGML gesetzt worden sind. Was ist SGML? Das Akronym kürzt **S**tandard **G**eneralized **M**arkup **L**anguage ab. SGML ist die Mutter aller Auszeichnungssprachen. Mit SGML ist ein Standard festgelegt worden, aus dem HTML, die **H**yper**T**ext **M**arkup **L**anguage, und später auch XML, die E**X**tensible **M**arkup **L**anguage, entwickelt worden sind. Doch zurück zu SGML. SGML ist eine sehr komplexe und wegen ihrer Komplexität sehr komplizierte Auszeichnungssprache. Als 1990 Tim Berners Lee das World Wide Web „erfand“ und ein Protokoll benötigte, das den Datentransfer kontrolliert, so dass das **H**yper**T**ext **T**ransfer **P**rotocoll ins Leben gerufen werden musste, wurde auch eine Auszeichnungssprache benötigt, die den Benutzern zur Verfügung steht, um die mittels HTTP zu übermittelnden Daten zu kodieren und zu dekodieren. Eben diese Auszeichungssprache ist HTML, die man ein vereinfachtes SGML nennen kann. Die Strukturen, aus denen HTML-Tags abgeleitet sind, sind SGML-Strukturen.

Weil das World Wide Web mit rasanter Geschwindigkeit wuchs, zahlreiche unterschiedliche Browser den Markt überfluteten, von denen jeder HTML ein wenig anders übersetzte, da zudem neue Anforderungen auftauchten wie die Notwendigkeit, Auszeichnungen für die Kodierung unterschiedlicher Datentypen wie *.avi's, *.jpg's oder *.swv's usw. zu implementieren, da weiterhin Mittel und Wege gesucht wurden, Datentransfer mittels interaktiver, dynamischer Websites zu ermöglichen und da für die Lösung all dieser Probleme je individuelle Wege eingeschlagen wurden,

[1] TEI P4, The TEI Consortium, Guidelines for Electronic Text Encoding and Interchange, edited by C.M. Sperberg-McQueen and Lou Bernard, 2002, University of Oxford, S. 6.

[2] P steht für Public Proposal, die sich anschlie?ende Zahl gibt die Versionsnummer an. Vgl. TEI P4, S.3.

hörten HTML-Daten bald auf, uneingeschränkt kompatibel zu sein. Es entstanden unterschiedliche HTML-Dialekte mit all den daraus sich ergebenden Folgen. Das Internet drohte seine Fähigkeit zu verlieren, eine Plattform für den Datenverkehr zur Verfügung zu stellen, die unabhängig von den unterschiedlichen Architekturen der einzelnen Clients universelle Kommunikation gewährleistete. Mit anderen Worten: Das Web hörte auf World Wide zu sein, um es überspitzt zu formulieren. 1998 wurden vom W3C Auswege aus diesem Dilemma gesucht. Der Ausweg heißt XML.

XML zeichnet sich gegenüber dem Vorgänger-SGML-Derivat HTML dadurch aus, dass es flexibel in der Anwendung ist, wie der Name schon sagt: die Auszeichnungssprache ist eben erweiterungsfähig. Jeder Client kann seine eigenen Regeln formulieren, die die Gestalt der von ihm verwendeten Dokumente festlegen. Das hört sich kompliziert an und ist es auch. Denn es gibt in Zukunft nicht mehr nur eine einheitliche Sprache wie HTML, die zwar den Nachteil hat, dass es unterschiedliche Dialekte gibt, aber: man versteht sich doch zumindest im Wesentlichen. Die einfachen Zeiten sind vorbei. Jetzt macht jeder seine eigene Sprache und sorgt dafür, dass es eine Übersetzungsmaschine gibt, die seine Sprache in HTML überträgt. Wessen bedarf es dazu? Jeder, der seine eigene Sprache benutzen möchte, muss eine Grammatik entwerfen, die benötigt wird, um die Regeln zu beschreiben, nach denen die Äußerungen der eigenen Sprache gebildet werden. In XML heißt eine solche Grammatik ein Schema oder eine **D**okument-**T**yp-**D**efinition (DTD), die in eigenen Dateien mit den Extension *.xsd oder *.dtd abgespeichert werden. Innerhalb einer Schema- bzw. Dokument-Typ-Definitions-Datei wird beschrieben, welche Elemente in den unterschiedlichen Äußerungen der Sprache vorkommen dürfen, die ebenfalls in Dateien gespeichert werden. Diese Dateien haben die Extension *.xml. Abschließend muss noch ein Verfahren beschrieben werden, mit dessen Hilfe die Äußerungen der jeweiligen Sprachen, die in XML-Dokumenten gespeichert sind, für alle, die die betreffende Sprache nicht sprechen, sondern z.B. nur HTML oder PDF verstehen, übersetzt werden können. Der Übersetzer ist ein XSLT-Prozessor, ein E**X**tensive **S**tylesheet **L**anguage **T**ransformer, die Vorgänge sind Extensive Stylesheet Language Transformations. Auch diese werden in eigenen Dateien gespeichert, die die Extension *.xsl bzw. *. xslt tragen.

Damit haben wir alle Komponenten beieinander, die benötigt werden, um unsere Daten kompatibel zu halten und deren universelle Darstellbarkeit zu garantieren. Wir brauchen eine **Grammatik (xml-schema.xsd)**, die die Regeln festlegt, denenzufolge die Äußerungen konstruiert sind, wir brauchen diese **Äußerungen** selbst **(xml-document.xml)** und eine **Übersetzung (xml-stylesheet.xsl)**, die deren universelle Verständlichkeit garantiert.

Übertragen wir das alles noch einmal auf die Ebene illustrierender Veranschaulichung: Sie sind Europäer und sprechen ihre jeweilige Nationalsprache, die aber nicht alle Europäer beherrschen. Europa hat sich nun doch darauf geeinigt, Esperanto zur europäischen Universalsprache zu erheben. So benutzt jeder Europäer zwar weiterhin die Grammatik und die dieser folgenden Äußerungen seiner Sprache, besitzt aber auch eine Übersetzungsmaschine, die alle Äußerungen in Esperanto übersetzt, wenn Europäer unterschiedlicher Nationalität zusammenkommen. Was hält den Laden nun zusammen? Esperanto. Esperanto aber ist nur benutzbar, wenn es Regelwerke gibt, die zeigen, wie die unterschiedlichen Nationalsprachen in Esperanto übertragen

werden.

Auf Seiten der XML-Sprachenfamilie bestehen diese Regelwerke aus den Richtlinien, die festlegen, wie die Ausdrücke der Sprachenfamilie gebildet werden. Es handelt sich um eine Metagrammatik, die beschreibt, nach welchen Regeln die Ausdrücke der einzelnen Schema-"Grammatiken", Äußerungs-Dokumente und Übersetzungs - Transformationen konstruiert werden.

Kommen wir zurück zur TEI. Nachdem zunächst gut zehn Jahre lang versucht wurde, die Standards der TEI direkt aus SGML abzuleiten, hat sich nach der Veröffentlichung der W3C-Richtlinien zur Festlegung des XML-Standards ein Wandel vollzogen. Die TEI hat sich im Jahre 2001 den W3C-Standards angeschlossen, so dass die Regeln, auf deren Basis TEI-Tagsets entworfen werden, XML-Regeln folgen. Die Version TEI P4 ist Ausdruck dieser Anpassung. Auf dieser Grundlage hat die TEI TagSets geschaffen, die speziell für die Textauszeichnung bzw. Annotation entwickelt worden sind. Mit anderen Worten: Die TEI hat auf der Grundlage der XML-Metagrammatik eine sehr komplexe Grammatik zur Entwicklung von XML-Dokumenten geschaffen, die erlaubt, die sehr unterschiedlichen Aufgaben im Bereich der Textannotation zu bewältigen. Es sind TagSets für die Bearbeitung von Prosa-, Dramen- und Verstexten, für die Herstellung von Wörterbüchern, die Übertragung gesprochener Sprache oder für die linguistische Textannotation geschaffen worden.

2.1 Basis-Wissen

2.1.1 HTML (statisch)

Grundlagen

Nahezu jeder heutzutage in einem Haushalt stehende Computer verfügt über einen Browser, kaum noch ein Computernutzer verzichtet auf die umfassenden Informationsmöglichkeiten, die das World Wide Web ihm bietet. Browser sind die Werkzeuge, die es uns ermöglichen, die Inhalte des Internets in unseren Rechnern darzustellen. Seitdem das Internet seinen Siegeszug angetreten hat, haben sich alle Komponenten, die dazu benötigt werden, die Inhalte des World Wide Web in den heimischen Rechner zu bringen, in einer kaum vorstellbaren Weise entwickelt. Ich kann mich noch gut daran erinnern, wie zu Beginn der 90iger Jahre während einer vom damals noch so genannten IPTS (Institut für Praxis und Theorie der Schule) angebotenen Fortbildung für Informatik-Lehrer Akustisk-Koppler als der letzte Schrei der DFÜ (Datenfernübertagung) „verkauft" wurden. (Man muss dazu sagen, dass viele Anwender seinerzeit mit Computern arbeiteten, deren Herzstück 8086/88-Prozessoren waren, die einen 16-Bit breiten Datenbus verarbeiteten. Mein erster 1986 gekaufter Rechner war ein Schneider-PC mit externer Festplatte, die über den damals gigantischen Speicherplatz von 20 MB (MegaByte) verfügte und deren Ventilator laut wie ein Staubsauger war. Das Betriebssystem war Microsoft DOS (Disk Operating System), das in der Lage war, unfassbare 640 KB (KiloByte) Arbeitsspeicher zu verwalten. Monitore waren seinerzeit monochrom; bernsteinfarbene, grellgrüne oder graue Schriften bzw. Grafiken bestimmten das Bild, dazugehörige Textverarbeitungssysteme wie Word oder WordPerfect zeichneten auf den schwarzen Monitor

ein z. B. graues Rechteck, in das hinein die Eingaben erfolgten. Unter dem rechteckigen Kasten befand sich die Menüleiste). Aber zurück zum Akustik-Koppler. Bei diesem Gerät handelt es sich um einen Analog-Digitalwandler, ein Gerät, das über zwei Einsteckplätze verfügt, in die Sprech- und Hörmuschel des Telephonhörers hineingedrückt werden. Die auf diesem Weg in den Rechner übermittelten Signale sind das Substrat, über das Netzwerke in diesen Tagen Daten austauschen. In den 90iger Jahren werden Akustik-Koppler zunächst durch noch sehr langsame, dann aber zunehmend schnellere Modems ersetzt, deren Leistungsfähigkeit sich im DSL-Zeitalter in einem kaum vorstellbaren Maß entwickelt hat.

Gleichzeitig mit der Hardware hat sich die Software entwickelt. Auch Browser sind Software. Jeder von uns hat seine besonderen Vorlieben hinsichtlich der Nutzung unterschiedlicher Software-Pakete. Das gilt natürlich auch für Browser. Der eine bevorzugt den Microsoft-Internet-Explorer, andere arbeiten lieber mit Mozilla's Firefox. Trotz erheblicher Unterschiede im Detail haben Browser eines gemeinsam. Sie sind Interpreter, d.h. Übersetzungsalgorithmen für HTML. HTML (für **H**yper**t**ext **M**arkup **L**anguage) ist eine Auszeichnungssprache. Das, was im Internet übertragen werden soll, muss, um im Browser darstellbar zu sein, in HTML ausgezeichnet sein. D.h., dass die Sprachelemente, die HTML bietet, dafür benutzt werden, die Inhalte zu markieren, die wir im Internet kommunizieren möchten, sie so aufzubereiten, dass sie durch Browser gelesen und ausgegeben, eben interpretiert werden können, um mit dieser Wortwahl einen Bezug zu der auf den ersten Blick merkwürdig erscheinenden Bezeichnung 'Interpreter' herzustellen. Im Folgenden sollen die grundlegenden Sprachelemente vorgestellt werden, die die Struktur von Hypertexten festlegen.

Beginnen wir mit einem einfachen Beispiel:

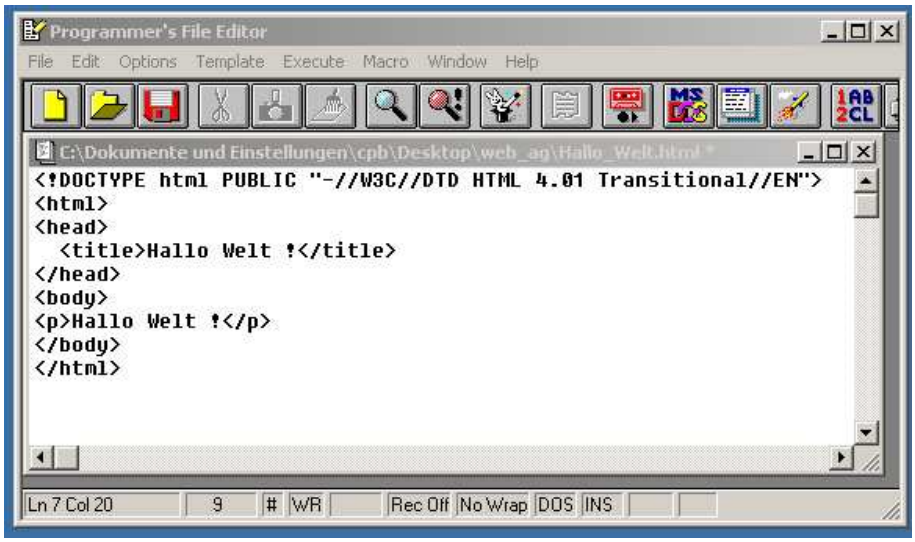


Abbildung 2.1: Beispiel eines einfachen HTML-Codes

Dieser Code zeigt grundlegende Elemente des Codes einer HTML-Datei. Bevor diese erläutert werden, soll zunächst gezeigt werden, was im Browser zu sehen ist, wenn dieser Code verarbeitet worden ist.

Wir sehen, dass nichts weiter als die Zeichenkette 'Hallo Welt !', die innerhalb

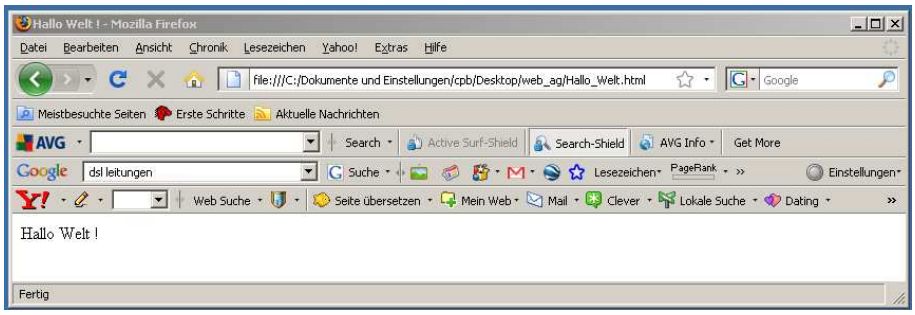


Abbildung 2.2: Darstellung dieses einfachen Codes im Browser

des <p>-Tags (p für paragraph = Absatz) des HTML-Codes steht, im Browser präsentiert wird. Woran liegt das? Betrachten wir der Reihe nach alles, was klärungsbedürftig ist. Zunächst zu der Frage, wie das HTML-Dokument in den Browser geladen worden ist. Im Browser-Adress-Fenster sehen wir den Ort, an dem die Datei gespeichert ist. Dass in diesem Fall die Datei durch Angabe des lokalen Ordners ausgewählt worden ist, innerhalb dessen sie auf dem Datenträger gespeichert worden ist, liegt daran, dass der Browser hier wie eine gewöhnliche Textverareitung benutzt worden ist. Der Browser Mozilla Firefox bietet, wie andere Browser auch, die Möglichkeit, Dateien durch einen 'Datei öffnen'-Dialog, der aus dem Datei-Menü heraus gestartet werden kann, in den Browser zu laden. Das ist natürlich nicht der standardmäßig gewählte Weg des Öffnens eines HTML-Dokuments, der darin besteht, über einen URL (Unified Ressource Locator), die sogenannte IP-Adresse (die Internet-Protokoll-Adresse), die HTML-Datei von irgendeinem Server des WWW herunterzuladen (Dazu später mehr). In unserem Kontext, in dem es erst einmal nur darum geht, die Grundstruktur eines HTML-Dokuments zu klären, können wir uns vorerst damit begnügen, HTML-Dateien auf diesem Weg zu laden. Innerhalb des Adress-Fensters sehen wir am Ende des Pfades, der den Weg angibt, auf dem die Datei gefunden werden kann, den Dateinamen, der aus zwei Teilen besteht, zunächst aus dem Namen des Dokuments im engeren Sinn, gefolgt von einer sogenannten Dateinamensextension, in diesem Fall *.html, die zeigt, von welchem Typ die Datei ist. Diese Dateinamensextensionen sind sehr wichtige Bestandteile eines Dateinamens, weil nur über diese Extensionen Anwendungen wie Browser erkennen, dass Dateien von dem Typ sind, den die Anwendung verarbeiten kann. So weit, so gut. Jetzt also ist die Datei gefunden und geladen.

Dass mit dem Code, der in unserer Beispiel-Datei enthalten ist, nur eine Zeile ausgegeben wird, überrascht vielleicht, weil der Code komplexer zu sein scheint. Wer den Code und die Browser-Ausgabe aufmerksam vergleicht, wird sehen, dass doch mehr ausgegeben wird als das, was im Hauptfenster des Browsers zu sehen ist. Der Text 'Hallo Welt !' wird noch einmal in der blau eingefärbten Titelzeile des Browsers wiederholt. Diese Ausgabe wird durch das <title>-Tag verursacht. Bevor dieses und weitere Tags erläutert werden, einige wenige Worte zu Tags im Allgemeinen. Das englische Wort 'tag' wird unterschiedlich übersetzt. Neben Übersetzungen wie 'Zipfel', 'Spitze', 'Zacken' finden sich auch Übertragungen wie 'Etikett' und 'Anhänger'. Diese Übersetzungen rechtfertigen, dass als Tags gelegentlich die spitzen Klammern angesprochen werden, in die Bezeichner eingeschlossen sind, wie auch die Gesamt-

heit aus Bezeichner und spitzen Klammern. Im Folgenden soll als Tag immer die Gesamtheit aus spitzen Klammern und Bezeichner benannt werden. Nun zu den Tags im Einzelnen.

Blicken wir der Reihe nach auf die unterschiedlichen im Dokument verwendeten Tags. Wir ignorieren vorerst die Prozessoranweisung zwischen spitzen Klammern <!...>, gefolgt von einem Ausrufezeichen, mit der der Code beginnt. Als erstes treffen wir auf das Tag <html>. Dieses Element eröffnet den HTML-Code. Grob gegliedert zerfällt jedes HTML-Dokument in zwei Bereiche, den Kopf, gekennzeichnet durch das <head>-Tag, und den Körper, durch das <body>-Tag markiert. Im Kopf eines HTML-Dokuments können unterschiedliche Informationen Platz finden, Hinweise zum Dokumenten-Titel, wie auch im hier behandelten Code, oder sogenannte Meta-Tags, auf die wir hier nicht näher eingehen. Dem <title>-Tag haben wir uns bereits genauso zugewendet wie dem <p>-Tag, so dass damit bereits alle Tags, die zur Grundausstattung eines HTML-Dokuments gehören, genannt sind. Es bleibt noch, darauf hinzuweisen, dass nur im Dokumenten-Körper, zwischen öffnendem und schließendem <body>-Tag, Informationen abgelegt werden können, die im Browser-Hauptfenster angezeigt werden. Gerade wurde erwähnt, dass es öffnende und schließende Tags gibt. Damit kommen wir zum Schluss dieser Einführung in die grundlegenden Tags von HTML. Etwas, das geöffnet wurde, muss auch wieder geschlossen werden; das gilt für alle Tags, erkennbar daran, dass jedem öffnenden Tag ein Tag der folgenden Form folgt: </Bezeichner>. Ein schließendes Tag ist dadurch gekennzeichnet, dass der führenden spitzen Klammer ein Slash (/) nachgestellt wird.

Hypertext

Was eigentlich ist das Besondere von Hypertext? Dessen Spezifika zeigen sich am ehesten im Vergleich mit seinem Gegenstück, Linearem Text. Linearer Text ist dadurch ausgezeichnet, dass er eben, wie der Name schon sagt, linear angelegt ist und verarbeitet wird. Linearen Text beginnen wir in der Regel links in der ersten Zeile der ersten Seite zu lesen und enden damit, dass wir den Lesevorgang auf der letzen Seite in der letzen Zeile rechts beschließen. So lesen wir in der Regel Bücher, vor allem Romane und Erzählungen, Texte also mit fortlaufend sich entwickelnder Handlung. Eine Biographie z. B. ist so angelegt, dass die Darstellung der Chronologie des Lebens folgt. Zunächst werden Herkunft und Kindheit, dann Erziehung, Ausbildung und Reifejahre, schließlich Lebensende und Wirkung eines Menschen geschildert. Der Leser folgt dieser Anlage in der Regel, weil er hofft, die Darstellung der Reifejahre besser zu verstehen, wenn er zuvor die Beschreibung der Kindheit gelesen hat. So folgt der Leser linear dem Aufbau des Geschilderten in der Hoffnung, das Spätere nach Verarbeitung des Früheren besser einordnen zu können. Hypertexte sind von anderer Machart. Sie empfehlen keinen linearen Lesepfad, sondern überlassen dem Leser, sich seinen eigenen Weg durch das Informationsangebot zu bahnen. Lexika sind so angelegt. Sie sind Hypertexte in Buchform. Wer ein Lexikon zu Rate zieht, liest die Informationen zu einem Stichwort, findet dort Verweise auf weit entfernt platzierte, dem Ausgangsthema mehr oder weniger nah verwandte Stichwörter, die zu lesen möglich, aber nicht zwingend ist. Wie bereits gesagt, es steht jedem frei, den Lesepfad zu wählen, den er einschlagen möchte. Die ideale Repräsentationsform für Hypertexte sind Web-Dokumente. Web-Dokumente sind

Prototypen dessen, was als Hypertext bezeichnet wird, weil die Verarbeitungsbedingungen, unter denen web-gestütze Hypertexte rezipiert werden, eine modulare Verarbeitung der Texte nahelegen. Der Desktop, die Arbeitsfläche, innerhalb derer gelesen wird, ist vergleichsweise klein, sie fasst in der Regel weniger Text als eine Buchseite. So empfiehlt es sich, im Web veröffentlichte Texte in kleinere Bausteine zu zerlegen, die so miteinander verbunden sind, dass sie nacheinander in den Browser geladen werden können. Das ist eine typische Hypertext-Struktur.

Schauen wir uns dazu ein Beispiel an:

Abbildung 2.3: Die Menueleiste der Homepage der Domschule-Website

Ein Blick auf die Startseite der Domschul-Website zeigt, welche Besonderheiten Hypertext gegenüber Linearem Text auszeichnen. Anders als ein Buch bietet die erste Seite nicht nur Titel, Autorennamen und Vergleichbares, später Inhaltsverzeichnisse usf.. Ein Hypertext ist anders strukturiert. Uns werden auf der Startseite schon wichtige Informationen über die Website präsentiert. Darüber hinaus aber finden wir zumeist am linken Rand oder am oberen Rand der Homepage auch Verweise auf weiter führende Informationsangebote. Diese Verweistexte, manchmal auch Bilder oder Grafiken, sind das, was im Hypertext-Jargon Link heißt, eine Verbindung

eben. Über einen Link können weitere Dokumente geöffnet werden, die ihrerseits mehr oder weniger zahlreiche Links enthalten können. Darin zeigt sich die besondere Struktur von Hypertext. Jedes Dokument bietet eine mehr oder weniger große Zahl an Verknüpfungen des aktuellen Dokuments mit anderen, so dass jeder Leser eines Hypertextes die Möglichekeit hat, aus diesem Angebot das ihn Interessierende auszuwählen, um so seinen eigenen Weg durch das Informationsangebot einzuschlagen. Im Zusammenhang mit dieser Wahlfreiheit innerhalb der Informationsverarbeitung ist auch schon kritisch von dem Phänomen gesprochen worden, dass Hypertext-Leser sich in dieser Informationsfülle verlieren können. Dieses Phänomen ist unter dem Stichwort „Lost in Hyperspace“ geläufig. Jedem von uns geschieht es immer wieder, dass er dann, wenn er nicht konzentriert und diszipliniert im Internet „surft“, sich von Angeboten, die auf ursprünglich nicht geplante Wege führen, einfach mitziehen lässt. Zurück zu unserem Thema.

Wie werden solche Angebote wie die von Hypertext unterbreiteten technisch realisiert? Blicken wir wieder auf die bereits untersuchte Homepage der Domschul-Website, diesmal aber auf den zugehörigen HTML-Quellcode:

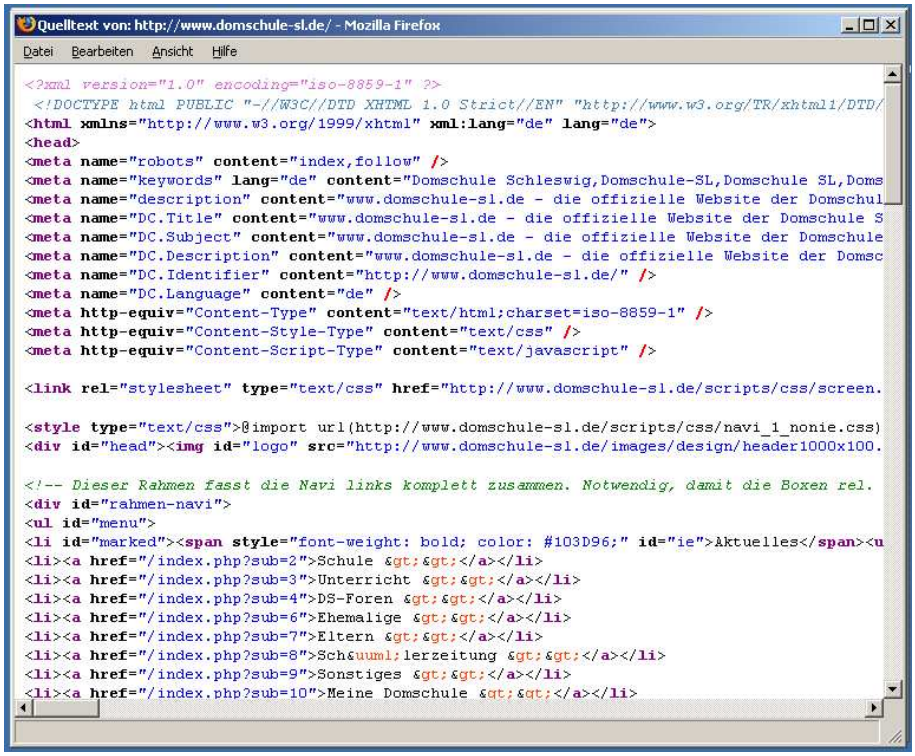


Abbildung 2.4: Ein Ausschnitt des HTML-Quellcodes der Homepage der Domschul-Website

Im unteren Drittel des Screenshots sehen wir Tags <ul> (ul = unordered list) und <li> (li = Listen-Element). In ein -Tag ist ein <a>-Tag (a für anchor = Anker) verschachtelt. Innerhalb des <a>-Tags nun finden wir einen Bezeichner href (href für Hypertext reference = Hypertext Bezugnahme), der vom Tagnamen auch farblich abgehoben ist. Diese Darstellung zeigt etwas, was für Tags typisch ist. In Tags, mit deren Hilfe Text in HTML-Dokumenten ausgezeichnet wird, sind Attribute eingebettet, die Eigenschaften der Elemente festlegen, zu denen sie gehören. In

diesem Fall enthält das <a>-Tag das Attribut href, innerhalb dessen festgelegt ist, wofür das <a>-Tag Anker ist; in der Regel ist es ein Ziel-Dokument, dessen Adresse im href-Attribut angegeben wird. Im Beispiel der Domschul-Homepage sehen wir das Verweisziel index.php?sub=2, dann die schließende spitze Klammer des öffnenden <a>-Tags. Darauf folgt der Verknüpfungstext, der oben in Abbildung ?? links unten zu sehen ist. Die besondere Struktur des Verweises muss uns im Augenblick nicht weiter beschäftigen, schauen wir uns erst einmal an, was geschieht, wenn der Verknüpfungstext angeclickt wird:

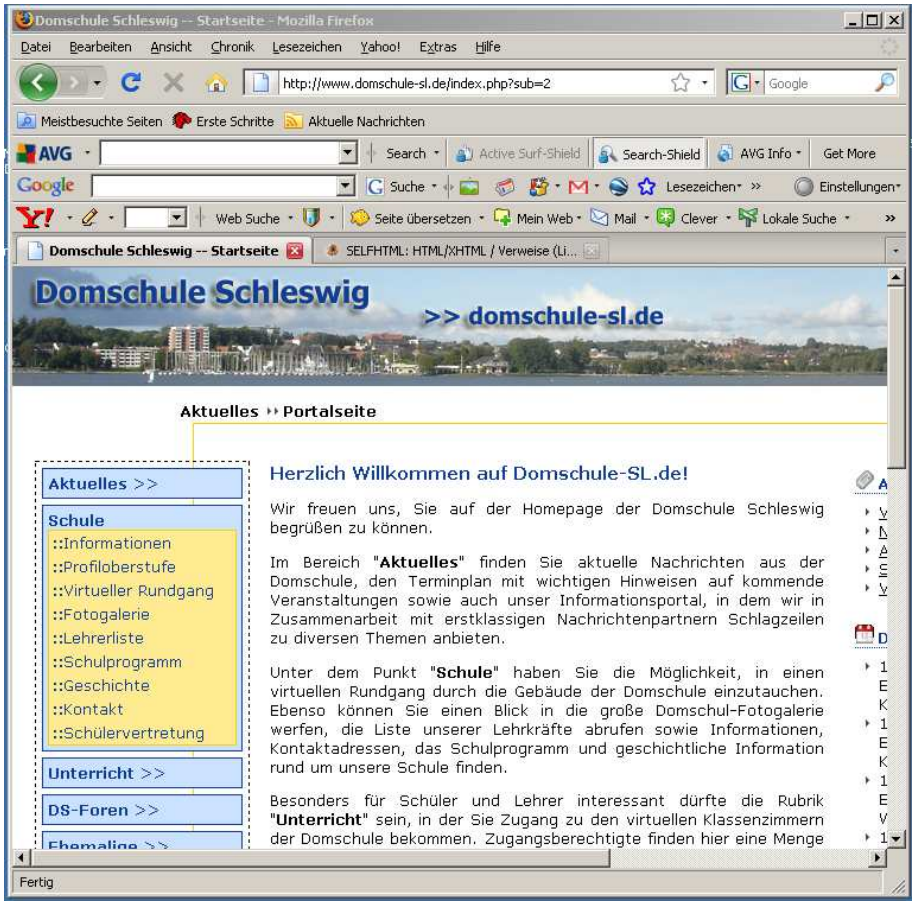


Abbildung 2.5: Verknüpfungsziel des Verknüpfungstextes 'Schule' der Startseite der Domschule-Website

Es zeigt sich, dass ein Untermenü geöffnet wird, das weitere Auswahlmöglichkeiten anbietet. Wer nach weiteren Informationen verlangt, wird weitere Verknüpfungstexte auswählen, sie anclicken, weiter führende Dokumente anzeigen lassen, bis die Informationen eingeholt sind, die Ziel der Recherche gewesen sind. Das ist das Wesen von Hypertext. Informationen werden in kleinere Bausteine zerlegt und diese werden miteinander verknüpft, wie es das oben abgebildete Beispiel zeigt. Mit dem Hinweis auf ein wichtiges eher am Rande erwähntes Ergebnis möge dieser kurze Überblick abgeschlossen werden. Es wurde kurz auf die innere Struktur der in Hypertexten verwendeten Elemente aufmerksam gemacht. Da diese Struktur grundlegend ist und auch in den Weiterentwicklungen von HTML wiederkehren wird, soll an dieser Stelle nochmals darauf hingewiesen werden.

Ein Tag ist grundsätzlich folgendermaßen aufgebaut. Es beginnt mit einer öffnenden spitzen Klammer, dann folgt ein Bezeichner als Elementname. Darauf dann können Attribute folgen, die Werte haben. Auch die Attribute werden durch einen Bezeichner benannt. Direkt auf den Attributnamen folgt ein Gleichheitszeichen, an das sich der in Anführungszeichen eingeschlossene Attribut-Wert anschließt. Hier noch ein Beispiel aus Abbildung **??** zur Illustration:

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de" lang="de">

Es handelt sich um das oben behandelte <html>-Tag, dem in diesem Beispiel drei Attribute angehören. Was es mit Bezeichnern wie 'xmlns' und 'xml' auf sich hat, werden wir weiter unten behandeln.

Wir beschließen hiermit die Einführung in die grundlegenden Verwendungsmöglichkeiten von HTML-Tags. Wer sich im Detail über die Verwendung der zahlreichen HTML-Tags unterrichten möchte, findet dazu im Internet eine vorzügliche Anleitung unter http://de.selfhtm.org.

2.1.2 HTML (interaktiv)

Die Leistungsfähigkeit von HTML erschöpft sich nicht darin, dafür zu sorgen, dass über das Internet verbreitete Texte auf unseren Rechnern zu Hause von uns gelesen werden können. In der Zwischenzeit ist HTML zu einem unverzichtbaren Werkzeug dafür geworden, Daten zwischen Rechnern so auszutauschen, dass Daten über weite Distanzen hinweg durch das Netz übermittelt werden, auf einem entfernten Rechner in Datenbanken eingelesen und dort weiter verarbeitet werden. Das ist hier lässig dahin gesagt, bedeutet aber in der Praxis eine Erhöhung der Komplexität des durch HTML zu steuernden Datenverkehrs, dass wir darauf an dieser Stelle nur auf grundlegenster Stufe eingehen können. Worum geht es konkret? Wir sind seit langem daran gewöhnt, unsere Bankgeschäfte, Bücherbestellungen und vieles mehr über das Internet abzuwickeln. Wir sitzen z. B. zu Hause am Computer, beschäftigen uns mit irgendeiner Frage, zu deren Beantwortung wir Literatur benötigen, sind bereits seit längerer Zeit Kunde bei Amazon, wo unsere persönlichen Daten einschließlich einer Bankverbindung gespeichert sind. Wo sind diese Daten gespeichert? In Datenbanken auf Rechnern der Firma. Immer dann, wenn wir Bestellungen abschicken, wird auf entsprechende Datensätze auf den Rechnern der Firma zugegriffen. Aus der Eigenart dieser Daten ist ersichtlich, wie sensibel diese Zugriffe sind. Wir übermitteln außerordentlich private und sicherheitsrelevante Daten, wenn wir auf diese Weise Geschäfte über das Internet abwickeln. Die hochkomplexen und hochkomplizierten Fragen der Datenverschlüsselung und -sicherung während solcher Transfers werden wir hier nicht ansprechen, sondern nur die elementarsten Schritte, die Voraussetzung dafür sind, einen Datenverkehr wie den oben beschriebenen zwischen Rechnern zu organisieren.

Formulare

Worum geht es im Grundsatz? Darum, dass von einem Rechner, nennen wir ihn einmal den Client (engl. client = Kunde), Daten an einen anderen Rechner, nen-

nen wir ihn den Server (engl. server wörtl. Dienender, in diesem Zusammenhang ein Rechner, der Dienste anbietet, z. B. Daten oder Vergleichbares) geschickt werden. Schauen wir uns zunächst erst wiederum ein Beispiel an:

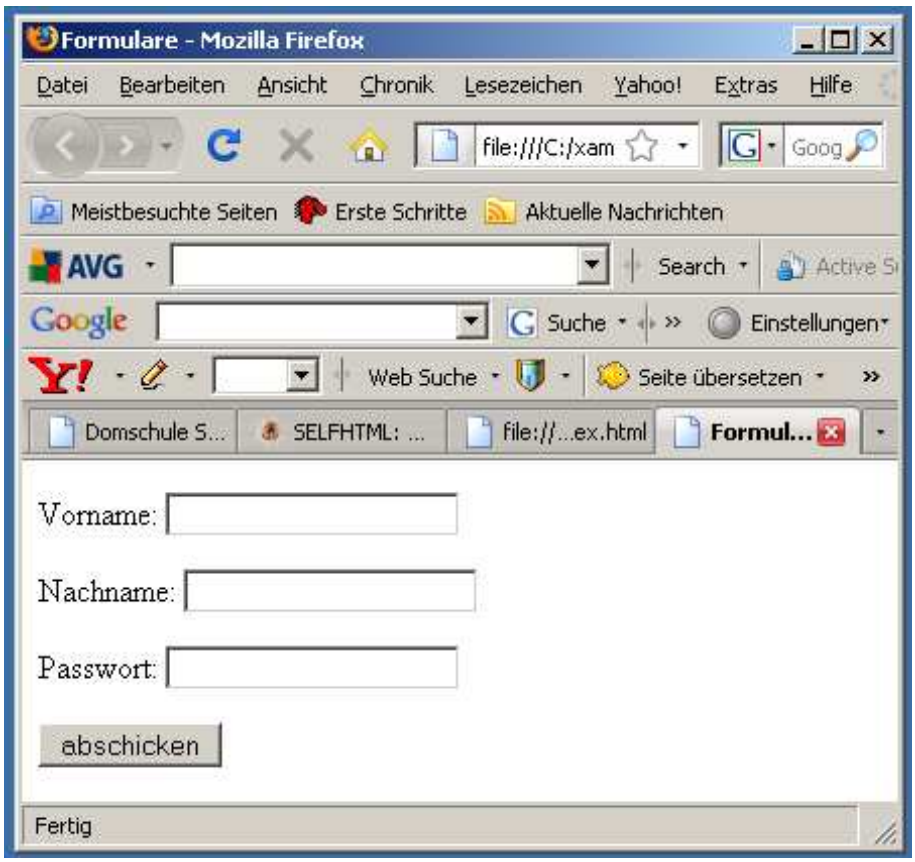


Abbildung 2.6: Ein HTML-Formular

Hier sehen wir ein Formular, wie es uns auch im realen Geschäftsverkehr im Internet begegnen könnte, wenn wir unsere persönlichen Daten übermitteln wollen. Dieses Formular ist in der Lage, in Textfelder Daten aufzunehmen und diese mit Hilfe des 'abschicken'-Buttons an ein Ziel weiterzugeben, das irgendwo im Quellcode angegeben sein muss. Wo das erfolgt und wie die Struktur des Quellcodes beschaffen ist, können wir ermitteln, wenn wir uns den zu diesem Dokument gehörigen HTML-Code anschauen.

Der für uns interessante Code-Abschnitt befindet sich direkt unterhalb des <body>-Tags. Dort ist das <form>-Tag mit den beiden Attributen 'method' und 'action'. Das method-Attribut hat den Wert post. Damit ist Folgendes gemeint: Wenn Daten zwischen Rechnern ausgetauscht werden sollen, dann sieht das **H**yper**t**ext **T**ransfer **P**rotocol (http) zwei unterschiedliche Methoden vor, die sogenannte Post-Methode und die Get-Methode. Die ausgewählte Methode wird als Wert des entsprechenden Attributs innerhalb des Formulars vermerkt, das die zu übermittelnden Daten enthält. Die Get-Methode sorgt dafür, dass die zu übergebenden Daten angehängt an den URL für jedermann im Browser-Adress-Fenster sichtbar versendet werden. Die zu übermittelnde Datenmenge ist begrenzt. Die Post-Methode vermeidet diese

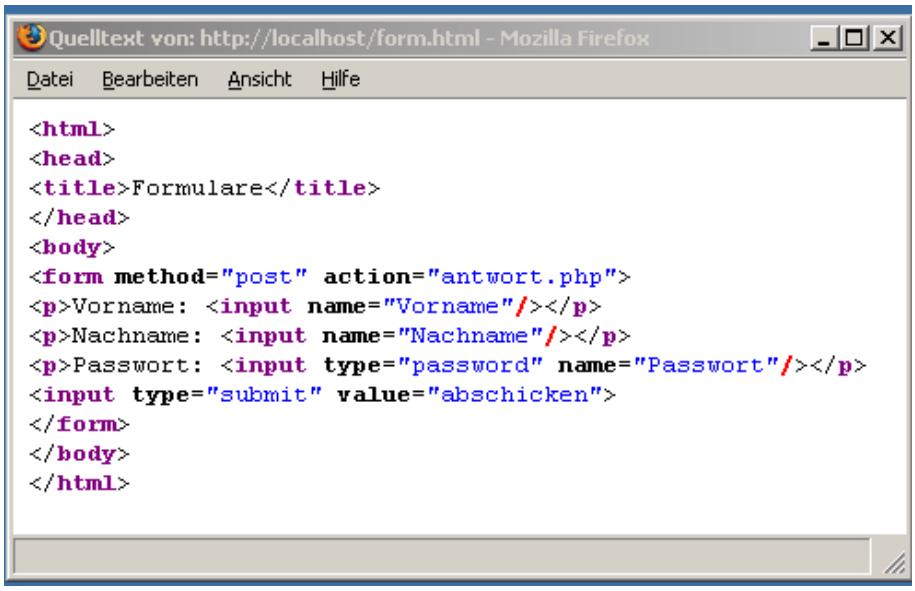


Abbildung 2.7: Der zum Formular gehörige Seitenquelltext

Risiken; die Daten, auch größere Textmengen, werden unsichtbar für den Nutzer gesendet. Das zweite wichtige Attribut ist 'action'. Der Wert dieses Attributs gibt darüber Auskunft, was geschehen soll, wenn die Formulardaten am Server angekommen sind. Es soll dann ein Skript namens antwort.php aufgerufen werden, an das die Daten übergeben werden, um von diesem weiter verarbeitet zu werden. Die folgenden drei input-Tags sind sogenannte Standalone-Tags, was daran zu erkennen ist, dass das öffnende Tag einen Slash / direkt vor der schließenden spitzen Klammer zeigt. Zwei dieser Input-Felder sind Textfelder, in die Text eingegeben wird, erkennbar am Attribut type, das in diesem Fall als Wert 'text' aufweist. Das dritte Input-Feld ist ein sogenanntes Passwort-Feld, wiederum an der Typzuweisung ablesbar. Ist ein Input-Feld von diesem Typ, dann wird die Passworteingabe maskiert. Während das Passwort eingetippt wird, sind z.B. nur Punkte als Stellvertreter der tatsächlich verwendeten Zeichen sichtbar. Das name-Attribut aller drei Input-Felder bewahrt den Namen, der dem Input-Feld gegeben wird, auf. Nun bleibt nur noch das letzte Input-Feld, das vom Typ 'submit' ist, zu erläutern. Der Typ 'submit' ist der Typ, der ein Input-Feld zu einem Button macht, der eine Datenübermittlung auslöst, sobald er gedrückt wird. Welchen Namen der Button trägt, steht im value-Attribut.

Nun stellt sich sofort die Frage, wohin die Daten, die das Formular aufgenommen hat, übermittelt werden. Knappe Antwort: An den Ort, wo sich das im action-Attribut genannte Skript befindet. Das bedeutet im vorliegenden Fall: Die Daten werden an das Skript antwort.php übergeben, das sich in demselben Verzeichnis auf demselben Rechner befindet wie das HTML-Dokument, das das Formular beherbergt, in dem sich die zu versendenden Daten befinden. (Zu diesen Fragen später mehr). Wenden wir uns also nun der Frage zu, was mit den Daten innerhalb des Skripts angestellt wird.

PHP-Grundlagen

Bevor wir uns das Skript anschauen, ein paar einführende Bemerkungen zu PHP. PHP steht für **H**ypertext-**P**re-**P**rocessor. Streng genommen sollte dies als HPP abgekürzt werden. PHP ist eine sogenannte Skript-Sprache, deren Code in Dateien mit der Dateinamensextension *.php hineingeschrieben werden kann, aber auch direkt in HTML-Dokumenten Verwendung findet, ohne dass eine eigens für php-Code mit der Extension *.php versehene Datei gespeichert wird. Wir konzentrieren uns hier darauf, Code zu untersuchen, der in selbständigen *.php-Dateien gespeichert ist. Bevor wir uns den Code anschauen, wollen wir, wie gewohnt, sehen, was er bewirkt. Zunächst werde das Formular ausgefüllt und abgeschickt, danach dann die Reaktion begutachtet, um anschließend zu untersuchen, wie es zu dieser Reaktion gekommen ist. Also erst einmal wird das ausgefüllte Formular gezeigt:

Abbildung 2.8: Das ausgefüllte Formular

Nach dem Ausfüllen wird es abgeschickt. So sieht die Reaktion aus:

Nun zu der interessantesten Frage: Wie kommt es zu dieser Ausgabe? Diese Frage

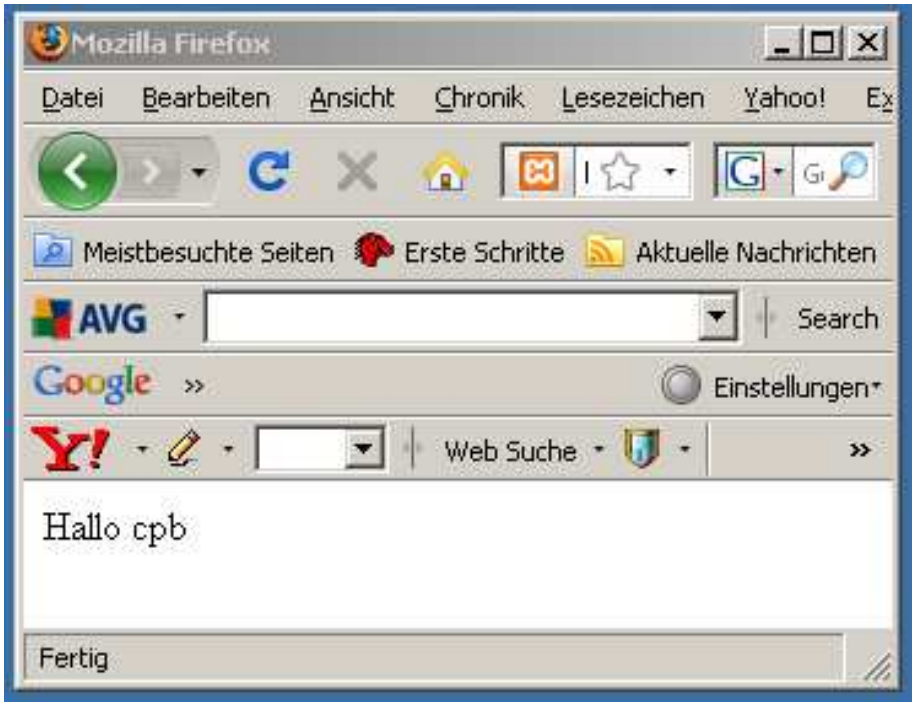


Abbildung 2.9: Die Ausgabe nach Verarbeitung der Formulardaten in antwort.php

beantwortet der Code der Datei antwort.php:

Der Code wird eingeleitet durch die sogenannte Prozessor-Anweisung <?php, die signalisiert, dass das Folgende als php-Code anzusehen ist. Auf diese Prozessor-Anweisung folgen drei Anweisungen, die Ausgaben erzeugen. Der echo-Befehl ist eine php-Anweisung, die dieses leistet. Sollen nur Zeichenketten ausgegeben werden, dann reicht es aus, diese zwischen Anführungszeichen hinter die echo-Anweisung zu schreiben. Jede php-Befehlszeile muss durch ein Semikolon abgeschlossen werden. Die beiden folgenden Zeilen zu verstehen erfordert, ein wenig auszuholen.

Wie in der Mathematik werden auch in der EDV Variablen benutzt. Sie sind genauso wie in der Mathematik Container, die unterschiedlichen Inhalt aufnehmen können. Nehmen wir unser Formular aus dem interaktiven HTML-Dokument, das wir geschrieben haben, um Namen und Passwort aufnehmen zu können. Wenn der Inhalt des Formulars auf dem Server, an den er geschickt worden ist, weiter verarbeitet werden soll, dann müssen auf dem Server Variablen, also Container, bereit stehen, die den Inhalt, den das Formular enthält, aufnehmen können.

Das Skript antwort.php enthält als solche Container die Variablen $_*POST*["*Vorname*"] und in $_*POST*["*Nachname*"]. $_*POST*-Variablen sind dazu da, die Inhalte aufzunehmen, die mittels der Methode POST vom Client an den Server geschickt worden sind. $_*POST*[...] ist ein Variablenname, also eine Bezeichnung für einen Container, der Inhalte aufnehmen, speichern und für die weitere Verarbeitung zur Verfügung stellen kann. Wichtig ist, die Formulardaten in Speicherzellen zur Verfügung zu halten, in die sie gehören. Um die richtige Zuordnung zu gewährleisten, werden die $_*POST*[...]-Variablen mit Hilfe von Eingaben innerhalb der eckigen Klammern differenziert. Im Skript antwortet.php gibt es zwei POST-Variablen, die mittels der Benennungen der Input-Felder des Formulars unterschieden werden. Ein

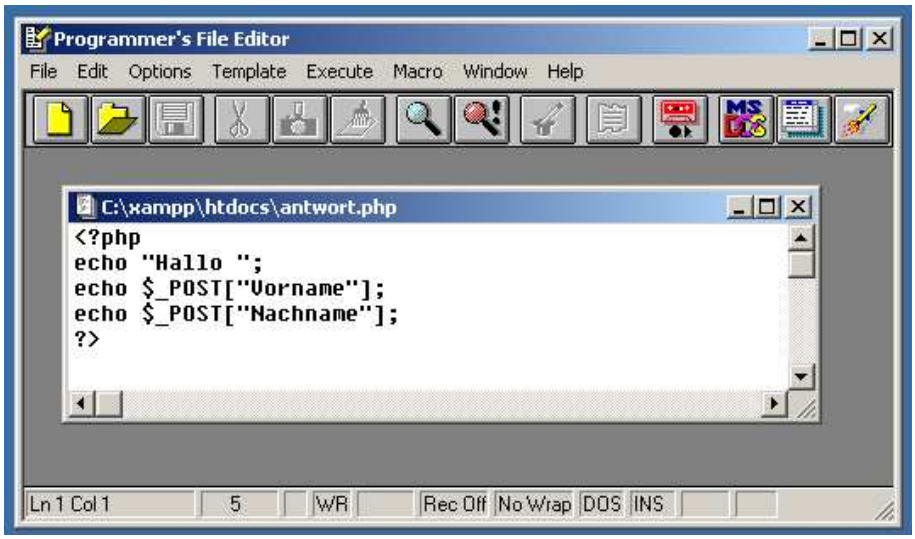


Abbildung 2.10: Der PHP-Quellcode der Datei antwort.php

Bezeichner wie $_*POST*["*Vorname*"] besagt: Nimm in diese POST-Variable die Daten auf, die aus dem Formular-Input-Feld stammen, dessen name-Attribut den Wert 'Vorname' hat. Damit ist die eindeutige Zuordnung von Formulardaten und Variableninhalten im PHP-Skript garantiert. Das Ende des PHP-Codes bezeichnet wiederum eine Prozessoranweisung ?>.

Fassen wir zusammen: Wir haben uns angeschaut, wie HTML-Dokumente aufgebaut sind, haben danach gesehen, wie Formulare innerhalb von HTML-Dokumenten benutzt werden, um Daten zwischen Client und Server zu übermitteln, um danach zu verfolgen, wie diese Daten serverseitig mit PHP-Code weiter verarbeitet werden. Wir haben dabei auf Server-Client-Rechner-Architekturen Bezug genommen, ohne detaillierter auf die Verarbeitungsbedingungen einzugehen, die solche Verarbeitungsprozesse voraussetzen. Zum Abschluss dieser kurzen Einführung in die Grundlagen der Webtechnologie sollen diese Aspekte einbezogen werden.

2.1.3 Client-Server-Architekur

Der Datentransfer innerhalb des WWW beruht darauf, dass Server bereit stehen, die die Daten zur Verfügung stellen, die von Clients weltweit nachgefragt werden.

Im Allgemeinen geht man davon aus, dass Server Maschinen, also Rechner, sind, die diese Arbeit leisten. Das ist natürlich in der Regel auch zutreffend. Viele Server sind oftmals sehr leistungsfähige Rechner, viel leistungsfähigere als durchschnittliche privat genutzte PC's. Dennoch ist es nur die halbe Wahrheit, zuerst an Maschinen zu denken, wenn man an Server denkt. Denn zunächst und zu allererst sind Server Softwarepakete. Das, was einen Rechner, sei er noch so leistungsfähig, zum Server macht, ist ein Softwarepaket. Auch die privat genutzte Maschine, die zu Hause als PC zum Spielen, Surfen, zum Schreiben von Texten oder zur Bild- bzw. Filmbearbeitung genutzt wird, kann als Server eingerichtet werden. Vorausgesetzt, dieser so eingerichtete Rechner wäre 24 Stunden online, dann könnte er auch als Plattform fungieren, mittels derer der eigene Web-Content im Netz verfügbar gemacht wird.

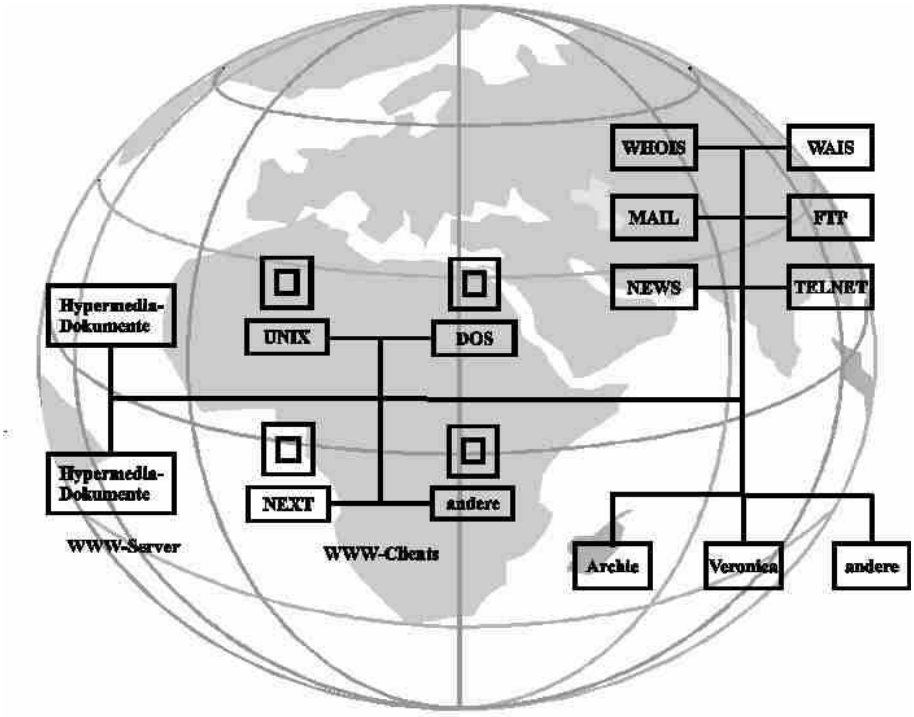


Abbildung 2.11: Server-Client-Architekturen

D. h., dass jeder Rechner gleichzeitig als Client und als Server fungieren kann. Solange er genutzt wird, um die oben beschriebenen Arbeiten zu verrichten, ist er Client, sobald er Daten im Internet, die von anderen Clients angefragt wurden, zur Verfügung stellen soll, ist er Webserver. In der Zwischenzeit gibt es zahlreiche vorkonfigurierte Softwarepakete, die die Installation eines Servers zum Kinderspiel machen. Unter der URL 'http://www.domschule-sl.de/vk.php?vkid=131p=materialcmd=neu' habe ich Videos auf der Schulwebsite ins Netz gestellt, die die Installation und Konfiguration eines xampp-Servers (xampp steht für einen unter windows betriebenen (x als Abkürzung für *.exe, die Dateinamensextension, die für executable steht; Dateien mit dieser Extension sind unter windows selbst ausführbar, eine Windows-Besonderheit, die es z. B. unter linux nicht gibt) apache (a), MySql (m), php (p)-Server) beschreiben.

An dieser Stelle sei nur noch soviel gesagt. Wenn der Webserver auf dem Rechner installiert ist, dann enthält das Verzeichnis, innerhalb dessen der Server installiert ist und das den Namen xampp trägt, ein Verzeichnis namens htdocs (htdocs steht für hypertext documents). Dieses Verzeichnis ist dasjenige, innerhalb dessen alle Dokumente abgelegt werden müssen, auf die der Webserver zwecks Veröffentlichung im WWW zugreifen soll. Ist ein Rechner als Webserver eingerichtet, dann kann er gleichzeitig als Server und als Client fungieren. Als Client ist er die Maschine, auf der z.B. ein Browser läuft, der Dokumente aus dem Netz herunterlädt und ausgibt. Ist der Webserver gleichzeitig auf dieser Maschine gestartet, dann kann der Browser unter dem URL 'http://localhost' auf das Verzeichnis C:\xampp\htdocs zugreifen, um alle dort abgelegten HTML-Dokumente in den Browser zu laden und darzustellen.

2.2 XML-Grundlagen

Nachdem nun HTML-Grundlagen behandelt worden sind, folgen jetzt Untersuchungen der Weiterentwicklungen, die, wie oben geschildert worden ist, notwendig geworden waren, um die Kompatibilität der im Internet publizierten Informationen zu erhalten. Nun soll der Blick auf die XML-Sprachenfamilie gerichtet werden, um mit der Erarbeitung der Grundkenntnisse die Vorausetzungen zu schaffen, die benötigt werden, um Textannotationen vornehmen zu können, die von XML-Strukturen Gebrauch machen.

2.2.1 Das XML-Dokument und seine Grammatik als Schema oder Dokument-Typ-Definition

Schauen wir uns auch hier zum Einstieg ein Beispiel an. Wie auch bei vielen anderen Gelegenheiten sei das so beliebte Hallo-Welt-Beispiel bemüht.[3] Wie sieht eine Äußerung in der Hallo-Welt-Sprache aus? Mit anderen Worten: Wie ist das XML-Dokument beschaffen, das das enthält, was ich auszeichnen möchte, um es dann in eine von allen verstandene Sprache, sprich: HTML, zu übersetzen? Das XML-Dokument gruss.xml könnte folgende Gestalt haben, wenn seine Struktur von einer DTD festgelegt wird. Wie eine XML-Datei beschaffen ist, die ein Schema auswertet, wird unten beschrieben:

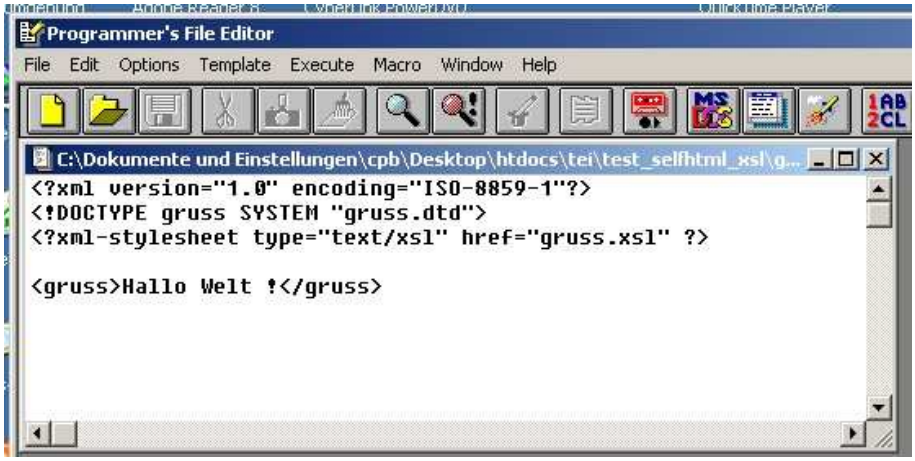


Abbildung 2.12: gruss.xml

In der ersten Zeile erfahren wir in einer processing instruction (Man erkennt processing instructions an dem Fragezeichen, das auf die öffnende spitze Klammer folgt, und jenem, das der schließenden vorangeht), einer Anweisung, die den in den Browser integrierten XML-Prozessor darüber informiert, dass er es mit einem xml-Dokument zu tun hat, das gemäß den Grundsätzen der XML-Version 1 gestaltet ist und den von der **I**nternational **O**rganisation of **S**tandardization (ISO) festgelegten Zeichensatz 8859-1 benutzt. Diese processing instruction enthält neben dem Bezeichner xml

[3] Die folgende Darstellung folgt den von Stefan Münz analysierten Beispielen. Vgl. [Selfhtml], XSLT-Beispiele.

zwei Attribute version und encoding, die ohne trennende Zeichen - mit Ausnahme der Leertaste - aufeinander folgen. Attributen werden Werte zugeordnet, die in Anführungszeichen eingeschlossen werden müssen. Die dritte Zeile enthält ebenfalls eine processing instruction, die einen Hinweis auf ein xml-stylesheet enthält, eine Gestaltungskomponente, zu der Anweisungen darüber gehören, wie der Inhalt des xml-Dokuments dargestellt werden soll. Auch hier finden sich zwei Attribute, die zum einen Auskunft darüber geben, von welchem Typ die Daten sind, die das Stylesheet präsentiert, nämlich Text einer xsl-Datei, und die in dem Attribut href (Hypertext-Referenz) darüber informiert, unter welchem Namen und wo das xsl-Dokument zu finden ist, nämlich als gruss.xsl in demselben Verzeichnis, innerhalb dessen sich auch das XML-Dokument gruss.xml befindet. Danach folgt dann das einzige Element des xml-Dokuments gruss. Genauso wie in HTML sind Elemente in spitze Klammern eingeschlossen, es gibt öffnende und schließende Tags. Die Syntax ist die aus HTML bekannte. Der Inhalt des Elements ist der String (die Zeichenkette) 'Hallo Welt !'.

Wie oben bereits ausgeführt, besteht die Aufgabe eines XML-Dokuments nur darin, Inhalte strukturiert auszuzeichnen. Die Art, wie die Struktur ausgegeben wird, wird den Festlegungen überlassen, die innerhalb einer anderen Datei, der xsl-Datei, enthalten sind. Die Beschaffenheit der Struktur wird wiederum in einer weiteren Art von Dokument festgelegt, einer Dokument Typ Definition mit der Extension *.dtd, oder einem Schema mit der Extension *.xsd. Der Stern vor dem Punkt, der die Extension vom Dateinamen trennt, bedeutet, dass beliebige Zeichenketten als Dateinamenbezeichner an dieser Stelle eingefügt werden können. Im oben verwendeten XML-Dokument wird in der zweiten Zeile auf eine Dokument-Typ-Definition verwiesen. Dieser Verweis beginnt hinter dem Ausrufezeichen, einem Sonderzeichen, das anzeigt, dass Anweisungen dazu folgen, wie der sich anschließende Code auszuführen ist. Dann schließt sich der Bezeichner DocType an, der signalisiert, dass jetzt Hinweise zur Dokument-Typ-Deklaration erscheinen. Im Anschluss an diesen Bezeichner findet sich der Name des Wurzel-Elementes des XML-Dokuments, dessen Dokument-Typ-Deklaration an dieser Stelle einbezogen wird. Der Bezeichner SYSTEM besagt, dass der Speicherort der *.dtd explizit angegeben wird. In unserem Beispiel befindet sie sich in dem Verzeichinis, innerhalb dessen auch das XML-Dokument liegt.

Es sollen jetzt die unterschiedlichen Festlegungen, die in der DTD bzw. dem Schema enthalten sind, genauer betrachtet werden. Beginnen wir mit der DTD. Es ist vielleicht eine kurze Anmerkung zur Geschichte dieser Dateitypen angebracht. Im Verlauf der Entwicklung der XML-Sprachenfamilie hat es, obwohl die Familie erst zehn Jahre alt ist, bereits einschneidende Veränderungen gegeben. Die DTD ist die erste Generation dieses Zweiges der Familie. Diese Familie unterschied und unterscheidet sich von anderen wie z.B. dem XML-Zweig dadurch, dass sie für die Erstellung ihrer Festlegungen eine andere Syntax benutzt. Schauen wir uns das an einem Beispiel an. Wie sieht die DTD gruss.dtd aus?

Sie besteht nur aus einer Zeile, die besagt, dass die XML-Datei, die eine gültige Instanz dieser DTD ist, aus einem Element besteht, das den Namen gruss trägt und vom Typ PCDATA ist, d.h. vom Typ einer strukturierten Zeichenkette oder einzelner Zeichen (**P**arsed **C**haracter **DATA**). Die XML-Datei wird daraufhin überprüft, ob sie die von der DTD gruss.dtd festgelegten Vorgaben erfüllt oder nicht. Ist dies der

Abbildung 2.13: gruss.dtd

Fall, dann ist das Instanzendokument gruss.xml ein gültiges Dokument.

Im Falle der Verwendung einer Schema-Datei gruss.xsd haben wir diese Anweisungsfolge:

Abbildung 2.14: gruss.xsd

Diese Datei beginnt mit der Adressierung eines Namensraumes, der innerhalb des Attributs xmlns (ns für **n**ame**s**pace) angegeben wird. Der Attribut-Bezeichner 'xmlns' enthält hinter dem Doppelpunkt ein Suffix, das als Abkürzung fungiert, die innerhalb der Schema-Datei verwendet wird. Die Abkürzung markiert innerhalb des Dokuments als Präfix verwendet den Typ der Elemente, nämlich den, Elemente einer Schema-Datei zu sein. In diesem Dokument sind es nur die Elemente 'schema' und 'element'. Damit wird angezeigt, dass die Elemente, denen dieses Präfix zugeordnet ist, dem angegebenen Namensraum angehören. In der zweiten Zeile finden wir dann die Festlegung in der Schema-Syntax, die wir auch schon aus der DTD kennen, allerdings jetzt mit Regeln angegeben, wie sie auch aus anderen Sprachen der XML-Familie vertraut sind. Die Schema-Datei verwendet nicht ihre eigene Syntax wie die DTD, sondern verwendet die allgemein gebräuchliche Syntax, wie sie auch in XML- oder XSLT-Dokumenten angewendet wird. Zunächst wird angegeben, dass es sich um ein Element handelt, mit dem wir es zu tun haben sollen. Zu diesem Element

'element' gehören zwei Attribute 'name' und 'type', die zeigen, welchen Namen das Element trägt und von welchem Typ es ist. In unserem Beispiel handelt es sich um den Namen 'gruss'; das Element soll Daten vom Typ 'string' aufnehmen können. Wie auch in HTML-Dokumenten geläufig endet das Dokument mit einem schließenden Tag.

Damit sind die Definition der Struktur und die Beschreibung der Daten, die gemäß dieser Struktur ausgezeichnet worden sind, abgeschlossen. Jetzt muss noch die Übersetzung folgen. Eine XML-Datei enthält noch keine Informationen darüber, wie die Daten dargestellt werden sollen. Um im Bild zu bleiben. Die Grammatik unserer Sprache ist bekannt, Äußerungen unserer Sprache haben wir auch. Allein, es fehlt noch der Übersetzungsalgorithmus, der dafür sorgt, dass die Äußerungen allgemein verständlich werden. Die Aufgabe dieser Übersetzung obliegt einer XSLT-Datei mit der Extension *.xsl.

2.2.2 Die Transformation durch XSLT-Dokumente

Wie sieht eine solche Datei aus? Schauen wir uns auch hier das zu unserem Beispiel gehörige Dokument an:

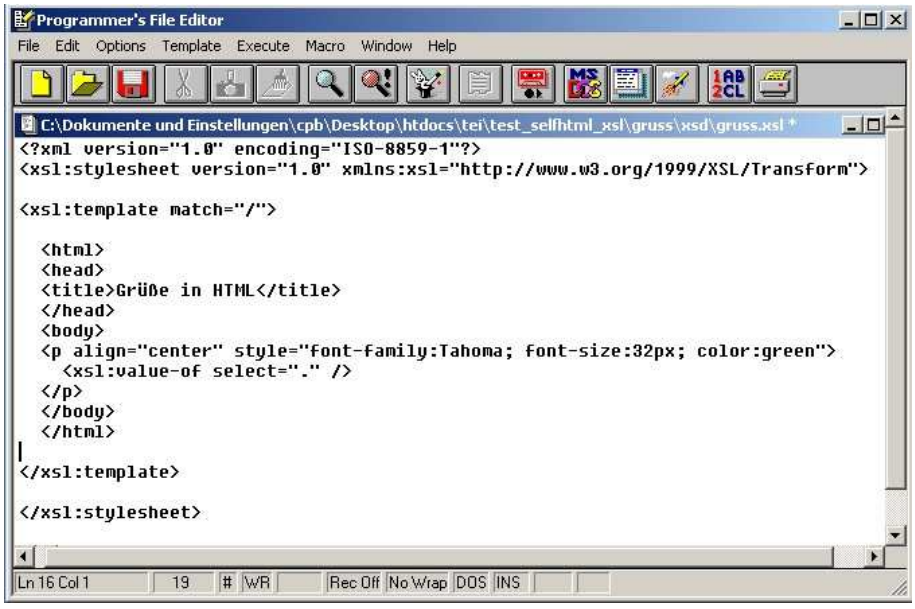


Abbildung 2.15: gruss.xsl

Da auch dieses Dokument den Regeln der XML-Syntax folgt, begegnen hier wiederum die bereits bekannten Anweisungen, die XML-Dokumente eröffnen. Darauf gehen wir hier jetzt nicht weiter ein, sondern konzentrieren uns auf die XSLT-spezifischen Sprachelemente wie <xsl:template> (engl. template: dt. Schablone) oder <xsl:value-of> mit dem Attribut select. Die erste in diesem Zusammenahng interessante Anweisung ist diejenige, die durch den Wert des match-Attributs geliefert wird. Im template-Element erhält das match-Attribut den Wert /. Der Slash bezeichnet das Wurzelelement des Dokuments. Demzufolge ist klar, dass alle folgenden Format-Anweisungen auf das Wurzelelement anzuwenden sind. Im Anschluss

daran wird beschrieben, welche Gestalt die Ausgabe, die das Wurzelelement betrifft, haben soll. Da wir ein HTML-Dokument erzeugen wollen, sind die HTML-üblichen Auszeichnungen vorzunehmen. Innerhalb des p-tags, das als Container für den von uns im XML-Dokument ausgezeichneten Inhalt fungiert, wird nun eine weitere XSLT-spezifische Anweisung eingefügt. Das XSLT-Element value-of ineins mit dem select-Atttribut, das den Wert ' . ' erhält, sorgt dafür, dass der vollständige Inhalt des aktuellen Elements, in diesem Fall des Wurzelelements, ausgewählt wird. Damit sind alle wesentlichen Inhalte des XSLT-Dokuments erklärt. Es bleibt noch zu zeigen, wie die verschiedenen Dokumente: das XML-, das XSD- und das XSL-Dokument zusammenarbeiten. Ausgangspunkt für die Analyse dieser Kooperation sollte das XML-Dokument sein, da bei diesem alle Fäden zusammenlaufen. Schauen wir uns das entsprechende Dokument also an:

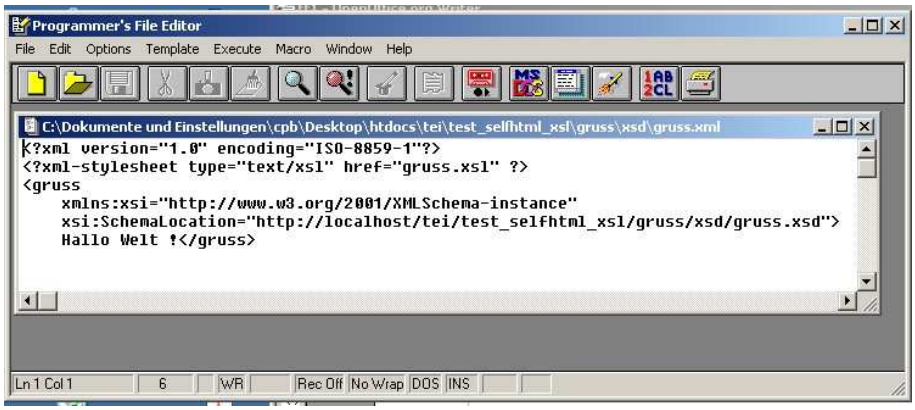


Abbildung 2.16: gruss.xml unter Verwendung einer Schema-Datei gruss.xsd

Man sieht, dass innerhalb des Wurzelelements ein Namensraum referenziert wird, der XMLSchema-Instance-Namensraum. Innerhalb der in diesem Zusammenhang relevanten Auszeichnungen wird angegeben, welchem Namensraum die mit dem Präfix *. xsi beginnenden Auszeichnungen zuzuordnen sind. Im Verlauf dieser Auszeichnungen ist auch der Ort anzugeben, an dem das Schema zu finden ist.

Von diesen Strukturen, XML-Strukturen, macht die TEI Gebrauch. Alle TagSets der TEI sind auf der Grundlage der Syntax der XML-Richtlinien definiert. Nachdem wir die grundlegenden Konzepte von XML untersucht haben, wenden wir uns jetzt also den TagSets zu, die im Zusammenhang unserer Interessen von Bedeutung sind. Da wir Prosatexte annotieren werden, sind also die diesbezüglichen TagSets genauer unter die Lupe zu nehmen.

2.3 Die TEI-TagSets 'Prosa'

Für den Anfang liegen die größten Schwierigkeiten erst einmal hinter uns. Was nun kommen wird, wird ein wenig bekömmlicher, weniger abstrakt und strukturell komplex als das, was wir uns zu XML erarbeitet haben. Es soll nun erst einmal darum gehen, die Text-Beschreibungselemente der TEI kennen zu lernen. Vergegenwärtigen wir uns zunächst den grundlegenden Aufbau einesXML-Dokuments, das die Tags der TEI benutzt.

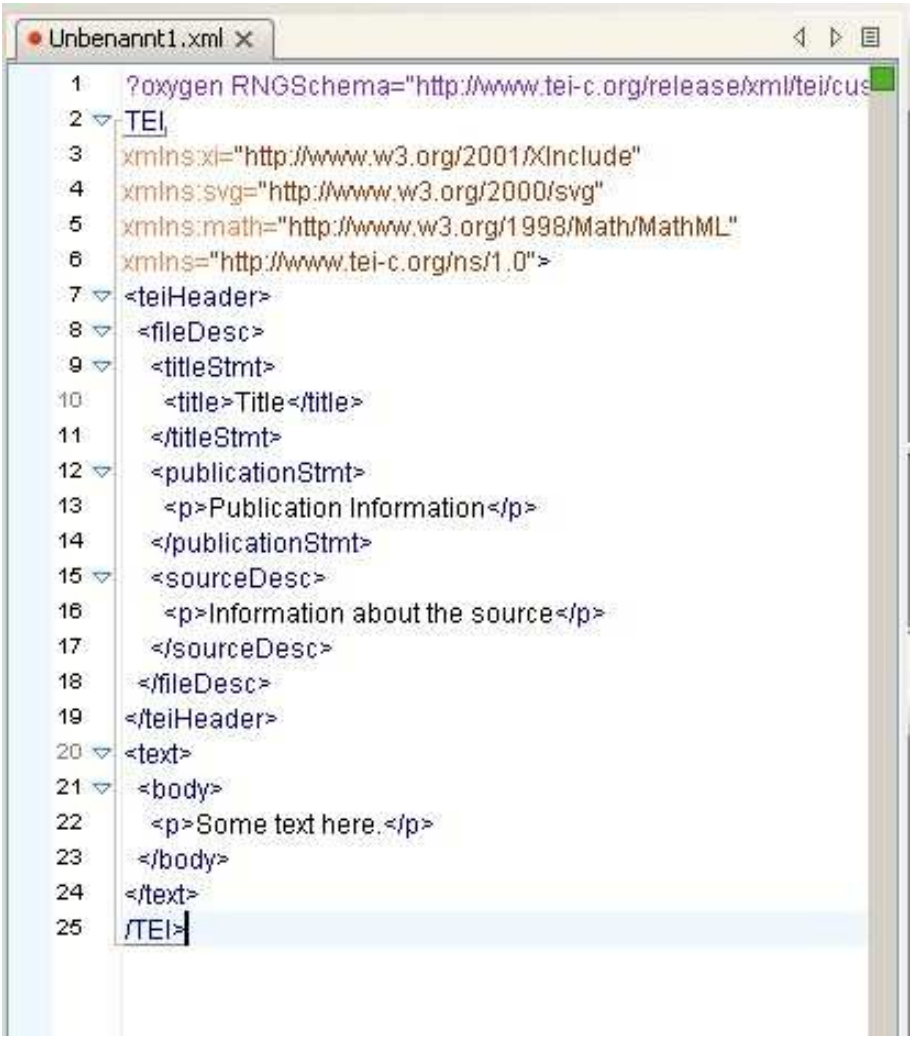


Abbildung 2.17: Der TEI-Header in der IDE des XML-Editors 'oXygen'

Diese Abbildung zeigt einen kleinen Ausschnitt der Entwicklungsumgebung des XML-Editors oXygen, auf den wir an anderer Stelle zurückkommen werden. Was wir sehen, ist die Arbeitsfläche, innerhalb derer wir ein XML-Dokument bearbeiten können. Das hier abgebildete Dokument zeigt die Struktur, die ihm durch die von der Text Encoding Initiative verwendeten Schemata mitgegeben wird. Ignorieren wir erst einmal die Instruktionen, die in den Zeilen 1-6 zu finden sind, um uns auf das in den Zeilen 7-24 Enthaltene zu konzentrieren. Was findet sich dort? Tags, die dem ähnlich sind, was aus HTML-Dokumenten vertraut ist.

2.3.1 Der TEI-Header

Im vorliegenden Fall sind das Tags wie <teiheader> oder <body>. In jedem HTML-Dokument gibt es ein <body>-Tag, das dasjenige umschließt, was im Browser als Inhalt eines HTML-Dokuments angezeigt werden soll. Darüber hinaus gibt es in HTML auch das <head>-Tag, das Informationen aufnimmt, die nicht im Browser-Fenster dargestellt werden, sondern entweder gar nicht oder aber in der Titel-Zeile

des Browsers, die den Titel des aktuell bearbeiteten HTML-Dokuments anzeigt. Hier zeigen sich grundlegende Gemeinsamkeiten zwischen TEI-XML- und HTML-Dokumenten. Besonderheiten des TEI-XML-Dokuments bestehen auf den ersten Blick vor allem in dem, was den Inhalt des TEI-Headers ausmacht. Dieser enthält das Tag <fileDesc> (file description), das seinerseits die Tags <titleStmt>(title statement), <publicationStmt> und <sourceDesc> umfasst. D.h. dass es im Header Informationen über das geben soll, was in bibliographischer Hinsicht[4] über die Datei gesagt werden kann, die bearbeitet wird. Im einzelnen sind dies Informationen, die den Titel und den Autoren des zu edierenden Werkes nennen (titleStmt); darüber hinaus soll auch etwas dazu gesagt werden, wer für die elektronische Publikation verantwortlich ist (publicationStmt) und last but not least können Informationen über die Beschaffenheit der Quelle, die Vorlage der elektronischen Edition ist, interessant sein (sourceDesc). Soviel zum TEI-Header, dessen Beschreibung mit der Abbildung einer Beispiel-Annotation beschlossen werden soll.

```
4  <teiHeader>
5   <fileDesc>
6    <titleStmt>
7     <title>Meister Eckhart, Deutsche Predigten</title>
8    </titleStmt>
9    <publicationStmt>
10    <p>Elektronische Publikation der deutschen Predigten Meister Eckharts in der Edition Josef
11     Quints. Die Paginierung der von Quint herausgegebenen Werk-Edition wird durch Annotat
12     vermerkt. Der Text ist entnommen der Digitalen Bibliothek Deutscher Klassiker auf CD-ROM,
13     Deutscher Klassiker Verlag, Frankfurt am Main 2003. Der Text folgt dessen Graphie, die von
14     derjenigen Quints an einigen Stellen, die nicht markiert werden, abweicht; die
15     elektronische Publikation wird eingerichtet von Claus-Peter Becke, Kiel, 2008.</p>
16   </publicationStmt><sourceDesc>
17    <p>Meister Eckharts Predigten Herausgegeben und übersetzt von Josef Quint Erster Band W.
18     Kohlhammer Verlag Stuttgart 1958</p>
19   </sourceDesc></fileDesc>
20  </teiHeader>
```

Abbildung 2.18: Das Tag <publicationStmt> im TEI-Header

2.3.2 Das Body-Tag

Welche Tags verwendet werden, um den Text des Dokuments zu annotieren, das hängt von den jeweiligen Interessen des Untersuchenden und von der Textstruktur ab. In der TEI werden zwei Bereiche von Tags unterschieden; derjenige der Kern-Tags (Core Tags) und derjenige der über diese hinausgehenden Tags. Der zuletzt genannte Bereich wird seinerseits in die grundlegenden (Base Tags) und die zusätzlichen (Additional Tags) unterteilt. Wir wollen uns zunächst mit den Tags beschäftigen, die benötigt werden, um eine "Default Text Structure"(Standard-Text-Struktur) zu beschreiben. Die Tag-Menge, die diese Gruppe bildet, besteht aus den Tags <text>, <front>, <body>, <group>, (division: Teile wie Kapitel u.ä.) und <p> (paragraph: Absätze), um die wichtigsten zu nennen. Wie man von diesen Tags Gebrauch macht, soll wiederum an einem Beispiel illustriert werden.

Der Textausschnitt zeigt einen Paragraphen, markiert durch das <p>-Tag. Bevor die Annotationen im Einzelnen erläutert werden, seien an dieser Stelle einige

[4] http://www.tei-c.org/release/doc/tei-p5-exemplars/html/teilite.doc.html.

```
<text>
 <front/>
 <group>
  <group>
   <text>
    <front>
     <head>Intravit Iesus in templum et coepit eicere vendentes et ementes. Mathaei</head
    </front>
    <body>
     <pb n="4"/>
     <div type="mhd_Predigt" n="1">
      <p> Wir lesen in dem heiligen êwangeliô die heilige schrift, daz unser herre gienc in
       den tempel und was ûzwerfende, die dâ kouften und verkouften, und sprach ze den
       andern, die dâ hâten tûben und sôgetâniu dinc veile: tuot diz hin, tuot diz <pb
        n="5"/>enwec! War umbe was Jêsus ûzwerfende, die dâ kouften und verkouften un
       hiez die hin tuon, die dâ hâten tûben? Er enmeinte anders niht, wan daz er den
       tempel wolte ledic hân, rehte als ob er spræche: ich hân reht ze disem tempel und
       wil aleine dar inne sîn und hêrschaft dar inne hân. Waz ist daz gesprochen? Dirre
       tempel, dâ got inne hêrschen wil gewalticlîche nâch sînem willen, daz ist des
       menschen sêle, die er sô rehte glîch nâch im selber gebildet und geschaffen hât, al
       wir lesen, daz unser herre sprach: machen wir den menschen nâch unserm bilde u
       unser glîchnisse. Und daz hât er ouch getân. Als glîch hât er des menschen sêle
       gemachet im selber, daz in himelrîche noch in ertrîche von allen hêrlîchen
       crêatûren, diu got sô wünnic<pb n="6"/>lich geschaffen hât, keiniu ist, diu im als
       glîch ist als des menschen sêle aleine. Her umbe wil got disen tempel ledic hân, da
       ouch niht mê dar inne sî dan er aleine. Daz ist dar umbe, daz im dirre tempel sô wol
       gevellet, wan er im alsô rehte glîch ist und im selber alsô wol behaget in disem
       tempel, swenne er aleine dar inne ist. </p>
```

Abbildung 2.19: Grundlegende Tags zur Annotation von Prosa-Texten

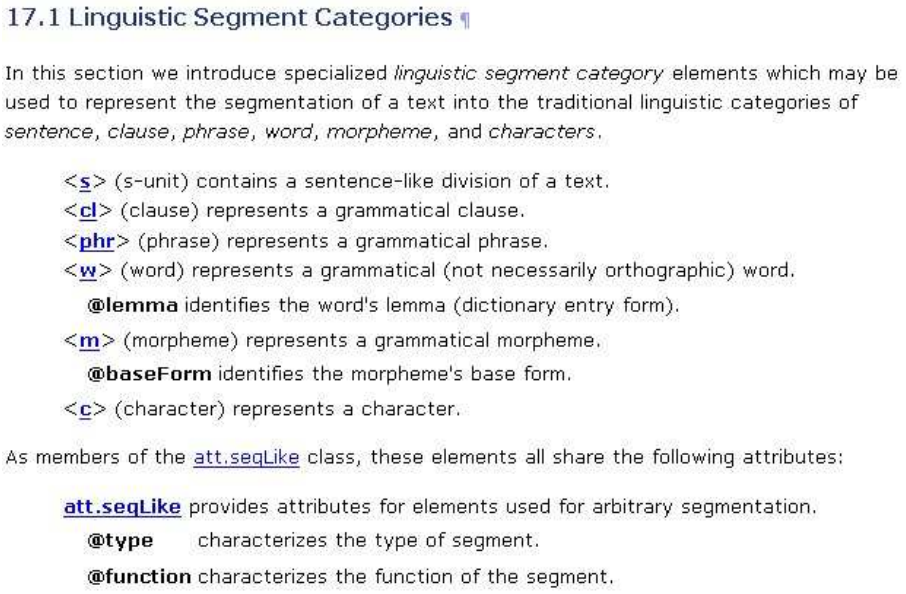
Bemerkungen zum Textinhalt eingefügt. Es handelt sich um einen mittelhochdeutschen Predigt-Text von Meister Eckhart, geboren um 1260 in Hochheim in Thüringen, Dominikaner, zweimals Dozent der Pariser Universität, ranghoher Amtsträger des Dominikaner-Orden, vom Papst nach Denunziation durch Ordensbrüder gegen Ende seines Lebens unter Häresieverdacht gestellt, gestorben in Avignon während des Prozesses gegen ihn. Um es zu präzisieren: Eckhart wurde nicht verdächtigt, Häretiker zu sein. Vielmehr wurden in seinen Schriften Lehrstücke gefunden, die als verdächtig galten. Einer Verurteilung ist Eckhart dadurch entgangen, dass er die Textstellen, die Anstoß erregten, widerrief oder in einer Rechtfertigungsschrift erklärte. Der oben abgebildete Textausschnitt enthält den Text einer Predigt, die nur in Nachschriften erhalten ist, nicht als von Eckhart selbst verfasstes Original. Es geht um einen Satz aus dem Matthäus-Evangelium, den Eckhart in seiner Predigt auslegt. Der lateinische Text heißt sinngemäß übersetzt so viel wie: Jesus ging in einen Tempel und warf diejenigen hinaus, die im Tempel Handel trieben. Eckhart legt diesen Evangeliumstext nun aus, indem er den Tempel mit der Seele gleichsetzt, um eine theologische Auslegung und Begründung anzuschließen.

Werfen wir jetzt einen Blick auf die Annotationen. Beachtung und Erklärung verdient erst einmal ein Tag wie <front/>, das an dieser Stelle leer bleibt, angezeigt durch den vor der schließenden spitzen Klammer stehenden Slash, der verwendet wird, um zu zeigen, das öffnendes und schließendes Tag zusammengeschoben werden, mit anderen Worten: dass es keinen Inhalt gibt zwischen öffnendem und schließendem Tag. In anderen Kontexten kann dieses Tag bei Bedarf dazu benutzt

werden, um z.B. Inhaltsverzeichnisse, Widmungen und verschiedene andere Textbausteine, die dem eigentlichen Dokumenttext vorangestellt werden sollen, aufzunehmen. In unserer Beispiel-Annotation ist ein weiteres <front>-Tag, das einem eingeschachtelten Text-Element zugehört, benutzt worden, um den Titel eines einzelnen Text-Elementes darzustellen. Um zu signaliseren, dass es sich um den Titel handelt, wird das <head>-Tag verwendet. Ein kurzer Blick auf die Struktur, die das Beispiel-Projekt präsentiert. Es gliedert sich zunächst in zwei große Gruppen, die Gruppe der mittelhochdeutschen Predigten Meister Eckharts und die Gruppe der neuhochdeutschen Übersetzungen dieser Predigten. Dass es zunächst darum geht, die mittelhochdeutschen Texte zu annotieren, wird u. a. dadurch kenntlich gemacht, dass das <division>-Tag mit zusätzlichen Daten ausgestattet wird, die Hinweise auf den Typ und die Ordnungszahl des Textabschnittes enthalten, der im aktuellen Kontext ediert wird. Im <division>-Tag können Angaben dazu gemacht werden, welche Art von Abschnitt bearbeitet wird, ein Kapitel innerhalb eines Buches, ein Gedicht innerhalb eines Zyklus, eine Erzählung innerhalb eines Bandes oder wie in unserem Fall, eine Predigt innerhalb einer entsprechenden Sammlung. Um welche Art von <division> es sich handelt, wird im type-Attribut angegeben, die Ordnungszahl der jeweiligen <division> im n-Attribut. Damit wird deutlich, dass auch die TEI-Elemente, genauso wie HTML-Elemente, Attribute enthalten können, die detaillierte Informationen über Charakteristika des Elementes liefern. Dass Attribute wertvolle Selektoren sein können, wenn es darum geht, die Informationen, die in XML-Dokumenten durch Annotationen bereit gestellt werden, in XSLT-Stylesheets auszuwerten, sei an dieser Stelle ohne weitere Erklärungen nur angemerkt. Im gegebenen Kontext sei jetzt nur noch auf das Pagebreak-Tag <pb> aufmerksam gemacht, das verwendet wird, um die Originalpaginierung des Textes zu bewahren, der der elektronischen Edition zugrunde liegt. Die jeweilige Seitenzahl wird in diesem Fall im n-Attribut eingetragen. Zum Abschluss noch der Hinweis auf ein Tag, das in der oben abgedruckten Beispiel-Annotation und in der Liste der Tags der default-text-structure nicht erwähnt ist, das <back>-Tag, das je nach Kontext, unterschiedlich genutzt werden kann. Am gebräuchlichsten ist wohl, die Verwendung dieses Element am Ende der Annotation des Dokumenttextes einzusetzen, um dort z.B. Raum für Anhänge, Indizes u.a. zu schaffen.

Die Richtungen, in die Annotationen weiterentwickelt werden, hängt von dem Interesse ab, das die Untersuchenden mitbringen. Wenn Interesse besteht, die linguistischen Eigenschaften des Textes zu erfassen, dann ist das Corpus der linguistischen Annotationskategorien ein geeigneter Fundus an Tags, die zur Beschreibung eingesetzt werden können. Auf der Website der TEI finden sich in den Guidelines Hinweise zur Verwendung dieser Tags. Der folgende Screenshot zeigt die einführenden Bemerkungen.

Eine Annotation eines kurzen Ausschnitts aus einer Predigt Meister Eckharts möge als illustrierendes Beispiel zeigen, wie syntaktische Annotationen angelegt werden können. Das Beispiel zeigt, wie von den unterschiedlichen Tags so Gebrauch gemacht werden kann, dass einige der syntaktischen Eigenschaften der Satzglieder erfasst werden. In diesem Fall sind insbesondere die beiden allen oben genannten Elementen <s>, <cl>, <phr>, <w>, <m> und <c> zur Verfügung stehenden Attribute type und function eingesetzt worden, um die Ergebnisse formaler und funktionaler Satz-



17.1 Linguistic Segment Categories ¶

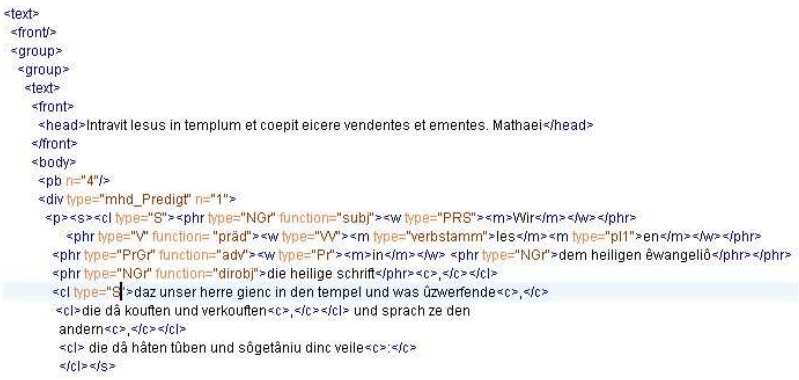
In this section we introduce specialized *linguistic segment category* elements which may be used to represent the segmentation of a text into the traditional linguistic categories of *sentence*, *clause*, *phrase*, *word*, *morpheme*, and *characters*.

<**s**> (s-unit) contains a sentence-like division of a text.
<**cl**> (clause) represents a grammatical clause.
<**phr**> (phrase) represents a grammatical phrase.
<**w**> (word) represents a grammatical (not necessarily orthographic) word.
@lemma identifies the word's lemma (dictionary entry form).
<**m**> (morpheme) represents a grammatical morpheme.
@baseForm identifies the morpheme's base form.
<**c**> (character) represents a character.

As members of the att.seqLike class, these elements all share the following attributes:

att.seqLike provides attributes for elements used for arbitrary segmentation.
@type characterizes the type of segment.
@function characterizes the function of the segment.

Abbildung 2.20: Erläuterungen zu den Tags der syntaktischen Annotation auf der TEI-Website http://www.tei-c.org



```
<text>
 <front/>
 <group>
  <group>
   <text>
    <front>
     <head>Intravit Iesus in templum et coepit eicere vendentes et ementes. Mathaei</head>
    </front>
    <body>
     <pb n="4"/>
     <div type="mhd_Predigt" n="1">
      <p><s><cl type="S"><phr type="NGr" function="subj"><w type="PRS"><m>Wir</m></w></phr>
          <phr type="V" function= "prād"><w type="VV"><m type="verbstamm">les</m><m type="pl1">en</m></w></phr>
       <phr type="PrGr" function="adv"><w type="Pr"><m>in</m></w> <phr type="NGr">dem heiligen êwangeliô</phr></phr>
       <phr type="NGr" function="dirobj">die heilige schrift</phr><c>,</c></cl>
       <cl type="S">daz unser herre gienc in den tempel und was ûzwerfende<c>,</c>
        <cl>die dâ kouften und verkouften<c>,</c></cl> und sprach ze den
        andern<c>,</c></cl>
        <cl> die dâ hâten tûben und sôgetâniu dinc veile<c>:</c>
        </cl></s>
```

Abbildung 2.21: Syntaktische Annotationen

analysen zu erfassen. Im vorliegenden Beispiel sind die von Eisenberg[5] benutzten Konstituenten- und Wortkategorien einerseits, Bezeichnungen syntaktischer Relationen andererseits verwendet worden, um die entsprechenden syntaktischen Eigenschaften des Satzes zu kennzeichnen. Für syntaktische Annotationen in der Schule empfiehlt sich die Verwendung der Kategorien, von denen der Schüler-Duden Grammatik Gebrauch macht.

[5] Vgl. Peter Eisenberg, Grundriss der deutschen Grammatik- Der Satz

3 Suchmaschinen

3.1 Einleitung: Information Retrieval

Suchmaschinen sind Systeme des Information Retrieval[1] (IR). Was ist IR? Es geht darum, Information wieder zugänglich zu machen, erneut aufzufinden, nachdem sie zuvor irgendwo abgelegt war. Das ist das Kernproblem. Überall dort, wo große Informationsmengen zur Verfügung stehen, die nicht ohne Weiteres zugänglich sind, muss nach Wegen gesucht werden, die Zugänglichkeit zu verbessern. Dieses Problem ist besonders dringlich angesichts der Suche nach Information im World Wide Web. Die über das WWW zugängliche Datenmenge ist unüberschaubar. Vielleicht ließe sich deren Unüberschaubarkeit am ehesten vergleichen mit derjenigen einer Bibliothek nicht übersehbarer Ausmaße, innerhalb derer sich jeder Besucher hoffnungslos verirren würde, stünden nicht überall Wegweiser, die Orientierung böten. Doch allein mit Wegweisern ist es innerhalb eines Systems solcher Dimensionen nicht getan. Ein Bibliotheksbesucher ist ein Leser, der die Wissensbestände, über die eine Bibliothek verfügt, erschlossen sehen möchte. Dazu werden Kataloge angelegt, die nach unterschiedlichen Ordnungsprinzipien strukturiert sind. Einerseits sind alphabetische Anordnungen der Einträge vorstellbar, andererseits systematisch nach Sachgebieten geordnete. Bibliothekare, die die Bestände der Bibliothek kennen, legen solche Kataloge an, in denen die Benutzer dann nach den Büchern suchen können, die sie interessieren. Was haben solche Kataloge mit Suchmaschinen gemeinsam? Das ist die Frage, die uns im Folgenden beschäftigen soll?

Die Wegweiser innerhalb der Bibliothek können den Histories, den gespeicherten Suchpfaden, innerhalb eines Browsers gleichgesetzt werden. Selbst dann, wenn wir irgendwann einmal nicht mehr erinnern, auf welchen Wegen wir zu der Website gekommen sind, innerhalb derer wir uns gerade befinden, können wir mit Hilfe der History diesen Pfad rekonstruieren, um damit dem zu entgehen, was in der Debatte über die Lern- und Arbeitsumgebung Hypertext häufig als „Lost in Hyperspace“ angesprochen wird, einem Phänomen, das ein ernsthafter Einwand dagegen ist, hypertextuelle Lernumgebungen insbesondere in einem Fachgebiet unerfahrenen Lernern zu empfehlen. Nachdem sich nun gezeigt hat, dass es Gemeinsamkeiten zwischen den Orientierungsschildern als Bestandteilen unserer virtuellen Bibliothek und in Browsern anzutreffenden Such- und Orientierungshilfen während einer Web-Recherche gibt, um damit zu illustrieren, wie Suchvorgänge im WWW ablaufen können, stellt sich jetzt erneut die Frage, ob auch Ähnlichkeiten zwischen Suchmaschinen und den von Bibliothekaren erstellten Katalogen bestehen. Um diese Frage beantworten zu können, müssen der Aufbau und die Funktionsweise von Suchmaschinen im Detail untersucht werden.

[1] Sebastian Erlhofer, Suchmaschinen-Optimierung, Bonn 2007, S.15.

3.2 Der Aufbau von Suchmaschinen

Suchmaschinen bestehen aus drei Komponenten[2]:

1. Instrumenten zur Datengewinnung (Webcrawler bzw. - robots)
2. Werkzeugen der Datenanalyse und -verwaltung (IR)
3. Verarbeitungen der Suchanfragen

3.2.1 Das Webcrawler-System

Webcrawler-Systeme[3] bestehen ihrerseits aus drei verschiedenen Komponenten[4]:

1. den Protokoll-Modulen (die einzelnen Crawler)
2. den Verarbeitungs-Modulen (der Storeserver und der Scheduler)
3. den Datenspeicher-Modulen (der Document Index und das Repository)

Was ist ein Webcrawler-System? Vereinfacht gesagt, ist es die Schnittstelle zwischen dem World Wide Web und den anderen Komponenten der Suchmaschine. Aufgabe des Webcrawlers ist es, das gesamte Web zu durchsuchen und dessen Dokumente zu erfassen und zu speichern. Angesichts der Datenmenge, die das WWW zur Verfügung stellt, ist das eine gewaltige Aufgabe. Wie löst ein Webcrawler diese Aufgabe? Beginnen wir die Analyse mit dem Blick auf die Rechner, die das WWW durchsuchen. Wie machen die das?

Um einen Eindruck davon zu bekommen, in welchen Dimensionen die Websuche des Crawler-Systems abläuft, ist es hilfreich, einen Blick auf die Rechnersysteme zu werfen, die Google einsetzt, um die hier beschriebene Aufgabe zu meistern. In den Jahren 2006/2007 setzte Google „hauptsächlich in den USA, aber auch in Irland“ in „über ein Dutzend Rechenzentren“ mehr als „10.000 Server“ ein; auf jedem einzelnen dieser 10.000 Server laufen „ca. 200 Crawler-Prozesse“ „Jeder einzelne Crawler-Prozess bearbeitet über 300 Verbindungen zu URL's gleichzeitig.“ Auf diese Weise können „Tausende von Ressourcen pro Sekunde heruntergeladen“ werden.[5]

Wie nun verläuft das Crawling im Einzelnen? Der Vorgang wird in vereinfachter Form als modellhafte Rekonstruktion beschrieben, ohne auf technische und organisatorische Details einzugehen. Ausgangspunkt für die Suchvorgänge ist das, was im **Dokumentenindex** der Suchmaschine enthalten ist. Dieser gilt auch als „URL-Datenbank“[6], die einerseits die Daten enthält, die als Rechercheergebnis des Crawling-Prozesses in die Datenbank eingetragen werden, die andererseits URL's umfasst, die direkt von Web-Autoren bei den Suchmaschinen angemeldet werden. Den Aufbau des Dokumentenindexes werden wir uns später anschauen. Bleiben wir

[2] A.a.O. S. 29 f.
[3] A.a.O. S. 71, Vgl. Abbildung **??**
[4] A.a.O. S. 70 f.
[5] A.a.O. S. 74 f.
[6] A.a.O. S. 72.

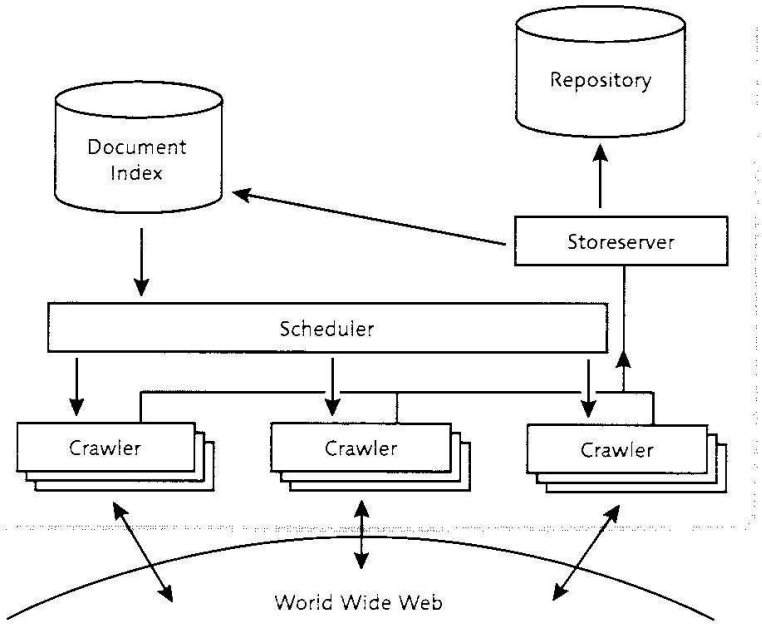


Abbildung 3.1: Das Webcrawler-System

zunächst beim Crawling. Der Crawling-Prozess zerfällt grob gesprochen in zwei unterschiedliche Aufgabenfelder. Zum Einen müssen die bereits erfassten URL's aktualisiert werden, zum Anderen müssen neue erfasst werden. Neue URL's werden einerseits durch direkte Anmeldung dem Dokumentenindex zur Kenntnis gebracht oder sind bereits in ihm enthalten. Waren sie bereits enthalten, so muss festgelegt werden, in welchen Rhythmen sie aktualisiert werden.

Diese zentrale Verwaltungsaufgabe erledigt der **Scheduler**, der den Einsatz der Rechner steuert, die das WWW durchsuchen, um die Websites herunterzuladen und für die Auswertung innerhalb der Suchmaschine zu speichern. Kurz gesagt, überprüft der Scheduler, welcher Crawler gerade frei und einsatzbereit ist und vergibt dann den jeweiligen Zielen, d.h. entweder aktualisieren oder neu erfassen, entsprechende Aufträge an die Rechner des Crawling-Systems.

Hat ein Crawler eine Web-Ressource, d.h. die hinter einer URL abgelegte Website erfasst, dann wird das Ergebnis an den **Storeserver** übergeben. Innerhalb des Storeservers werden die Daten erst einmal gesichert. Dieser Sicherungsvorgang besteht daraus, den HTTP-Response-Header auszuwerten, den Dokumentenindex zu aktualisieren und letztlich die Ressource die Filter durchlaufen zu lassen, die überprüfen, ob sie einer weiteren Untersuchung durch die Suchmaschine unterzogen werden soll oder nicht. Hat eine Ressource den Filter passiert, dann wird sie in das **Repository** aufgenommen. Das ist der „Datenspeicher", der „überwiegend Webseiten mit HTML-Code enthält". Beim „Eingang" wird der Website „eine fortlaufende Kennzahl" zugeordnet, die „DocID", die ihr gemäß ihrer Aufnahme in den Dokumentenindex zugewiesen wird. Zusätzlich werden die „Länge des URL, der URL selbst und

schließlich die Länge bzw. Größe der Ressource mit aufgenommen."[7]

So lässt sich zusammenfassend sagen: „Das Repository beinhaltet gewissermaßen das Resultat der Arbeit des Webcrawler-Systems. Alle relevanten Dokumente liegen hier auf Vorrat im Originalzustand bereit und werden im nächsten Schritt, der Datenaufbereitung, zu einer durchsuchbaren Struktur weiterverarbeitet."[8]

3.2.2 Datenanalyse und Aufbau der Datenstruktur

Wenn das Webcrawler-System seine Arbeit mit dem Ergebnis abgeschlossen hat, dass im Repository die Dokumente, die auf eine Suchanfrage hin als Angebote auf diese Anfrage ausgegeben werden sollen, zur Verfügung stehen, beginnt die nächste Phase der Dokumentenverarbeitung. Zu welchem Zweck? Könnte man nicht auch den Weg beschreiten, auf das Repository als die Datenbank zuzugreifen, aus der die anlässlich einer Anfrage gesuchten Dokumente ausgewählt werden, um nach deren Auffinden im Repository in der Antwort auf die Suchanfrage deren URL auszugeben? Wenn wir uns einmal vor Augen halten, welche unüberschaubar großen Datenmengen im Repository aufbewahrt werden, dann zeigt sich sofort, dass jede Suchanfrage sehr viel Zeit in Anspruch nehmen müsste, wenn die Dokumente unbearbeitet, so wie sie nach dem Crawling-Prozess im Repository gespeichert werden, daraufhin überprüft werden sollten, ob sie ein angemessenes Angebot auf eine Suchanfrage hin bilden oder nicht. Aus diesem Grund ist eine zweite Verarbeitungsstufe in die Suchmaschine integriert, die folgendermaßen in das Gesamtsystem eingepasst ist[9]:

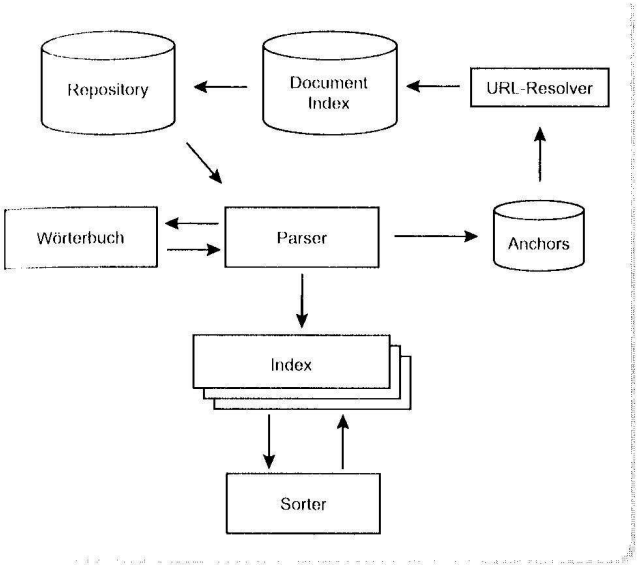


Abbildung 3.2: Dokumentenanalyse

Die Abbildung zeigt, dass ein Parser das Kernstück dieser Verarbeitungsstufe darstellt. Was ist ein Parser und wie arbeitet er? Bevor am Leitfaden der Suchmaschinenuntersuchung das hier vor sich gehende Parsing genauer unter die Lupe

[7] A.a.O. S. 81.
[8] Ebenda.
[9] A.a.O. S.83, Vgl. Abbildung ??

genommen wird, sei vorweg kurz etwas zur Entstehungsgeschichte dieses Ausdrucks gesagt. To parse ist ein englisches Verb und heißt soviel wie 'zerlegen'. Gemeint ist insbesondere die Zerlegung umfangreicherer sprachlicher Ausdrücke wie die von Sätzen in ihre Bestandteile. Das Wort hängt etymologisch, d.h. wortgeschichtlich, mit dem lateinischen 'pars' (dt. Teil) zusammen. In der antiken und mittelalterlichen Grammatik hat man die Texte in 'partes orationis', d.h. in die Bestandteile der Rede, gegliedert und damit insbesondere Wortartenbestimmungen gemeint. In der modernen Grammatik, vor allem in der maschinellen Sprachverarbeitung, hat sich der Terminus 'Parsing' für syntaktische Analysen eingebürgert, die zum Ziel haben, zusammenhängende Wortgruppen als solche zu entdecken und für weiter gehende syntaktische Analysen oder Übersetzungen zu präparieren. In einer Suchmaschine hat der Parser die folgenden Aufgaben zu bewältigen, die wir uns zunächst im Überblick und anschließend im Detail ansehen werden[10]:

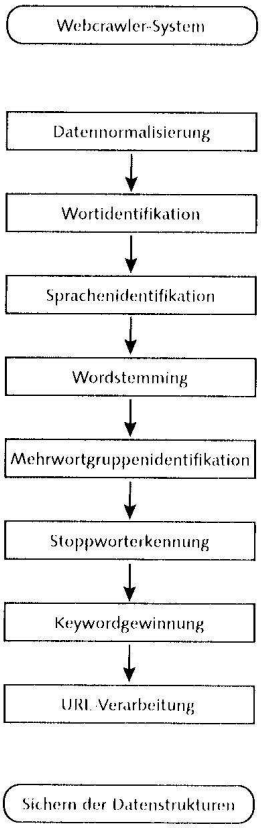


Abbildung 3.3: Parser

Schauen wir uns die einzelnen Phasen dieses Parsing-Prozesses genauer an. Zunächst müssen wir uns erst noch einmal vergegenwärtigen, dass die Suchmaschine es mit HTML-Dokumenten zu tun hat. Diese Dokumente enthalten Annotationen des natürlich-sprachlichen Textes, der Gegenstand der Untersuchung innerhalb des Parsers ist. Da für die Auswertung des Dokuments innerhalb der Suchmaschine ausschließlich der Textinhalt von Bedeutung ist, müssen erst einmal alle Sonderzeichen,

[10] A.a.O. S. 86, vgl. Abbildung **??**

wie HTML-Tags oder Javascript-Code eliminiert werden. Dieser Vorgang heißt **Datennormalisierung**[11]. Um zu illustrieren, was während dieses Prozesses vor sich geht, werfen wir einen Blick auf unsere oben verwendete Datei 'gruss.xml', die nach der Transformation in ein HTML-Dokument folgenden Quellcode präsentiert:

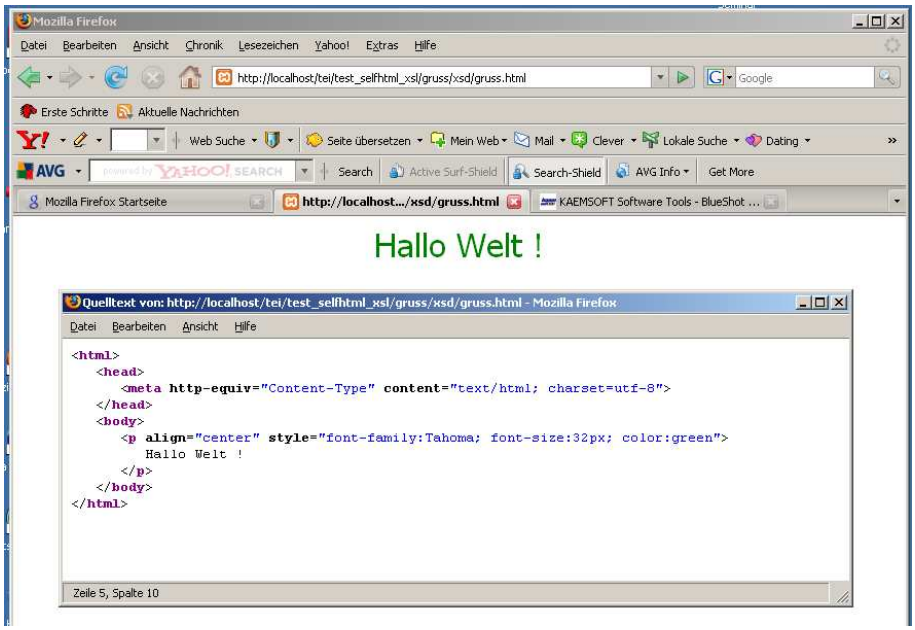


Abbildung 3.4: Datennormalisierung

Der Quellcode enthält die üblichen Tags, die nichts zum allein interessierenden Textinhalt beitragen und insofern beseitigt werden. übrig bleibt nur der Text: Hallo Welt !. Dieser wird weiter verarbeitet. Im nächsten Schritt müssen die einzelnen Wörter gefunden werden, die den Text bilden. Für einen Menschen ist das kein Problem. Eine Maschine aber verfügt nicht über das Sprachverständnis, das ein Mensch im Lauf seines Lebens erworben hat. Der Computer ist darauf angewiesen, exakte Angaben darüber zu erhalten, wie er Daten bearbeiten soll. Und ein Text wie 'Hallo Welt !' ist für den Computer nichts anderes als eine Datenmenge, die innerhalb des Speichers durch Nullen und Einsen repräsentiert ist. Nehmen wir, um dies zu illustrieren, einmal nur das H, um zu sehen, wie dieses im Rechner dargestellt wird. Der Buchstabe H wird im ASCII (American Standard Code for Information Interchange: dt. Amerikanischer Standard-Code für den Informations-Austausch) - Code durch den Dezimalwert 72 vertreten, dem der Hexadezimalwert 48 entspricht. Eine Hexadezimalzahl beruht, wie der Name sagt, auf der Basis 16, nicht auf der Basis 10 wie eine Zahl innerhalb des Dezimalsystems. Die Hexadezimalzahl 48 ist zweistellig. An letzter Stelle die Einer-Stelle, definiert durch die Potenz 16^0, an vorletzter Stelle die 16er-Stelle, definiert durch die Potenz 16^1. Wie sich die Dezimalzahl 72 ergibt, indem ich $7 * 10^1 + 2 * 10^0$ rechne, so die Hexadezimalzahl 48 als Dezimalzahl durch $4 * 16^1 + 8 * 16^0$. Berechnen wir diesen Wert, so erhalten wir 72, die Dezimalzahl. Umgekehrt lässt diese sich in die Hexadezimalzahl 48 umrechnen, wenn wir fragen, wie oft 16 in 72 enthalten ist, nämlich 4 Mal zuzüglich eines

[11] A.a.O. S. 87.

Rests von 8. Versuchen wir nun dezimal 72 oder hexadezimal 48 in eine Binärzahl zu verwandeln, dann müssen wir die Basis 2 zugrunde legen. 01001000 entspricht $0 * 2^7 + 1 * 2^6 + 0 * 2^5 + 0 * 2^4 + 1 * 2^3 + 0 * 2^2 + 0 * 2^1 + 0 * 2^0$. Genau das ist die Repräsentation des Buchstabens bzw. des ASCII-Zeichens H. So denken Computer, wenn man das so nennen möchte. Weil es nur diese Art der Repräsentation von Zeichen innerhalb von Rechnern gibt, kann der Computer keine Bedeutungen erkennen wie wir Menschen. Er weiß nicht, dass die Zeichenkette 'Hallo' ein Grußwort repräsentiert. Der Rechner weiß noch nicht einmal, dass das Leerzeichen einen Wortzwischenraum und somit Wortgrenzen markieren kann. All das muss ihm mühsam z.B. so beigebracht werden, dass ihm Folgendes gesagt wird: Immer dann, wenn du zuvor eine Reihe von aufeinanderfolgenden Zeichen der ASCII-Codes $65 - 90_{dez}$ (A..Z) oder $97 - 122_{dez}$ (a..z) zu einer Zeichenkette zusammengefasst hast und auf das ASCII-Zeichen 32_{dez} (Spacebar) stößt, beende die vorhergehende Zusammenfassung der Zeichen und beginne eine neue. Diesen Prozess nennt die Suchmaschine **Tokenizing**[12] oder **Wortidentifkation**. Als Separatoren sind nicht nur Leerzeichen, sondern auch der Apostroph und Satzzeichen zulässig. Das hat zur Folge, dass gegebenenfalls auch einzelne Zeichen wie ein durch Apostroph geführtes s in die Wortliste aufgenommen werden. Um dem entgegenzuwirken, empfiehlt es sich, Kurzwörter, die nur aus ein, zwei oder drei Zeichen bestehen, aus der Liste zu löschen. Wie sich später noch zeigen wird, resultiert daraus in der Regel kein Verlust an Aussagekraft des Extraktes. Steht eine aus einem Dokument generierte Wortliste zur Verfügung, ist als Nächstes zu prüfen, in welcher Sprache das Dokument verfasst ist. Das lässt sich am leichtesten durch Vergleich der Wortliste mit einem Lexikon oder mehreren Lexika herausfinden. Diese Ermittlung der Sprache[13] oder **Sprachidentifikation** ist tatsächlich ein aufwändiger Vorgang, der hier nicht im Einzelnen beschrieben werden soll, da zum Zweck einer solchen Darstellung komplizierte mathematische Modelle wie Hidden Markov Modelle[14] eingeführt werden müssten.

Um die Wortliste so aussagekräftig wie möglich bei gleichzeitig möglichst geringem Umfang zu machen, werden Dubletten bzw. Ableitungen von Wortstämmen gleicher Herkunft gelöscht. Das zuletzt genannte Verfahren wird **Stemming**[15] genannt, gemäß der Bezeichnung Stemma für den Wortstamm. Nach Abschluss dieses Verfahrens bleibt dann, wenn ein Text die Wörter Auto und Autos enthält, nur noch das Wort Auto übrig.

Jetzt naht das Ende der Dokumentenbearbeitung. Bevor die Früchte geerntet werden können, derart, dass aus dem Dokument eine Wortliste gewonnen worden ist, die den Inhalt des Dokuments repräsentiert, müssen nur noch die so genannten **Stoppwörter**[16] beseitigt werden. Bei diesen handelt es sich um inhaltsarme Funktionswörter wie aber, als, das usw., die wenig zum Gehalt des Textes beitragen.

---

[12] A.a.O. S. 88.

[13] A.a.O. S. 90 ff.

[14] A.a.O. S. 92.

[15] A.a.O. S. 93. Die Untersuchung der Mehrwortgruppenidentifikation wird hier übergangen, da es sich bei diesem Verfahren um eine Spezialuntersuchung handelt, die zum Verständnis des Grundsätzlichen nichts Neues beiträgt. Bei der Mehrwortgruppenidentifikation handelt es sich um die Zerlegung von Komposita nach dafür vorgesehenen Verfahren.

[16] A.a.O. S. 98.

Sind diese Wörter entfernt, dann ist als nächstes die **Keyword-Extrahierung**[17] an der Reihe. Der linguistischen Textanalyse zufolge werden „Inhalte am ehesten durch Substantive dargestellt“[18]. So werden als Schlüsselwörter bzw. keywords Substantive extrahiert, die als geeignete Repräsentanten des Textgehaltes gelten. Unter der Annahme, „dass die empirisch gefundene Häufigkeit von Wörtern eines ausreichend langen Textes mit dem Rang ihrer Bedeutung korreliert“[19], wird die Häufigkeit des Auftretens eines Schlüsselwortes zum Maß seines Gewichtes für die Bestimmung des Textinhaltes. Im Klartext: Je häufiger ein Schlüsselwort innerhalb eines Textes auftaucht, desto größer ist die Wahrscheinlichkeit, dass der Inhalt des Textes davon handelt, was Bezeichnetes des so häufig genannten Schlüsselwortes ist. Genaueres zur Gewichtung der Schlüsselwörter folgt im nächsten Abschnitt.

Bevor nun aus dem bis hierher gewonnenen Material eine „Datenstruktur entworfen“[20] wird, sei der Blick abschließend noch auf den **URL-Resolver**[21] gerichtet, der die wichtige Aufgabe erfüllt, die Links, die innerhalb eines Dokumentes vorkommen, zu sammeln und im Dokumentenindex abzuspeichern, bevor solche Linklisten vom Scheduler genutzt werden, um die Webcrawler erneut das Web durchsuchen zu lassen.

Nachdem die Extraktion der Keywords bzw. Schlüsselwörter abgeschlossen ist, beginnt der nicht minder wichtige Prozess der Aufbereitung der bis dahin gewonnenen Daten, um sie für den Suchprozess nutzbar machen zu können. Die zu erstellende Datenstruktur besteht aus drei Komponenten:

- einer Hitlist (i)
- einem direkten Index (ii)
- einem invertierten Index (iii)[22]

Wenden wir uns zunächst der **Hitlist** (i) zu. Um mit der Beschreibung eines konkreten Beispiels zu beginnen: Google verwendet zu deren Speicherung 2 Bytes. Zu den Informationen, die die Hitlist enthalten sollte, gehören vor allem Hinweise darauf, welchen Rang ein Schlüsselwort innerhalb des Dokuments hat. Eine zugrunde liegende Annahme besteht darin, davon auszugehen, dass „ein Schlüsselwort umso wichtiger für die inhaltliche Aussage ist, je weiter oben es auftritt.“[23] So wird demzufolge festgehalten, ob das Schlüsselwort im Titel, im Kopfbereich oder im Dokumentkörper auftritt. Da Schriftgröße und Formatierung auch eine Rolle spielen können, werden auch Informationen dazu festgehalten.

Der **direkte Index** (ii) nimmt die Daten der Hitlist in sich auf. Aber nicht nur die. Darüber hinaus enthält er Hinweise auf das Dokument, dem die Schlüsselwörter entstammen und die Schlüsselwörter selbst. Hierzu ein Beispiel[24]:

[17] A.a.O. S. 99 ff.
[18] A.a.O. S. 102.
[19] A.a.O. S. 103.
[20] A.a.O. S. 105.
[21] A.a.O. S. 104.
[22] A.a.O. S. 105.
[23] A.a.O. S. 106.
[24] A.a.O. S. 108f.

Dokument	Schlüsselwort	Hitlist
http://www.literatur.de/zitate/shakespeare.html	sein	[1, 4], ...
http://www.literatur.de/zitate/shakespeare.html	oder	[2], ...
http://www.literatur.de/zitate/shakespeare.html	nicht	[3], ...

Tabelle 3.1: Einfaches Beispiel eines direkten Index ohne Codierung

DocID	WordID	Hitlist
000034	004532	$CE625$
000034	000235	$67F24$
000034	001256	$E42C4$

Tabelle 3.2: Direkter Index im codierten Zustand

Das Dokument möge als einzige Textzeile den Satz „Sein oder nicht Sein“[25] enthalten. In der Hitlist würden dann die oben abgedruckten Hinweise auf den Platz des Schlüsselwortes innerhalb des Dokumentes und die übrigen relevanten Daten, von denen oben gesprochen wurde, stehen. Um die Dokumentenbezeichnung möglichst effektiv und Platz sparend zu gestalten, empfiehlt es sich, aus dem Dokumentenindex die zugehörige DocId zu wählen, um das Dokument eindeutig zu kennzeichnen.

Da jedoch kein effizienter Vergleich von Suchanfrage und Index möglich ist, wenn der Index nach dem Dokumentenindex geordnet ist, muss eine Umkehrung der Einträge vorgenommen werden, so dass es zunächst das Schlüsselwort ist, nach dem gesucht werden kann. Das leistet der Aufbau des **invertierten Indexes** (iii), der folgendermaßen[26] aussieht:

WordID	DocID
004532	000034, 001243, 000872, 007360, 083729
002176	000237, 371613, 002371, 927872, 298109
093278	000281, 287301, 984590, 298732, 029491

Tabelle 3.3: Invertierter Index

Hier sieht man, dass zu jedem Schlüsselwort die Dokumente angegeben werden, innerhalb derer sich das Schlüsselwort befindet. Wäre ein Suchmaschinenanbieter mit diesem Ergebnis zufrieden, dann wäre die Arbeit hier schon abgeschlossen, da es jetzt möglich wäre, die Liste der Dokumente auszugeben, die gefunden worden sind. Weil auf diesem Wege aber auch Dokumente gelistet würden, die mit Blick auf die aktuelle Suchanfrage irrelevant sind, müssen weitere Schritte unternommen werden, um die Ergebnisse zu verfeinern. Dies geschieht so, dass mit Hilfe des invertierten

[25] Ebenda.
[26] A.a.O. S. 110 f.

Indexes auf den direkten zugegriffen wird, so dass nun die Hitlist und andere Eigenschaften wie die Vorkommenshäufigkeit des Schlüsselwortes in Dokumenten zur Bewertung der Relevanz eines Dokumentes herangezogen werden können.

An dieser Stelle besteht Gelegenheit auf das in der Einleitung zu diesem Abschnitt zum Thema Information Retrieval benutzte Bild der virtuellen Bibliothek unüberschaubarer Ausmaße zurückzukommen. Jetzt kann die Frage beantwortet werden, welche Bestandteile der Suchmaschine den Katalogen einer Bibliothek entsprechen. Es ist der invertierte Index, der zu Schlüsselbegriffen, die über WordID's zugänglich sind, die Dokumente, die ihrerseits über die DocID identifizierbar sind, auflistet, in denen diese Schlüsselwörter vorkommen. Damit wird der Bestand an Dokumenten, der innerhalb des World Wide Web verfügbar ist, genauso erschlossen wie die Buchbestände, die in den Regalen einer Bibliothek stehen.

3.2.3 Relevanzgewichtung

Erlhofer zufolge bedeutet Relevanz eines Dokumentes „die Ähnlichkeit eines Dokuments zu der Suchanfrage“[27]. Wie ist diese feststellbar? Nachdem zunächst mit Hilfe des invertierten Indexes alle Dokumente „in die nähere Auswahl einbezogen“ worden waren, „die ein Mindestmaß an Ähnlichkeit zu der Anfrage besitzen“, gilt es nun, „eine nach Relevanz sortierte Liste zu gewinnen“[28], die ein Höchstmaß an Ähnlichkeit garantiert. Um dieses Ziel zu erreichen, werden a) vektorielle Analysen, b) Häufigkeitsberechungen und c) das Pagerank-Verfahren (letzteres insbesondere zunächst bei Google, später auch bei anderen Suchmaschinenanbietern) eingesetzt.

Vektorielle Analysen

Vektorräume sind aus der Mathematik geläufig. Innerhalb eines dreidimensionalen Koordinatensystems z. B. lassen sich alle Punkte des Raumes durch Vektoren markieren, nachdem die Achsen des Koordinatensystems als Vektoren definiert worden sind, die den Vektorraum aufspannen. Ebenso lassen sich innerhalb unserer Dokumenten- und Suchräume Vektorräume aufspannen. In der Suchmaschinenalgebra hat es sich eingebürgert, die Suchanfrage dazu zu nutzen, den Suchvektorraum aufzuspannen, indem diesem Vektorraum ebenso viele Dimensionen zugeordnet werden, wie es Suchbegriffe innerhalb der Anfrage gibt. Um das zu illustrieren, greifen wir wieder auf das oben eingeführte Beispiel zurück: „Sein oder nicht Sein. Denn Sein ohne Sinn ist nicht Sein.“

In diesem Dokumententext kommt das Wort „Sein“ viermal vor. Das Wort „Sinn“ einmal. Ein Dokumentvektor, der als Komponenten die jeweilige Anzahl des Vorkommens der die Dimensionen des Vektors bildenden Schlüsselwörter enthält, sähe folgendermaßen aus:

dokumentvektor-$A = (4, 1)$.

Nehmen wir an, in einem zweiten Dokument kämen beide Schlüsselwörter viermal vor, dann sähe der entsprechende Vektor so aus:

dokumentvektor-$B = (4, 4)$.

[27] A.a.O. S. 115.
[28] A.a.O. S. 116.

Gäbe es einen Suchvektor mit den Komponenten $(5, 2)$, dann könnte das folgende Diagramm die Abstände zwischen den Vektoren wiedergeben.[29]

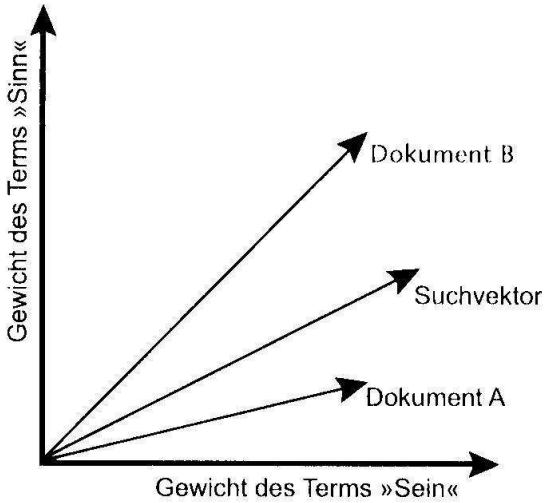


Abbildung 3.5: Gewichtetes Vektorraummodell mit zwei Termen

Im Ergebnis würde der Dokumentenvektor als der dem Suchvektor ähnlichste ermittelt, der den geringsten Abstand hinsichtlich Länge und Richtung vom Suchvektor hat. Die für die Ermittlung der Komponenten der Dokumentenvektoren benötigten Daten werden aus der Hitlist bezogen, die Auskunft über die Platzierung der Schlüsselwörter innerhalb des Dokuments, ihre Formatierung und Häufigkeit gibt.

Haeufigkeitsberechnungen

Erlhofer zufolge sind vektorielle Analysen nur eine Komponente innerhalb der Berechnung der Relevanz eines Dokumentes hinsichtlich einer Suchanfrage. Hinzukommen Berechnungen der relativen Häufigkeit eines Schlüsselwortes innerhalb eines Dokuments. Auch hier werden die Daten der Hitlist herangezogen, da diese Auskunft darüber geben, wie häufig ein Wort innerhalb des Textes vorkommt. Um die Relevanz eines Dokumentes im Vergleich verschiedener Dokumente bewerten zu können, ist es nötig, die Vorkommenshäufigkeit eines Schlüsselwortes innerhalb eines Dokumentes mit Blick auf die Gesamtzahl der vorkommenden Wörter zu relativieren. So lassen sich unterschiedliche Häufigkeiten in unterschiedlich langen Dokumenten miteinander vergleichen. Ein Beispiel: Nehmen wir ein Dokument (1) zum Faust, das 168 Wörter enthält. Das Schlüsselwort "Gretchen"komme in diesem Dokument fünfmal vor. Mit Hilfe der Formel:

(Term Frequency) TF = Häufigkeit eines Schlüsselwortes im Dokument / Anzahl der Wörter im Dokument ergäbe sich $0,03$. In einem längeren Dokument (2), das 1032 Wörter enthält, wobei das Schlüsselwort „Gretchen“ $20-mal$ vertreten ist, läge

[29] A.a.O. S. 121.

TF bei 0,019. So haben wir in (1) eine TF von 3%, während sie in (2) bei nur knapp 2% liegt. Demzufolge wäre (1) das relevantere Dokument zu einer Suchanfrage, die als Schlüsselwort „Gretchen“ enthält. Allgemein gilt: Je höher TF, desto relevanter das Dokument.

Das Pagerank-Verfahren

Dieses Verfahren hat den Siegeszug der Suchmaschine Google mitbegründet. Entwickler sind Lawrence Page, nach dem das Verfahren benannt worden ist, und Sergey Brin[30], die dessen Grundlagen in einer wissenschaftlichen Arbeit an der Stanford University entwickelt haben. Dem Verfahren liegt ein sehr einfacher Gedanke zugrunde. Page und Brin haben das in der Wissenschaft angewendete Kriterium zur Messung der Bedeutung einer wissenschaftlichen Arbeit auf das Web übertragen, indem Sie sagten: „Je mehr eingehende Links eine Seite zählt, desto bedeutsamer ist ihr Inhalt.“[31] Das entspricht dem in der Wissenschaft angewendeten Kriterium, demzufolge eine wissenschaftliche Arbeit umso bedeutender ist, je häufiger sie zitiert wird. Page und Brin zufolge drückt der Pagerank „die bestimmte Wahrscheinlichkeit aus, mit der ein Surfer eine Webseite besucht. Dieser typische Benutzer wird als >Random Surfer< bezeichnet, weil er sich von einer Seite zur nächsten bewegt und dabei einen beliebigen Link nutzt, ohne dabei auf dessen Inhalt zu achten. Die Wahrscheinlichkeit, einen bestimmten Link zu verfolgen, ergibt sich demnach einzig und allein aus der Anzahl der zur Verfügung stehenden Links auf einer Seite.“[32] Der Pagerank-Algrorithmus liefert die Daten, um abschließend zu gewichten, inwiefern eine Website einer Anfrage ähnlich ist oder nicht, nachdem zuvor vektorielle und Häufigkeitsanalysen auf der Grundlage der Keyword-Extraktion die Basis für diese abschließende Beurteilung zur Verfügung gestellt haben. Wie der Pagerank-Algorithmus im Einzelnen arbeitet, das soll mit Hilfe des im Anhang A.1 abgedruckten Materials von Markus Sobeck, das im World Wide Web unter dem URL http://pr.efactory.de/d-index.shtml erarbeitet werden.

3.2.4 Der Suchvorgang

Der Vergleich zwischen einer Suchanfrage und der Datenbasis der Suchmaschine läuft im Grundsatz folgendermaßen ab: Der im entsprechenden Dialogfeld der Suchmaschine eingetragene Suchstring wird im Prinzip genauso bearbeitet wie die Dokumente der Datenbasis bearbeitet werden. Am Ende steht eine Schlüsselwortliste, die in einen Suchvektor konvertiert wird, der mit den Dokumentenvektoren der in der Datenbasis gespeicherten Dokumentendaten verglichen wird gemäß den oben beschriebenen Verfahren. Die Dokumente, deren Vektoren dem Suchvektor am ähnlichsten sind, werden als Ergebnis der Anfrage zurückgeliefert. Die zurückgelieferte Trefferliste enthält die der Suchanfrage ähnlichsten Dokumente am Beginn der Liste.[33]

[30] A.a.O. S. 126.
[31] Ebenda.
[32] A.a.O. S. 127.
[33] A.a.O. S. 149.

4 Formale Sprachen

4.1 Grundbegriffe

In der sich anschließenden Zusammenstellung von Grundbegriffen[1] sollen diejenigen Begriffe eingeführt und erklärt werden, die in den folgenden Darstellungen benutzt werden. Um diesen Abschnitt nicht über Gebühr zu gewichten, werden alle nicht zwingend benötigten, aber dennoch nicht minder wichtigen Grundbegriffe nicht ausdrücklich behandelt. Es wird aber dringend empfohlen, diese in einschlägigen Sammlungen zu erarbeiten.

4.1.1 Mengenlehre und Algebra

Mengen

Beginnen wir mit dem Begriff der Menge. Er ist so elementar, dass er in der Mengenlehre als undefinierter Grundbegriff eingeführt wird, dessen Bedeutung intuitiv einleuchtet. Obwohl der Mengenbegriff alles andere als einfach ist, scheint dieses Vorgehen doch gerechtfertigt angesichts der Tatsache, dass wir uns den Begriff erklären können, indem wir zunächst Beispiele für Mengen aufzählen. Da sind zum Einen endliche Mengen wie

- die Bewohner der Stadt Schleswig
- das lateinische Alphabet
- die Wörter innerhalb dieses Readers

oder zum Anderen unendliche Mengen wie

- die natürlichen Zahlen
- die reellen Zahlen

Zu den unendlichen Mengen Mengen wie die Atome im Universum zu rechnen ist riskant, da niemand weiß, ob das Universum endlich oder unendlich ist. Obwohl die Anzahl der Atome in einem endlichen Universum unvorstellbar groß wäre, wäre die Menge dennoch nicht unendlich, da sie begrenzt und abzählbar wäre.

Verglichen mit den zuletzt genannten Mengen werden wir es im Folgenden ausschließlich mit eher überschaubaren Mengen zu tun haben, den Mengen von Wörtern, die wir mit Hilfe von Grammatiken erzeugen können, oder denen, die zum

[1] Die folgenden Darstellungen sind weitgehend auf Klenk 1980, S. 1-20 gegründet.

Wortschatz unserer Sprache gehören. Dass wir es im Feld der Untersuchung Formaler Sprachen mit einem Wortbegriff zu tun haben, der sich von dem alltäglichen Verständnis unterscheidet, zeigt sich schon darin, dass davon die Rede war, dass Wörter mit einer Grammatik erzeugt werden sollen. Wenn wir von unserer alltäglich uns begegnenden Sprache sprechen, dann haben wir es mit einem breits existierenden Wortschatz zu tun, den wir nicht erst herstellen wollen. Dazu später mehr.

Kurioserweise betrachtet man in der Mengenlehre auch Mengen, die keine Elemente enthalten. Solche Mengen heißen leere Menge. Das Symbol, das dazu dient, kurz und knapp eine leere Menge zu bezeichnen ist $\oslash$.

Element-Beziehung

Zurück zu den Mengen. Mengen bestehen aus Elementen. Wenn jemand Bewohner der Stadt Schleswig ist, dann ist er ein Element dieser Menge. In Abkürzung:

$\{x \mid x \in \text{Bewohner Schleswigs}\}$

Alltagssprachlich formuliert: Die Menge aller x, für die gilt: x ist Bewohner Schleswigs. Die Variable x bezeichnet eine Menge von Individuen, die die Eigenschaft besitzen, zur Menge der Bewohner Schleswigs zu gehören. Will man das Gegenteil sagen, will man also von allen Nicht-Bewohnern Schleswigs sprechen, dann schreibt man:

$\{x \mid x \notin \text{Bewohner Schleswigs}\}$

Mit diesem Ausdruck bezeichnet man den Rest der Weltbevölkerung, der nicht in Schleswig wohnt.

Diese mengentheoretische Schreibweise und Bezeichnung empfiehlt sich immer dann, wenn es nicht mehr möglich ist, in einer informellen Darstellung die einzelnen Elemente einer Menge durch Aufzählung anzugeben, wie dies z.B. problemlos bei der Angabe der Kleinbuchstaben des lateinischen Alphabets möglich ist. Diese Menge könnte wie folgt angegeben werden:

Die Menge der Kleinbuchstaben des lateinischen Alphabets = {a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z}.

Kardinalzahl

Will man etwas über die Anzahl der Elemente einer Menge sagen, dann kann dazu die Kardinalzahl der Menge angegeben werden. Die Kardinalzahl ist eine natürliche Zahl. Am 27.8.2008 gehören zur Menge 'Bewohner Schleswigs' 24462 Einwohner[2]. Um diese Kardinalität anzugeben, gibt es folgende Notation:

card(Bewohner Schleswigs) = 24462

[2] http://www.trendmile.de/staedte/schleswig.php

Allgemein wird die Kardinalität einer Menge M als card(M)= n, mit n $\in \mathbb{N}$ angegeben.

Inklusions-Beziehung

Wollen wir nur einen Teil der Menge der Bewohner Schleswigs auswählen, dann können wir dies mit Hilfe der Teilmengen-Relation tun: In formaler Schreibweise sieht dies folgendermaßen aus:

Bew-Sl sei die Menge aller Bewohner Schleswigs,
Bew-StJ sei die Menge aller Bewohner des Schleswiger Stadtteils St.Jürgen, dann können wir schreiben:

Bew-Sl :=[3] {x | x $\in$ Bewohner Schleswigs}
Bew-StJ := {y | y $\in$ Bewohner St.Jürgens}

Da card(Bew-StJ) $<$ card(Bew-Sl), gilt

Bew-StJ $\subset$ Bew-Sl,

d.h. die Menge der Bewohner St. Jürgens ist eine echte Teilmenge der Bewohner Schleswigs. Sind die Inklusionsbeziehungen, die zwischen Mengen bestehen, nicht so beschaffen, dass von vornherein klar ist, dass die Kardinalität einer Obermenge M größer als die Kardinalität einer Untermenge N ist, dann kann die Enthaltensein-oder-gleich-Relation $\subseteq$ verwendet werden. Mit anderen Worten:

Wenn N $\subseteq$ M, dann gilt: card(N) $\leq$ card(M).

Die Enthaltensein-oder-gleich-Relation lässt sich nutzen, um die Gleichheit von Mengen N und M zu definieren:

Wenn N $\subseteq$ M und M $\subseteq$ N, dann gilt: N = M.

Mengen-Vereinigung

Wenn wir über Mengen sprechen, deren Elemente wir zusammenfassen möchten, dann steht die Operation 'Vereinigung' $\bigcup$ zur Verfügung. Nehmen wir einmal an, wir wollen über die Mengen der Bewohner Schleswigs (Bew-Sl) und Londons (Bew-L) sprechen, so können wir beide Mengen vereinigen, indem wir Folgendes hinschreiben:

Bew-Sl $\cup$ Bew-L := {x | x $\in$ Bew-Sl oder x $\in$ L).

[3] definitionsgemäß gleich

Potenzmenge

Mengen lassen sich in Teilmengen zerlegen. Nehmen wir einmal eine Buchstabenmenge M = {a, b, c}. Zu dieser Menge gehören die folgenden Teilmengen. So ist etwa ⊘ eine Teilmenge von M, da die leere Menge Teilmenge jeder Menge ist, sogar ihrer selbst. Daher gilt: ⊘ ⊆ ⊘. Aber auch die Menge, die a als Element enthält{a} ist eine Teilmenge von M, oder die Menge {b, c}. Wenn es gelungen ist, alle Teilmengen einer Menge M aufzuzählen, dann ist es gelungen, deren Potenzmenge P(M) anzugeben. D.h., die Potenzmenge P einer Menge M ist die Menge aller Teilmengen von M. In einer formalen Notation mit A und M als Variablen, die beliebige Mengen bezeichnen:

$P(M) := \{A \mid A \subseteq M\}$

Die Potenzmenge der oben angegebenen Beispielmenge ist folgende: P(M)= {⊘, {a}, {b}, {c}, {a, b}, {b, c}, {a, c},{a, b, c}}

Geordnete Mengen

Bisher, solange wir mit Mengen operierten, ist es nicht darauf angekommen, in welcher Reihenfolge die Elemente der Mengen aufgezählt worden sind. Zwar wurden die Kleinbuchstaben des lateinischen Alphabets in der Reihenfolge aufgezählt, die wir gelernt haben. Aber das war Zufall. Wir hätten ebenso gut eine andere Reihenfolge wählen können, ohne dass wir damit über eine andere Menge gesprochen hätten. So ist die Menge der Primzahlen < 10 immer dieselbe, gleichgültig ob ich {2,3,5,7} oder {7,5,3,2} oder {5,3,7,2} schreibe. In jedem der drei Fälle sprechen wir über ein und dieselbe Menge.

Gelegentlich ist es aber nötig, über Mengen in bestimmten Anordnungen zu sprechen. Je nach Anzahl der Elemente wird dann von Paaren oder 2-Tupeln (2 Elemente), Tripeln oder 3-Tupeln (3 Elemente), Quadrupeln oder 4-Tupeln (4 Elemente), Quintupeln oder 5-Tupeln (5 Elemente) usw. gesprochen. Geordnete Mengen werden anders notiert als Mengen, sie werden nicht in geschweifte, sondern in runde Klammern eingeschlossen. Die einzelnen Elemente einer geordneten Mengen werden Komponenten der geordneten Menge genannt. Wir werden später Grammatiken und Automaten als geordnete Mengen definieren. In den Definitionen werden die Bedeutungen der Komponenten festgelegt. Wenn dann einzelne Grammatiken bzw. Automaten untersucht werden, werden die Bestandteile dieser Objekte auf der Grundlage der Festlegungen innerhalb der Definitionen aufgezählt.

Die Notation, innerhalb derer wir eine Grammatik beschreiben, könnte dann folgendermaßen aussehen:

G = ({S, NP, VP, DET, N, V}, {die, Katze, Maus, sieht}, {S → NP VP, NP → DET N, VP → V NP, V → sieht, DET → die, N → {Katze, Maus}}, S).

Aufgrund der Tatsache, dass Grammatiken als Quadrupel folgendermaßen definiert werden können,

G = (V_N, V_T, R, S), mit V_N als Menge der Nicht-Terminalsymbole, V_T als Men-

ge der Terminalsymbole, R als Menge der Regeln und S als Startsymbol,

wissen wir, dass in der oben angegebenen Grammatik gilt:

V_N = {S, NP, VP, DET, N, V},
V_T = {die, Katze, Maus, sieht}
R = {S → NP VP, NP → DET N, VP → V NP, V → sieht, DET → die, N → $\{Katze, Maus\}$
$S = S$.

Kartesisches Produkt und Relationen

Die oben beschriebene Grammatik zeigt, dass die Komponenten eines Tupels in systematisch geordneten Beziehungen zueinander stehen. Die Grammatik-Regeln als dritte Komponente der Grammatik geben an, in welcher Beziehung V_N zu V_T stehen. Um solche Tupel herzustellen, werden Kartesische Produkte benutzt. Bevor wir auf die allgemeine Bestimmung dessen, was unter einem Kartesischen Produkt zu verstehen ist, eingehen, wollen wsir uns ein Beispiel anschauen:

„Auf der Menge der natürlichen Zahlen ist die kleiner-als-Relation $<$ definiert durch alle Zahlenpaare, an deren 1. Stelle von links eine Zahl steht, die in der Reihenfolge des Zählens vor der an der 2. Stelle stehenden Zahl kommt. so ist (1,10) aus der Relation $<$, (10,1) nicht (also $(1,10) \in <$, $(10,1) \notin <$).“[4] Wenn wir dieses Beispiel verallgemeinern, dann können wir sagen: Relationen zwischen Objekten geben Beziehungen an, die festgelegt werden, indem das Kartesische Produkt der Mengen gebildet wird, zu denen diese Objekte gehören. „Eine n-stellige Relation R zwischen $M_1, \ldots, M_n$ ist eine Teilmenge von $M_1 \times \ldots \times M_n$.“[5]

Das Kartesische Produkt von $M_1, \ldots, M_n$ Mengen schreiben wir dann als

$M_1 \times \ldots \times M_n := \{(x_1, \ldots, x_n) | x_1 \in M_1$ und ... und $x_n \in M_n\}$.

Die Relation $<$ würde folgendermaßen definiert:

$<$: M × N := {(x, y) | x ∈ M und y ∈ N mit x in der Reihenfolge des Zählens vor y}.

Kartesisches Produkt und Funktionen

Funktionen sind eine Teilmenge der Relationen, für die besondere Bedingungen gelten. Relationen sind Funktionen nur dann, wenn die Zuordnungen zwischen den Mengen, die aufeinander abgebildet werden, in der Weise eindeutig sind, dass jedem Element des Definitionsbereichs höchstens ein Element des Wertebereichs zugeordnet ist. Schauen wir uns das an einem Beispiel an:

[4] Klenk 1980 S. 7.
[5] Ebenda.

Nehmen wir den Definitionsbereich der reellen Zahlen, $\mathbb{R}$, die Zuordnungsvorschrift sei $f_{(x)} = x^2$, der Wertebereich sei ebenfalls der Bereich der reellen Zahlen, $\mathbb{R}$. Die Zuordnungsvorschrift legt fest, dass jedem x $\in D_f$ genau ein $f_{(x)} \in W$ zugeordnet wird. Oder abstrakter formuliert: „Da eine Abbildung f: A $\rightarrow$ B vollständig durch die geordneten Paare aus den jeweils einander zugeordneten Elementen von A und B beschrieben wird, kann man eine Abbildung auch als spezielle Relation definieren: Eine Relation $f \subseteq A \times B$ heißt eine Abbildung (Funktion), wenn sie *linksvollständig* und *rechtseindeutig* ist, d.h., wenn gilt:

1. Für jedes $a \in A$ existiert ein $b \in B$ mit $(a, b) \in f$. (Wir schreiben f: a $\longmapsto$ b statt $(a, b) \in f$.)

2. Sind $(a, b_1), (a, b_2) \in f$, so ist $b_1 = b_2$.“[6]

Vergleichen wir die Funktion $f_{(x)} = x^2$ mit einer Relation, die jedem Element der Menge der natürlichen Zahlen $\mathbb{N}$ die Menge ihrer Teiler T zuordnet, dann zeigt sich sofort, dass diese Relation keine Funktion ist, da außer dem Element 1 jedem Element aus dem Vorbereich $\mathbb{N}$ mehr als ein Element des Nachbereichs T zugeordnet ist.

Rekursive Definitionen

Wir machen von rekursiven Definitionen Gebrauch im Abschnitt **??**. Um das Wesen einer rekursiven Derfinition zu erfassen, empfiehlt es sich, von einem Beispiel auszugehen. Man stelle sich vor, eine rekursive Definition für die Menge der natürlichen Zahlen $\mathbb{N}$ geben zu wollen ausgehend von 1, indem die Zahlen von $\mathbb{N}$ durch Strichmengen dargestellt werden. Das sieht dann so aus, dass die 1 durch |, die 2 durch ||, die 3 durch ||| usw. dargestellt werden. Eine rekursive Definition dieser Zahlen besteht aus zwei Schritten:

1. „| ist eine natürliche Zahl,

2. Für alle x gilt: Ist x eine natürliche Zahl, dann ist auch x| eine natürliche Zahl.

1. und 2. zusammen ergeben eine Definition der natürlichen Zahlen. [...] Mit 1. wird | konstruiert, mit 2. dann ||. Da || wiederum für x eingesetzt werden kann, erhält man durch 2. ||| und durch weitere Anwendung von 2. ||||, ||||| usw. Das Beispiel zeigt, was für eine rekursive Definition einer Menge charakteristisch ist. Sie besteht aus einem (oder mehreren) Anfangsschritt(en), dem *Rekursionsanfang*, wodurrch zunächst bestimmte zu der Menge gehörende Elemente festgelegt werden. Wir wollen diese hier Anfangselemente nennen. Auf den Rekursionsanfang folgen ein oder mehrere weitere Schritte, die *Rekursionsschritte*. Diese zeigen, wie man ausgehend von den Anfangselementen weitere Elemente der Menge konstruiert.“ [7]

[6] Duden 1990 S. 9 f.
[7] Klenk 1980 S. 11 f.

4.1.2 Grundbegriffe der Theorie Formaler Sprachen

Alphabet

„Ein Alphabet ist eine endliche nicht-leere Menge von Symbolen.“[8] Beispiele für Alphabete sind die

- die Kleinbuchstaben des lateinischen Alphabets
- die Wörter dieses Readers
- die Ziffern des Dezimalsystems

Wort

Sei V ein Alphabet. Eine geordnete Menge $(a_1, \ldots, a_n)$ mit $a_i \in$ V für 1≤i≤n ist eine Symbolkette (kurz Kette) bzw. ein Wort über V. Der Einfachheit halber schreibt mann statt $(a_1, \ldots, a_n)$ auch einfach $a_1, \ldots, a_n$.

Die Symbolkette mit n = 0 heißt leeres Wort, bezeichnet durch ϵ.[9]

4.2 Grammatiken

Wir haben, als wir Texte annotierten, Tags wie <s>, <cl> oder <phr> und manch andere kennen gelernt und eingesetzt. Ein Tag wie <phr> , das wir mit Blick auf die in der Duden-Grammatik gebräuchlichen Termini verwendet haben, kann nun seinerseits auch mit Blick auf andere Grammatikmodelle auf andere Art weiter eingeteilt werden. In der Grammatiktheorie der vergangenen 50 Jahre haben sich insbesondere im amerikanischen Strukturalismus Bezeichnungen wie NP für Nominalphrase, DET für Determinierer (damit wird eine Menge von Ausdrücken zusammengefasst, zu denen Artikel und Pronomina gerechnet werden), N für Nomen, VP für Verbalphrase, V für Verb, AdvP für Adverbialphrase, Adv für Adverb, PP für Präpositionalphrase, P für Präposition und andere eingebürgert. Mit Hilfe solcher Kategorien (Einteilungsbegriffe) lassen sich Regeln formulieren, die zur Erzeugung von Sätzen benutzt werden können. Schauen wir uns ein Beispiel an:

Die Regeln

(1)	S	→	NP	VP
(2)	NP	→	DET	N
(3)	VP	→	V	PP
(4)	PP	→	P	NP

Tabelle 4.1: Phrasenstrukturregeln

können einen Satz erzeugen wie

[8] Klenk 1980 S. 15.
[9] Klenk 1980 S. 15.

(5) Die Katze liegt auf der Matte.

Vorausgesetzt wir ergänzen unsere Regeln um weitere der folgenden Art:

(6)	N	→	{Katze, Matte}
(7)	DET	→	{die, der}
(8)	V	→	{liegt}
(9)	P	→	{auf}

Tabelle 4.2: Lexikalische Regeln

Die geschweiften Klammern auf der rechten Seite der Regeln (6) bis (9) sind Zeichen dafür, dass in sie Mengen von Ausdrücken eingeschlossen werden, die für die Zeichen auf der linken Regelseite eingesetzt werden können. Je umfangreicher die Mengen sind, die den einzelnen Bezeichnern zugeordnet werden, desto größer ist die Anzahl der Sätze, die mit Hilfe nur einer Regel gebildet werden können. Erhöhen wir die Anzahl der Regeln, so lassen sich entsprechend größere Satzmengen produzieren. Weil eine Grammatik dieser Art die Eigenschaft besitzt als Erzeugungssystem zu fungieren, heißt sie entsprechend Erzeugungs- bzw. Produktionssystem.

Sehen wir uns die Anwendung einer solchen Grammatik an, indem wir den Satz (5) mit den zur Verfügung stehenden Regeln ableiten. Wiederholen wir den Satz hier der Bequemlichkeit halber als

(10) Die Katze liegt auf der Matte.

und schauen wir, wie sich eine solche Ableitung durchführen lässt. Wenn wir mit dem Satz als der zu analysierenden Zeichenkette beginnen, könnten wir versuchen, die Struktur des Satzes schrittweise zu ermitteln. Wir müssten dementsprechend, ausgehend von der Zeichenkette, fragen, ob sich die Ausdrücke der Zeichenkette mit den zur Verfügung stehenden Regeln syntaktisch, d.h. bezogen auf den Satzbau, erklären lassen. Wenn wir links im Satz beginnen, zeigt sich, dass wir die Zeichenkette 'die' ableiten können (vorausgesetzt, wir ignorieren die Unterschiede zwischen Groß- und Kleinschreibung, auf die es im gegebenen Kontext nicht ankommt, da es sich bei diesem Unterschied zunächst nur ein die Orthographie betreffendes Phänomen handelt, das keinen unmittelbaren Einfluss auf die Frage hat, ob die Zeichenkette 'die' an dieser Stelle syntaktisch zulässig ist oder nicht). Nach Anwendung der Regel

(7) DET → {die, der}

könnten wir schreiben

(11) **DET** Katze liegt auf der Matte (7)

In (11) haben wir der Deutlichkeit halber noch einmal die angewendete Regel hinzugeschrieben, um zu zeigen, dass diese angewendet wurde, um (11) abzuleiten.

Wenn wir dieses Verfahren wiederholt anwenden, dann können wir den Satz folgendermaßen weiter ableiten (Vgl. Tabelle **??**).

(12)	DET **N** liegt auf der Matte	(6)
(13)	**NP** liegt auf der Matte	(2)
(14)	NP liegt auf **DET** Matte	(7)
(15)	NP liegt auf DET **N**	(6)
(16)	NP liegt auf **NP**	(2)
(17)	NP liegt **P** NP	(9)
(18)	NP liegt **PP**	(4)
(19)	NP **V** PP	(8)
(20)	NP **VP**	(3)
(21)	**S**	(1)

Tabelle 4.3: Ableitung fuer Beispielsatz (10)

Das Vorgehen besteht aus der Anwendung des immer gleichen Verfahrens. Wir schauen, ob unsere Zeichenkette Ausdrücke oder eine Ausdrucksfolge enthält, die als Input einer Regel fungieren können. Ist dies der Fall, dann wird die entsprechende Zeichenfolge durch das auf der linken Seite stehende Symbol ersetzt. Ein mechanischer Vorgang, wie es den Anschein hat, der nur auf Mustervergleichen beruht. Dass es sich bei diesem Vorgehen um einen recht anspruchsvollen Musterkennungsprozess handelt, werden wir im Folgenden sehen, wenn wir einen Automaten analysieren, der diese Aufgabe vollständig mechanisch zu lösen versucht.

Betrachten wir einige weitere Eigenschaften unserer Ableitung. Zunächst sehen wir, dass offensichtlich alle Regeln (1)-(4) und (6)-(9) angewendet wurden. Das heißt, dass unsere Grammatik gerade ausreichend komplex ist, um einen Satz wie unseren Beispielsatz abzuleiten. Des Weiteren sehen wir, dass wir offensichtlich Bottom Up vorgegangen sind. D.h., wir sind von unserer Zeichenkette ausgegangen, um diese schrittweise abzuleiten, indem wir Schritt für Schritt die Eingaben auf den rechten Regelseiten durch das Symbol der zugehörigen linken Regelseite ersetzt haben, bis wir das Startsymbol erreicht haben. Gelingt dies, können wir also eine Zeichenkette durch regelkonforme Ersetzungen bis hin zum Startsymbol ableiten, dann gilt der analysierte Satz gemäß der angewendeten Grammatik als akzeptiert. Dass eine solche Ableitung auf der Grundlage der von uns verwendeten Beispielgrammatik auch misslingen kann, möge das folgende Beispiel illustrieren:

(22) Die Katze jagt die Maus.

(23)	**DET** Katze jagt die Maus	(7)
(24)	DET **N** jagt die Maus	(6)
(25)	**NP** jagt die Maus	(2)
(26)	NP jagt **DET** Maus	(7)
(27)	$\diamond$	

Tabelle 4.4: Ableitung fuer Beispielsatz (22)

In (27) bricht die Ableitung, gekennzeichnet durch $\diamond$, ab. Es gibt keine Eingaben für eine der rechten Seiten der geltenden Regeln, die eine Ersetzung zuließe. Natürlich lassen sich die Regeln leicht so modifizieren, dass eine Ableitung des Satzes (22) terminiert.

Die Struktur einer Grammatik wie der oben beschriebenen lässt sich auf der Grundlage der bis dahin beschriebenen Elemente folgendermaßen skizzieren. Offensichtlich besteht unsere Grammatik G aus einer Menge von Nicht-Terminalsymbolen V_N(S, NP, VP, PP, V, P, DET, N), einer Menge von Terminalsymbolen V_T (die, der Katze, Matte, liegt), einer Menge von Regeln R (vgl. (1) bis (4) und (6) bis (9)) und einem ausgezeichneten Symbol, dem Startsymbol S. Die Menge der Terminalsymbole können wir auch das Lexikon nennen. Auf diese Wörter kann unsere Grammatik mit Hilfe der lexikalischen Regeln zugreifen, die erlauben, die Wörter unseres Lexikons einer Teilmenge aus V_N, den lexikalischen Kategorien DET, N, V und P, zuzuordnen. Denen stehen die sogenannten phrasalen Kategorien S, NP, VP und PP gegenüber. Die Menge der Nicht-Terminalsymbole lässt sich demzufolge in die Teilmengen der lexikalischen und der phrasalen Kategorien zerlegen. Abgekürzt lässt sich unsere Grammatik jetzt so hinschreiben:

(28) G = (V_N, V_T, R, S).

Eine solche Grammatik heißt kontextfrei, wenn sie die folgenden Bedingungen erfüllt:

- Sie enthält ein Lexikon, innerhalb dessen den im Lexikon vorkommenden Wörtern die entsprechenden lexikalischen Kategorien zugeordnet sind,
- sie enthält Regeln der Form $A \rightarrow \varphi$ wobei A für ein Nicht-Terminalsymbol (in unserem Beispiel also S, NP, VP, PP, DET, N, V, und P) steht und φ aus einem Ausdruck besteht, der entweder aus Nicht-Terminalsymbolen oder aus Terminalsymbolen bestehen darf. Der Pfeil hat die Bedeutung 'wird expandiert zu' oder 'besteht aus'.[10]

Die Verwendung des Terminus „kontextfrei“ als eines Ausdrucks zur Charakterisierung der Eigenschaften einer Grammatik ist schwer einzuordnen, auch wenn die Definition dessen, was unter „kontextfrei“verstanden wird, bekannt ist. Warum nennt man eine Grammatik, die über die oben beschriebenen Eigenschaften verfügt, eine kontextfreie Grammatik. Die Bedeutung einer solchen Kennzeichnung wird klarer, wenn man sich die alternativen Bezeichnungen für andere Grammatiktypen vor Augen hält, von denen die kontextfreie Grammatik abgegrenzt wird. Da wäre zunächst die sogenannte kontextsensitive Grammatik zu nennen. Gerade in Gegenüberstellung zu diesem Typ werden die Besonderheiten der kontextfreien Grammatik besonders klar. Ohne in die Einzelheiten zu gehen, soll hier nur soviel gesagt werden, dass eine kontextsensitive Grammatik weniger restrikitv hinischtlich der Festlegung dessen, was in Regeln vorkommen darf, ist als die kontextfreie Grammatik. Während kontextfreie Grammatiken auf der linken Seite einer jeden Regel

[10] Vgl. Klenk 1980 S. 27 f. bzw. Sag 2003 S. 26.

nur Nicht-Terminalsymbole zulassen, dürfen kontextsensitive Regeln in ihren linken Seiten Kombinationen von Terminal- und Nicht-Terminalsymbolen enthalten. So ist folgender Regeltyp denkbar:

(29) xAy → xzy

Das bedeutet, dass das das Nicht-Terminalsymbol A nur im Kontext $x...y$ in z übertragen werden darf. Nehmen wir, unter der stillschweigenden Voraussetzung all der übrigen zu ergänzenden Produktionen einschließlich des Lexikons, die Regeln

(30) ein N → ein Hund, (31) eine N → eine Katze,

dann wird durch diese Regeln festgelegt, dass N in den unterschiedlichen Kontexten entweder zu Hund oder zu Katze expandiert wird. Kontextsensitive Grammatiken gehören zu einem Grammatiktyp, den man eine Typ-1-Grammatik nennt. Demzufolge sind die kontextfreien Grammatiken Typ-2-Grammatiken. In beide Richtungen kennt dieses System Erweiterungen: Auf die kontextsensitiven Grammatiken folgen die unbeschränkten Regelgrammatiken, Typ-0-Grammatiken, Grammatiken, die Sprachen erzeugen, die nicht entscheidbar sein können, d.h., dass es keinen Algorithmus gibt, mit dessen Hilfe angegeben werden kann, ob ein mit einer Typ-0-Grammatik erzeugter Ausdruck zu der von der Grammatik erzeugten Sprache gehört oder nicht[11]. Diesen schwierigen Eigenschaften der Typ-0- und auch der Typ-1-Grammatiken wird im Folgenden nicht weiter nachgegangen. Wir konzentrieren uns auf die Typ-2-, die kontextfreien, und die Typ-3-Grammatiken, die sogenannten linearen Grammatiken, deren Struktur die Möglichkeit eröffnet, Automaten zu konstruieren, die in endlich vielen Schritten entscheiden, ob ein von einer Typ-2- bzw. Typ-3-Grammatik erzeugter Ausdruck zu der von der jeweiligen Grammatik hervorgebrachten Sprache gehört oder nicht.

4.3 Automaten

Kommen wir wieder zurück auf die Textannotation, an die wir zu Beginn des vorigen Abschnitts erinnerten. Als wir in oXygen Texte syntaktisch annotierten, haben wir dies als kompetente Sprecher unserer Sprache gemacht, die zudem noch eine Grammatik beherrschen, um mit deren Hilfe die gegebenen Sätze syntaktisch zu strukturieren. Nun kann man sich vorstellen, dass eine solche Aufgabe durch eine Maschine erledigt wird. Was muss man eine Maschine „lehren“, die über entsprechende Fähigkeiten verfügen soll? Diese Frage wollen wir jetzt untersuchen.

4.3.1 Deterministische endliche Automaten (DEA)

Zunächst müssen wir den Begriff der Automatisierung klären. Was ist das? Etwas zu automatisieren heißt, einen mechanischen Ablauf zu ermöglichen. Ein mechanischer Ablauf ist ein solcher, der ohne Eingriff von außen selbst gesteuert zum Abschluss

[11] Klenk 2003, S.79.

kommt. Wie kann ein solcher Ablauf aussehen, wenn es um von Grammatiken produzierte Zeichenketten geht? Man kann sich vorstellen, dass ein Automat existiert, der Zeichenketten den Regeln der Grammatik entsprechend hervorbringt. Das wäre ein Erzeugungsautomat. Andererseits ist ein Automat denkbar, der erkennt, ob eine gegebene Zeichenkette den Regeln einer gegebenen Grammatik entspricht oder nicht. Das wäre ein Automat als ein sogenannter Recognizer. Wenden wir uns zunächst dem Recognizer zu. Dessen Struktur soll jetzt erklärt werden. Wir beginnen mit der basalen Beschreibung der Elemente eines Automaten.

Zwei Elemente arbeiten zusammen: ein Leseband und eine Steuereinheit. Das Leseband ist eine Folge von Zeichen, die Steuereinheit befindet sich in Zuständen (Vgl. Abbildung ??[12]).

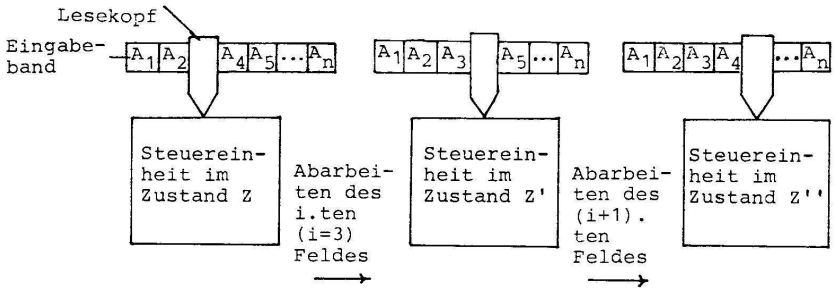


Abbildung 4.1: Steuereinheit und Lesekopf eines einseitigen endlichen Automaten

Es gibt einen Anfangs- (Z) und einen Endzustand (Z") und eine Übergangsfunktion, die angibt, unter welchen Bedingungen der Automat von einem Zustand in den anderen übergeht, bis er, ausgehend vom Anfangszustand, den Endzustand erreicht hat. Um kurz vorauszublicken. Innerhalb eines Recognizers könnte Folgendes ablaufen: Auf dem Leseband befindet sich eine Zeichenkette ($A_1 ... A_n$), über die geurteilt werden soll. Der Automat befindet sich in einem Anfangszustand (Z), liest das erste Zeichen (in Abbildung ?? A_3). Die Übergangsfunktion legt fest, was geschehen soll, wenn der Automat ein Zeichen in einem bestimmten Zustand liest, mit der Folge, dass dieses Ereignis, z.B. der Übergang in einen anderen Zustand (Z'), eintritt, nachdem das Zeichen gelesen worden ist. Das setzt sich so lange fort, bis ein Endzustand (Z") erreicht ist.

Nun aber zu präzisen Definitionen[13]:

Definition 4.3.1.1 des **d**eterministischen **e**ndlichen **A**utomaten (DEA).
Ein DEA ist ein Quintupel (Z, A, δ, z_0, E), für das Folgendes gilt: Z ist eine endliche Menge von Zuständen, A ein endliches Eingabealphabet, z_0 aus Z der Startzustand, E $\subseteq$ Z eine Menge von Endzuständen und δ die Übergangsfunktion, die Z$\times$A auf Z abbildet, so dass δ(z, a) einen Zustand z für jeden Zustand z und jedes Eingabezeichen a festlegt.

[12]Klenk 1980 S. 67
[13]Die Darstellung folgt im Wesentlichen Hopcraft 1996 und Klenk 1980.

Wenn wir eine Definition so wörtlich nehmen, wie sie dies verlangt, haben wir bisher nur den Übergang von jeweils einem Zustand in einen anderen beschrieben, nachdem die Eingabe eines einzelnen Symbols vorausgegangen ist. Nun wollen wir aber auch festlegen, wie sich der Automat verhält, wenn er nicht mehr nur mit einem einzelnen Symbol , sondern mit einer längeren Zeichenkette, die mehr als ein Symbol enthält, konfrontiert ist. Zu diesem Zweck wird die Übergangsfunktion δ mit Hilfe einer rekursiven Definition erweitert:

Zunächst wird ein Ausgangspunkt beschrieben. Was geschieht, wenn der Automat das leere Wort ϵ verarbeitet? Dann verbleibt er in dem Ausgangszustand. Das ist intuitiv leicht nachvollziehbar. Wenn kein Symbol eingegeben wird, dann verändert sich auch nichts.

Innerhalb der rekursiven Definition der Übergangsfunktion δ hat dieser Rekursionsanfang die folgende Gestalt:

4.3.1.2. $\delta(z, \epsilon) = z$.

Im zweiten Schritt dann kann gezeigt werden, wie sich die Übergangsfunktion verhält, wenn Zeichenketten beliebiger Länge zu verarbeiten sind. Dazu stellen wir uns vor, dass unser Automat zunächst hat eine Zeichenkette w und zusätzlich ein weiteres Symbol unseres Alphabets a zu verarbeiten hat. Das kann formal folgendermaßen dargestellt werden:

4.3.1.3. $\delta(z, wa) = \delta(\delta(z, w), a)$.

Schauen wir uns das einmal an einem Beispiel 4.3.1.1 an. Unser Automat möge die folgende Gestalt haben: Er bestehe aus den Zuständen $Z = \{z_0, z_1, z_2, z_3\}$, aus dem Eingabealphabet $A = \{0, 1\}$, aus dem Endzustand E = $\{z_0\}$ und der Übergangsfunktion δ, deren Verhalten in der folgenden Tabelle dargestellt sei:

Zustände	Eingaben	
	0	1
z_0	z_2	z_1
z_1	z_3	z_0
z_2	z_0	z_3
z_3	z_1	z_2

Tabelle 4.5: Zustandstafel zur Uebergangsfunktion im DEA aus Beispiel 4.3.1.1

Aus der Tabelle **??** erfahren wir, dass der Automat dann, wenn er sich im Zustand z_0 befindet und auf eine 0 als Eingabe trifft, in den Zustand z_2 übergeht. Trifft er im Zustand z_0 auf eine 1, dann geht er in z_1 über usw.

Analysieren wir auf dieser Grundlage das Verhalten des Automaten anlässlich der Eingabe der Zeichenkette w = 110101. Wir können das Wort zerlegen in w'=11010 und das Symbol a' = 1, diese wiederum in w" = 1101 und a" = 0 usw. So können wir unter Ausnutzung unserer Festlegungen in 4.3.1.2 Folgendes hinschreiben:

$\delta(z_0, w) = \ \delta\ (\delta\ (\delta\ (\delta\ (\delta\ (\delta\ (z_0, a''''''), a''''''), a''''), a'''), a'')a'')$

oder

$$\delta(z_0, 110101) = \delta\ (\delta\ (\delta\ (\delta\ (\delta\ (\delta\ (z_0, 1), 1), 0), 1), 0), 1), 1)$$

Wenn wir diese geschachtelte Ableitung in anderer Darstellung betrachten, dann sehen wir Folgendes:

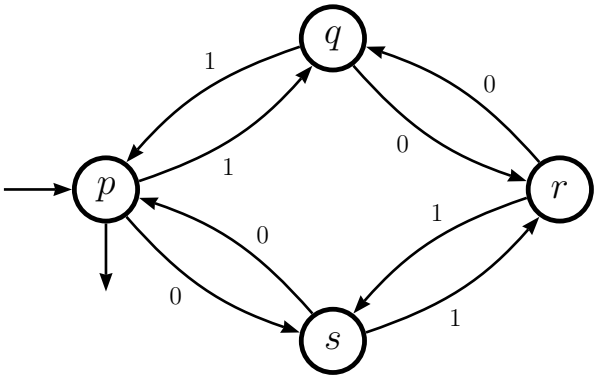
$$z_0^1\ z_1^1\ z_0^0\ z_2^1\ z_3^0\ z_1^1\ z_0.$$

Das heißt in ausführlicher Darstellung nichts anderes als Folgendes: In z_0 mit 1 erreichen wir z_1. Von dort aus mit 1 nach z_0 zurück. Wiederum von z_0 mit 0 nach z_2. Von z_2 mit 1 nach z_3. Von dort aus mit 0 erneut nach z_1, und zum Schluss dann von dort aus mit 1 nach z_0. Das zeigt, dass unsere Zeichenkette uns z_0, den einzigen zugelassenen Endzustand, erreichen lässt. Auf dem Weg dorthin lässt sich mit jeder Eingabe ein Zustand erreichen, von dem es einen Übergang in einen weiteren Zustand gibt, bis der Automat im Endzustand anhält. Wenn eine Zeichenkette einen solchen Ablauf ermöglicht, dann bedeutet das, dass die Zeichenkette zu denjenigen gehört, die von unserem Automaten erkannt werden. Wenn wir die Zeichenkette als Bestandteil einer Sprache ansehen, die wir mit L bezeichnen (wegen des L's als des Anfangsbuchstaben des lateinischen Wortes für Sprache 'lingua'), dann können wir sagen, dass w zu der Sprache L gehört, die von unserem Automaten $\mathscr{A}$ erkannt wird. Mit anderen Worten: w $\in$ L($\mathscr{A}$).

Wäre das letzte Zeichen der Zeichenkette w nicht die 1, sondern die 0, so dass der Automat eine Zeichenkette v = 110100 zu verarbeiten hätte, dann endete unser Automat mit der Konsequenz in z_2, einem Zustand, der nicht zur Menge der Endzustände gehört, dass die Zeichenkette v = 110100 nicht vom Automaten erkannt würde, demzufolge nicht Teil der Menge L($\mathscr{A}$) wäre.

Das folgende Zustandsdiagramm zeigt den Automaten $\mathscr{A}$ mit einigen wenigen Umbenennungen, die bedauerlicherweise notwendig sind[14].

[14] Leider ist es mit dem Paket Vaucanson-G, das benutzt wird, um die Automatendiagramme zu zeichnen, nicht möglich, tiefgestellte Ziffern in der Benennung der Zustände darzustellen. Deswegen müssen folgende Umbenennungen vorgenommen werden: z_0 = p, z_1 = q, z_2 = r und z_3 = s.. Der von links auf Zustand p treffende Pfeil markiert den Startzustand, der p nach unten verlassende Pfeil den Endzustand des Automaten.



Mit diesen Ausführungen sind die grundlegenden Fakten zu deterministischen endlichen Automaten festgehalten, so dass wir uns im nächsten Schritt den nicht-deterministischen endlichen Automaten zuwenden können.

4.3.2 Nicht-deterministische endliche Automaten (NEA)

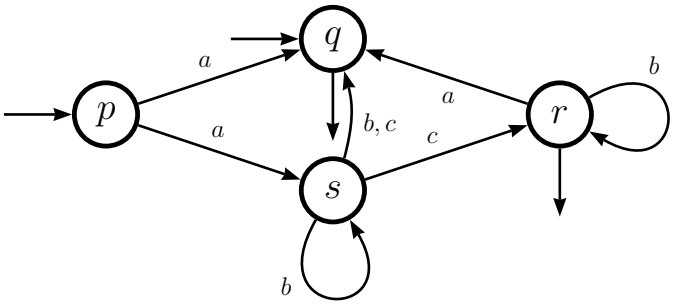
Da sich Nicht-deterministische und deterministische endliche Automaten nur in einigen, wenn auch wichtigen Details unterscheiden, lassen sich einige definierende Merkmale des zuerst genannten Automaten-Typs hier wörtlich wiederholen.

Definition 4.3.2.1 des **N**icht-deterministischen **e**ndlichen **A**utomaten (NEA)
Ein NEA ist ein Quintupel (Z, A, δ, X, E), für das Folgendes gilt: Z ist eine endliche Menge von Zuständen, A ein endliches Eingabealphabet, X $\subseteq$ Z eine Menge von Startzuständen, E $\subseteq$ Z eine Menge von Endzuständen und δ die Übergangsfunktion, die Z$\times$A auf 2^Z, die Potenzmenge von Z, abbildet, so dass δ(z, a) eine **Zustandsmenge** beschreibt, in die von Zustand z bei Eingabezeichen a übergegangen wird.

Ein Vergleich der Definitionen des DEA und des NEA zeigt, worin sich beide Automaten-Typen unterscheiden. Zum einen räumt der NEA die Möglichkeit ein, von mehreren Startzuständen auszugehen, zum Anderen lässt er Übergänge zu **Zustandsmengen** zu, während der DEA nur den Übergang zu einem einzelnen Zustand ermöglicht. Wenn wir diese Beoabchtung auswerten, kann gesagt werden, dass der NEA eine Verallgemeinerung des DEA ist. Der DEA ist ein NEA mit Ausgängen von einelementigen Startzuständen und Übergängen in einelementige Zustandsmengen. Bevor wir weiter gehende Analysen anstellen, betrachten wir zunächst erst einmal ein Beispiel 4.3.2.1.

Gegeben sei der NEA $\mathscr{A}$[15] mit (Z, A, δ, X, E), wobei Z = $\{p, q, r, s\}$, A = $\{a, b, c\}$, X = $\{p, q, \}$, E = $\{q, r\}$ und δ durch folgendes Zustandsdiagramm:

[15] Vgl. Klenk 1980 S. 72 f.



In einer Zustandstafel lassen sich die vom Automaten zugelassenen Übergänge so darstellen:

δ	a	b	c
p	{s, q}	$\oslash$	$\oslash$
q	$\oslash$	$\oslash$	$\oslash$
s	$\oslash$	{s, q}	{r, q}
r	{q}	{r}	$\oslash$

Tabelle 4.6: Zustandstafel zur Uebergangsfunktion im NEA aus Beispiel 4.3.2.1

Zustandsdiagramm und Zustandstafel machen deutlich, welche besonderen Eigenschaften den NEA gegenüber dem DEA auszeichnen. Es ist möglich, nach Eingabe ein und desselben Eingabesymbols in unterschiedliche Zustände wechseln zu können, was eben auch auschlaggebend für die Benennung dieses Typs von Automaten ist. Nicht-deterministisch heißt hier soviel wie nicht eindeutig festgelegt. Zu erklären bleibt, wie die Frage entschieden wird, ob ein Automat bei Angabe einer Zeichenkette einen Endzustand erreicht oder nicht. Offensichtlich gibt es für Zeichenketten, wenn sie denn ableitbar sind, in NEA'en unterschiedliche Ableitungspfade. Das geht aus dem Umstand hervor, dass es Übergänge gibt, die mehrere Zustände enthalten können. Wenn im Verlauf einer Ableitung diese eine solche mehrelementige Zustandsmenge erreicht, müssen mehrere Fortsetzungeen der Ableitungen geprüft werden. Wie ist dies technisch darstellbar? Ein gangbarer Weg, diese Ableitungspfade zu rekonstruieren, besteht darin, die Übergangsfunktion als eine Vereinigung von Abbildungen zu verstehen. Ursula Klenk beschreibt dies folgendermaßen:

„Sei $a_1 \ldots a_i$ für ein i, $1 \leq i \leq n$, bereits analysiert und M_i die Menge der Übergangszustände, die dann erreicht ist. M_i ist entweder leer, oder es gilt $M_i = \{z_{i,\,1}, \ldots, z_{i,\,k}\}$ mit $z_{i,\,1}, \ldots, z_{i,\,k} \in Z$. Falls M_i nicht leer, wird $\delta(z_{i,\,1}, a_{i+1}) \cup \ldots \cup \delta(z_{i,\,k}, a_{i+1})$ gebildet, die Menge aller Übergangszustände, die man ausgehend von den Zuständen aus M_i mit a_{i+1} erreicht. Wir bilden die Übergangszustandsmengen solange, bis M_n gebildet ist oder die Menge leer wird. Ist M_n ein Endzustand, so wird $a_1 \ldots a_n$ vom

NEA akzeptiert, in allen anderen Fällen nicht."[16]

Um es noch einmal weniger abstrakt, aber hoffentlich genauso präzise auszudrücken: Angenommen, die Verarbeitung einer Zeichenkette $a_1 \ldots a_7$ sei bis zu einem Eingabesymbol a_5 gelangt. Der Automat möge sich im Zustand z_4 befinden, so dass als nächstes a_5 zu verarbeiten sei. Nun sei weiterhin vorausgesetzt, dass nach der Verarbeitung von a_5 in z_4 zugleich zwei unterschiedliche Zustände $z_{5,1}$ und $z_{5,2}$ erreicht seien. Die Übergangszustandsmenge M_5 umfasst dann $M_5 = \{z_{5,1}, z_{5,2}\}$ wegen $M_5 = \{\delta(z_{4,1}, a_5) \cup \delta(z_{4,1}, a_5)\}$.

Eine Beispielanalyse möge die Untersuchung des NEA beschließen. Gegeben sei der oben in Beispiel 4.3.2.1 beschriebene Automat. Die Zeichenkette w = ac soll abgleitet werden. Terminiert der Automat, d.h. erreicht er in endlich vielen Schritten einen Endzustand?

Da der Automat über zwei Startzustände verfügt, muss das erste Zeichen von diesen beiden Zuständen verarbeitet werden:

$\delta(p, a) \cup \delta(q, a)\} = \{\text{s,q}\}$,
$\delta(s, c) \cup \delta(q, c)\} = \{\text{r,q}\}$.

Da sowohl r als auch q Endzustände des Automaten sind, ist die Zeichenkette abgeleitet. Die Zeichenkette wird über die Zustandsfolgen p, s, q und p, q, r akzeptiert.

4.4 Syntaxanalyse natuerlich-sprachlicher Saetze

Bisher haben wir von unseren Automaten nur Zeichenketten, die aus Ziffernfolgen 110101 oder Buchstabenfolgen abca bestehen, untersuchen lassen. Wie die Darstellungen im Abschnitt Grundbegriffe gezeigt haben, gelten solche Folgen in der Theorie der Formalen Sprachen als Wörter bzw. als Zeichenketten einer Sprache. Die Sprache L(G) wird definiert durch die Grammatik G, deren Struktur die Produktion einer bestimmten Ausdrucksmenge zulässt. Die zugelassene Ausdrucksmenge wird festgelegt durch die in der Grammatik vorkommenden Terminalsymbole, Nicht-Terminalsymbole und Regeln. Zu einer Grammatik können wir Automaten entwerfen. Die Automaten sollen die Fähigkeit besitzen, die Ausdrücke, die von der Grammatik G erzeugt worden sind, abzuleiten. Wenn ein Automat alle Zeichenketten, die mit G produziert werden können, abzuleiten fähig ist, dann dürfen wir sagen, dass der Automat $\mathscr{A}$ die Ausdrucksmenge L($\mathscr{A}$), die der Sprache L(G) äquivalent ist, erkennt.

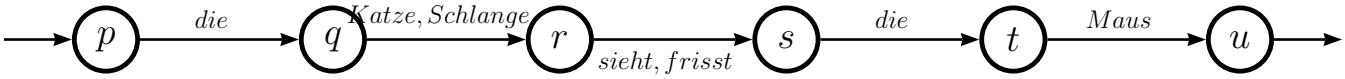
Sofern wir als Terminalsymbole nur Ziffern oder Buchstaben wählen, dann erhalten wir in der Regeln Zeichenketten, von denen wir sagen, dass sie für uns bedeutungslos sind. Es bleibt zu fragen, ob wir einen Automaten entwerfen können, von dem wir sagen dürfen, dass er eine Sprache, die für uns bedeutungsvolle Sätze enthält, abzuleiten in der Lage ist.

Schauen wir uns die folgenden Sätze in Beispiel 4.4.1 an:

[16] Klenk 1980 S. 73. Die Notation ist den hier verwendeten Symbolen angepasst worden.

(32) die Katze sieht die Maus
(33) die Schlange sieht die Maus
(34) die Katze frisst die Maus
(35) die Schlange frisst die Maus

Diese Satzmenge möge die Menge der Sätze sein, die mit einer Grammatik, die wir später entwerfen wollen, konstruiert werden kann. Konstruieren wir zunächst den NEA A, den wir benötigen, um Sätze dieser Struktur abzuleiten. Beginnen wir mit dem Alphabet A, das aus folgenden Elementen besteht: A = {Katze, Schlange, Maus, sieht, frisst, die}. Darüber hinaus benötigen wir eine Zustandsmenge, Anfangs- und Endzustände und eine Übergangsfunktion. Legen wir zuerst einmal letztere fest. Daraus wird sich dann alles Weitere ergeben. δ sei durch das folgende Zustandsdiagramm festgelegt:



Aus diesem Zustandsdiagramm lassen sich die übrigen Elemente des Automaten gewinnen. Offensichtlich verfügt der Automat über die Zustandsmenge Z = {p ,q, r, s, t, u}. Der Startzustand besteht aus der einelementigen Menge X = {p}. Der Endzustand aus der ebenfalls einelementigen Menge E = {u}. Die Übergangsfunktion δ lässt sich gemäß Zustandsdiagramm mit der folgenden Zustandstafel aus Tabelle ?? beschreiben:

δ	die	Katze	Maus	Schlange	frisst	sieht
p	{q}	⊘	⊘	⊘	⊘	⊘
q	⊘	{r}	⊘	{r}	⊘	⊘
r	⊘	⊘	⊘	⊘	{s}	{s}
s	{t}	⊘	⊘	⊘	⊘	⊘
t	⊘	⊘	{u}	⊘	⊘	⊘
u	⊘	⊘	⊘	⊘	⊘	⊘

Tabelle 4.7: Zustandstafel zu Beispiel 4.4.1

Wir sehen, dass mit Hilfe dieses Automaten alle Sätze der oben angegebenen Ausdrucksmenge (32) - (35) abgeleitet werden können. Versuchen wir jetzt einmal eine Grammatik zu finden, die dem Automaten äquivalent ist. D.h., dass wir eine Grammatik suchen, die fähig ist, alle die Sätze zu erzeugen, die unser Automat abzuleiten in der Lage ist. Wenn wir die Bezeichnungen der Zustände in Großschreibung dazu verwenden, die Nicht-Terminalsymbole der Grammatik zu benennen, dann können wir Grammatik und Automaten einander so ähnlich wie möglich beschreiben. Innerhalb der Grammatik gibt es genauso wie im Automaten ein Startsymbol. Wählen

wir für das Startsymbol innerhalb der Grammatik P. Dann lässt sich eine erste Regel formulieren. Unser Startsymbol P expandieren wir zu 'die Q'. Q seinerseits lässt sich zu 'Katze R' und 'Schlange R' entwickeln. R zu 'frisst S' oder 'sieht S'. S zu 'die T' und zum Abschluss T zu 'Maus U'.

In Tabellenform sieht das folgendermaßen aus:

(36)	P	→	die	Q
(37)	Q	→	Katze	R
(38)	Q	→	Schlange	R
(39)	R	→	frisst	S
(40)	R	→	sieht	S
(41)	S	→	die	T
(42)	T	→	Maus	

Tabelle 4.8: Regeln der Grammatik zu Beispiel 4.4.1

Analysieren wir die Grammatik-Regeln, so zeigt sich, dass sie die folgende Form haben: Auf der linken Regelseite finden wir jeweils ein Nicht-Terminalsymbol gefolgt vom Expansionspfeil, einem Terminalsymbol und einem Nicht-Terminalsymbol auf der rechten Seite der Regel. Wir entwickeln eine Zeichenkette, indem wir das am weitesten rechts stehende Nicht-Terminalsymbol in einer Folgeregel auf der linken Regelseite antreffen, so dass auch dieses nach demselben Muster expandiert werden kann. Eine Grammatik solchen Zuschnitts trägt in der Theorie der Formalen Sprachen die Bezeichnung 'rechts-linear'. Diese Grammatik gehört zu den Typ-3-Grammatiken, von denen oben die Rede war. Es lässt sich beweisen, dass die äquivalenz der Übergangsfunktionen und der Typ-3-Grammatik-Regeln nicht auf Zufall gegründet ist, sondern ein Wesensmerkmal endlicher Automaten und linearer Grammatiken ist (Vgl. zu den Beweisen Anhang A.2 Formale Sprachen). Aber auch ohne die entsprechenden Beweise zeigen die Parallelen zwischen Grammatik-Regeln und Übergangsfunktionen in Tabelle **??** deutlich, dass endliche Automaten und lineare Grammatiken einander äquivalent sind.

Grammatik-Regeln	Übergangsfunktionen
(43) P → die Q	$q \in \delta(p, die)$
(44) Q → Katze R	$r \in \delta(q, Katze)$
(45) Q → Schlange R	$r \in \delta(q, Schlange)$
(46) R → frisst S	$s \in \delta(r, frisst)$
(47) R → sieht S	$s \in \delta(r, sieht)$
(48) S → die T	$t \in \delta(s, die)$
(49) T → Maus U	$u \in \delta(t, Maus)$

Tabelle 4.9: Parallelen zwischen Grammatik-Regeln und Uebergangsfunktionen

Damit ist dargestellt, wie sich zu einer Menge natürlich-sprachlicher Sätze eine Grammatik, die die Sätze erzeugt, und ein Automat, der die Sätze erkennt, herstellen lassen. Wenn wir die von uns in diesem Zusammenhang entwickelte lineare Grammatik mit derjenigen vergleichen, die wir oben im Grammatik-Abschnitt entworfen

haben, dann sehen wir, dass die hier verwendete Grammatik erheblich einfacher ist als die früher benutzte. Das hat seinen Grund. Die oben benutzte komplexere Grammatik gehört zur Gruppe der Typ-2-Grammatiken, der sogenannten kontextfreien Grammatiken, die wir oben kurz charakterisiert haben. Um Satzmengen, die von solchen Grammatiken hervorgebracht werden, parsen zu können, kommen wir nicht mit endlichen Automaten aus. Um Sätze dieser Struktur maschinell verarbeiten zu können, werden sogenannte Kellerautomaten benötigt. Diese haben ihren Namen deswegen erhalten, weil sie zusätzlich zu den Elementen, die auch aus endlichen Automaten bekannt sind, einen Stapel, einen Zwischenspeicher, besitzen, auf dem Zwischenergebnisse abgelegt werden können.

Literaturverzeichnis

[Erlhofer '07] Erlhofer, Sebastian, *Suchmaschinen-Optimierung*, Bonn, 2007.

[Hopcraft 1996] Hopcraft, John E., Ullman, Jeffrey D., *Einfuehrung in die Automatentheorie, Formale Sprachen und Komplexitaetstheorie*, Bonn, Paris, Mass. [u.a.], 1996.

[Klenk 1980] Klenk, Ursula,*Formale Sprachen*, Tuebingen 1980.

[Klenk 2003] Klenk, Ursula, *Generative Syntax*, Tuebingen 2003.

[Lobin 2004] Lobin, Henning, Lothar Lomnitzer, *Texttechnologie*, Tuebingen 2004.

[Selfhml] Muenz, Stefan, *Selfhtml*, http://de.selfhtml.org, Version 8.12 vom 1.3.2007.

[Sag 2003] Sag, Ivan A., Thomas Wasow, Emily Bender, *Syntactic Theory*, CSLI Publications, Stanford.

[Sob 2002] Sobeck, Markus, *Überblick über das PageRank-Verfahren der Suchmaschine Google*, http://pr.efactory.de/d-index.shtml, 2002/03.

[Duden 1990] Schüler Duden, *Die Mathematik I*, Mannheim, Zürich, Wien 1990.

[TEI] Text Encoding Initiative, <*Text Encoding Initiative*>, http://www.tei-c.org.

A Anhang

A.1 Pagerank

Markus Sobeck (*http://pr.efactory.de/d-index.shtml*)

Überblick über das PageRank-Verfahren der Suchmaschine Google

Im Verlauf der letzten Jahre hat sich Google weltweit zur bedeutendsten Suchmaschine entwickelt. Maßgebend verantworlich hierfür war neben einer hohen Performance und einer großen Benutzerfreundlichkeit vor allem die anderen Suchmaschinen teilweise weit überlegene Qualität der Suchergebnisse. Diese Qualität der Suchergebnisse beruht ganz wesentlich auf dem PageRank-Verfahren.

An dieser Stelle soll ein möglichst breiter Überblick über alle Aspekte des PageRank-Verfahrens wiedergegeben werden. Unser Überblick stützt sich dabei im Kern auf Veröffentlichungen der Google-Gründer Lawrence Page und Sergey Brin aus ihrer Zeit als Graduiertenstudenten an der Stanford University.

Vielerorts wird angeführt, dass seit den Forschungsarbeiten am PageRank - Verfahren vor allem angesichts der Dynamik des Internets zu viel Zeit vergangen ist, als dass die veröffentlichten Dokumente immer noch für die Bewertungsmethodik der Suchmaschine Google maßgebend sind. Es soll auch nicht bezweifelt werden, dass im Verlauf der letzten Jahre mit großer Wahrscheinlichkeit zahlreiche Änderungen, Anpassungen und Modifikationen am ursprünglichen PageRank-Algorithmus stattgefunden haben. Allerdings war gerade das PageRank-Verfahren ein wichtiger Faktor für den Erfolg der Suchmaschine Google, womit zumindest das Konzept des PageRank-Verfahrens immer noch grundlegend sein sollte.

Das PageRank-Konzept

Im Zuge der Entwicklung des World Wide Webs wurden verschiedene Verfahren zur Bewertung von Webseiten mit dem Ziel der Relevanzbeurteilung durch Suchmaschinen entwickelt. Ein aus unmittelbar einleuchtenden Gründen auch heute immer noch von praktisch allen Suchmaschinen genutzter Maßstab ist das Vorkommen eines Suchbegriffs in den Inhalten einer Webseite. Dieses Vorkommen wird nach den verschiedensten Kriterien wie etwa der relativen Häufigkeit des Vorkommens (der sog. Keyword-Dichte), den Stellen des Vorkommens des Suchbegriffs oder auch der Exponiertheit des Suchbegriffs im Dokument gewichtet.

Aus der Absicht, Suchmaschinen resistent gegen Webseiten zu machen, die auf der Basis von Analysen der inhaltsspezifischen Bewertungskriterien generiert wurden (Doorway Pages), entstand das Konzept der Link-Popularität. Dabei fließt die Anzahl der eingehenden Links für ein Dokument als ein grundsätzliches Kriterium

für die Bedeutung einer Webseite in die Relevanzbeurteilung ein. Diesem Ansatz liegt zu Grunde, dass ein Dokument um so wichtiger ist, je häufiger es von anderen verlinkt wird. Hierdurch wird weitestgehend verhindert, dass automatisch generierte "suchmaschinenoptimierte" Webseiten ohne jeglich Einbindung in das WWW oben in den Suchmaschinenergebnissen erscheinen. Es zeigte sich allerdings, dass auch das Konzept der Link-Popularität schnell von Webmastern antizipiert werden konnte, indem sie von ebenso unbedeutenden, automatisch generierten Seiten eingehende Links für Doorway Pages schufen.

Im Gegensatz zum Konzept der Link-Popularität nutzt das PageRank-Konzept nicht einfach die absolute Anzahl eingehender Links für die Beurteilung der Bedeutung einer Webseite. Die Argumentation der Google-Gründer gegen das Konzept der einfachen Link-Popularität war, dass ein Dokument zwar bedeutsam ist, wenn es von vielen anderen verlinkt wird, nicht jedes verlinkende Dokument ist jedoch gleichwertig. Vielmehr sollte einem Dokument - völlig unabhängig von seinen Inhalten - ein hoher Rang zugewiesen werden, wenn es von anderen bedeutenden Dokumenten verlinkt wird.

Die Bedeutsamkeit eines Dokuments bestimmt sich im Rahmen des PageRank-Konzepts also aus der Bedeutsamkeit der darauf verlinkenden Dokumente. Deren Rang wiederum bestimmt sich ebenfalls aus dem Rang verlinkender Dokumente. Die Bedeutsamkeit eines Dokuments definiert sich stets rekursiv aus der Bedeutsamkeit anderer Dokumente. Da - wenn auch über viele hintereinanderfolgende Links hinweg - der Rang eines jeden Dokuments eine Auswirkung auf den Rang eines jeden anderen hat, beruht das PageRank-Konzept letztlich auf der Linkstruktur des gesamten Webs. Obwohl diese ganzheitliche Betrachtung des WWW es nicht vermuten lässt, gelang es Page und Brin das PageRank-Konzept mittels eines relativ trivialen Algorithmus umzusetzen.

Der PageRank-Algorithmus

Der ursprüngliche PageRank-Algorithmus wurde von Lawrence Page und Sergey Brin mehrfach beschrieben. Er hat die folgende Form:

$$PR(A) = (1-d) + d(PR(T1)/C(T1) + ... + PR(Tn)/C(Tn))$$

Hierbei ist: PR(A) der PageRank einer Seite A, PR(Ti) der PageRank der Seiten Ti, von denen ein Link auf die Seite A zeigt, C(Ti) die Gesamtanzahl der Links auf Seite Ti und d ein Dämpfungsfaktor (Damping Factor), wobei $0 <= d <= 1$ ist.

Das PageRank-Verfahren bewertet damit grundsätzlich nicht Websites in ihrer Gesamtheit, sondern basiert ausschließlich auf der Beziehung einzelner Webseiten zueinander. Der PageRank einer Seite A bestimmt sich dabei rekursiv aus dem PageRank derjenigen Seiten, von denen ein Link auf die Seite A zeigt.

Der PageRank der Seiten Ti, die auf eine Seite A verlinken, fließt nicht gleichmäßig in den PageRank von Seite A ein. Der PageRank einer Seiten T wird stets anhand der Anzahl C(T) der von Seite T ausgehenden Links gewichtet. Das bedeutet, dass je mehr ausgehende Links eine Seite T hat, umso weniger PageRank gibt sie an Seite A weiter.

Der anhand der Anzahl an ausgehenden Links gewichtete PageRank der Seiten Ti wird nun addiert. Dies hat zur Folge, dass jeder zusätzliche eingehende Link für eine Seite A stets den PageRank dieser Seite A erhöht.

Schließlich wird die Summe der gewichteten PageRanks der Seiten Ti mit dem Dämpfungsfaktor d, der stets zwischen 0 und 1 liegt multipliziert. Hierdurch wird das Ausmaß der Weitergabe des PageRanks von einer Seite auf einer andere verringert.

Das Random Surfer Modell

Lawrence Page und Sergey Brin bieten in ihren Veröffentlichungen eine sehr einfache, intuitive Rechtfertigung des PageRank-Algorithmus an. Sie betrachten PageRank-Verfahren als ein Modell zur Abbildung von Benutzer-Verhalten. Hierzu führen sie einen Zufalls-Surfer an, der von einer Webseite zur nächsten jeweils beliebige Links verfolgt, ohne dabei auf Inhalte zu achten.

Der Zufalls-Surfer befindet sich mit einer bestimmten Wahrscheinlichkeit auf einer Website, die sich aus deren PageRank herleiten lässt. Die Wahrscheinlichkeit, dass der Zufalls-Surfer nun einen bestimmten Link verfolgt, ergibt sich dann einzig und allein daraus, aus wievielen Links er die Auswahl hat. Aus diesem Grunde fließt der PageRank einer verlinkenden Seite stets nach der Anzahl Ihrer ausgehenden Links gewichtet in die PageRank Berechnung einer verlinkten Seite ein.

Die Wahrscheinlichkeit, dass der Zufalls-Surfer auf eine Seite gelangt, ist also die Summe der Wahrscheinlichkeiten, mit der er von einer verlinkenden Seite den entsprechenden Link verfolgt. Nun wird allerdings die Wahrscheinlichkeit, mit der der Zufalls-Surfer auf eine Seite gelangt, um den Faktor d gedämpft. Dies hat im Rahmen des Random Surfer Modells den Hintergrund, dass der Zufalls-Surfer nicht unendlich viele Links verfolgt. Nach einer bestimmten Zeit wird er gelangweilt und ruft eine beliebige andere Webseite auf.

Die Wahrscheinlichkeit, mit der der Zufalls-Surfer die Verfolgung von Links nicht abbricht und somit weiterklickt, wird durch den Dämpfungsfaktor d angegeben, der abhängig von der Höhe der Wahrscheinlichkeit einen Wert von 0 bis 1 annimmt. Je höher d ist, um so wahrscheinlicher ist es, dass der Zufalls-Surfer Links verfolgt. Da der Zufalls-Surfer nach dem Abbruch der Link-Verfolgung eine beliebige Seite aufruft, geht die Wahrscheinlichkeit mit er er dies tut, mit dem Wert (1-d) als Konstante in die Berechnung des PageRanks einer jeden Seite ein.

Abweichende Formulierung des PageRank-Algorithmus

Lawrence Page und Sergey Brin bieten in ihren Veröffentlichungen zwei unterschiedliche Versionen des PageRank-Algorithmus an. In dieser zweiten Version bestimmt sich der PageRank einer Seite A wie folgt:

$$PR(A) = (1-d)/N + d(PR(T1)/C(T1) + ... + PR(Tn)/C(Tn))$$

Hierbei ist N die Anzahl aller Seiten des Webs. Diese zweite Version des PageRank-Algorithmus unterscheidet sich allerdings nicht grundlegend von der ersten. In der

zweiten Version beschreibt der PageRank einer Seite im Sinne des Random Surfer Modells lediglich die tatsächliche Wahrscheinlichkeit, mit der der Zufalls-Surfer nach dem Verfolgen vieler Links eine Seite erreichen wird. Dieser Algorithmus bildet damit eine Wahrscheinlichkeitsverteilung über alle Seiten des Webs ab. Die Summe aller PageRank-Werte des Webs ist damit bei dieser Version des Algorithmus gleich 1.

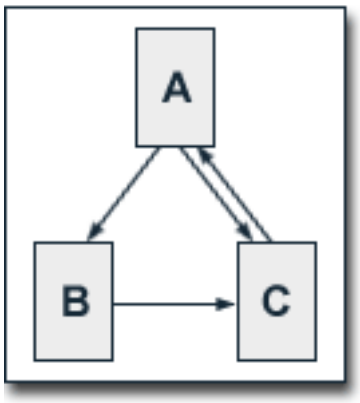
In der oben genannten, ersten Version erfolgt eine Gewichtung der Wahrscheinlichkeit des Besuchs einer Seite nach der Anzahl der Seiten des Webs. Demnach ist der PageRank in dieser Version im Grunde der Erwartungswert für den Besuch des Zufalls-Surfers auf einer Seite, wenn er hierfür Anläufe in genau der Höhe der Anzahl der Seiten des Webs nimmt. Bestände das Web also aus 100 Seiten, und eine Seite hat einen PageRank von 2, so würde der Zufalls-Surfer sie bei 100 SSurfgängenïm Mittel zweimal erreichen.

Wie bereits erwähnt, unterscheiden sich die beiden Versionen des Algorithmus sich nicht grundlegend. Letztlich muss der PageRank einer Seite aus der Algorithmus-Version 2 lediglich mit der Anzahl der Webseiten multipliziert werden, um zum PageRank der Algorithmus-Version 1 zu gelangen. Selbst Page und Brin ist in Ihrer wohl bekanntesten Veröffentlichung "The Anatomy of a Large-Scale Hypertextual Web Search Engine"der Fehler unterlaufen, die erste Version des PageRank-Algorithmus als Wahrscheinlichkeitsverteilung zu charakterisieren, bei der die Summe der PageRank-Werte aller Seiten gleich eins sei.

Im Folgenden wird für die weiteren Betrachtungen der oben zuerst genannte Algorithmus verwandt. Dies hat den einfachen Hintergrund, dass Berechnungen hiermit wesentlich einfacher sind, da die Größe des Webs vollkommen auñer Acht gelassen werden kann.

Die Eigenschaften des PageRank

Die Eigenschaften des PageRank sollen jetzt anhand eines Beispieles veranschaulicht werden.



Hierzu wird ein kleines 3-Seiten-Web aus den Seiten A, B und C betrachtet, wobei Seite A sowohl auf Seite B als auch auf Seite C verlinkt. Seite B verlinkt lediglich auf Seite C und Seite C wiederum verlinkt auf Seite A. Der Dämfungsfaktor d wird Angaben von Lawrence Page und Sergey Brin zufolge für tatsächliche Berechnungen üblicherweise auf 0.85 gesetzt. Der Einfachheit halber wird d an dieser Stelle

ein Wert von 0.5 zugewiesen, wobei die Höhe von d zwar Auswirkungen auf den PageRank hat, das hier zu erläuternde Prinzip jedoch nicht beeinflusst. Es ergeben sich die folgenden Gleichungen für den PageRank der einzelnen Seiten:

$PR(A) = 0.5 + 0.5PR(C)$
$PR(B) = 0.5 + 0.5(PR(A)/2)$
$PR(C) = 0.5 + 0.5(PR(A)/2 + PR(B))$

Dieses Gleichungssystem lässt sich sehr einfach für den PageRank der einzelnen Seiten lösen. Es ergeben sich die folgenden Werte:

$PR(A) = 14/13 = 1.07692308$
$PR(B) = 10/13 = 0.76923077$
$PR(C) = 15/13 = 1.15384615$

Es zeigt sich, dass die Summe der PageRanks aller Seiten gleich drei und somit gleich der Anzahl der Seiten ist. Dies ist keine spezifisches Ergebnis für unser Beispiel, da der PageRank Algorithmus einen Erwartungswert für den Besuch von Seiten bei Anläufen in Höhe der Anzahl der Seiten darstellt.

Für ein kleines 3-Seiten-Beispiel lässt sich ein Gleichungssystem unproblematisch lösen. Das tatsächliche WWW besteht jedoch mittlerweile aus mehreren Milliarden Webseiten, so dass die Lösung eines entsprechenden Gleichungssystems nicht mehr möglich ist.

Die iterative Berechnung des PageRank

Aufgrund der Größe des Webs erfolgt in der Praxis der Suchmaschine Google eine näherungsweise, iterative Berechnung des PageRank. Dies bedeutet, dass zunächst jeder Seite ein PageRank zugewiesen wird, und anschließend der PageRank aller Seiten in mehreren Berechnungsrunden ermittelt wird. Diese näherungsweise Berechung soll wiederum anhand unseres kleinen Beispiels demonstriert werden, wobei als Ausganswert für den PageRank einer jeden Seite zunächst 1 angenommen wird.

Es zeigt sich, dass sich in unserem Beispiel bereits nach sehr wenigen Iterationen eine sehr gute Näherung an die tatsächlichen Werte ergibt. Für die Berechnung des PageRanks für das komplette WWW werden von Lawrence Page und Sergey Brin ca. 100 Iterationen als hinreichend genannt.

Entscheidend ist, dass die Summe der PageRanks aller Seiten nach der Durchführung der iterativen Berechnung gegen die Anzahl aller Seiten konvergiert. Der durchschnittliche PageRank aller Seiten geht mithin gegen 1. Jede Seite hat einen minimalen PageRank von $(1 - d)$. Der theoretisch maximale PageRank einer Seite beträgt $dN + (1 - d)$, wobei N die Anzahl aller Webseiten ist. Dieser theoretische Wert käme zustande, wenn sämtliche Webseiten ausschließlich auf eine Seite verlinken, und auch diese wiederum ausschließlich auf sich selbst verlinkt.

Iteration	PR(A)	PR(B)	PR(C)
0	1	1	1
1	1	0.75	1.125
2	1.0625	0.765625	1.1484375
3	1.07421875	0.76855469	1.15283203
4	1.07641602	0.76910400	1.15365601
5	1.07682800	0.76920700	1.15381050
6	1.07690525	0.76922631	1.15383947
7	1.07691973	0.76922993	1.15384490
8	1.07692245	0.76923061	1.15384592
9	1.07692296	0.76923074	1.15384611
10	1.07692305	0.76923076	1.15384615
11	1.07692307	0.76923077	1.15384615
12	1.07692308	0.76923077	1.15384615

Die Implementierung des PageRank in die Suchmaschine Google

Für die Implementierung des PageRank ist von zentraler Bedeutung, auf welche Art und Weise der PageRank in die generelle Bewertung von Webseiten durch die Suchmaschine Google einfließt. Das Verfahren wurde von Lawrence Page und Sergey Brin mehrfach in ihren Veröffentlichungen beschrieben. Ursprünglich basierte die Seitenbewertung durch Google auf drei Faktoren:

- Seitenspezifische Faktoren
- Ankertext eingehender Links
- PageRank

Zu den seitenspezifischen Faktoren zählen neben den konkreten Textinhalten etwa auch der Inhalt des Title-Tags und die URL einer Seite. Es ist mehr als wahrscheinlich, dass seit der Veröffentlichung dieser Punkte weitere Faktoren hinzugekommen sind. Dies soll an dieser Stelle jedoch nicht interessieren.

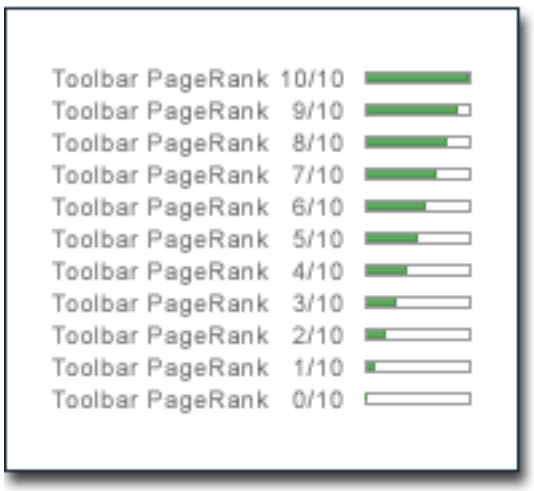
Bei Suchanfragen wird aus den seitenspezifischen Faktoren und den Ankertexten eingehender Links für den Suchbegriff eine nach Position und Grad der Hervorhebung gewichteter IR-Wert berechnet. Die Bewertung für die Relevanz einer Webseite für eine konkrete Suchanfrage wird nun mit dem PageRank als Indikator für die ganz allgemeine Bedeutsamkeit der Webseite kombiniert. Dieses Kombinieren erfolgt in multiplikativer Form. Dass hier kein additives Verfahren eingesetzt wird ist unmittelbar einleuchtend, da ansonsten Seiten mit einem sehr hohen PageRank auch auf Suchanfragen hin gefunden werden könnten, obwohl sie in keinerlei Zusammenhang zum gesuchten Begriff stehen.

Insbesondere bei aus mehreren Begriffen bestehenden Suchanfragen zeigt sich ein deutlich größerer Einfluss der inhaltsspezifischen Bewertungskomponenten. Der Einfluss des PageRank hingegen wird eher bei unspezifischen, aus lediglich einem Suchbegriff bestehenden Anfragen deutlich. Gerade für Mehr-Begriffs-Anfragen ist es möglich, mit den klassischen Mitteln der Suchmaschinen-Optimierung Listungen vor Seiten zu erlangen, die einen weitaus höheren PageRank-Wert inne haben.

Bei der Optimierung für Suchbegriffe, für die in den Suchmaschinen ein großer Wettbewerb herrscht, ist ein hoher PageRank-Wert unerlässlich für eine hohe Suchmaschinen - Position, selbst wenn die Seite selbst den klassischen Kriterien der Suchmaschinen - Optimierung folgt. Dies liegt darin begründet, dass die Wertung des zusätzlichen Vorkommens eines Suchbegriffs innerhalb eines Dokuments sowie in den Ankertexten von eingehenden Links mit der Häufigkeit des Vorkommens abnimmt, um Spam durch oftmalige Keyword-Wiederholungen zu vermeiden. Damit sind die Möglichkeiten zur Seitenoptimierung im klassischen Sinne beschränkt, und bei hohem Wettbewerb in Suchmaschinen für einen Suchbegriff wird der PageRank zum ausschlaggenden Faktor.

Die PageRank Anzeige der Google Toolbar

Einen großen Bekanntheitsgrad erlangte der PageRank durch seine Anzeige in der Google Toolbar. Die Google Toolbar ist ein Browser-Plug-In für den Microsoft Internet Explorer, das von der Google Website herunter geladen werden kann und zahlreiche Erleichterungen für die Google-Suche bereithält.



Die Google Toolbar zeigt den PageRank einer Seite auf einer Skala von 0 bis 10 an. Zunächst ist der PageRank an der Breite des grünen Balkens in der Anzeige ersichtlich. Führt der Benutzer mit der Maus über die Anzeige, gibt die Toolbar darüberhinaus den Wert des Toolbar-PageRank an. Vorsicht: Die PageRank-Anzeige zählt zu den „Advanced Features“ der Google Toolbar. Sobald diese „Advanced Features“ aktiviert sind, sammelt Google über die Toolbar Daten über das Benutzerverhalten. Außerdem führt die Toolbar selbstständig Updates durch, ohne dass der Benutzer über das Herunterladen der neuen Version informiert wird. Dies bedeutet letztlich, dass Google Zugriff auf die Festplatte des Benutzers hat.

Der tatsächliche PageRank, der für eine Seite theoretisch maximal einen Wert von $dN + (1 - d)$ annehmen kann, wobei N die Anzahl aller Seiten des Webs ist und d üblicherweise auf 0.85 gesetzt wird, muss für die Anzeige in der Google Toolbar skaliert werden. Es wird im Allgemeinen davon ausgegangen, dass die Skalierung nicht linear sondern logarithmisch erfolgt. Bei einem Dämpfungsfaktor von 0.85 und einem damit verbundenen minimalen PageRank von 0.15 sowie einer angenommenen

logarithmischen Basis von 6 ergäbe sich das folgende Bild für die Skalierung:

Toolbar-PR	Tatsächlicher PR
0/10	0.15 - 0.9
1/10	0.9 - 5.4
2/10	5.4 - 32.4
3/10	32.4 - 194.4
4/10	194.4 - 1,166.4
5/10	1,166.4 - 6,998.4
6/10	6,998.4 - 41,990.4
7/10	41,990.4 - 251,942.4
8/10	251,942.4 - 1,511,654.4
9/10	1,511,654.4 - 9,069,926.4
10/10	9,069,926.4 - 0.85 $\times N + 0.15$

Ob tatsächlich eine mathematisch strikte logarithmische Skalierung erfolgt ist natürlich ungewiss. Wahrscheinlich erfolgt eine manuelle Skalierung, die einem logarithmischen Schema folgt, damit Google die volle Kontrolle darüber behält, wie viele Seiten einen bestimmten Toolbar-PageRank inne haben. Diesem Schema dürfte allerdings eine logarithmische Basis von 6 bis 7 zu Grunde liegen, was sich etwa ansatzweise aus der Anzahl der von Google angezeigten eingehenden Links mit einem Toolbar-PageRank größer 4 für Seiten mit einem sehr hohen Toolbar-Pagerank herleiten lässt.

Die Datenkommunikation der Toolbar

Auch Webmaster, die aufgrund von Sicherheitsbedenken die Google Toolbar oder auch den Internet Explorer nicht dauerhaft nutzen möchten, haben eine Möglichkeit zum Einblick in die PageRank-Werte ihrer Seiten. Google übermittelt den PageRank in einfachen Textdateien an die Toolbar. Früher geschah dies per XML. Der Wechsel zu Textdateien fand im August 2002 statt.

Die PageRank-Textdateien können direkt von der Domain www.google.com abgerufen werden. In ihrer Grundform sehen die URLs der Dateien folgendermaßen aus (ohne Zeilenumbrüche):

http://www.google.com/search?client=navclient-auto
ch=0123456789features=Rankq=info:http://www.domain.com/

Die PageRank-Dateien bestehen aus einer Zeile. Der PageRank-Wert ist die letzte Ziffer in dieser Zeile.

Die oben in der URL dargestellten Parameter sind unerlässlich für die Anzeige der PageRank-Dateien im Browser. So identifiziert sich mit dem Wert „navclient-auto“ für den Parameter „client“ die Toolbar; mit dem Parameter „q“ wird die abgefragte URL übermittelt. Der Wert „Rank“ für den Parameter „features“ legt fest, dass die

PageRank-Dateien abgerufen werden. Wird dieser Parameter weggelassen, werden auch weiterhin XML-Dateien übermittelt. Der Parameter „ch“ wiederum übergibt eine Prüfsumme für die URL, wobei sich diese Prüfsumme im Zeitablauf für einzelne URLs lediglich bei Updates der Toolbar ändern kann.

Um die Prüfsummen einzelner URLs herauszufinden ist es damit erforderlich, die Toolbar zumindest einmal zu installieren. Hierbei wird dann vielerorts der Einsatz von Packet Sniffern, lokalen Proxies und ähnlichem empfohlen, um die Kommunikation zwischen Toolbar und Google aufzuzeichnen. Dies ist allerdings nicht zwingend erforderlich, da die PageRank-Dateien vom Internet Explorer gecached werden und somit die Prüfsummen im Ordner Temporary Internet Files eingesehen werden können. Die PageRank-Dateien können hiermit dann auch z.B. in anderen Browsern als dem Internet Explorer angezeigt werden, ohne dass Googles 36-Jahres-Cookies akzeptiert werden müssen.

Da die PageRank-Dateien im Browser-Cache gespeichert werden und somit offen einsehbar sind, und sofern eine Abfrage nicht automatisiert erfolgt, sollte dies keine Verletzung von Googles Dienstleistungsbedingungen darstellen. Es ist allerdings Vorsicht geboten. Die Toolbar übermittelt einen eigenen User-Agent an Google. Es ist:

Mozilla/4.0 (compatible; GoogleToolbar 1.1.60-deleon; OS SE 4.10)

Hierbei ist 1.1.60-deleon eine Toolbar-Version, die sich natürlich ändern kann, und OS das Betriebssystem des jeweils eingesetzten Rechners. Google kann also nachprüfen, ob eine direkte Anfrage über den Browser erfolgt, sofern kein Proxy zwischengeschaltet und der User-Agent entsprechend modifiziert wird.

Beim Blick in den Cache des IE wird man in der Regel feststellen, dass die PageRank-Dateien nicht von der Domain www.google.com, sondern von IPs wie z.B. 216.239.33.102 abgerufen werden. Ebenso enthalten die URLs häufig einen weiteren Parameter „failedip“ mit Werten wie z.B. „216.239.35.102; 1111“. Die IPs sind jeweils einem der derzeit sieben sich im Einsatz befindlichen Rechenzentren Googles zugeordnet. Wozu der Parameter „failedip“ tatsächlich genutzt wird, ist unklar. Hintergrund der unmittelbaren Abfrage der PageRank-Dateien bei einzelnen IPs ist wohl der Versuch, die PageRank-Anzeige insbesondere in den Zeiten des „Google Dance“ besser zu steuern.

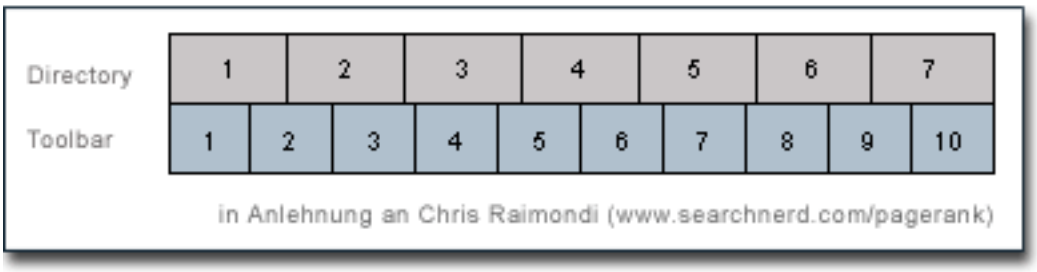
Die PageRank Anzeige der Google Directory

Denjenigen, denen der Abruf der PageRank-Dateien zu kompliziert ist, bleibt schließlich mit der Google Directory (directory.google.com) noch eine eingeschränkte Möglichkeit, etwas über den PageRank ihrer Site zu erfahren.

Bei der Google Directory handelt es sich um einen Dump des Open Directory Projects (dmoz.org), der neben den Seiteneinträgen ähnlich der Google Toolbar den skalierten PageRank für die in das ODP eingetragene Seite in Balkenform anzeigt. Allerdings erfolgt die PageRank-Anzeige in der Google-Directory auf einer Skala von 1 bis 7. Der exakte Wert wird nicht angezeigt, kann aber über die zweigeteilte Balkengrafik bzw. die Breite von deren Einzelgrafiken bestimmt werden, falls der

Betrachter sich beim einfachen Augenschein unsicher ist.

Durch den Vergleich des Toolbar-PageRanks mit dem Directory-PageRank kann vom tatsächlichem PageRank von Seiten, die in das ODP eingetragen sind, ein etwas genauerer Eindruck gewonnen werden. Dieser Zusammenhang wurde zuerst von Chris Raimondi (www.searchnerd.com/pagerank) aufgezeigt.



Insbesondere für Seiten mit einem Toolbar-PageRank von 5 oder 6 ergibt sich hier die Möglichkeit der Einschätzung, ob sich die Seite eher am unteren oder am oberen Ende eines Bereichs der Toolbar-Skalierung befindet. Es sei an dieser Stelle angemerkt, dass für die Darstellung des Vergleichs der beiden PageRank-Anzeigen der Toolbar-PageRank von 0 nicht berücksichtigt wurde. Dass dies sinnvoll ist, kann anhand von Seiten mit einem Directory-PageRank von 3 nachvollzogen werden. Hier ist allerdings zu berücksichtigen, dass zur Überprüfung eine Seite der Google Directory mit einem Toolbar-PageRank von maximal 4 ausgewählt werden sollte, da sich sonst in der Regel keine von dort verlinkten Seiten mit einem Toolbar-PageRank von 3 finden lassen.

Der Effekt eingehender Links

Es wurde bereits gezeigt, dass ein jeder eingehender Link auf ein Webseite deren Pagerank stets erhöht. Bei oberflächlicher Betrachtung des ursprünglichen PageRank-Algorithmus

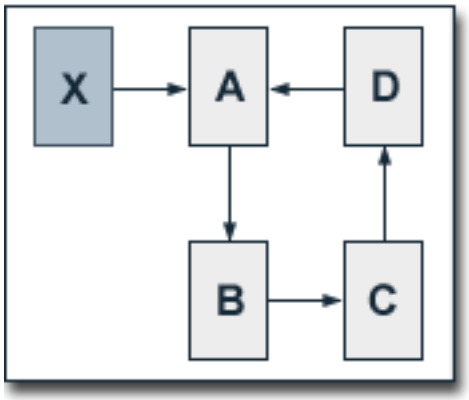
$$PR(A) = (1-d) + d(PR(T1)/C(T1) + ... + PR(Tn)/C(Tn))$$

möchte man meinen, ein zusätzlicher eingehender Link erhöht den PageRank der verlinkten Seite um

$d \times PR(X)/C(X)$

wobei PR(X) der PageRank der verlinkenden Seite X und C(X) deren Anzahl ausgehender Links ist. Bei genauerer Betrachtung zeigt sich allerdings, dass eine Webseite, die einen zusätzlichen eingehenden Link erhält, selbst auch auf andere Webseiten verlinken kann. Diese erhalten nunmehr ebenfalls einen höheren PageRank, den sie gegebenfalls wiederum über Links an unsere Seite mit dem zusätzlichen eingehenden Link zurückgeben.

Die einzelnen Effekte zusätzlicher eingehender Links sollen nun anhand eines Beispiels beschrieben werden.



Wir betrachten eine kleine Website aus den Seiten A, B, C und D, die im Kreis verlinkt sind und nicht selbst auf andere Webseiten verlinken. Ohne eingehende Links von externen Seiten ergibt sich ganz offensichtlich ein PageRank von 1 für jede der betrachteten Seiten. Wir wählen nunmehr eine Seite X, für die ein PageRank PR(X) von 10 angenommen wird. Seite X verlinkt auf Seite A und sonst auf keine andere Seite. Bei einem willkürlich gewählten Dämpfungsfaktor von 0.5 ergibt sich das folgende Gleichungssystem für den PageRank der einzelnen Seiten unserer Site:

$PR(A) = 0.5 + 0.5(PR(X) + PR(D)) = 5.5 + 0.5PR(D)$
$PR(B) = 0.5 + 0.5PR(A)$
$PR(C) = 0.5 + 0.5PR(B)$
$PR(D) = 0.5 + 0.5PR(C)$

Da die Anzahl der ausgehenden Links jeder Seite gleich 1 ist, müssen diese hier nicht berücksichtigt werden. Die Lösung des Gleichungssystems ergibt folgende Werte für den PageRank der einzelnen Seiten:

$PR(A) = 19/3 = 6.33$
$PR(B) = 11/3 = 3.67$
$PR(C) = 7/3 = 2.33$

$PR(D) = 5/3 = 1.67$

Der unmittelbare Effekt des zusätzlichen Links auf Seite A in Höhe von

$d \times PR(X)/C(X) = 0,5 \times 10/1 = 5$

setzt sich also über die Verlinkung der einzelnen Seiten untereinander fort.

Der Einfluss des Dämpfungsfaktors

Der Grad der Weitergabe von PageRank ist vor allem auch abhängig von der Höhe des Dämpfungsfaktors d. Wird für diesen beispielsweise ein Wert von 0.75 angenommen, ergibt sich für das obige Beispiel das folgende Gleichungssystem:

$PR(A) = 0.25 + 0.75(PR(X) + PR(D)) = 7.75 + 0.75PR(D)$
$PR(B) = 0.25 + 0.75PR(A)$
$PR(C) = 0.25 + 0.75PR(B)$
$PR(D) = 0.25 + 0.75PR(C)$

Die Lösung dieses Gleichungssystems ergibt folgende Werte für den PageRank der einzelnen Seiten:

$PR(A) = 419/35 = 11.97$
$PR(B) = 323/35 = 9.23$
$PR(C) = 251/35 = 7.17$
$PR(D) = 197/35 = 5.63$

Es zeigt sich zunächst ein wesentlich höherer unmittelbarer Effekt des zusätzlichen eingehenden Links auf den PageRank von Seite A in Höhe von

$d \times PR(X)/C(X) = 0.75 \times 10/1 = 7.5$

Dieser Effekt setzt sich nun aber noch verstärkt durch die interne Verlinkung der Seiten fort, so dass der PageRank von Seite A bei einem Dämpfungsfaktor von 0.75 beinahe doppelt so hoch ist wie bei einem Dämpfungsfaktor von 0.5. Ist der PageRank von Seite A bei einem Dämpfungsfaktor von 0.5 noch beinahe viermal größer als der Pagerank von Seite D, so ist er bei einem Dämpfungsfaktor von 0.75 nur noch etwas mehr als doppelt so groß. Je höher der Dämpfungsfaktor ist, um so stärker ist einerseits der Effekt auf den PageRank der den Link erhaltenden Seite und um so gleichmäßiger verteilt sich andererseits der PageRank auf die anderen Seiten der Site.

Der tatsächliche Effekt eingehender Links

Die Summe der PageRank-Werte aller Seiten bei einem Dämpfungsfaktor von 0.5 beträgt in unserem Beispiel

$$PR(A) + PR(B) + PR(C) + PR(D) = 14$$

Dadurch, dass eine Seite mit einem PageRank von 10 mit ihrem einzigen Link auf eine Seite der Beispiel-Site verlinkt, erhöht sich also deren aufaddierter PageRank um 10. (Vor Erhalt des Links hatte jede Seite einen PageRank von 1.) Bei einem PageRank von 0.75 beträgt die Summe der PageRank-Werte

$$PR(A) + PR(B) + PR(C) + PR(D) = 34$$

Der aufaddierte PageRank erhöht sich also um 30. Es zeigt sich, dass sich die Summe des PageRanks stets um

$$(d/(1-d)) \times (PR(X)/C(X))$$

erhöht, wenn X die verlinkende Seite, PR(X) deren PageRank und C(X) die Anzahl der ausgehenden Links von Seite X ist. Dieser Wert ist allerdings daran gebunden, dass die Verlinkung in ein geschlossenes Sytem von Webseiten, also etwa eine Website ohne ausgehenden Link erfolgt. Sofern von der Website Links auf andere, externe Webseiten gesetzt sind, verringert sich der Faktor entsprechend.

Die Begründung für den oben angegebenen Wert liefert uns Raph Levien und sie bezieht sich auf das Random Surfer Modell. Die Länge eines Surf-Vorgangs des Zufalls-Surfers ist eine Exponentialverteilung mit einem Mittel von $(d/(1-d))$. Wenn also der Zufalls-Surfer einen Link auf ein geschlossenes System von Webseiten verfolgt, besucht er im Schnitt genau $(d/(1-d))$ Seiten innerhalb dieses geschlossenen Systems. Und genau so viel mehr PageRank der ursprünglich verlinkenden Seite - gewichtet nach der Anzahl der ausgehenden Links - muss damit an das geschlossene System übertragen werden.

Lawrence Page und Sergey Brin geben regelmäßig einen Dämpfungsfaktor von 0.85 für die tatsächliche PageRank-Berechnung an. Damit ergibt sich ein Faktor für die Erhöhung des aufaddierten PageRanks einer geschlossenen Site durch einen zusätzlichen eingehenden Link von Seite X in Höhe von

$$(0.85/0.15) \times (PR(X)/C(X)) = 5.67 \times (PR(X)/C(X))$$

Eingehende Links haben also einen weitaus größeren Effekt auf den PageRank als man bei oberflächlicher Betrachtung annehmen mag.

Die PageRank-1 Regel

Viele Nutzer der Google Toolbar stellen fest, dass oftmals Seiten mit einem bestimmten Toolbar-PageRank eine darauf verlinkende Seite mit einem Toolbar-PageRank aufweisen, der um den Wert 1 höher ist als der der verlinkten Seite. Diese Beobachtung dient vielfach dazu, den hier präsentierten PageRank-Algorithmus in Frage zu stellen. Dagegen soll an dieser Stelle gezeigt werden, dass die Beobachtung vollkom-

men im Einklang mit dem hier präsentierten PageRank-Algorithmus steht.

Zuallererst stützt die PageRank-1 Regel das grundlegende Konzept des PageRank-Verfahrens. Webseiten sind genau dann bedeutsam, wenn andere bedeutsame Webseiten auf sie verweisen. Es ist nicht erforderlich, dass eine Website viele eingehenden Links erhält, um einen hohen PageRank zu bekommen. Ein einzelner Link von einer Website mit einem hohen PageRank reicht hierzu aus.

Dafür, dass die PageRank-1 Regel auch mit dem hier präsentierten PageRank-Algorithmus in Einklang steht, sind mehrere Faktoren verantwortlich. Zunächst ist Toolbar-PageRank eine logarithmisch skalierte Version des tatsächlichen PageRank. Wenn der PageRank einer verlinkenden Seite im Sinne der Toolbar um eins höher ist als derjenige der verlinkten Seite, so kann ihr tatsächlicher PageRank immer mindestens um einen Faktor höher sein, der der Basis des für die Skalierung eingesetzten Logarithmus entspricht. Ist also die Basis des Logarithmus gleich 6, und der Toolbar-PageRank der verlinkenden Seite gleich 5, so kann der tatsächliche PageRank der verlinkten Seite immer mindestens 6 Mal kleiner sein, damit diese in jedem Fall noch einen Toolbar-PageRank von 4 erreicht.

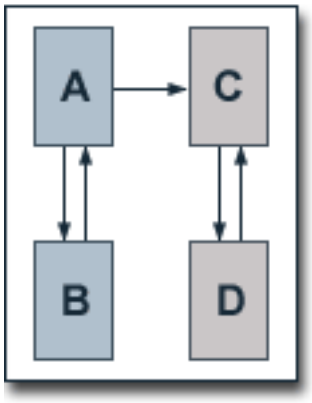
Der Basis des Logarithmus wirkt nun die Anzahl der ausgehenden Links auf der verlinkenden Seite entgegen, da deren PageRank praktisch unter allen verlinkten Seiten aufgeteilt wird. Es wurde allerdings oben auf dieser Seite bereits gezeigt, dass der über einen Link an eine Seite weitergegebene PageRank weitaus größer sein kann, als der im Algorithmus erscheinenden Term $d(PR(Ti)/C(Ti))$ vermuten lässt. Dies hängt damit zusammen, dass intern in der Regel weitere Seiten auf die von außen verlinkte Seite verlinken und somit weiteren PageRank auf diese Seite verteilen. Gehen wir nun etwa davon aus, dass die logarithmische Basis für die Skalierung 6 beträgt und weiterhin bei einem hohen Dämpfungsfaktor das doppelte des nach ausgehenden Links gewichteten PageRank der verlinkenden Seite auf die verlinkte Seite übertragen wird, so könnte die verlinkende Seite mindestens 12 ausgehende Links haben, damit die verlinkte Seite immer noch einen Toolbar-PageRank aufweist, der maximal um 1 kleiner ist als derjenige der verlinkenden Seite.

Eine Zahl von 12 ausgehenden Links erscheint an dieser Stelle zugegebenermaßen sehr gering. Allerdings ist es in aller Regel so, dass wenn eine Webseite von außen verlinkt wird, dies nicht nur von einer einzelnen Seite geschieht, und der betrachteten Seite somit noch weiterer PageRank übertragen wird. Falls sich Beispiele finden, bei der eine Seite durch einen einzigen externen Link ein PageRank übertragen wird, der der PageRank-1 Regel entspricht, und die verlinkende Seite eine hohe Zahl ausgehender Links hat, so ist dies vor allem ein Indiz dafür, dass der PageRank der verlinkenden Seite sich im oberen Bereich ihres Toolbar-PageRank-Wertes befindet. Schließlich könnte die verlinkende Seite aus unserem Beispiel eine "hohe"5 und die verlinkte Seite eine "tiefe"4 sein. In diesem Falle könnte die verlinkende Seite bis zu 72 ausgehende Links aufweisen. Diese Zahl würde sich weiter erhöhen, wenn wir von einer größeren logarithmischen Basis für die Skalierung des Toolbar-PageRanks ausgehen.

Der Effekt ausgehender Links

Da das PageRank-Verfahren die Link-Struktur des gesamten Webs abbildet, ist es

unausweichlich, dass wenn eingehende Links einen Einfluss auf den PageRank haben, das gleiche auch für ausgehende Links gilt. Zur Darstellung der Effekte ausgehender Links soll wieder ein kleines Beispiel dienen.



Betrachtet wird ein Web aus zwei Websites, die jeweils zwei Seiten beinhalten. Die eine Site besteht aus den Seiten A und B, die andere aus den Seiten C und D. Die beiden Seiten einer jeden Site verlinken sich jeweils gegeneinander. Es wird unmittelbar deutlich, dass jede der Seiten einen ursprünglichen PageRank von 1 inne hat. Nun wird Seite A ein externer Link auf Seite C hinzugefügt. Für den PageRank der einzelnen Seiten ergeben sich bei einem angenommenen Dämpfungsfaktor d von 0.75 die folgenden Gleichungen:

$$PR(A) = 0.25 + 0.75PR(B)$$
$$PR(B) = 0.25 + 0.375PR(A)$$
$$PR(C) = 0.25 + 0.75PR(D) + 0.375PR(A)$$
$$PR(D) = 0.25 + 0.75PR(C)$$

Die Lösung dieses Gleichungssystems ergibt die folgenden Werte:

$$PR(A) = 14/23$$
$$PR(B) = 11/23$$

und somit einen aufsummierten PageRank von 25/23 für die erste Site,

$$PR(C) = 35/23$$
$$PR(D) = 32/23$$

und damit einen aufsummierten PageRank von 67/23 für die zweite Site. Der aufsummierte PageRank beider Sites in Höhe von $92/23 = 4$ bleibt also erhalten. Das Hinzufügen von Links hat also keinen Einfluss auf den aufsummierten PageRank des Webs. Ferner ist damit der Gewinn der verlinkten Site genauso groß wie der Verlust der anderen.

Der tatsächliche Effekt ausgehender Links

Wie bereits gezeigt, ist der Gewinn eines geschlossenen Systems auf das ein zusätzlicher Link gesetzt wird, gegeben durch

$$(d/(1-d)) \times (PR(X)/C(X)),$$

wobei X die verlinkende Seite, PR(X) deren PageRank und C(X) die Anzahl der ausgehenden Links von Seite X ist. Dieser Wert beschreibt damit auch den PageRank-Verlust, der einem vormals geschlossenen System daraus entsteht, dass einer Seite X innerhalb dieses Systems ein ausgehender Link hinzugefügt wird.

Bedingung für die angegebene Formel ist, dass die verlinkte Site nicht etwa direkt wieder auf die verlinkende Site zurückverlinkt, da die verlinkende Site wieder einen Teil des verlorenen PageRanks zurückgewinnen würde. Solange eine Rückverlinkung sich über eine gar nicht so große Anzahl von Webseiten erstreckt, ist dieser Effekt jedoch durch die Wirkungsweise des Dämpfungsfaktors zu vernachlässigen. Ferner Bedingung für die Gültigkeit der Formel ist, dass die verlinkende Site nicht bereits vorher ausgehende Links besitzt. Sollte dies jedoch der Fall sein, vermindert sich die Höhe des Verlustes der betrachteten Site, und gleichzeitig haben die bereits verlinkten Webseiten einen entsprechenden Verminderung des PageRank hinzunehmen.

Selbst wenn für eine tatsächlich existierende Website die PageRank-Werte der einzelnen Webseiten bekannt wären, könnte allerdings dennoch nicht ohne weiteres im Vorhinein ermittelt werden, wie sehr das Hinzufügen eines externen Links den PageRank der einzelnen Seiten vermindert, da die oben genannten Formel den Status nach der Verlinkung betrachtet.

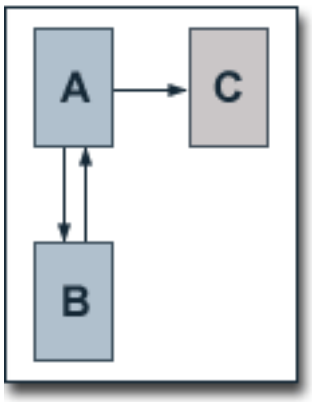
Intuitive Begründung für den Effekt ausgehender Links

Intuitiv lässt sich der Verlust von PageRank für die eigenen Seiten damit erklären, dass der Zufalls-Surfer aus dem Random Surfer Modell durch das Hinzufügen eines externen Links mit einer geringeren Wahrscheinlichkeit einen Link auf eine der internen Seiten verfolgt. Damit sinkt in der Folge auch die Wahrscheinlichkeit, mit der sich der Surfer auf einer derjenigen Seiten aufhält, die wiederum auf diejenige Seite verlinken, der der externe Link hinzugefügt wurde, womit auch deren PageRank sinken muss.

Es bleibt letztlich festzuhalten, dass ausgehende externe Links den aufsummierten PageRank aller Webseiten einer Website und gegebenenfalls auch den PageRank jeder einzelnen Seite einer Site vermindern. Da jedoch die Verlinkung zwischen Websites gerade die Grundlage des PageRank-Verfahrens darstellt und für sein Funktionieren unabdingbar ist, besteht durchaus die Möglichkeit, dass ausgehende Links an einer anderen Stelle innerhalb der Bewertung von Webseiten durch die Suchmaschine Google positiven Einfluss nehmen. Schließlich machen gerade auch relevante ausgehende Links die Qualität einer Website aus, und Webmaster, die Links auf andere Websites setzen, beziehen gewissermaßen deren Content in das eigene Web-Angebot mit ein.

Dangling Links

Ein nicht ganz unwichtiger Aspekt ausgehender Links ist das Fehlen ausgehender Links. Sobald einzelne Webseiten keine ausgehenden Links aufweisen, versickert der PageRank gewissermaßen an diesen Stellen. Lawrence Page und Sergey Brin bezeichnen Verweise auf derartige Seiten als "Dangling Links".



Die Auswirkungen von Dangling Links sollen anhand eines kleinen Beispiels veranschaulicht werden. Wir betrachten eine Website die aus aus den drei Seiten A, B und C besteht. Die Seiten A und B verlinken sich gegenseitig. Seite A verlinkt zudem auf Seite C, die ihrerseits jedoch keine ausgehenden Links aufweist. Für den PageRank der einzelnen Seiten ergeben sich bei einem angenommenen Dämpfungsfaktor d von 0.75 die folgenden Gleichungen:

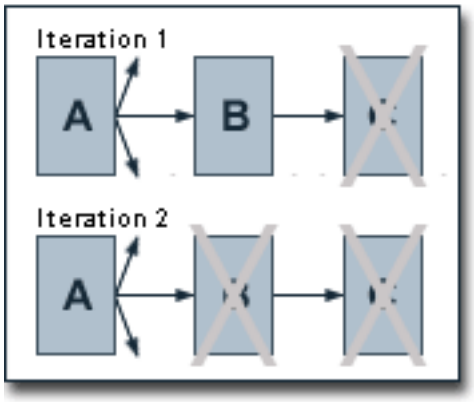
$PR(A) = 0.25 + 0.75PR(B)$
$PR(B) = 0.25 + 0.375PR(A)$
$PR(C) = 0.25 + 0.375PR(A)$

Die Lösung dieses Gleichungssystems ergibt die folgenden PageRank-Werte:

$PR(A) = 14/23$
$PR(B) = 11/23$
$PR(C) = 11/23$

Damit beträgt der aufaddierte PageRank aller Seiten 36/23, also nur etwas mehr als die Hälfte dessen, was zu erwarten gewesen wäre, wenn Seite C auf eine der beiden Seiten A oder B verlinkt hätte. Die Anzahl von Dangling Links ist nach Angaben von Page und Brin nicht unbeträchtlich - und sei es, weil zahlreiche verlinkte Seiten von Google nicht indexiert sind, etwa weil die Indexierung per robots.txt verhindert wird. Hier ist zusätzlich zu berücksichtigen, dass Google mittlerweile auch andere Dokumenten-Typen als HTML wie zum Beispiel PDF oder Word Dateien indexiert, die keine wirklichen ausgehenden Links haben. Dangling Links könnten also nicht unbeträchtliche Folgen für das PageRank-Verfahren haben.

Um die negativen Effekte von Dangling Links auszuschalten, werden diese Angaben von Page und Brin zufolge vor der PageRank-Berechnung aus der Datenbank



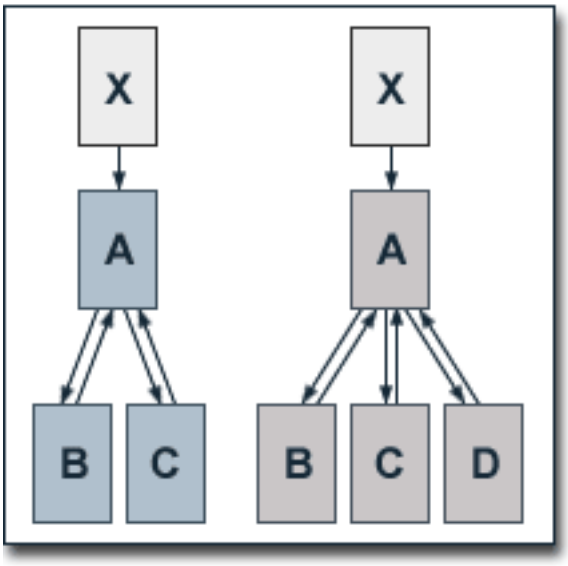
unter Anpassung der jeweiligen Anzahl von ausgehenden Links entfernt bis alle PageRank-Werte berechnet sind. Bei der Entfernung von Dangling Links handelt es sich um einen iterativen Vorgang, da das Entfernen wiederum neue Dangling Links erzeugen kann, wie aus unserer einfachen Abbildung ersichtlich. Nachdem die eigentliche PageRank-Berechnung abgeschlossen ist, wird auch den Dangling Links PageRank - auf der Basis der PageRank-Werte der auf sie verweisenden Seiten und unter Rückgriff auf den PageRank-Algorithmus - zugewiesen. Dies erfordert ebenso viele Iterationen wie bei der Entfernung der Dangling Links. Um bei unserer Abbildung zu bleiben, könnte schließlich Seite C vor Seite B bearbeitet werden. Seite B weist dann im ersten Bearbeitungsdurchlauf bei der Bearbeitung von Seite C noch keinen PageRank auf, womit Seite C wiederum ein PageRank von 0 zugewiesen würde. Erst anschließend erhält Seite B ihren PageRank und im zweiten Bearbeitungsschritt würde Seite C einen tatsächlichen PageRank zugewiesen bekommen.

Für unser ursprüngliches Beispiel hat das Entfernen von Seite C aus der Datenbank zur Folge, dass die Seiten A und B nach Abschluss der Berechnungen jeweils einen PageRank von 1 erhalten. Seite C wird dann im Anschluss ein PageRank in Höhe von $0.25 + 0.375PR(A) = 0.625$ zugewiesen. Damit enspricht der aufaddierte PageRank zwar nicht der Anzahl der Seiten, doch zumindest diejenigen Seiten mit ausgehenden Links nehmen keinen Schaden durch Dangling Links.

Durch die Eliminierung von Dangling Links haben diese also keinen negativen Einfluss auf den PageRank der übrigen Seiten. Und wie bereits erwähnt, sind Verweise auf Dokumententypen, die keine ausgehenden Links aufweisen können, grundsätzlich Dangling Links. Damit wird auch unmittelbar deutlich, dass etwa Links auf PDF-Dokumente den PageRank einer darauf verlinkenden Seite bzw. Site nicht reduzieren können. PDF-Dokumente können also ein sehr gutes Instrument der Suchmaschinenoptimierung für Google sein.

Der Einfluss der Anzahl der Seiten auf den PageRank

Da der aufaddierte PageRank aller Seiten des Webs gleich der Anzahl der Seiten ist, folgt unmittelbar, dass eine zusätzliche Seite den aufaddierten PageRank des Webs um eins erhöht. Wesentlich interessanter als die Auswirkungen zusätzlicher Seiten auf den aufaddierten PageRank des gesamten Webs sind die Auswirkungen auf den PageRank der Seiten einer konkreten Site.



Um die konkreten Auswirkungen zusätzlicher Seiten zu veranschaulichen, betrachten wir zunächst eine hierarchisch strukturierte Beispielsite bestehend aus den drei Seiten A, B und C, der auf der unteren Ebene eine zusätzliche Seite D hinzugefügt wird. Die Site hat keine ausgehenden Links. Auf Seite A verlinkt eine externe Seite X mit einem PageRank von 10 durch ihren einzigen ausgehenden Link. Bei einem Dämpfungsfaktor d in Höhe von 0.75 ergeben sich vor dem Hinzufügen von Seite D die folgenden Gleichungen für den PageRank der einzelnen Seiten:

$$PR(A) = 0.25 + 0.75(10 + PR(B) + PR(C))$$
$$PR(B) = PR(C) = 0.25 + 0.75(PR(A)/2)$$

Die Lösung des Gleichungssystems ergibt die folgenden PageRank-Werte:

$$PR(A) = 260/14$$
$$PR(B) = 101/14$$
$$PR(C) = 101/14$$

Nach dem Hinzufügen von Seite D lauten die Gleichungen für die PageRank-Berechnung folgendermaßen:

$$PR(A) = 0.25 + 0.75(10 + PR(B) + PR(C) + PR(D))$$
$$PR(B) = PR(C) = PR(D) = 0.25 + 0.75(PR(A)/3)$$

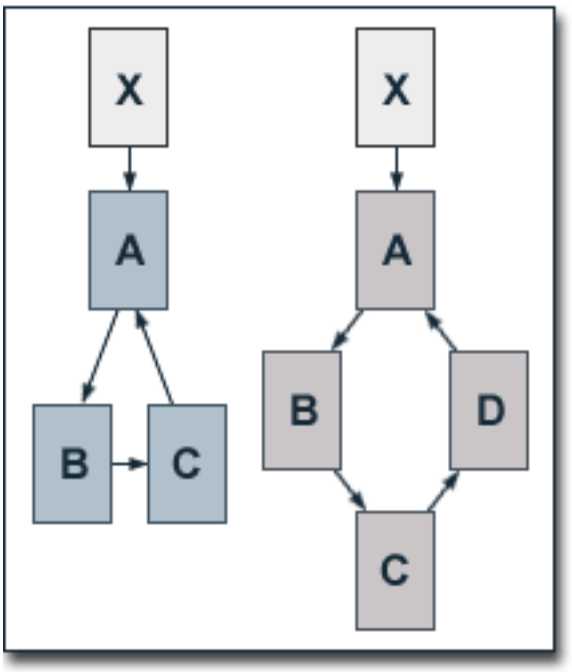
Die Lösung dieses Gleichungssystems ergibt die folgenden PageRank-Werte:

$$PR(A) = 266/14$$
$$PR(B) = 70/14$$
$$PR(C) = 70/14$$
$$PR(D) = 70/14$$

Da unsere Beispielsite keine ausgehenden Links aufweist, steigt der aufaddierte PageRank aller Seiten nach dem Hinzfügen von Seite D erwartungsgemäß um genau 1 von 33 auf 34. Ferner steigt der PageRank von Seite A marginal an. Der PageRank der Seiten B und C jedoch sinkt um ein beträchtliches Maß.

Die Reduzierung des PageRanks durch zustätzliche Seiten

Bei dem Hinzufügen zusätzlicher Seiten zu einer Website mit strikt hierarchischer Struktur sind die Auswirkungen auf den PageRank der bereits bestehenden Seiten uneinheitlich. Welche Auswirkungen das hinzufügen von Seiten zu Sites mit anderer Struktur hat, soll wiederum anhand eines Beispiels erläutert werden.



Betrachtet wird jetzt eine Website, deren Seiten A, B und C untereinander im Kreis verlinken und der eine zusätzliche Seite D hinzugefügt wird, die sich in die bestehende Linkstruktur einpasst. Die Site hat ebenfalls keine ausgehenden Links. Auf Seite A verlinkt wiederum eine externe Seite X mit einem PaeRank von 10 durch ihren einzigen ausgehenden Link. Bei einem Dämpfungsfaktor d in Höhe von 0.75 ergeben sich vor dem Hinzufügen von Seite D die folgenden Gleichungen für den PageRank der einzelnen Seiten:

$PR(A) = 0.25 + 0.75(10 + PR(C))$
$PR(B) = 0.25 + 0.75 \times PR(A)$
$PR(C) = 0.25 + 0.75 \times PR(B)$

Die Lösung des Gleichungssystems ergibt die folgenden PageRank-Werte:

$PR(A) = 517/37 = 13.97$
$PR(B) = 397/37 = 10.73$

$PR(C) = 307/37 = 8.30$

Nach dem Hinzufügen von Seite D lauten die Gleichungen für die PageRank-Berechnung folgendermaßen:

$PR(A) = 0.25 + 0.75(10 + PR(D))$
$PR(B) = 0.25 + 0.75 \times PR(A)$
$PR(C) = 0.25 + 0.75 \times PR(B)$
$PR(D) = 0.25 + 0.75 \times PR(C)$

Die Lösung dieses Gleichungssystems ergibt die folgenden PageRank-Werte:

$PR(A) = 419/35 = 11.97$
$PR(B) = 323/35 = 9.23$
$PR(C) = 251/35 = 7.17$
$PR(D) = 197/35 = 5.63$

Wiederum steigt der aufaddierte PageRank aller Seiten nach dem Hinzfügen von Seite D um genau 1 von 33 auf 34. Jetzt allerdings verlieren alle bereits vorher existierenden Seiten an PageRank. Dieser Effekt zeigt sich stets um so eher, je gleichmäßiger der PageRank auf die einzelnen Seiten einer Site verteilt werden.

Damit wird auch deutlich, dass der PageRank-Algorithmus grundsätzlich kleinere Websites bevorzugt. Dies ist allerdings dadurch zu relativieren, dass Sites mit mehr Content dies ausgleichen können, indem andere Seitenbetreiber um so eher auf sie verlinken.

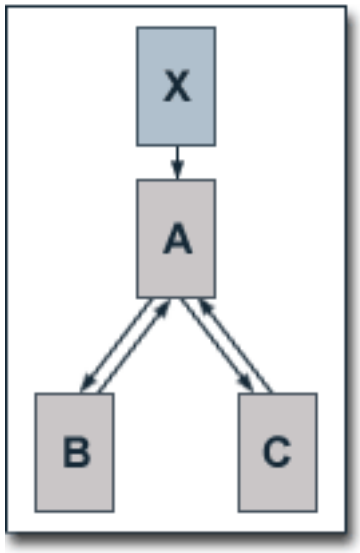
Es ist allerdings auch möglich, durch zusätzliche Seiten den PageRank bereits existierender Seiten zu steigern. Hierbei ist jedoch darauf zu achten, dass auf die zusätzlichen Seiten möglichst wenig PageRank verteilt wird.

Die Distribution von PageRank

Bislang wurde erörtert, wie durch die Anzahl von ein- und ausgehenden Links sowie durch die Anzahl der Webseiten einer Site der PageRank beinflusst werden kann. An dieser Stelle hingegen soll hauptsächlich besprochen werden, wie mittels der internen Link-Struktur einer Site zum Zwecke der Suchmaschinenoptimierung Einfluss auf den PageRank genommen werden kann.

In den meisten Fällen sind Websites zumindest bedingt hierarchisch struktieriert. Dabei ist in der Regel die Startseite für den wichtigsten Suchbegriff bzw. die wichtigste Suchphrase optimiert. In unserem Beispiel erhölt die optimierte Startseite A einen eingehenden Link von einer Seite X mit einem PageRank von 10 und einem einzigen ausgehenden Link. Die Seiten B und C erhalten einen Link von Seite A und verlinken auch wieder auf diese zurück. Hieraus ergeben sich bei einem angenommenen Dämpfungsfaktor d in Höhe von 0.5 die folgenden Gleichungen für die PageRank-Berechnung:

$PR(A) = 0.5 + 0.5(10 + PR(B) + PR(C))$

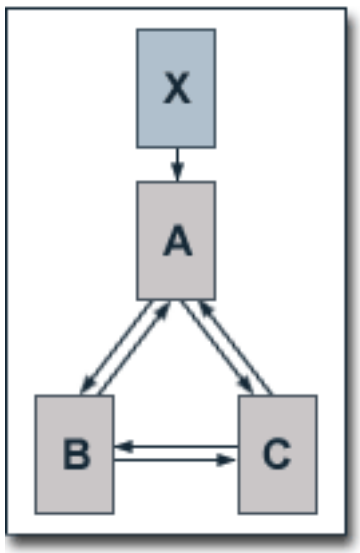


$PR(B) = 0.5 + 0.5(PR(A)/2)$
$PR(C) = 0.5 + 0.5(PR(A)/2)$

Die Lösung dieses Gleichungssystems ergibt die folgenden PageRank-Werte:

$PR(A) = 8$
$PR(B) = 2.5$
$PR(C) = 2.5$

Nun ist es in der Regel nicht ausreichend, im Rahmen der Suchmaschinenoptimierung lediglich die Startseite für einen Suchbegriff zu optimieren. Es ist vielmehr ratsam, alle Seiten auf die Optimierung für jeweils unterschiedliche Suchbegriffe auszurichten.



Sobald die Startseite für den optimierten Suchbegriff hinreichend gute Suchma-

schinenergebnisse erzielt, die anderen Seiten hingegen noch nicht, empfiehlt gegenenfalls die Linkstruktur entsprechend der folgenden Vorgehensweise bei unserer Beispielsite zu modifizieren. Die hierachisch nachgeordneten Seiten B und C verlinken sich nunmehr gegenseitig, und bei ansonsten gleichen Bedingungen ergibt sich das folgende Gleichungssystem für die PageRank-Berechnung:

$$PR(A) = 0.5 + 0.5(10 + PR(B)/2 + PR(C)/2)$$
$$PR(B) = 0.5 + 0.5(PR(A)/2 + PR(C)/2)$$
$$PR(C) = 0.5 + 0.5(PR(A)/2 + PR(B)/2)$$

Hieraus ergeben sich nun die folgenden PageRank-Werte für die einzelnen Seiten:

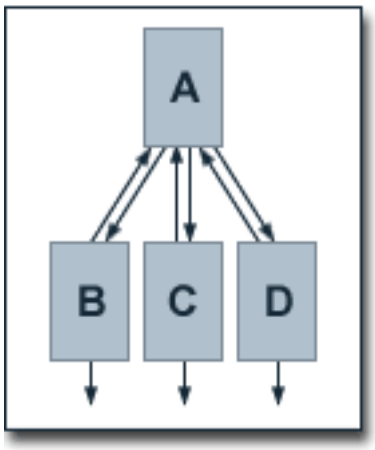
$$PR(A) = 7$$
$$PR(B) = 3$$
$$PR(C) = 3$$

Es zeigt sich, dass die PageRank-Werte für die Seiten B und C steigen, so dass diese wahrscheinlich für die optimierten Suchbegriffe höher in den Suchmaschinenergebnissen erscheinen werden. Andererseits besteht die Möglichkeit, dass die Startseite in den Suchmaschinenergebnissen absinkt.

Grundsätzlich zeigt sich, dass sich im Rahmen der Suchmaschinenoptimierung der PageRank innerhalb einer Seite um so gleichmäßiger verteilt, je stärker die hierarchisch nachrangigen Seiten untereinander verlinkt sind.

Gezielte Distribution von PageRank durch Konzentration der ausgehenden Links

Dass ausgehende Links sich grundsätzlich eher negativ auf den PageRank der Seiten einer Website auswirken, wurde bereits gezeigt. An dieser Stelle soll erörtert werden, wie dieser Effekt durch die gezielte Platzierung der ausgehenden Links minimiert werden kann.



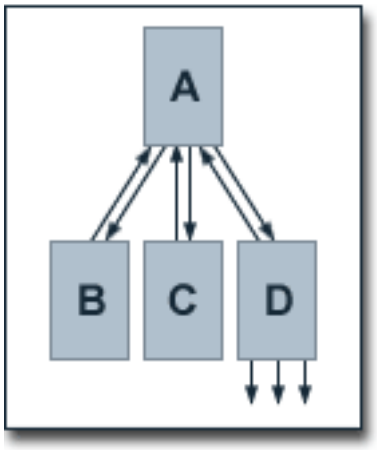
Betrachtet wird nun eine Beispielsite aus den Seiten A, B, C und D, wobei Seite A auf die anderen Seiten verlinkt, und diese neben einem Link auf Seite A jeweils auch

noch einen ausgehenden Link haben. Bei einem angenommen Dämpfungsfaktor d in Höhe von 0.5 ergeben sich die folgenden Gleichungen für die PageRank-Berechnung:

$$PR(A) = 0.5 + 0.5(PR(B)/2 + PR(C)/2 + PR(D)/2)$$
$$PR(B) = PR(C) = PR(D) = 0.5 + 0.5(PR(A)/3)$$

Die Lösung dieses Gleichungssystems ergibt die folgenden PageRank-Werte:

$$PR(A) = 1$$
$$PR(B) = 2/3$$
$$PR(C) = 2/3$$
$$PR(D) = 2/3$$



Nunmehr wird die Beispiel-Website so modifiziert, dass bei ansonsten gleichen Voraussetzungen nurmehr Seite D alle ausgehenden Links auf sich vereint und die Seiten B und C keinerlei ausgehenden mehr besitzen. Bei einem Dämpfungsfaktor d in Höhe von 0.5 ergeben sich die folgenden Gleichungen für die PageRank-Berechnung:

$$PR(A) = 0.5 + 0.5(PR(B) + PR(C) + PR(D)/4)$$
$$PR(B) = PR(C) = PR(D) = 0.5 + 0.5(PR(A)/3)$$

Die Lösung dieses Gleichungssystems ergibt die folgenden PageRank-Werte:

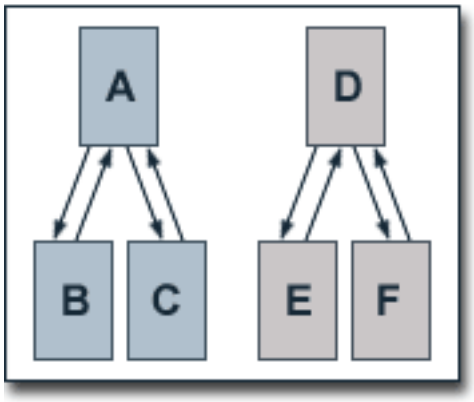
$$PR(A) = 17/13$$
$$PR(B) = 28/39$$
$$PR(C) = 28/39$$
$$PR(D) = 28/39$$

Es zeigt sich unmittelbar, dass für unsere Beispiel-Website die PageRank-Werte aller vier Seiten erhöhen. Vor dem Hintergrund der Suchmaschinenoptimierung kann es also durchaus ratsam sein, die ausgehenden Links einer Website auf einer einzelnen

Seite zu konzentrieren, wobei allerdings durchaus nicht vergessen werden darf, dass dies der Benutzerfreundlichkeit abträglich sein kann.

Linktausch zum Zwecke der Suchmaschinenoptimierung

Viele Webmaster streben zum Zwecke der Suchmaschinenoptimierung den Linkaustausch mit möglichst vielen anderen Websites an, um auf diese Weise ihre Link-Popularität zu erhöhen. Da das Hinzufügen von Links allerdings keinerlei Effekte auf den aufaddierten PageRank innerhalb geschlossener Systeme von Webseiten hat, stellt sich die Frage, in wie fern ein Linkaustausch zwischen Websites überhaupt Auswirkungen auf den PageRank hat.



Wir betrachten zwei hierarchisch strukturierte Websites aus den Seiten A, B und C bzw. D, E und F. Seite A verlinkt auf die Seiten B und C und diese wiederum verlinken zurück auf Seite A. Da die zweite Site exakt gleich strukturiert ist, ergeben sich für sie die gleichen PageRank-Werte, die deshalb an dieser Stelle nicht berücksichtigt werden müssen. Bei einem Dämpfungsfaktor d in Höhe von 0.5 ergeben sich die folgenden Gleichungen für die PageRank-Berechnung:

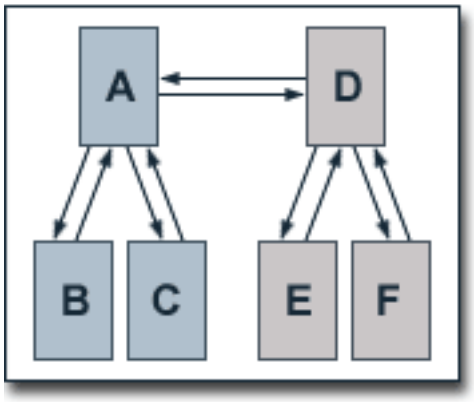
$$PR(A) = 0.5 + 0.5(PR(B) + PR(C))$$
$$PR(B) = PR(C) = 0.5 + 0.5(PR(A)/2)$$

Die Lösung des Gleichungssystems ergibt die folgenden PageRank-Werte für die einzelnen Seiten:

$$PR(A) = 4/3$$
$$PR(B) = 5/6$$
$$PR(C) = 5/6$$

und analog

$$PR(D) = 4/3$$
$$PR(E) = 5/6$$
$$PR(F) = 5/6$$



Nunmehr findet für unsere Beispiel-Websites ein Linktausch statt. Seite A verlinkt auf Seite D und umgekehrt. Bei ansonsten gleichen Voraussetzungen mit einem Dämpfungsfaktor d in Höhe von 0.5 ergibt sich jetzt das folgende Gleichungssystem für die Berechnung der PageRank-Werte:

$$PR(A) = 0.5 + 0.5(PR(B) + PR(C) + PR(D)/3)$$
$$PR(B) = PR(C) = 0.5 + 0.5(PR(A)/3)$$
$$PR(D) = 0.5 + 0.5(PR(E) + PR(F) + PR(A)/3)$$
$$PR(E) = PR(F) = 0.5 + 0.5(PR(D)/3)$$

Die Lösung des Gleichungssystems ergibt die folgenden PageRank-Werte:

$$PR(A) = 3/2$$
$$PR(B) = 3/4$$
$$PR(C) = 3/4$$
$$PR(D) = 3/2$$
$$PR(E) = 3/4$$
$$PR(F) = 3/4$$

Es zeigt sich also, dass durch den Linktausch die beiden Seiten A und D profitieren und die hierarchisch nachgeordneten Seiten PageRank einbüßen. Für die Suchmaschinenoptimierung bedeutet dies in erster Linie, dass hier ein genau entgegengesetzter Effekt wie bei der stärkeren internen Verlinkung von hierarchisch nachgeordneten Seiten stattfindet. Der Linktausch bietet sich also insbesondere an, wenn nur mit einer Seite auf einen bestimmten Suchbegriff abgezielt werden soll.

Voraussetzung für die genannten positiven Wirkungen durch einen Linktausch ist in jedem Falle, dass die jeweils verlinkenden Seiten einen ähnlich großen PageRank an die jeweils andere Site weitergeben. Sollte etwa eine Seite einen wesentlich größeren PageRank oder aber wesentlich weniger ausgehende Links haben, so besteht die Möglichkeit, dass alle Seiten ihrer Site an PageRank einbüßen. Ein nicht zu unterschätzender Einflussfaktor ist hier auch die Größe der beiden Sites. Je mehr Seiten eine Website besitzt, um so mehr des PageRanks aus eingehenden Links wird auf andere Seiten der Site verteilt, unabhängig davon, weie viele ausgehende Links die am Linktausch beteiligte Seite hat. Damit profitiert die am Linktausch beteiligte Seite

selbst relativ wenig vom Linktausch, und kann an die andere am Linktausch beteiligte Seite nur relativ wenig PageRank zurückgeben. Letzlich sollten die genannten Faktoren stets gegeneinander abgewogen werden, bevor ein Linktausch eingegangen wird.

Abschliekend bleibt anzumerken, dass ein Linktausch auch positive Effekte für alle Seiten einer Site haben kann, ohne dass die andere am Linktausch beteiligte Site geschädigt wird. Dies kann der Fall sein, wenn die am Linktausch beteiligte Seite bereits eine bestimmte Anzahl ausgehender Links auf Seiten aufweist, die nicht in direkter oder indirekter Form auf die betrachtete Site zurückverlinken. Mit dem Linktausch geht der betrachteten Site dann weniger PageRank durch die bereits vorher existierenden ausgehenden Links verloren.

A.2 Formale Sprachen

Beweis der Äquivalenz linearer Grammatiken und endlicher Automaten [Klenk 1980]: S. 75 - 77.

Wir zeigen nun, daß es zu jeder einseitig-linearen Grammatik G einen FNA A und umgekehrt für jeden FNA A eine einseitig-lineare Grammatik G gibt, so daß L(G) = L(A). Dieser Satz ist bedeutsam, da aus ihm hervorgeht, daß endliche Automaten einseitig-lineare Sprachen akzeptieren, und da es, wie wir gesehen haben, dafür einfache Analyseverfahren gibt. Um den Satz zu beweisen, führen wir einen speziellen Typ von einseitig-linearen Grammatiken ein, die regulären Grammatiken.

Definition 3.1.3. (reguläre Grammatik): Eine rechts-lineare (links-lineare) Grammatik $G = (V_N, V_T, R, S)$ heißt *rechts-(links-)-regulär* gdw jede Regel aus R von folgender Form ist:

1) $S \rightarrow \varepsilon$
oder 2) $A \rightarrow aB$ $(A \rightarrow Ba)$
oder 3) $A \rightarrow a$
mit $A,B \in V_N$, $a \in V_T$, wobei, falls $S \rightarrow \varepsilon \in R$, $B \neq S$ gilt.

Die Menge der rechts-regulären und die der links-regulären Grammatiken bilden vereinigt die Menge der *regulären* Grammatiken.

Es gilt

Satz 3.1.2.: Zu jeder einseitig-linearen Grammatik $G = (V_N, V_T, R, S)$ gibt es eine reguläre Grammatik $G_1 = (V_{N1}, V_T, R_1, S_1)$, so daß $L(G) = L(G_1)$.

Abbildung A.1: Beweis der Aequivalenz linearer Grammatiken und endlicher Automaten 1

Wir geben eine kurze Skizzierung des Beweises.

Beweisskizze: Wir können auf Grund von Satz 2.2.4.1.7. davon ausgehen, daß G rechts-linear ist, und, da G kontextfrei ist, auf Grund der Sätze 2.2.3.1. und 2.2.4.1.4., daß R keine Regeln der Form $A \rightarrow \varepsilon$ mit $A \neq S$ und der Form $A \rightarrow B$ mit $A,B \in V_n$ enthält. In R sind also außer eventuell $S \rightarrow \varepsilon$ nur Regeln der Form $A \rightarrow xB$ und $A \rightarrow x$ mit $A,B \in V_N$ und $x \in V_T^* - \{\varepsilon\}$. R_1 wird, wie folgt, konstruiert: Die Regeln aus R der Form $A \rightarrow aB$ und $A \rightarrow a$ mit $a \in V_T$ gehören zu R_1. Für jede Regel r aus R der Form $A \rightarrow xB$ oder $A \rightarrow x$ mit $x = a_1 \ldots a_n$, $n > 1$ und $a_1,\ldots,a_n \in V_T$ werden n-1 neue bisher nicht verwendete Nicht-Terminalsymbole $A_1,\ldots,A_{n-1}$ eingeführt und statt r die neuen Regeln $A \rightarrow a_1A_1$, $A_1 \rightarrow a_2A_2$, ..., $A_{n-2} \rightarrow a_{n-1}A_{n-1}$ sowie für den Fall $A \rightarrow xB$ die Regel $A_{n-1} \rightarrow a_nB$, für den Fall $A \rightarrow x$ die Regel $A_{n-1} \rightarrow a_n$. Diese Regeln liefern dasselbe Ableitungsergebnis wie r. Damit ist R_1 konstruiert. V_{N1} ist die Vereinigungsmenge aus V_N und der Menge aller neu eingeführten Nicht-Terminalsymbole.

Satz 3.1.3. Hauptsatz für endliche Automaten: Für jede einseitig-lineare Sprache L gibt es einen endlichen Automaten A, so daß L(A) = L. Für jeden endlichen Automaten A ist L(A) eine einseitig-lineare Sprache.

Beweis:

Teil 1: Nach Satz 3.1.2. gibt es für jede einseitig-lineare Sprache L eine rechts-reguläre Grammatik G mit L = L(G). Sei $G = (V_N,V_T,R,S)$. Wir nehmen an, daß $\varepsilon \notin L(G)$. Es wird ein FNA $A = (K,V,\delta,F,X)$ folgendermaßen konstruiert: $K = V_N \cup \{T\}$, wobei $T \notin V_N$, $V = V_T$, $F = \{T\}$, $X = \{S\}$. Ist r eine Regel aus R der Form $B \rightarrow aC$, so gilt $C \in \delta(B,a)$. Ist r von der Form $B \rightarrow a$, so gilt $T \in \delta(B,a)$. Damit ist δ gegeben. Wir zeigen, daß L(G) = L(A).

Sei $a_1 \ldots a_n$ ein beliebiges Element aus L(G) mit $a_1,\ldots,a_n \in V_T$. Dann entsteht $a_1 \ldots a_n$ durch sukzessive Anwendung von Regeln $S \rightarrow a_1C_1$, $C_1 \rightarrow a_2C_2$, ..., $C_{n-1} \rightarrow a_n$ mit $C_1,\ldots,C_n \in V_N$. Entsprechend gilt für A $C_1 \in \delta(S,a_1)$, $C_2 \in \delta(C_1,a_2)$, ..., $C_{n-1} \in \delta(C_{n-2},a_{n-1})$, $T \in \delta(C_{n-1},a_n)$. Also wird $a_1 \ldots a_n$ von A akzeptiert. Es gilt also $L(G) \subseteq L(A)$.

Sei nun $a_1 \ldots a_n$ ein beliebiges Element aus L(A). Dann gilt $C_1 \in \delta(S,a_1)$, $C_2 \in \delta(C_1,a_2)$, ..., $C_{n-1} \in \delta(C_{n-2},a_{n-1})$,

Abbildung A.2: Beweis der Aequivalenz linearer Grammatiken und endlicher Automanten 2

$T \in \delta(C_{n-1},a_n)$ für irgendwelche $C_1,\ldots,C_{n-1} \in K$. Dazu gibt es Regeln $S \rightarrow a_1C_1$, $C_1 \rightarrow a_2C_2$, ..., $C_{n-1} \rightarrow a_n$ aus R . Diese erzeugen $a_1\ldots a_n$. Daher gilt auch $L(A) \subseteq L(G)$.

Also gilt $L(A) = L(G) = L$.

Ist $\varepsilon \in L(G)$, so wird $X = \{S,T\}$ gesetzt und der Übergang $T \in \delta(T,\varepsilon)$ eingeführt. Dann ist auch ε aus L(A) (vgl. Definition 3.1.2.). Damit ist gezeigt, daß es für jede einseitig-lineare Sprache L einen endlichen Automaten A mit $L(G) = L(A)$ gibt.

Teil 2: Sei $A = (K,V,\delta,F,X)$ ein FNA. Ohne Beschränkung der Allgemeinheit können wir davon ausgehen, daß $K \cap V = \emptyset$. Zu A wird eine rechts-reguläre Grammatik $G = (V_N,V_T,R,S_1)$ folgendermaßen konstruiert: $V_N = K \cup \{S_1\}$, wobei $S_1 \notin K$. $V_T = V$. Für jedes $\delta(B,a)$ mit $\delta(B,a) = \{B_1,\ldots,B_m\}$ für irgendwelche Zustände $B, B_1,\ldots, B_m \in K$ und irgendein $a \in V$ sind 1) die Regeln $B \rightarrow aB_1$, ..., $B \rightarrow aB_m$ aus R. 2) Gibt es ein B_i $(1 \leq i \leq m)$ mit $B_i \in F$, so ist auch $B \rightarrow a$ aus R. Ist $B \in X$, so sind neben den nach 1) und 2) konstruierten Regeln auch 3) $S_1 \rightarrow aB_1$, ..., $S_1 \rightarrow aB_m$ aus R und, wenn zugleich ein B_i aus F ist, auch 4) $S_1 \rightarrow a$. Ist $\varepsilon \in L(A)$, so ist $S_1 \rightarrow \varepsilon$ aus R. Damit ist G konstruiert.

Wir zeigen, daß $L(G) \subseteq L(A)$. Sei $a_1\ldots a_n$ mit $n > 1$ ein beliebiges Element aus L(G) mit $a_1,\ldots,a_n \in V_T$. Dann wird $a_1\ldots a_n$ durch sukzessive Anwendung von Regeln $S_1 \rightarrow a_1C_1$, $C_1 \rightarrow a_2C_2$, ..., $C_{n-1} \rightarrow a_n$ aus R erzeugt. Die Regel $S_1 \rightarrow a_1C_1$ ist aus einer Regelmenge, die nach 3) entstanden ist, d. h. es gibt ein $B \in X$, so daß $C_1 \in \delta(B,a_1)$. Die Regel $C_{n-1} \rightarrow a_n$ ist nach 2) entstanden, d. h. es gibt ein $B \in F$, so daß $B \in \delta(C_{n-1},a_n)$. Die übrigen Regeln sind nach 1) entstanden, d. h. für jedes $C_i \rightarrow a_{i+1}C_{i+1}$ gilt $C_{i+1} \in \delta(C_i,a_{i+1})$. Also ist $a_1\ldots a_n \in L(A)$. Ist $a \in L(A)$ mit $a \in V_T$, so wird a durch $S_1 \rightarrow a$ erzeugt. Diese Regel entsteht nach 4), d. h. es gibt ein $B \in X$ und ein $C \in F$, so daß $C \in \delta(B,a)$. Damit ist $a \in L(A)$. Ist $\varepsilon \in L(G)$, so kann ε nur durch $S_1 \rightarrow \varepsilon$ erzeugt werden. Diese Regel ist nur dann aus R, wenn $\varepsilon \in L(A)$. Damit ist gezeigt, daß für alle $x \in L(G)$ gilt: $x \in L(A)$. Also gilt $L(G) \subseteq L(A)$.

Es gilt auch umgekehrt $L(A) \subseteq L(G)$. Wir lassen den Beweis als Übung für den Leser (vgl. Kapitel 3.4. Übung 4 und Kapitel 5). Also gilt

Abbildung A.3: Beweis der Aequivalenz linearer Grammatiken und endlicher Automaten 3