Universität zu Köln
Philosophische Fakultät
Institut für Linguistik, Abt. Sprachliche Informationsverarbeitung

Paradigmenbildung in einem selbstlernenden System

Version 0.41

Magisterarbeit

vorgelegt bei Herrn Prof. Dr. Jürgen Rolshoven

von
Pascal Christoph
Kalscheurer Weg 31
50969 Köln
Matrikelnummer: 3611531
9. Semester
Sprachliche Informationsverarbeitung
Phonetik
Allgemeine Sprachwissenschaft

Köln, im März 2006

Inhaltsverzeichnis

1. Einleitung

»The Ultimate information system will work so well that finding information is as easy as remembering it.«

Gregory B. Newby

In der sogenannten *Informationsgesellschaft* lebend bedarf es Mittel und Techniken, die Inhalte, Informationen und Dokumente verschiedenster Art zu organisieren, um mehr als nur zufälligen Zugriff auf das Gewünschte zu erhalten.[1] Mittlerweile hält nicht mehr nur das weltweite Internet eine unglaubliche Fülle an Dokumenten bereit, auch der heimische Rechner ist längst zur Bibliothek geworden, in der sich zehn Jahre Tageszeitung neben Teilen der Gutenbergbibliothek[2], der Wikipedia[3] und einer Menge anderer Bücher, Aufsätze etc. befinden. Der massive Zuwachs an digitalen Dokumenten lässt für deren Organisation keine manuelle, sondern nur eine automatische Methode zu. Dabei sind zumindest für die textuellen Dokumente neben den bestehenden Werkzeugen von dem Sprachlichen Informationsverarbeiter weitere Werkzeuge entwickelbar, wie das in der vorliegenden Arbeit entwickelte Programm beispielhaft aufzeigt. Eine Möglichkeit läge darin, die Meta-Tags[4] in HTML-Dokumenten um Keywords zu erweitern, die nicht explizit Bestandteile des originären Dokuments sind. Ein mögliches Verfahren dazu ist die Generierung von Paradigmen, denn Worte sind im gleichen Paradigma wenn sie „gegeneinander austauschbar“[5] sind, also z.B. die Wörter *Orange* und *Apfelsine*[6] oder die Wörter *Obst* und z.B. *Orange*[7]. Die auf diese Weise ausgezeichneten Dokumente, in denen etwa im eigentlichen Text stets nur von *Apfelsine* die Rede ist, würden auch dann noch durch eine Suchmaschine auffindbar sein, wenn der Benutzer als Suchbegriff *Orange* eingegeben hätte. Für die Analyse und maschinelle Erzeugung von Wissen ist demnach

1 Mit zunehmender Dokumentenzahl steigt die Unwahrscheinlichkeit des Rechercheerfolges. Ein ausführlicher Artikel zum Thema findet sich unter:

http://de.wikipedia.org/wiki/Dokumentenmanagement (letzter Zugriff: 21.02.2006)

2 http://www.gutenberg.org/ (letzter Zugriff: 21.02.2006)

3 http://de.wikipedia.org/wiki/Hauptseite (letzter Zugriff: 21.02.2006)

4 *Meta-Tags* sind versteckte Elemente auf einer Webseite. Sie enthalten Metadaten über das betreffende Dokument.

5 Diese Definition gilt für die Domäne Linguistik. Es gibt andere Lesarten von *Paradigma*.

6 Der Fachterminus einer solchen Relation lautet *Synonym***.** Dabei hat das Beispiel nur für „die Nordhälfte Deutschlands Gültigkeit und ist in Österreich und der Deutschschweiz als Teutonismus markiert. In Bayern würde der Gebrauch des Wortes Apfelsine einen "Zugereisten" oder Urlauber kennzeichnen.“ Vgl. http://de.wikipedia.org/wiki/Synonym (letzter Zugriff: 08. 01. 2006)

7 Der Fachterminus einer solchen Relation lautet *Hyperonym* (vgl. Kapitel 2.12)**.**

eine Akzentverschiebung in Richtung Semantik[8] erfoderlich.

In jüngster Zeit wird die automatisierte Paradigmenbildung als wichtiger Bestandteil zur Informationsorganisation fokussiert.[9] Dieser Umstand ist insbesondere der taxonomischen Eigenschaft von Paradigmen geschuldet: sowohl Elemente innerhalb eines Paradigmas, als auch Paradigmen als Ganzes (vgl. dazu Schwiebert 2004:9 f.), stehen immer in Relation zueinander.[10] Das XML-Magazin veröffentlichte zu dem Thema Taxonomie und Topic Maps einen Aufsatz von Thomas Bandholtz. Hier beobachtet Bandholtz (2002):

> Internationale Firmen (wie z.B. Accenture, BP, HP, IBM, Microsoft, Nokia, Royal Dutch/Shell, Schlumberger, Siemens, Toyota oder Xerox) wetteifern im Aufbau ihrer Master Classification und weisen terminologisch versierten Mitarbeitern eine Rolle als Taxonomist zu.

Auch in mittelständischen Firmen steigt der Bedarf an Organisation digitaler Dokumente.[11]

Ferner findet sich in dem Artikel von Bandholtz ein Querverweis auf den in *The Bulletin: Seybold News & Views On Electronic Publishing* veröffentlichten Text von Luke Cavanagh, in dem betont wird, dass Taxonomie Managemet eine Schlüsselposition im Content Management erhält. Cavanagh (2002) stellt fest: „The categorization software business is developing as we speak, and the software being created may well be the next big must-buy item in your organization."

Wie der Titel der vorliegeden Arbeit „Paradigmenbildung in einem selbstlernenden System" impliziert, ist zentrales Ziel dieser Arbeit die Entwicklung einer Software, die automatisch Paradigmen auf der Grundlage eines Korpus generiert. Das dazu notwendige linguistische Fundament wird in Kapitel 2 dargestellt. Im Anschluss an Jürgen Rolshovens (2002:3) Feststellung „Die Codierung in der parole ist sehr unvollständig."[12], wird in diesem Kapitel u.a. den Fragen nachgegangen, inwiefern sich diese Unvollständigkeit auf die richtige Bestimmung von Paradigmen auswirkt. Wo liegen die Grenzen der Computabilität von Paradigmen bzw. welche Schwierigkeiten gibt es? Bereits die

8 Die Semantik ist ein Synonym zu *Bedeutungslehre*.

9 Im deutschsprachigen Raum wurde das Projekt des Leipziger Wortschatz entwickelt. Vgl.: http://wortschatz.uni-leipzig.de/ (letzter Zugriff: 21.02.2006)

10 Auch wenn Paradigmen *per se* keine wissenschaftliche Klassifizierung der außersprachlichen Realität darstellen, so organisieren sie doch linguistische Klassifikationen.

11 Vgl. auch: http://www.knowledgebusiness.com/knowledgebusiness/ (letzter Zugriff: 21.02.2006)

12 Der Begriff *parole* ist von Ferdinand de Saussure geprägt und bezeichnet den tatsächlichen Sprachgebrauch, also z.B. Sätze, im Gegensatz zur *langue*, die das allgemeine Regelwerk von Sprache beschreibt, wie etwa die Grammatik.

Geschichte der Sprachwissenschaft liefert darauf eine erste Antwort. Dies aufzuzeigen, ist u.a. die Aufgabe des 2. Kapitels. Zusätzlich liefert das Kapitel 3.3 mit einer Diskussion über die technischen Grenzen und Probleme praxisnahe Einsicht in das weite Feld der strukturellen Semantik.

Es besteht ein weitverbreitetes Interesse der Wirtschaft sowie öffentlicher Institutionen an der Bereitstellung einfach zu bedienender Schnittstellen zwischen Mensch und Maschine. Aus diesem Grund war der Einsatz entsprechender Softwaretechnologien sowie die Konzeption und Programmierung einer „niedrigschwelligen" Schnittstelle ein zentrales Anliegen vorliegender Arbeit. Die Verwendung des Programms soll für den Benutzer möglichst unkompliziert sein.

Das Ziel der Arbeit besteht folglich in der Erzeugung von *deklarativem Wissen* mittels *funktionalen Wissens.*[13] Die maschinelle Analyse eines beliebigen Korpus[14] erzeugt durch einen Algorithmus[15] Daten[16], die in einer Datenbank gespeichert werden.[17] Das auf diese Art generierte sprachliche Wissen, die Zuordnung von Wörtern zu einem Wortparadigma, sollte im Idealfall dem intuitiven Wissen eines Muttersprachlers nicht widersprechen.[18] Ein wichtiges Anliegen ist das in der Datenbank persistierte deklarative Wissen jedem Anwender frei zur Verfügung zu stellen und den Zugriff auf diese Daten möglichst einfach[19] zu halten. Im Kapitel 3 wird das aus der theoretischen Vorarbeit entstandene Programm *PaGe*[20] vorgestellt und seine Handhabung erläutert.

Im letzten Kapitel werden zusammenfassend die Ergebnisse der Arbeit dargelegt sowie offene Fragestellungen und Wünsche aufgezeigt.

Vor dem Übergang zum nächsten Kapitel wird an dieser Stelle die in der Arbeit verwendete Notation erläutert:

Kursiver Text soll das Rezipieren erleichtern. Außer in Kapiteln vorangestellten Zitaten

13 Qualitativ unterscheidet sich deklaratives Wissen vom Output funktionalen Wissens lediglich durch die Latenzzeit. *Latenzzeit* beschreibt den Zeitraum zwischen einer Aktion (hier: der Anfrage, in welchem Paradigma ein Wort liegt) und dem Eintreten einer Reaktion (hier: die Antwort der Zuordnung des Wortes zu einem Wortparadigma).

14 Einschränkungen hinsichtlich des Aufbaus des Korpus werden in Kapitel 3.2 erläutert.

15 Der *Algorithmus*, also ein exakt definierter Handlungsablauf, stellt das funktionale Wissen dar.

16 Die *Daten* stehen für das deklarative Wissen. Sie sind entweder axiomatisch oder (wie im vorliegenden Fall) Produkte der Anwendung anderer Daten auf funktionales Wissens.

17 Siehe hierzu Kapitel 3 und Kapitel 4

18 Ergebnisse des PaGe werden in Kapitel 4 diskutiert.

19 Siehe hierzu den Exkurs in Kapitel 3.1 zur Barrierefreiheit

20 PaGe ist ein aus den beiden Anfangsbuchstaben der Worte *Paradigmen* und *Generator* gebildetes Akronym.

soll der kursive Text einen Fokus erzeugen. Am besten lässt sich das durch eine prosodische Hervorhebung mitlesen. Die Funktion so markierten Textes kann dabei durchaus verschieden sein, sei es um das Wort als Eigennamen, als inhaltlslose Oberflächenerscheinung, ganze Wortgruppen als Zitat oder einfach als inhaltlich *besonders* bedeutsam zu kennzeichnen. Die genaue Funktion sollte sich aus dem Kontext ergeben.

2. Linguistische Fundierung

> *»Der Strukturalismus beschreibt auf Rechnern umsetzbare Algorithmen, die einen Unterschied zwischen dem* Lesen des Weines *und dem* Lesen des Buches *erkennen.«*

In der Sprachwissenschaft existieren viele verschiedene Denkrichtungen. Jedes Modell hat seine eigenen Fachtermini, Axiome und Ansprüche. Dabei werden oftmals gleichlautende Termini verwendet, die aber inhaltlich sehr verschieden sind.[21]

Die vorliegende Arbeit ist aus strukturalistischer Sichtweise geschrieben. Dabei gibt es selbst innerhalb der strukturalistischen Schule mindestens drei Hauptschulen,[22] die sich ihrem Wesen nach voneinander unterscheiden,[23] nämlich die

1. Prager Schule unter Mathesius, Trubetzkoy (1939) und Jakobson und die
2. Kopenhagener Schule unter Louis Hjelmslev (1953/1961) sowie den
3. amerikanischen Strukturalismus, der sich in den Deskriptivismus von Bloomfield (1933) und den Distributionalismus Zellig Harris' (1951/1960) aufspaltet.

Grundsätzlich aber kann der sprachwissenschaftliche Strukturalismus als Weiterentwicklung der Ideen von Ferdinand de Saussure (1916)[24] angesehen werden. Der Strukturalismus[25] bezeichnet demnach

21 Vgl. z.B. die differenten Lesarten des Begriffs *Grammatik* unter deskriptiver, mentalistischer oder normativer Sichtweise.

22 Vgl. Hans-Jürgen Sasse (2003)

23 Letztlich unterscheiden sich sogar die einzelne Autoren selbst über die Jahre in ihren Ansichten und Definitionen. Daher auch z.B. die Ausdrücke „der frühe Chomsky“ oder „der Chomsky der späten 60er Jahre“.

24 Die erste französische Ausgabe erschien 1916. In der Arbeit wird die deutsche Fassung von 1967 verwendet (vgl. Literaturverzeichnis).

25 Synonyme sind strukturale/strukturelle/strukturalistische Sprachwissenschaft.

> [...] eine Gruppe unterschiedlicher sprachwissenschaftlicher Richtungen, die in der ersten Hälfte des 20. Jahrhunderts entwickelt wurden. Wesentliche gemeinsame wissenschaftstheoretische Prämissen sind das Postulat, dass alle linguistischen Aussagen als die Struktur betreffende Aussagen zu formulieren sind (daher der Name) und die Auffassung von Sprache als ein relationales System von Elementen, deren interne Beziehungen zueinander ohne Rückgriff auf psychologische oder geisteswissenschaftliche (z.B. auch historische) Erklärungshilfen beschrieben werden müssen. (Sasse 2003:69)

Die vorliegende Arbeit orientiert sich dabei an jenen Strukturalisten, die sich nicht nur um deskriptive Techniken zur Analyse von Sprachdaten bemühten, sondern daneben eine rigorose Form der Sprachtheorie entwickelten.[26] Da sich diese strukturalistischen Methoden ihrer Form nach an den Naturwissenschaften orientieren, ihre Formalismen und Termini sich also der Logik und der Mathematik entlehnen, sind diese Methoden für den Einsatz in computerlinguistische Modelle prädestiniert. Die zum Einsatz kommenden heuristischen Methoden sind vor allem der Substitutionstest[27] für die Kookkurrenzanalyse und die Gewichtung der Aussagen durch quantitative Relationen, um Distributionsverhältnisse zu ermitteln. Wie diese Heuristiken funktionieren und auf welche Weise sie im PaGe umgesetzt sind, ist in Kapitel 3.3 beschrieben.
In Kapitel 2 wird der Begriffsverortung und der geschichtlichen Auseinandersetzung mit dem Begriff *Paradigma* innerhalb der Linguistik Platz eingeräumt. Dies kann nicht geschehen ohne sich mit der bereits in der Einleitung erwähnten *Semantik* auseinanderzusetzen.

2.1. Semantik

> »*Die Semantik, die Lehre von den Inhalten und Bedeutungen der Wörter und Sätze, ist ein weites Feld.«*
>
> *K.D. Bünting*

Die eingangs umschriebene Lesart von Semantik als *Bedeutungslehre,* die z.B. auch die Beziehung zwischen Sprache und Denken und Welt fasst, muss im Kontext dieser Arbeit

26 Wichtige Vertreter dieser mathematoiden Richtung sind der Behaviorist Louis Hjelmslev und Zellig S. Harris, der Lehrer von Noam Chomsky. Vgl.: Sasse (2003:71)

27 Der Substitutionstest, oder *Austauschtest*, ist ein Mittel der Distributionsanalyse. Der Begriff *Distribution* ist ein Synonym für *Verteilung* und meint die Beschreibung der Elemente aufgrund der anderen sie umgebenden Elemente. Ein Element innerhalb einer Elementenkette wird gegen ein anderes Element ausgetauscht. Die so entstandene Kette wird auf Wohlgeformtheit überprüft, und das Ergebnis trägt zur Definition (lat: de=ab; finis=Grenze, also Definitio=Eingrenzen) des Elements bei.

auf den spezielleren Begriff *Wort*bedeutungslehre eingeengt werden und ist damit abgegrenzt von z.B. emotiver, situativer, expressiver und sozialer Bedeutung. *Semantik* wird hier also als *Wissenschaft von den sprachlichen (vorwiegend lexematischen[28]) Inhalten* gebraucht. Der Begriff war (und ist) in der Linguistik nicht immer derart eingegrenzt verwendet worden. Überlegungen eines zu weit gefassten Semantikbegriffs können bis zum Ausschluss der Semantik aus linguistischen Theorien führen. Auf diese Tatsache wird unter anderem im folgenden Kapitel eingegangen. Auf *Semantik* im allgemeineren, umfassenderen Sinne wird im Unterkapitel 2.1.3 eingegangen.

2.1.1. Genealogische Skizze

> *»Was sich wie vieles andere als gesunder Menschenverstand maskiert, ist in Wirklichkeit äußerst kompliziert und liegt gar nicht weit ab von den Spekulationen antiker und mittelalterlicher Philosophen.«*
>
> *Bloomfield*

Als einer der frühesten Quellennachweise der Reflektion über die Bedeutung von sprachlichen Zeichen dienen uns die Gedanken griechischer Philosophen. Zum Begriff *Bedeutung* schreibt John Lyons einleitend:

> Die scholastischen[29] Philospohen waren wie die Stoiker[30] an der Sprache als Werkzeug zur Analyse der Struktur der Wirklichkeit interessiert. Deshalb war es gerade die Frage der Bedeutung oder der „Signifikation“, der sie das Größte Gewicht beimaßen. (Lyons 1971: 15)

Die Denkrichtung der sich auf Aristoteles berufenden Scholastiker ist die des Objektivismus. Dieser geht davon aus, dass der Verstand als Spiegel der (*einen*) Wirklichkeit zu betrachten ist, und dass linguistische Symbole zu Einheiten und Kategorien in der Welt korrespondieren, sich mithin Sprache, Denken usw. auf eine Logik zurückführen liesse (siehe dazu u.a. Lakoff 1987).
Die Stoiker dagegen

> glaubten jedoch nicht, daß Sprache eine direkte Widerspiegelung der „Natur“ sei. Die meisten unter ihnen waren „Anomalisten“ die eine Entsprechung von Wörtern und Dingen verneinten und auf den unlogischen Aspekten der Sprache bestanden. (Lyons 1975: 12)

28 Ein *Lexem* ist ein Eintrag im Lexikon. Eigenschaften eines Lexems sind u.a. Merkmale der syntaktischen Klassifikation (z.B. Verb oder Nomen).

29 Die Blütezeit der Scholastik lag im 13. Jahrhundert.

30 Der antike Ursprung lag in Athen um 300 vor Christus.

Die hier deutlich gewordenen gänzlich unterschiedlichen Auffassungen wiederspiegeln einen alten, bis in die Gegenwart anhaltenden Disput in der Philosophie.[31]

Die neuere Geschichte der lexikalischen Semantik wurde bis zum Anfang des 20. Jahrhunderts fast gänzlich unter *diachronischem*[32] Aspekt betrachtet.[33] Die Beschreibung der Semantik zu einem gegebenen Zeitpunkt, also die *synchrone* Betrachtung, wurde vor allem 1931 durch Jost Triers Habilitation „Der deutsche Wortschatz im Sinnbezirk des Verstandes. Die Geschichte eines sprachlichen Feldes“ revolutioniert. Zur Entwicklung seiner Lehre erklärt Jost Trier: „Im ganzen der Auffassung fühle ich mich am stärksten verpflichtet FERDINAND DE SAUSSURE, am stärksten verwandt LEO WEISGERBER.“[34] (Trier 1931: 11 zit. nach Geckeler 1971: 89; Hervorhebung im Original) Doch war Trier zweifelsohne nicht der erste Denker dieser Idee. Eugenio Coseriu macht auf Karl Wilhlem L. Heyse aufmerksam, dessen Buch „System der Sprachwissenschaft“ in Berlin 1856 posthum herausgegeben wurde. Heyse unternimmt darin eine „beinahe vollkommene strukturelle, wenn auch mit anderen Absichten durchgeführte Inhaltsanalyse [des Wortfeldes ‚Schall‘].“[35] (Coseriu 1967:489 f. zit. nach Geckeler 1971: 88)

Ein Grund für die (trotz Trier) weitgehende Ausklammerung der Semantik aus der Linguistik bis 1950 war sicherlich die Datenflut, die hätte verarbeitet werden müssen. Auf phonologischer Ebene ist die Anzahl[36] der Elemente, und damit die Anzahl möglicher

31 Aus den sogenannten *Naturalisten* und den ihnen entgegengesetzt denkenen *Konventionalisten* entwickelten sich später die sogennanten *Analogisten* und *Anomalisten* (vgl.: Lyons 1971:4 ff.).

32 Der von Saussure geprägte Begriff *diachron* stammt aus dem Griechischen *(dia*[δια]*=durch, hindurch* und chronos[χρονος]=Zeit). Ein diachronischer Prozess ist als ein Prozess der über einen längeren Zeitabschnitt hinweg stattfindet (z.B. der Wandel eines mittelhochdeutschen zum neuhochdeutschen Wort).

33 Wobei „die Semantik schon im Rahmen der historisch ausgerichteten Sprachwissenschaft des 19. und des frühen 20. Jahrdunderts als letzte der klar umrissenen Disziplinen entstand.“ (Geckeler 1971:26 f.)

34 Diesem Selbstzeugnis fügt Geckeler Wilhelm von Humboldt als einen weiteren Vordenker hinzu, da Weisgerber schließlich „in keiner seiner Arbeiten einen Zweifel über die entscheidende Rolle [aufkommen läßt], die Humboldts Ideen für seine sprachwissenschaftliche Konzeption spiele.“ (Geckeler 1971:89)

35 Die explizite Bezeichnung Wortfeld findet sich in Heyses Werk noch nicht. Spätestens 1910 findet der Feldbegriff Verwendung. Geckeler (1971:88) verweist hierbei auf das im Jahr 1910 erschiene Buch von Adolf Stöhr „Lehrbuch der Logik in psychologisierender Darstellung“.

36 Natürliche Sprachen haben zwischen 13 und etwa 80 Phoneme. So gibt es im Sobei, einer ozeanischen Sprache, 15 Konsonanten und 5 Vokale. Das deutsche Phoneminventar hat etwa die Größe von 40.

Distributionen[37], relativ beschränkt.[38] Auf der Wortebene ist die Anzahl der differenten Elemente um ein Vielfaches größer – und zudem dynamischer als auf phonologischer Ebene[39]. Dem amerikanischen Strukturalisten Leonard Bloomfield, dem oftmals Bedeutungs*feindlichkeit* vorgeworfen wurde, beschreibt dies so:

> In order to give a scientifically accurate definition of meaning of every form of a language, we should have to have a scientifically accurate knowledge of everything in the speakers' world. The actual[40] extent of human knowledge is very small, compared to this. (Bloomfield 1965: 139 f.)

Erst seit Beginn der 50er Jahre des 20. Jahrhunderts regt sich großes Interesse an der Semantik. Dazu trug, so eine These der vorliegenden Arbeit, der nach Alan Turing benannte Turing-Test bei:

> Der Turing-Test wurde 1950 von Alan Turing vorgeschlagen, um die Frage „Können Maschinen denken?“ zu entscheiden. Der aus der Anfangszeit des Informatik-Teilbereichs Künstliche Intelligenz stammende und seither legendäre Test trug dazu bei, den alten Mythos von der denkenden Maschine für das Computerzeitalter neu zu beleben. [41]

Die digitale, maschinelle Sprachdatenverarbeitung, deren praktischer Ursprung im während des zweiten Weltkrieges speziell zur Dechiffrierung von militärischen Nachrichten der Deutschen gebauten Colossus gesehen werden kann,[42] war zwar noch im Anfangsstadium, doch die Möglickeiten und Entwicklungen wurden durch theoretische Überlegungen, wie etwa Turings, vorweggenommen und beflügelte sicherlich Theoretiker anderer Disziplinen.

Wenn auch der Zusammenhang von *Informatik* und *Semantik* u.a. in Geckeler (1971) nicht genannt wird, scheint es doch mehr als zufällig, dass

> [i]m Jahre 1951 [...] ein Aufsatz von E.A. Nida mit dem für die damalige Situation des nordamerikanischen Strukturalismus ungewöhnlichen Titel: „A System for the Description of Semantic Elements“ [erschien]. Diese Arbeit schlägt zum ersten Mal eine umfassende und kohärente Terminologie für die Beschreibung der Bedeutung

37 Die „ungünstigste“ Anzahl der möglichen Distributionen errechnet sich aus der Anzahl der Elemente potenziert mit der „Ordnung“ (oder: Stelligkeit) der Distributionsanalyse. Die wichtigsten Aussagen sind schon mit Analysen zweiter Ordnung, also der Beschreibung eines Elementes und seines direkten Nachfolgers, zu erreichen. Beispielsweise gilt für das Deutsche: Dem /p/ kann ein /f/ folgen, aber kein /t/.

38 Der Strukturalismus hat sich anfangs fast gänzlich auf den Bereich Phonologie konzentriert.

39 Sprache ist ein offenes, lebendiges System und konstituiert ständig neue Worte. Mit Ludwig Wittgensteins Worten „Sprache ist eine Lebensform“.

40 Diese Äußerung stammt aus dem Jahre 1933.

41 http://de.wikipedia.org/wiki/Turing-Test (letzter Zugriff: 21.02.2006)

42 http://de.wikipedia.org/wiki/Colossus (letzter Zugriff: 21.02.2006)

(„meaning“) vor. (Geckeler 1971: 32)

Die Wiederentdeckung der Semantik innerhalb die Linguistik ist, so eine These dieser Arbeit, dem Umstand der Erfindung der maschinellen Datenverarbeitung geschuldet.

2.1.2. Definitionen

«Aliquid stat pro aliquo.«[43]

Die *lexikalische Semantik* beschäftigt sich mit der Bedeutung von Wörtern wie auch der inneren Strukturierung des Wortschatzes insgesamt. Die Strukturierung besteht in der Darstellung der Relationen der Wörter zu anderen Wörtern und zeigt damit Verwandtschaft zur Kategorienlehre des Aristoteles sowie zu deren Weiterentwicklung, der Prototypentheorie Eleanor Roschs und auch der Kategorialsemantik.[44] Als Bezeichnung solch einer Relation hat sich der Begriff Wortfeld etabliert. Der Begriff *Wortfeld* wurde vor allem von Trier und Weissgerber geprägt und suggeriert eine Sammlung von Wörtern, die durch ein gemeinsames semantisches Merkmal gekennzeichnet sind. Trier geht von einer Gliederung des Wortschatzes aus, bei der sich die Glieder, also die Wörter, gegenseitig voneinander abgrenzen und so ihre Bedeutung durch ihre Stellung innerhalb dieses Systems erhalten. Bei Saussure findet sich im Kapitel „Der sprachliche Wert, § 2“ ein Grundgedanke der Feldforscher:

> Innerhalb einer und derselben Sprache begrenzen sich gegenseitig alle Worte, welche verwandte Vorstellungen ausdrücken: Synonyma wie *denken, meinen, glauben* haben ihren besonderen Wert nur durch ihre Gegenüberstellung; wenn *meinen* nicht vorhanden wäre, würde sein ganzer Inhalt seinen Konkurrenten zufallen. (Saussure 1967:138)

Vor der Definition komplexerer Begriffe ist es sinnvoll, grundlegende Begriffe[45] zu erklären. Diese entstammen der Semiotik[46]:

- **signe** (oder sprachliche Zeichen): im Sinne de Saussures bestehend aus:
    - **signifié** (oder *Signifikat*): entspricht dem *Bezeichneten,* ist also die *begriffliche Inhaltsseite* des signe

43 Etwas, das für etwas anderes steht.

44 Für ein Beispiel einer solchen Hierarchie siehe Abbildung 6

45 Die fünf unten dargestellten Begriffe finden ihre Entsprechung bei den Stoikern sowie bei dem Scholastiker Augustinus (vgl.: Geckeler 1971:79f).

46 Ein Synonym zu *Semiologie*

- **signifiant** (oder *Signifikant*): entspricht dem *Bezeichnenden,* ist also die *Ausdrucksseite* des signe
- **valeur** (oder *Wert*): entspricht der strukturellen Position in Beziehung zu anderen signe
- **chose** (oder *aussersprachliche Realität*): das *signe* steht als Einheit einer außersprachlichen Realität gegenüber

Geckeler beschreibt die Bestandteile des Wortfeldes nach Coseriu wie folgt:

> Im Hinblick darauf, daß Lexeme in Wortfeldern funktionieren, definieren wir die „Bedeutung“ (wir denken besonders an die lexikalische Bedeutung) als reine Beziehungen auf der Inhaltsebene, als Verhältnisse von „signifiés“ zueinander. (Geckeler 1971: 79 f.)

Begriffe, die die Beziehungen von Elementen innerhalb des Wortfeldes bezeichnen, sind:

- **Synonyme**: Wörter, deren s*ignifié (annähernd)* gleich ist, z.B. *Orange* und *Apfelsine*
- **Antonyme**: Wörter, deren *signifié* gegensätzlich ist, z.B. *Tag* und *Nacht*
- **Hyponyme**: Wörter, deren *signifié* in anderen inkludiert ist, z.B. *Tulpe* und *Blume*
    - **superordinierte** Hyponyme: Wörter, deren *signifié* andere inkludieren[47]
- **Hyperonyme**: Synonym zu *superordinierte Hyponyme*
- **Kohyponyme**: Wörter, deren *signifié* in ähnlicher Relation zu einem anderen steht, deren s*ignifié* untereinander aber in verschiedenen Relationen stehen, z.B. die Worte *purpurrot* und *zinnoberrot* in Relation zu *rot*

2.1.3. Problematik

»MEANING is one of the most ambigous and most controversial terms in the theory of language.«

Stephen Ullmann

In allen wissenschaftlichen Disziplinen, ausgenommen der Mathematik, sind Termini durch unterschiedliche Bedeutungsrichtungen geprägt, abhängig von den jeweiligen wissenschaftlichen Diskursen und historischer Situierung. Die signifiés der jeweiligen Begriffe sind stark ko(n)textabhängig. Dies wird im Folgenden anhand von einigen

47 Der Begriff entstammt dem Buch Lyons und lässt die dichotone Art der Definitionen Saussures vermissen. So existiert zwar ein superordiniertes Hyponym, aber kein subordiniertes (da Hyponomie an sich schon eine subordinierte Relation ist). Lyons zieht diese nicht stringente Definition der Verwendung des Begriffs *Hyperonyme „wegen der akustischen Verwechslungsmöglichkeiten“* vor. (Lyons 1975:465)

Beispielen erläutert. Zudem wird die für Computerlinguisten interessante Heuristik der Generierung von Semantik durch Transkription skizziert. Außerdem wird in einem Exkurs auf den aktuellen Diskurs im Bereich Künstliche Intelligenz eingegangen.

Im Folgenden wird der konträren Interpretationen des saussureschen Terminus *signifié* nachgegangen. Nach Martin Kuester (2001: 584) ist

> [d]as Signifikat [...] eine psychische Vorstellung, die für Saussure nur auf der Ebene des Denkens existiert und sich auf den materiellen Referenten in der Wirklichkeit bezieht.

Diese Definition von Kuester, verknüpft mit der Definition Coserius vom Wortfeld (vgl. Kapitel 2.1.2) bedeutet, dass der im PaGe verwendete Algorithmus mit *psychischen Vorstellungen* arbeiten müsste. Dies ist nach dem gegenwärtigen Forschungsstand nicht möglich. Geckeler schließt die sogenannten „Konnotationen", die die Verbindung zu den psychischen Vorstellungen herstellen, aus dem Sprachsystem explizit aus:[48]

> Konnotationen gehören, unserer Meinung nach, nicht zur Ebene des Sprachsystems, und zwar nicht allein deshalb, weil ihnen die intersubjektive Gültigkeit fehlt, sondern weil sie nicht die distinktiven Funktionen betreffen. (Geckeler 1971: 78)[49]

Im Gegensatz zu Geckeler unterstützt Daniel Chandler den Standpunkt, dass den *signifiés* Konnotationen zugrunde liegen:[50]

> [...] [*P*]*aradigmatic analysis* seeks to identify the various paradigms (or pre-existing sets of signifiers) which underlie the manifest content of texts. This aspect of structural analysis involves a consideration of the positive or negative connotations of each signifier [...].

Festzuhalten bleibt folglich die ambige Interpretation des saussureschen Terminus.[51]
Für computerlinguistisches Arbeiten bedarf es mathematisch exakt definierbarer Methoden und Begriffe. Für die Generierung von Semantik werden im Folgenden

48 Unter anderem Halliday (1966) versteht den Begriff *Konnotation* genau im Gegegenteil als grundlegend für strukturelle Bedeutung: „Die auf dem konnotativen Bedeutungsaspekt beruhende strukturelle Bedeutung umfaßt demgegenüber sämtliche sprachinternen Relationen (wie etwa Synonymie, Hyperonymie u.a.), welche die Position eines Zeichens innerhalb des Sprachsystems bedingen." (Mehler 2004)

49 Das 3. Kapitel beschäftigt sich u.a. auch mit der Glossematik Hjelmslevs und zeigt auf, dass Konnotationen sogar in Bloomfields Hauptwerk *Language* einen beachtliche Rolle spielen.

50 Das Buch befindet sich als Online-Version unter: http://www.aber.ac.uk/media/Documents/S4B/sem05.html (vgl. Kapitel *Paradigmatic Analysis)* (letzter Zugriff: 28.02.2006). Chandler beruft sich explizit auf Saussure.

51 Wenn sich diese verschiedenen Interpretationen auch ofttmals auf die verschiedenen Kontexte *Linguistik* und *Kulturwissenschaft* zurückführen lassen, kann dies keine genügende „Erklärung" sein.

Herangehensweisen an diese Problematik dargestellt.

Für Saussure entsteht die Bedeutung eines Zeichens durch die Opposition zu anderen Zeichen. Er spricht daher von der Negativität des - in sich bedeutungslosen - Zeichens ("nullité du sème en soi"). Diesen systemischen Aspekt der differenzlogischen Bestimmung von Bedeutung bezeichnet Saussure als *valeur*, als *systemischen Wert* des Zeichens, als *Bedeutung im System*.[52] Geckeler schreibt über den Begriff *valeur* lediglich in einer Fussnote zur Definition des Begriffs *Wortfeld:* „F. de Saussures Begriff der „valeur“ scheint sich dagegen aus den Verhältnissen der sprachlichen Zeichen zueinander zu bestimmen.“ (Geckeler 1971:80, Anm. 149)

Der Wert ist einerseits Bestandteil der Bedeutung, andererseits kann er auch die Bedeutung beeinflussen (Saussure 1967:136). Unter der Verwendung des *valeur,* so eine These dieser Arbeit*,* könnte ein *Wert*feld erzeugt werden, dass bar jeder *psychischen Konnotation* und *ohne jede Interpretation* rein maschinell erzeugt werden kann.[53]

Als ein weiteres Verfahren zur maschinellen Generierung von Semantik kann die Heuristik der Transkription verwendet werden. Transkription, also das „Überschreiben, Übertragen“, kann als Überbegriff zur Bedeutungsgewinnung gelten. Dabei beobachtet George Lakoff : „Lewis is right about the inadequacy of doing semantics by translating one set of symbols to another.“ (Lakoff 1987:205)

Ein Bedeutungsgewinn ist durch einfaches Ersetzen eines Symbols durch ein anderes, gleichbedeutendes (bzw. zuerst inhaltsleerem) Symbol nicht zu erreichen. Werden allerdings Symbole unterschiedlicher signifiés oder valeur in Beziehung gesetzt, verhält es sich anders. Ludwig Jäger erläutert zur Generierung von Semantik:

> Der Begriff des Transkribierens, der als ein heuristischer und operativ zu erprobender Leitbegriff medienwissenschaftlichen Forschens angesehen werden kann, steht dabei für ein Verfahren wechselseitiger intermedialer *Um*, *Ein*- und *Über*-Schreibungen, das als eine basale Strategie für die Generierung kultureller Semantik zu fungieren scheint [...].
>
> Bedeutungsgenerierung [ist] die wechselseitige Bezugnahme (...) symbolischer Mittel desselben Systems [(gemeint ist die Sprache)] aufeinander (...). Transkribieren in Bezug auf die Semantik natürlicher Sprachen [ist] ein *intra*mediales Verfahren, das [...] ihre [der Sprachen] Eigenschaft nutzt, mit Sprache über Sprache zu kommunizieren und so den Verwendungssinn von Äußerungen zu erschließen [...]. (Jäger 2004:71 f.)[54]

52 Vergleiche dazu Saussure (1967), Kapitel 4 „Der sprachliche Wert“.

53 In der Konsequenz wäre das Wertfeld das, was hier als Wortfeld bezeichnet wird.

54 Transkriptive Verfahren unter intra- und intermedialer Bezugnahmen sind nach Jäger z.B. Konkordanzen, Glossare, Register, Lexika, Enzyklopädien, Datenbanken und Suchmaschinen.

In diesem Sinne hat Ross Quillian im Jahre 1968 die auf Otto Selz zurückgehende Idee des *semantischen Netzwerks* in LISP umgesetzt. Zum Beispiel wird darin einem Wortsymbol [*Table*] unter anderem folgende Attribute zugeschrieben:
superordinate: FURNITURE
part: LEGS
Philip Johnson-Laird formuliert dazu: „By itself, this Symbol [(Table)] is meaningless, but it appears to become more meaningful if several labelled associations are linked to it." (Johnson-Laird 1993:328)
Weshalb sollte ein bedeutungsloses Symbol an Bedeutung gewinnen, wenn es mit weiteren bedeutungslosen Symbole vernetzt ist? Die (strukturalistische) Antwort darauf kann nur sein:
der Bedeutungsgewinn ist die so entstandene *Relation zwischen den untereinander verknüpften Symbolen*.

Ein weiterer Aspekt der Problematik ist, dass sich Algorithmen stets in abgeschlossenen Systemen bewegen. Dies steht im schroffen Gegensatz zu Menschen, die „gödelisieren"[55], d.h. ihre Gedanken bewegen sich über definierte Systemgrenzen hinweg. Das geschieht oft unbewusst. Dabei stellt Quine[56] fest, dass diese Systeme insgesamt wieder ein geschlossenes System bilden: „Sätze dürfen sich nicht einzeln, sondern als geschlossene Formation dem Erfahrungstest stellen." (zit. nach Putnam 1999:35)
Wenn beispielsweise die gleichen signifiants in unterschiedlichen Ko(n)texten stehen, so sind ihre signifiés unterschiedlich. Das Problem, das daraus erwachsen kann, nämlich eine stark vereinfachte Sicht der Dinge, ist nicht die Tatsache an sich, sondern das dies unbewusst geschieht. Der Logiker Hilary Putnam macht diese mit einem Beispiel deutlich: Wenn bekannt ist, dass der Dieb durch ein Fenster eingestiegen ist, und außerdem der Boden vor dem Fenster matschig sei, so wird daraus „gefolgert", dass es in dem Matsch Fußabdrücke gibt. Dies ist jedoch keine *logische Folgerung* aus den dargelegten Fakten, denn es wurde eine unausgesprochene Hilfshypothese verwendet, die besagt: *Wenn der Dieb durch dieses Fenster eingestiegen ist, ist er über den Boden gegangen, um zu dem Fenster zu gelangen*, und außerdem kommen noch allgemeine Informationen hinzu. Sagt ein Zeuge nun aus: „Nein, er ist auf Stelzen gegangen", so wird damit gerechnet, anstelle der Fußabdrücke andere geformte Vertiefungen im Matsch zu

55 Für eine nähere Erläuterung siehe Hofstadter (2001)

56 Willard Van Orman Quine ist einer der einflussreichsten amerikanischen Philosophen und Logiker des 20. Jahrhunderts.

finden.

> Es ist eben die geschlossene Formation der Aussagen, die Erfahrungsbedeutung hat, und diese Bedeutung ist nicht die bloße Summe der Erfahrungsbedeutungen der Einzelaussagen. (Putnam 1999:35)

Exkurs: Künstliche Intelligenz und Automatenmenschen

Immer dann, wenn der Semantikbegriff sehr eingeschränkt verwendet wird, entstehen Theorien für die Übertragbarkeit des „Denkens" auf Maschinen. So stellt Hillary Putnam in den 1950er Jahren die These auf, der Computer sei das adäquate Modell für das Bewusstsein.[57] In *Repräsentation und Realität*[58] versucht er zu beweisen, dass seine These des sogenannten *Funktionalismus* nicht stimmen kann und schlussfolgert: „Kurz, wenn der Funktionalismus zuträfe, würde er den Behaviourismus implizieren!" (Putnam 1999:218)

Viele Forscher der künstlichen Intelligenz gehen dennoch soweit zu behaupten, dass das Denken und Fühlen eines Menschen in einen Automaten programmierbar sei.[59]

Renommierte Neuro-Wissenschaftler proklamieren in einem Manifest in der vom Spektrum Verlag herausgegebenen Zeitschrift „Gehirn & Geist" die Programmierbarkeit des Menschen: „Geist und Bewusstsein – wie einzigartig sie von uns auch empfunden werden – fügen sich also in das Naturgeschehen ein und übersteigen es nicht."[60]

So wichtig und nützlich die Erforschung der Intelligenz auch ist, so gefährlich ist sie im Hinblick auf das von ihr induzierte Menschenbild. Joseph Weizenbaum dazu:

> Daß die *Artificial-Intelligence*-Elite glaubt, Gefühle wie Liebe, Kummer, Freude, Trauer und alles, was die menschliche Seele mit Gefühlen und Emotionen aufwühlt, ließen sich einfach mir nichts dir nichts in einen Maschinenartefakt mit Computergehirn transferieren, zeigt, wie mir scheint, eine Verachtung für das Leben, eine Verleugnung ihrer eigenen menschlichen Erfahrung, um es vorsichtig auszudrücken. (Weizenbaum 2001: 42)

Zusammenfassend lässt sich festhalten:

Bei aller Beschäftigung mit Semantik, die ab Mitte des 20. Jahrhunderts und im Kontext digitaler Rechner eine Renaissance erlebte, wird oft vergessen, dass letztlich nur der Mensch durch seine Interpretation Symbolen, Relationen oder Netzwerken usw. *Sinn* zuspricht: ohne einen interpretierenden Menschen wird *Bedeutung* bedeutungslos.

---

57 Die Theorie erhielt den Namen *Funktionalismus*

58 Der Beweis erschien das erste Mal 1981 in „Reason, Truth, and History" und wurde bekannt als *Putnams Theorem*.

59 Ähnlich der Gestalt *Datas* aus der bekannten Science Fiction Serie *Star Trek – The Next Generation* und ähnlch dem Androiden aus dem Film *Bladerunner* (der auf dem Roman „Do Androids Dream of Electric Sheep?" von Philip K. Dick beruht).

60 http://www.gehirnundgeist.de/blatt/det_gg_manifest (letzter Zugriff: 28.02.2006)

2.2. Paradigma

»Äpfel und Birnen lassen sich vergleichen.«
Abwandlung eines alten deutschen Sprichwortes

Elemente eines linguistischen Paradigmas müssen sich in der gleichen lexikalischen Klasse befinden – dies wird in Kapitel 2.2.2 näher beleuchtet. Zudem ist nach Coseriu „ein lexikalisches Paradigma in struktureller Hinsicht ein Wortfeld“ (Geckeler 1971: 192). Diese Einordnung gilt auch für die vorliegende Arbeit. Einige Grundbegriffe des Wortfeldes wurden in Kapitel 2.1.2 bereits definiert. Dort und auch im einleitenden Kapitel dieser Arbeit wurden die Leistungen des Wortfeldes skizziert. Im folgenden wird vor allem die Erzeugung des Wortfeldes und die Zuordnung von Worten zu lexikalischen Klassen beschrieben.

2.2.1. Genealogische Skizze

Die griechischen *Analogisten* verwendeten den Begriff *Paradigma*[61] oft, allerdings generell als Begriff vom *Modell* gebrauchend, also als etwas, das Vorbildcharakter hat. Ferdinand de Saussure, der die aktuelle linguistische Definition von *Paradigma* geprägt hat, definiert den Begriff als Gegenüberstellung zur syntaktischen Ebene.[62]
Bei Plato (ca. 429-347 v. Chr.) ist der Unterschied zwischen Nomina und Verb - und damit die Kategorisierung von Worten zu lexikalischen Klassen – dokumentiert. (vgl.: Lyons 1975: 11). Die lexikalischen Klassen gehören zur *primären paradigmatischen* Struktur, welche wiederum der *lexematischen* Struktur subsumiert ist.[63]
Van Zaanen diskutiert die Substituierbarkeit von syntaktisch differenten, funktional aber äquivalenten Wörtern und schlägt daher vor, zwischen funktionalem und syntaktischem Typ einer linguistischen Einheit zu unterscheiden (vgl. dazu van Zaanen 1999: 9 nach Schwiebert 2004: 9).

61 Paradigma: griechisch: παράδειγμα [parádeigma] (*para* = neben und *deiknynai* = zeigen, begreiflich machen), Modell, Muster, Beispiel

62 Zur näheren Erläuterung siehe das folgende Kapitel 2.2.2

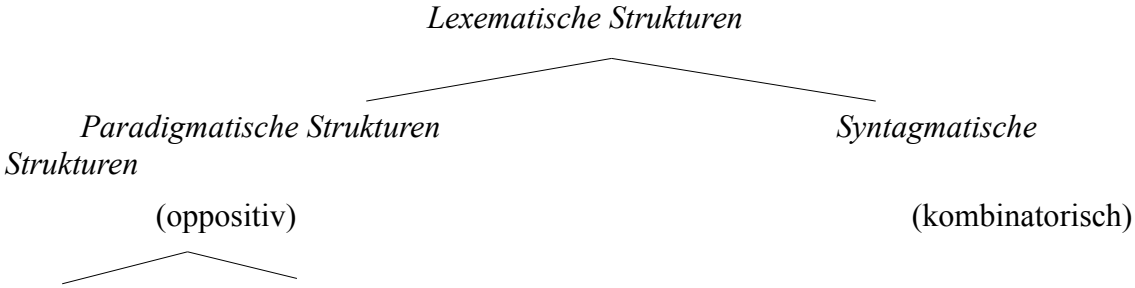
63 Nach Coseriu (1968)

2.2.2. Definitionen

»You shall know a word by the company it keeps.«

John R. Firth

Zu Beginn der Definitionen soll ein Schaubild die Gliederung einiger der verwendeter Begriffe sichtbar machen. Die Details der Syntagmatischen Strukturen (rechte Seite in Abbildung 1) sind nur der Vollständigkeit halber aufgeführt. Das Schaubild ist dem von Geckeler (1971: 191) nachgezeichnet:



Primäre Strukturen	*Sekundäre Strukturen*	
- Wortfeld	- Modifizierung	- Affinität
- Lexikalische Klasse	- Entwicklung	- Selektion
	- Komposition	- Implikation

Abbildung 1: Lexematische Strukturen

Das *Paradigma* ergibt sich aus bestimmten gemeinsamen Merkmalen von Lexemen. Ein Paradigma kann als eine Zuordnung von Kategorisierungen zu Wortformen betrachtet werden. Hierzu erläutert Bernd Wiese (1999) in *Linguistik online*:

> Lexikalische Wörter können als Paare aus Paradigmen und Wortbedeutungen aufgefaßt werden. Wortparadigmen sollen also die nicht-semantischen Komponenten von Wörtern sein.

Im Gegensatz dazu *sind* nach Coseriu semantische Merkmale *Teil* des Paradigmas, insofern als das Wortfeld den paradigmatischen Strukturen subsumiert ist, und Wortfeld

als eine Sammlung von Wörtern mit gemeinsamen semantischem Merkmal definiert ist.[64]

In dieser Arbeit soll die Synthese aus diesen beiden Definitionen die Arbeitsgrundlage sein: Wörter innerhalb eines Wortparadigmas müssen sowohl den gleichen funktionalen Kategorien angehören als auch ein gemeinsames, sprachsystemimmanentes semantisches[65] Merkmal haben.

Ein Paradigma wird durch distributionelle Tests ermittelt. PaGe verwendet ausschließlich den Substitutionstest als Kriterium für Akzeptanz. Die sogenannten distributionellen Tests gehen explizit auf Harris zurück. Bereits bei de Saussure finden sich heirzu grundlegende Ansätze: In der strukturellen Sprachwissenschaft saussurescher Prägung sind die Elemente der Sprache, seien es nun beispielsweise Buchstaben als Elemente der Wörter oder auch Wörter als Elemente von Sätzen[66], wie in einem kartesianischen Koordinatensystem angeordnet. Die Anreihungen der Elemente auf der horizontalen X-Achse werden von de Saussure (1967: 147) als *Syntagmen*[67] bezeichnet:

> Einerseits gehen die Worte infolge ihrer Verkettung beim Ablauf irgendwelcher Aussagen Beziehungen unter sich ein, die auf dem linearen Charakter der Sprache beruhen [...]. [D]iese Kombinationen, deren Grundlage die Ausdehnung ist, können Anreihungen oder *Syntagmen* genannt werden.

Die Anreihungen der Elemente auf der vertikalen Y-Achse bezeichnet er als *assoziative Beziehungen.* Im weiteren Verlauf der Arbeit wird der gebräuchlichere, sinnäquivalente Begriff *paradigmatische Relation*[68] verwendet:

> Andererseits aber assozieren sich außerhalb des gesprochenen Satzes die Wörter, die irgend etwas unter sich gemeinsam haben, im Gedächtnis, und so bilden sich Gruppen, innerhalb deren sehr verschiedene Beziehungen herrschen. (Saussure 1967: 147)

Die Assoziationen sind semantischer Art. Folgendes Beispiel soll dies verdeutlichen:

---

64 Vergleiche Kapitel 2.1.2 sowie Geckeler (1971:192 ff)

65 Semantik ist im weiteren als *Semantik*, *frei von Konnotationen* gemeint.

66 Es existieren weitere Ebenen wie z.B. die *Phrase*.

67 *Syntagma*, griechisch: σύνταγμα [sʏn'takma] = Zusammengesetztes. Ries (1894:84) verwendete noch den sinngleichen Begriff *Wortgefüge* (vgl. Vater 1996:105)**.** Der Begriff *Syntax* (griechisch, σύνταξις [sʏn'taksɪs] := die Zusammenstellung) bezeichnet die *Lehre von den Syntagmen.*

68 Ein *Paradigma* ist also eine vollständige Sammlung von in paradigmatischer Relation stehender Worte.

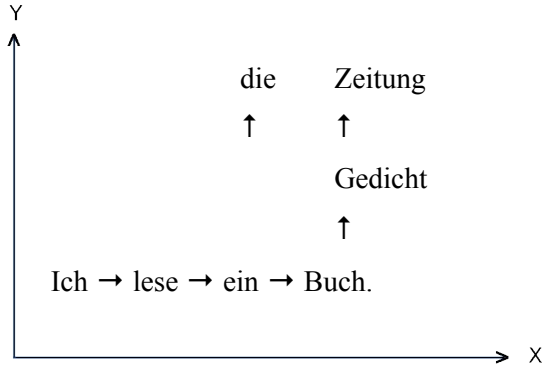


Abbildung 2: Kartesianisches Koordinatensystem

Die Wörter *Zeitung* und *Buch* gehören zwar zum gleichen Wortfeld, da sie aber nicht einfach gegeneinander ausgetauscht werden können, wenn sie in ein Syntagma höherer Ordnung eingeordnet werden sollen, bilden sie kein gemeinsames Paradigma:

* Ich lese ein *Zeitung*.

Der syntagmatischen Regeln wegen muss der bestimmende Artikel, der zum Kasus von *Zeitung* passt, mitsubstituiert werden:

Ich lese *eine Zeitung*.

Das Syntagma *eine Zeitung* wäre damit im gleichen Paradigma wie das Syntagma *ein Buch*. Es ist wichtig hervorzuheben, dass der korrekte deutsche Satz:

Ich lese Wein.

nicht impliziert, dass *Wein* im gleichen Paradigma wie *Gedicht* und *Buch* liegt. Denn semantisch liegt *Wein* zu Schriftstücken sehr weit entfernt.[69] Die Oberflächengleichheit der Wörter darf nicht darüber hinwegtäuschen, dass dem signifiant *lesen* unterschiedliche signifiés zugrunde liegen. Auf welche Weise soll ein selbstlernendes Sytem, ein Algorithmus, erkennen, dass das Lesen des *Weines* ein anderes Lesen ist als das des *Buches*? Der Strukturalismus kann darauf nur antworten: distributionell. Diese strukturalistische These lässt sich in Anlehnung an Schwiebert (2004) als *Wenn-Dann Regel* formulieren:

> *Wenn ein in den Kotexten[70] K stehendes Syntagma S1 hinreichend oft[71] durch ein Syntagma S2 ausgetauscht werden kann, dann stehen S1 und S2*

69 Auch wenn in beiden bekanntlich Wahrheit zu finden ist

70 Kotext wird hier im Gegensatz zu (dem oft synonym verwendeten Wort) *Kontext* verwendet: das zuerst Genannte bezieht sich auf den text*internen* Zusammenhang, also auf den Text in unmittelbarer Umgebung, während letzeres als Text*externes*, also als Weltwissen, oder lediglich als Synonym von *Zusammenhang* verstanden wird.

71 Es ist nicht notwendig, dass zwei differente linguistische Einheiten jeweils in *allen* Kotexten der anderen Einheit vorkommen müssen, um demselben Paradigma zugeordnet werden zu können. Wie hoch der Schwellwert sein muss, um tatsächlich von einer paradigmatischen Relation sprechen zu können, bleibt an dieser Stelle offen (vgl. dazu auch das nächste Kapitel).

in paradigmatischer Relation.

Die Distributionen, hier also die Beschreibung der Worte aufgrund anderer sie umgebender Worte, definieren die Wortbedeutung – dadurch ist der Begriff *Wortbedeutung* genau im Sinne Wittgensteins (1953) definiert, nämlich als „Bedeutung im Gebrauch“. Einzig mit der Kotextanalyse, genauer mit der Kookkurrenzanalyse, werden also Wortfelder generiert:

> Da die syntagmatischen und paradigmatischen Dimensionen voneinander abhängen, können latente paradigmatische Relationen durch die Berechnung der manifesten syntagmatischen Ketten zum Ausdruck gebracht werden.
>
> (Buyko 2005: 2)

Auf das Beispiel der zwei Lesarten von *lesen* bezogen würde ein Algorithmus durch Kookkurrenzanalyse des Korpus festhalten, dass *lesen* einmal im Kotext *Wein*, der sich wiederum im Kotext von u.a. *trinken* befindet, und einmal in Kotext von *Büchern*, die u.a. *ausgeliehen*, niemals aber *getrunken* werden, steht.[72] Dieser Umstand kann als ein Indiz – weitere ließen sich finden - für die zwei verschiedenen Lesarten genommen werden.[73]
Die Kookkurrenzanalyse wird in Kapitel 4 eingehender beschrieben. Das Ergebnis einer solchen Analyse könnte etwa so aussehen:

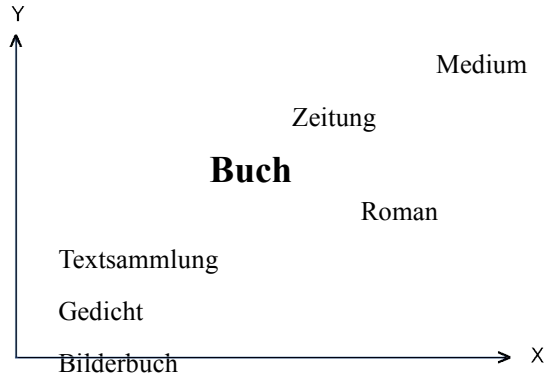


Abbildung 3: Wortfeld von Buch[74]

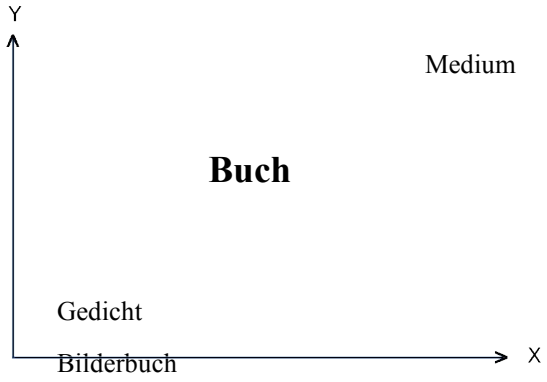


Abbildung 4: Paradigma von Buch

Die X-Achse stellt die relative Größe des Feldes der Worte dar,[75] auf der Y-Achse ist die

72 Kurz: Die Wortbedeutung von *Buch* hat einen sehr großen Abstand zu der Wortbedeutung von *Wein*.

73 Tasächlich bedarf es dazu immenser Rechenschritte. Auch aus diesem Grund ist die Disambiguierung nicht Bestandteil des PaGe geworden.

74 Dieses Ergebnis kann nur Folge einer Analyse sein, die entweder ein artikelfreies Korpus zur Grundlage hätte (durch Verwendung von Stoppwörtern), oder die z.B. nur den rechten Kotext des Wortes berücksichtigt hätte (davon ausgehend, dass die Artikel immer links vom Wort stehen) oder die eine Sprache zur Grundlage hätte, die Genus durch Flektion am Wort markiert.

75 In diesem Beispiel ist die Größe eines Feldes durch die Häufigkeit des Auftretens differenter Okkurrenzen determiniert.

Ordinierung vermerkt. Ein Wort steht zu *Buch* in Superordination, wenn es einen größeren Y-Wert hat, und ist subordiniert, wenn sein Y-Wert kleiner als der von Buch ist.[76] Das Wort *Bilderbuch* hat also ein kleines Feld[77] und ist dem Begriff *Buch* subordiniert. Eine Schnittmengendarstellung aus diesem Beispiel für die Beziehung der Worte Bilderbuch und Buch könnte so aussehen:

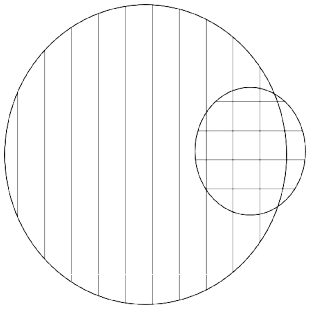


Abbildung 5: Buch und Bilderbuch

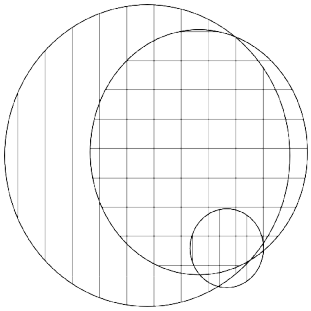


Abbildung 6: Medium, Buch und Bilderbuch

Nach einer „Normalisierung“[78] der Werte für *Medium, Buch* und *Bilderbuch* würden die subordinierten Begriffe tatsächlich zur Gänze dem superordinierten Begriff untergeordnet sein. Dann liesse sich z.B. Abbildung 6 auch durch folgende Notationen repräsentieren (die Zahlen stellen die Größe der Schnittmenge des Wortes in Bezug auf die höchstordinierte Basisklasse dar):

1. (Medium 1.0 (Buch 0.7 (Bilderbuch 0.1))
2. Medium 1.0
   Buch 0.7
   Bilderbuch 0.1

Es handelt sich also um sogenannte Baumstrukturen (siehe Wirth 1975: 220).[79]

76 Dies ist nur eine von vielen möglichen Darstellungen eines Wortfeldes. Es wiederspiegelt die Ergebnisse der Berechnungen des PaGe. Im Allgemeinen ist ein Wortfeld ein multidimensionales Feld, wobei die Dimensionen des Feldes bestimmte Schlüsselkookkurrenzen darstellen (im Idealfall der Index eines Korpus. Dies ist aber aus Performanzgründen in der Praxis undurchführbar). In den Dimensionen werden dann die Abstände des Wortes zu diesen Schlüsselwörtern eingetragen.

77 Gemeint ist die Häufigkeit differenter Vorkommen. Es gibt andere Definitionen für die *Größe eines Wortfeldes*. Z.B. kann eine Wortfeldgröße auch durch die Anzahl unterschiedlicher Worte innerhalb des Feldes bestimmt werden.

78 Eine Normalisierung geschieht über einen Parameter. Als Effekt ergibt sich eine grobere, qualitativere Klassifizierung. Ein Beispiel wäre die uneingeschränkte Zuordnung von *Bilderbuch* unter *Buch*.

79 Sie ließen sich auch dann durch eine Baumstruktur repräsentieren wenn sie noch einen zweiten, „unscharfen“ (Nicht)Zugehörigkeitswert behielten.

2.2.3. Problematik

»Wir dürfen jetzt nur nicht den Sand in den Kopf stecken.«
Lothar Matthäus

Im Folgenden wird zwischen zwei Typen von Problematiken unterschieden, nämlich zwischen einer *generellen* Problematik und einer *speziellen*. Ein generelles Problem für die hier dargestellten Methoden zur Paradigmenbildung betrifft diejenigen Sprachen, deren syntaktische Freiheiten groß sind, also deren Wortanordnungen schlimmstenfalls beliebig sind. Mit dem hier vorgestellten Instrumentarium lassen sich diese nicht bearbeiten. Es gilt: je größer die Gewichtung der Syntax einer Sprache, desto besser lassen sich mit Hilfe der Kookkurrenzanalyse Paradigmen berechnen. Das heißt, dass eine Sprache wie das Englische besser zu analysieren ist als z.B. das Deutsche, Russische oder gar das stark flektierende Lateinische. Des Weiteren sind prinzipiell Daten aus einem Korpus gesprochener[80] Sprache besser geeignet als Daten aus der Schriftsprache, da diese eine größere Komplexität und Variabilität auf syntaktischer Ebene vorweisen.

Ein weiteres Problem ist die Abhängigkeit vom Datenmaterial. Die Qualität des Outputs einer Funktion steht in starker Dependenz zur Qualität des Inputs. Davon ausgehend, dass die Korpora „im Großen und Ganzen“ von akzeptabler Qualität sind,[81] können die „Fehler“ im Material auch durchaus als plausible Grundlage für die Möglichkeit einer Sprachevolution, also Sprachveränderung, angesehen werden: Fehler sind grundlegend für evolutive Prozesse. Allerdings sollte dieser Überlegung in diesem Kontext nicht zuviel Beachtung geschenkt werden.

Wie im vorigen Kapitel dargelegt, ist die lexikalische Disambiguierung möglich. Ein *spezifisches Problem* dabei ist allerdings der enorme Rechenbedarf, der bei einem größeren Korpus entstehen kann (vgl. dazu Kapitel 3.3). Um diesen zu verringern lassen sich Schwellwerte einführen, ab denen die Berechnungen abgebrochen werden. Allerdings wäre dadurch das Ergebnis verfälscht.
Lexikalische Ambiguitäten, die satzübergreifend sind, lassen sich mit PaGe nicht direkt auflösen. In den 1960er Jahren hat ein früher Forscher der maschinellen Übersetzung

80 Die gesprochenen Worte müssten allerdings ins Hochdeutsche übertragen werden – bei einer zu engen Transkription würde die Anzahl der Worte explodieren (vgl. etwa die verschiedenen Entsprechungen für das Wort *nichts*: nix, niks, nüscht, nöscht u.a.).

81 Aufgrund der Einfachheit des Algorithmus wirken sich einige fehlerhafte Stellen im Korpus kaum aus.

namens Bar-Hillel ein als mittlerweile klassisch eingestuftes Beispiel der Schwierigkeit von (maschineller) Übersetzung anhand des Satzes *The pen is in the box* gegeben. Die Lesarten sind unter anderem die folgenden:

1. Der Kugelschreiber ist in der Schachtel.
2. Der Laufstall ist in der Loge.

Ohne den Kontext der voherigen Sätze, d.h. ohne die Kenntnis des Topiks, bleibt die Lesart unaufgelöst.[82]
Neben lexikalischer Ambiguität existieren weitere Ambiguitätstypen. Ein Beispiel für syntaktische Ambiguität ist der Satz *Ich sehe den Mann mit dem Fernrohr.* Die Lesarten werden durch die Klammernotation deutlich:

1. Ich sehe (den Mann (mit dem Fernrohr)).
2. Ich sehe (den Mann) (mit dem Fernrohr).

Im ersten Fall ist *Fernrohr* possessiv verwendet, da es im Besitz des Mannes ist, während im zweiten Fall das Fernrohr instrumental eingesetzt ist. Auch wenn PaGe für dieses Beispiel kein „falsches" Paradigma berechnen würde, so zeigt sich jedoch, dass kategorielle Feinheiten nicht so einfach zu bestimmen sind. Für die Disambiguierung syntaktischer Ambiguität, sowie auch anderer Uneindeutigkeiten, wie etwa der pronominaler Art, bedarf es satzübergreifender Kotextanalysen. Dies kann PaGe nicht leisten.[83]

Ein anderes Problem als das der Performanz zeigt Abbildung 5. Obwohl *Bilderbuch* in Bezug auf *Buch* tatsächlich ein *Hyponym* ist, es also von *Buch* inkludiert wird, gibt die Grafik die Inklusion nicht zu 100 Prozent wieder. Es existieren Kotexte im Korpus, in denen zwar *Bilderbuch* auftritt, nicht aber Buch, z.B. „Das *Bilderbuch* von Moebius ist in der Vadeboncoeur Collection erschienen."[84] Auch hier ließe sich durch Einführung eines Schwellwertes das Ergebnis nachträglich in dem etwas unscharfen Sinne des *Hyponyms* verbessern.

---

82 Eine kleine Sammlung von Sprachübersetzungsprobleme findet sich unter der folgenden URL: http://www.ttt.org/theory/barker.html (letzter Zugriff: 26.02.2006)

83 Es ist natürlich vorstellbar, dass zu jedem in der Datenbank abgespeicherten Kotext ein Spalteneintrag für den Topik verwirklicht wird. Die Ermittlung des Topik wäre Aufgabe eines Präprozessors.

84 Und nicht: „Das *Buch* von Moebius ist in der Vadeboncoeur Collection erschienen".

Es ist nicht notwendig, dass zwei differente linguistische Einheiten jeweils in *allen* Kotexten der anderen Einheit vorkommen. Wie hoch der Schwellwert sein muss, um tatsächlich von einer paradigmatischen Relation sprechen zu können, bleibt an dieser Stelle offen. Der Algorithmus von PaGe listet lediglich eine Reihenfolge der (nach den Regeln des Algorithmus berechneten) *besten* Einheiten auf. Es kann sich dabei durchaus ergeben, dass für eine Einheit kein sinnvolles Paradigma vorhanden ist – trotzdem würde ein solches behauptet (wenn auch aus der Darstellung der geringen Schnittmenge dem Interpreten die Qualität der Aussage bewusst gemacht sein sollte).

Ein weiteres Problem ist das des *Antonyms*:
Nicht zu jedem Wort gibt es auch ein Antonym. Welche Antonyme hätten z.B. die Worte *Rechner, Buch, Maus, orange* et cetera ?

2.3. Selbstlernende Systeme

»Was Hänschen nicht lernt, dass lernt eben Hans«
Abwandlung eines alten deutschen Sprichwortes

Unter dem Begriff *selbstlernendes System* ist im weitesten Sinne jedes System mit der Eigenschaft des Sich-selbst-veränderns subsumiert.
Die Domäne selbstlernender Systeme ist facettenreich, und so wird sich hier auf die knappe Darstellung existierender selbstlernender Softwaresysteme für Paradigmenbildung beschränkt.
Die Geschichte selbtlernender Softwaresysteme beginnt mit der 5. Computergeneration, also ungefähr ab 1980. In dieser Zeit enstehen unter anderem die ersten *neuronalen Netze*, *adaptiven Regler, hierarchisierende Expertensysteme* und *lernende Entscheidungsbäume*. Ein bekanntes System ist das 1984 entstandene, unter dem Begriff künstliche Intelligenz subsumierte *Cyc*[85]-Projekt.

2.3.1. Definitionen

Das Charakteristikum adaptiver Prozesse ist die Möglichkeit zur Verhaltensänderung. In

85 Gegründet wurde Cycorp 1984 in Austin, Texas, von Computerwissenschaftler Professor Douglas Lenat. Siehe http://www.cyc.com (letzter Zugriff: 26.02.2006)

der Psychologie Jean Piagets wird dies als *Akkomodation* bezeichnet und ist damit das Antonym zu *Assimilation*. Änderungen können die eigene Funktionalität betreffen oder auch nur Änderung gespeicherten deklarativen Wissens bedeuten. Für den PaGe trifft lediglich letzteres zu, das heißt es handelt sich nicht um ein seinen Algorithmus selbstständig veränderndes Programm. PaGe zieht ausschließlich deduktive Schlussfolgerungen aus gegebenen Input, wobei die Qualität und Quantität des Inputs über die Qualität des Outputs bestimmt.
Selbstlernende Systeme werden in zwei Hauptkategorien eingeteilt:

- *unsupervised learning*
- *supervised learning*

PaGe verwendet das erstere, also das unkontrollierte Lernverfahren. Zu keiner Zeit werden Schlussfolgerungen des Systems vom Menschen evaluiert und beeinflusst.
Auf der Distributionsanalyse basierende Modelle, die zur Generierung von Paradigmen eingesetzt werden, sind zusätzlich ABL (van Zaanen 2002), SP (Dennis & Harrington 2001), ADIOS (Solan et al 2002) und SOG (Schwiebert 2004) zu nennen. Alle diese Ansätze arbeiten auf der Basis unüberwachten Lernens mit rohen, nicht weiter prozessierten Texten, und (sind zumindest theoretisch) sprachunabhängig.[86] Zur eingehenden Darstellung derjenigen Algorithmen, die im Laufe der Entwicklung von PaGe eingesetzt wurden, siehe Kapitel 3.3.

2.3.2. Problematik

Häufiges Problem selbstlernender Systeme ist die Datenabhängigkeit. Qua Definition hängt die Qualität des Outputs vom Input ab, und zwar sowohl von dessen Qualität als auch dessen Quantität.
Der Aussagenlogik nach kann aus etwas „Wahrem“ nichts „Falsches“, aber aus etwas „Falschem“ Beliebiges gefolgert werden: Berechnungen valider Paradigmen können nur auf Basis valider Kotexte geschehen. Ergebnisse der Algorithmen müssen mithin stets in Bezug zu ihrer Datengrundlage interpretiert werden. Von prinzipiellen Aussagen der Art „Wort A ist in Wort B inkludiert“ ist also abzuraten, wobei mit zunehmender (qualitativ hochwertiger) Menge des Inputs der normative Charakter des Outputs steigen sollte. Eine gewisse „Fehlertoleranz“ der Algorithmen, bedingt durch den probabilistischen Charakter

86 Zur erläuternden Darstellung der Algorithmen siehe unter anderem Schwiebert (2004).

der Aussagen, ist dabei gegeben.

Selbst bei einem größeren Korpus liegen unter Umständen für ein einzelnes Wort nur wenig Kotexte vor. Die Abhängigkeit, probabilistische Aussagen aus einer eher spärlichen Menge gegebener Daten zu formulieren, wird als *Data-Sparseness-Problem* bezeichnet: eine Extrapolation auf Grundlage weniger Daten muss ungenau bis unmöglich, weil beliebig, werden.

Den beiden genannten korpusabhängigen Problemen kann durch entsprechende Textwahl begegnet werden, wobei das prinzipielle Postulat dieser Arbeit darin besteht, möglichst auf vorselektierte (und damit „künstliche“) Korpora zu verzichten und stattdessen eine möglichst große Textsammlung breitgefächerter Form und unspezifischen Inhalts zu verwenden. „Bessere“ datengrundlegende Texte als die in Kapitel 2.2.3 erwähnte Quelle gesprochener Texte (im Gegensatz zur elaborierteren Schriftsprache) wären aber jedenfalls Materialien, die zum Sprachlernen eingesetzt werden. Ein dort zur Verwendung kommendes didaktisches Mittel besteht in der Gegenüberstellung leicht zu vergleichender Sätze, also der Bereitstellung ähnlicher Kotexte. Der Einsatz eines solchen Methode ist zudem ein Indiz für die Änhlichkeit des Prinzips des in Kapitel 3.3 vorgestellten Lernalgorithmus und dem Lernen eines Menschens. Das Prinzip des Lernens durch Analyse von Distributionsverhältnisse stellt auch Thimm (2003: 200) in ihrer Beobachtung fest:

> Wann immer sie Kategorien bilden, arbeitet ihr Gehirn wie ein nimmermüder Statistiker. Permanent wertet das Baby aus, was es hört, sieht oder fühlt, und ordnet die Eindrücke nach Wahrscheinlichkeit: was häufig vorkommt muss bedeutsam sein. Was regelmäßig zusammen auftaucht – Artikel und Hauptwort, süße Pampe und Löffel – wird schon irgendwie zusanmengehören. (Thimm 2003: 200)

Die grundlegende Herangehensweise zur Lösung beider Probleme, nämlich der Benutzung einer möglichst großen Textsammlung, zieht ein weiteres Problem nach sich. Mit der Eingabe einer zunehmenden Datenmenge wächst der Ressourcenbedarf des Programms. Dem Speicherplatzproblem kann, wie im Falle des PaGe, mit der Auslagerung der Daten in eine Datenbank begegnet werden. Allerdings steigt schon damit der Bedarf an Rechenzeit für die Bearbeitung der gleichen Menge von Daten im Vergleich zur Berabeitung bei Datenhaltung im RAM. Gravierender ist aber die Tatsache, dass die Relation von Datenmenge und der Rechenleistung, die für deren Verarbeitung aufgebracht werden muss, in einem nicht-linearen Verhältnis stehen. Im Falle von z.B. SOG ist das Verhältnis von Rechenzeit zur Anzahl der eingelesenen Sätze ungefähr quadratisch. In frühen Versionen von PaGe kamen Algorithmen zum Einsatz, deren Rechenbedarf *deutlich* höher lag (siehe hierzu das Kapitel 3.3).

3. Die Webapplikation *PaGe*

»Am Anfang war alle Software frei.«

Die Programmierarbeit entstand fast gänzlich unter Einsatz von *Freier-*[87] und *Open Source-*[88] Software, welche auch für den Betrieb des laufenden Programms eingesetzt wird. Lediglich die verwendete Programmiersprache, nämlich Java von Sun, ist kein Open Source Projekt. Außerdem wurde die Arbeit zum größten Teil unter dem Betriebssystem Mac Os X 10.3.9 erstellt, welches ebenfalls nicht Open Source ist.[89]
PaGe unterliegt der *GNU General Public License (GPL)*[90] und ist damit ebenfalls *Freie Software*. Damit wird einer Tradition der Informatiker gefolgt, die *in toto* bis Ende der 1970er Jahre den freien Gedankenaustausch mit anderen Programmierern als Selbstverständlichkeit erachtet haben.[91]

Das Programm PaGe lässt sich prinzipiell aus drei Perspektiven betrachten:

1. Entwicklersicht: dies ist die komplexeste Perspektive. Für einen Programmierer, der PaGe weiterentwickelt, sind alle weiteren Kapitel interessant.
2. Administratorsicht: der PaGe-Administrator installiert den HTTP-Server, den PaGe und die MySQL-Datenbank. Dies anzuleiten ist vor allem Aufgabe des Kapitels 3.1.
3. Benutzersicht: Nutzer des PaGe benötigen lediglich einen Internetzugang und einen Browser. Die sich bietenden Steuerungsmöglichkeiten werden in Kapitel 3.2.3 beschrieben.

Das folgende Unterkapitel geht auf die Schritte zur Installation und Pflege des Programms ein. Zudem werden das Technologiekonzept der Benutzeroberfläche, die Verteilung durch das Applikationsmodell sowie die Datenhaltung dargestellt und die sich daraus

87 „*Freie Software* hat etwas mit Freiheit zu tun, nicht mit dem Preis. Um das Konzept zu verstehen, ist an *frei* wie in *freier Rede*, und nicht wie in *Freibier* zu denken.“ (siehe http://www.gnu.org/philosophy/free-sw.de.html letzter Zugriff: 26.02.2006).

88 Einblick in den Quelltext eines Programms haben zu können, sowie die Erlaubnis zu haben, diesen Quellcode auch beliebig weiterzugeben oder zu verändern, ist die Minimaldefinition von Open Source. Dass die derart lizensierte Software kostenlos sein muss, ist ein oft anzutreffendes Voruteil, auch wenn dies für die hier eingesetze Open Source- (und Freie-) Software zutrifft.

89 Auch wenn der darunter liegende Kernel, ein FreeBSD-Unix Derivat, zur *Freien Software* gehört.

90 http://www.gnu.org/licenses/gpl.html (letzter Zugriff: 26.02.2006)

91 Zur weiteren Information über *Frei Software*, z.B. über Sicherheitsaspekte und weitere, gesamtgesellschaftliche Implikationen siehe u.a. http://freie-software.bpb.de/ (letzter Zugriff: 26.02.2006)

ergebenden Implikationen für den Benutzer in einem Exkurs zur „Barrierefreiheit“ aufgezeigt.

In den Kapiteln ab 3.2 wird vor allem die Bedienung, die Administration der Daten über das Webinterface und die Arbeitsweise einiger Methoden des PaGe ausführlich beleuchtet. Dem Algorithmus zur Datenstrukturierung und der Deduktion neuen Wissens aus diesen Daten ist das Kapitel 3.3 gewidmet.

3.1. Softwareumgebung

Die Applikation[92] *PaGe* folgt in ihrem Aufbau dem *3-Tier Modell*[93], welches eine Abstraktion des bekannten *Modell-View-Controller (MVC)* Prinzips darstellt:

> MVC is a software architecture that separates an application's data model, user interface, and control logic into three distinct components so that modifications to one component can be made with minimal impact to the others.[94]

Damit folgt die Realisation der Applikation einem aktuellen Paradigma[95] für die Programmierung von Anwendungen.

In der Mitte von Abbildung 7 sind diese drei Ebenen dargestellt: Client-, Geschäfts- und Persitenzebene.[96]

92 *Applikation* ist ein Synonym für Anwendungsprogramm und steht damit „im Gegensatz zum Betriebssystem und allen System- und Hilfsprogrammen, die "nur" den Betrieb ermöglichen, aber noch keinen "Nutzen" bringen.“ vgl. http://de.wikipedia.org/wiki/Anwendungsprogramm (letzter Zugriff: 26.02.2006)

93 http://en.wikipedia.org/wiki/Three-tier_(computing) (letzter Zugriff: 26.02.2006)

94 http://en.wikipedia.org/wiki/Model-view-controller (letzter Zugriff: 26.02.2006)

95 Hier wird *Paradigma* in seiner anderen Lesart, der des *wissenschaftlichen Paradigmas*, verwendet und ist somit dann dem Begriff „Leitbild“ oder „Modellvorstellung“ synonym.

96 Hall und Brown (2004) beschreiben die JSP-Seite als *View*, während in Kapitel 1 „Overview Distributed Multitiered Applications“ unter http://java.sun.com/j2ee/1.4/docs/tutorial/doc/ die JSP-Seite der Geschäftslogik zugeordnet wird: „[...] multitiered applications are generally considered to be three-tiered applications because they are distributed over three locations“. Tatsächlich nimmt JSP eine Zwischenstufe ein, da es einerseits zu einem Servlet auf dem Server umgewandelt wird, es aber andererseits in HTML eingebettet ist und immer einen View erzeugt.

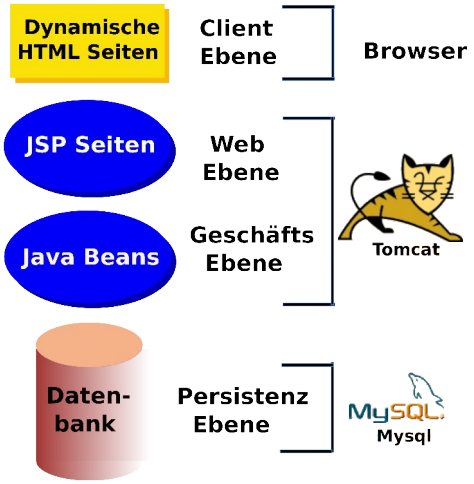


Abbildung 7: Applikationsmodell

Auf der linken Seite der Abbildung sind die erzeugten Softwarekomponenten dargestellt, also etwa Java Beans. Sie werden in Kapitel 3.2 erläutert. Auf der rechten Seite sind die zu ihrer Interpretation oder Steuerung eingesetzten Softwarewerkzeuge aufgeführt, also etwa der Tomcat von Apache, der als Container für Servlet und Java Server Page (JSP)s fungiert.[97] Auf diese *Softwareumgebung* wird im Folgenden näher eingegangen.

3.1.1. Server und Servlet Container

Wie bereits erwähnt sind die eingesetzten Softwarewerkzeuge *Open Source* oder, wie im Falle des Webservers *Tomcat*,[98] sogar *Freie Software.* Das impliziert, dass die Werkzeuge auf jeder neueren Rechnerplattform zur Verfügung stehen oder selbst portiert werden können. MySQL beispielsweise gibt es auch für den Amiga[99], während aber Tomcat nur auf Rechnern mit Java-Unterstützung[100] existiert.

Auf dem Server läuft die eigentliche Programmlogik (die sogenannte *Geschäftsebene* (englisch Business Tier)). Diese ist durch Java-Beans implementiert sowie über Servlets und JSP, die über den Tomcat-Container als Webanwendung über HTTP[101] kommuniziert.

Die Installation des Tomcat ist auf *Redhat-Fedora*[102] und auf *Mac Os X 10.3*[103] unproblematisch.[104] Es müssen lediglich einige Klassenpfade gesetzt werden. Der

97 http://wiki.apache.org/tomcat/ (letzter Zugriff: 26.02.2006)

98 Es existieren weitere Servlet Container, unter anderem Jetty, Resin, Jrun, Weblogic und Winstone.

99 Homecomputer aus dem Jahre 1985

100 http://java.sun.com/j2se/ (letzter Zugriff: 26.02.2006)

101 Das „Hyper Text Transfer Protocol“ wird hautsächlich für den Transfer von Daten über das World Wide Web (www) eingesetzt. (Für Spezifikationen siehe http://www.w3.org/Protocols/ letzter Zugriff: 26.02.2006)

102 „Fedora is an operating system and platform, based on Linux, that is always free for anyone to use, modify and distribute, now and forever. It is developed by a large community of people who strive to provide and maintain the very best in free, open source software and standards. Fedora is a collection of projects sponsored by Red Hat, Inc.“ (siehe http://fedoraproject.org letzter Zugriff: 26.02.2006)

103 http://www.apple.com/ (letzter Zugriff: 26.02.2006)

104 Die Installation sollte auf anderen Betriebssystemen ähnlich einfach sein.

voreingestellte Port[105], den die Firewall auf der Seite des Servers offen halten muss, damit die Außenwelt mit ihm kommunizieren kann, hat die Nummer 8080. Nun muss das mitgelieferte *Web Application Archive* (*Paradigmenbildung.war*)[106] in das Verzeichnis $CATALINA_HOME/webapps/[107] kopiert werden. Damit ist der PaGe einsatzbereit – lediglich die Datenbank bedarf der Installation.

3.1.2. Datenbank

Als Datenbank kommt MySQL 5.0[108] zum Einsatz. Die Installation ist zwar einfach, doch setzt die Benutzerverwaltung, zumindest in Unix-Umgebungen, Kenntnisse im Umgang mit Mehrbenutzersystemen voraus. Da die Entwickler der „populärsten Open-Source-Datenbank der Welt“ ein über 1500 seitiges Handbuch zur Verfügung stellen, sollten keine Fragen mehr offen bleiben.[109]

Es ist möglich den MySQL Server und die Datenbank auf einem anderen Rechner als dem der Webanwendung laufen zu lassen. Doch würde dies erhebliche Performanzeinbußen bedeuten. Auch aus Sicherheitsgründen[110] wird dies nicht empfohlen.[111]

Da das Abfragen der Datenbank der „Flaschenhals“ der Datenverarbeitung ist,[112] ließe sich die Performanz, die bei mehreren simultanen Zugriffe leidet, durch eine verteilte Architektur, bei der die Datenbank auf einem Cluster von Workstations vorgehalten wird, verbessern. PaGe in Version 0.3 verhindert die simultanen Zugriff auf die Datenbank, da ansonsten jegliche Prozesse stark verlangsamt würden.[113]

---

105 „Ports sind Adresskomponenten, die in Netzwerkprotokollen eingesetzt werden, um Datenpakete den richtigen Diensten (Protokollen) zuzuordnen.“(http://de.wikipedia.org/wiki/Port_%28Protokoll%29 letzter Zugriff: 26.02.2006)

106 Das Archiv liegt auch auf: http://pascal.selfip.org/uni/spinfo/Paradigmenbildung.war

107 $CATALINA_HOME ist die Variable für das Heimverzeichnis der Tomcatinstallation.

108 http://www.mysql.de/ (letzter Zugriff: 26.02.2006)

109 Falls sich trotzdem Probleme ergeben, hilft ein Besuch des Forums unter: http://forums.mysql.com/ (letzter Zugrif: 26.02.2006).

110 Es muss ein weiterer Port geöffnet werden.

111 Sinnvoll wäre dies falls es tatsächlich mehrere Installationen auf verschiedenen Servern gäbe, die ihre Paradigmenberechnungen dadurch einer Datenbank eines einzelnen Servers mitteilen könnten. Dadurch wären die Ergebnisse an einer zentralen Stelle gebündelt.

112 Das heißt, dass der Datenbankzugriff relativ viel Zeit benötigt.

113 Bei einem simultanen Zugriff wird eine HTML-Seite aufgerufen, die den Benutzer auf die Datenbankauslastung mit dem Zusatz hinweist, die Anfrage nach zwei Minuten zu wiederholen. Die Einrichtung einer *Queue,* also einer Warteschlangedatenstruktur, wäre sicherlich eine bessere Alternative.

3.1.3. Client

Der Benutzer kann die Webapplikation durch Eingabe der URL in einem beliebigen Browser aufrufen.[114] Hat er die Applikation wie im vorherigen Kapitel beschrieben selbst installiert, lautet die lokale URL: http://localhost:8080/Paradigmenbildung. Die Anwendung ist auch unter der URL http://pascal.selfip.org/Paradigmenbildung erreichbar.[115]

Der Einsatz eines Thin-Client[116] ermöglicht u.a. eine niedrigschwelligere Mensch-Maschine Schnittstelle. Dadurch kann das Projekt der Maxime des „Barrier Free Web" folgen.

Exkurs: „Barrier Free"

Was bedeutet das gerade im Zusammenhang mit Computertechnologie häufig benutzte Modewort „Barrierefreiheit" (Barrier Free)? Grundlage für die folgende Darlegung bilden die Empfehlungen des W3C, die im Rahmen der Web Content Accessibility Guidelines 1.0[117] aufgeführt sind. Hinter dem Begriff verbirgt sich die Forderung des für die Benutzer möglichst einfachen Zugangs zu Internetinhalten, wobei die sog. All-Inklusion angestrebt wird.[118] Dieser Terminus kann nur relative Bedeutung haben. So sind Menschen ohne eigenen Rechner oder ohne Netzanschluss per Definitionem vom Internet exkludiert. Dies wird wiederum durch andere Zugangsmöglichkeiten, z.B. durch öffentliche Bibliotheken oder Internetcafés, relativiert.[119]

Sind die hardwaretechnischen Hürden genommen, erscheinen softwaretechnische Barrieren – das fängt an mit dem generellen Ausschluss an Teilhabe aufgrund einer spezifischen Rechnerplattform (sogenannte „Betriebssystemabhängigkeit") und hört

114 Meldungen wie der des Max-Planck-Instituts für Psycholinguistik in Nijmegen unter der URL http://www.mpi.nl/world/ (dem Link „Online Experiments" folgend): „**Browser not compatible.** Your internet browser is not compatible with our website. Please use Microsoft Internet Explorer (version 6 or above) or Mozilla Firefox." sind vermeidbar.

115 Der dort eingesetzte Server wird vom Autor auf seinem privaten Laptop sowie einem Linux-Ersatzsystem betrieben. Die Netzanbindung ist recht schmalbandig, deshalb können sich längere Wartezeiten ergeben.

116 Dies im Gegensatz zum Beispiel zu einem Rich-Client wie etwa einem Java-Applet. Vgl. auch http://www.torsten-horn.de/techdocs/webanwendungen.htm (letzter Zugriff: 26.02.2006)

117 http://www.w3.org/TR/WCAG10/ (letzter Zugriff: 26.02.2006)

118 Die Bedeutsamkeit der All-Inklusion mündet in einer der zentralen Funktionen der Massenmedien: "Ihre [bezogen auf Massenmedien] gesellschaftliche Primärfunktion", so Niklas Luhmann, "liegt in der Beteiligung aller an einer gemeinsamen Realität oder, genauer gesagt, in der Erzeugung einer solchen Unterstellung, die dann als operative Funktion sich aufzwingt und zur Realität wird." (Luhmann 1993:320)

119 Zur Inklusions/Exkusionsproblematik vgl. exemplarisch Stichweh (1997).

vielleicht mit dem „freiwilligen“ Verzicht des Benutzers aufgrund von Bedienungsunfreundlichkeit des Dienstes auf.
Die Nutzung von PaGe benötigt auf Seiten des Clients lediglich einen mit dem Internet verbundenen Rechner auf dem ein (auch nicht-grafischer) Browser installiert ist.[120] Der verwendete HTML-Code genügt mindestens den Ansprüchen der Kategorie A des W3C, wobei versucht wurde, möglichst der Kategorie AA zu entsprechen.[121]
Die Schnittstelle zur Bedienung des Programms existiert in der Version 0.3 in drei Sprachversionen: Englisch[122], Deutsch und Russisch.
Auf die Verwendung von UTF-8[123] wurde Wert gelegt. Damit lassen sich die meisten Zeichensätze der ca. 5000 menschlichen Sprachen verarbeiten und darstellen - zum Beispiel auch Inuktitut, eine Silbensprache der Inuit. (vgl. dazu auch Kapite 3.2.1)

3.2. Aufbau und Bedienung

Das Applikationsmodell in Abbildung 7 zeigt die Verzahnung von Softwareumgebung und PaGe. Im Folgenden werden einzelne Klassen und Methoden der Klassen aus den Packages *org.selfip.pascal.servlet.** und *org.selfip.pascal.paradigmen.**, aus denen sich insbesondere die Datengewinnung aus den Koprpora ableitet, näher betrachtet. Des Weiteren wird die Bedienung der Webanwendung erläutert.

3.2.1 Servlet und Java Beans

Laut Wikipedia ist ein Server „ein Programm, welches auf die Kontaktaufnahme eines Client-Programmes wartet und nach Kontaktaufnahme mit diesem Nachrichten austauscht.“[124]

Der in Kapitel 3.1.1 beschriebene Tomcat HTTP-Server, der vor allem auch als Servlet

120 Über den Umweg eines lokal installierten Terminals, von dem aus z.B. Lynx auf einem Server gestartet wie (eine solche Funktionalität stellen etwa einige Server der Universität zu Köln und der RWTH Aachen Studierenden zur Verfügung), kann u.a. auch ein C64 (Homecomputer aus dem Jahre 1982) zur Benutzung eingesetzt werden.

121 Das W3C definiert insgesamt drei konforme Kategorien.

122 Englisch ist die voreingestellte Sprache.

123 Zur Dokumentation dieses Formates siehe http://www.unicode.org (letzter Zugriff: 26.02.2006)

124 http://de.wikipedia.org/wiki/Server (letzter Zugriff: 26.02.2006)

Container fungiert, leitet HTTP-Anfragen an das PaGe-Servlet weiter. Das Servlet dient also als eine Erweiterung des Servers, denn es wartet auf Daten, die der Benutzer über HTTP sendet[125] und ruft daraufhin die entsprechenden Programmkomponenten auf, die ihrerseits Daten über den Tomcat an den Browser des Benutzers senden (siehe auch Kapitel 3.4). Bei den Komponenten handelt es sich um gewöhnliche Java-Beans – auch der Code des Servers ist gewöhnliches Java, doch ist die Klasse eine Ableitung und implementiert zudem eine Schnittstelle, muss also dementsprechend angepasst werden:

```
public class Server extends javax.servlet.http.HttpServlet
implements javax.servlet.Servlet { ... }
```

Genaueres ist dem Quellcode zu entnehmen.

Zur Erstellung des Korpus muss der Benutzer sich über den Client zuerst authentifizieren. Der authentifizierte Benutzer kann ein Korpus auswählen und dieses in eine neu anzulegende Datenbank einlesen lassen.[126] Der Server startet die dazu notwendigen Programmmodule. Es sollte auch ein Korpus auswählbar sein, das sich nicht nur lokal auf dem Rechner des Servers befindet. Der Ort soll lediglich durch Angabe einer beliebigen URL spezifiziert werden . Allerdings ist diese Option in Version 0.3 nicht realisiert.
Zum Einlesen des Korpus müssen folgende Bedingungen erfüllt sein:
Die Texte sind entweder HTML-Dokumente oder einfache Texte (*.txt) in ASCII oder UTF-8. XML-Dokumente werden nur ungenügend eingelesen – es fehlt eine ausreichende Präprozessierung.[127]

Der Algorithmus[128] des Tokenizers:

1. Wenn das Dokument zu Ende gelesen wurde, geht das Programm zu Punkt 4, sonst überliest der Tokenizer solange Zeichen aus einem Dokument bis er einen Buchstaben

125 Siehe hierzu Abbildung 8: Die HTML-Seiten für den Client .

126 Diese Option ist noch nicht implementiert. Die Korpuswahl erfolgt also zur Zeit über die Quelltext-veränderung. Damit einher geht die Notwendigkeit der Kompilierung des Codes.

127 Ähnliches gilt auch für HTML-Dokumente. Schlichte Texte in UTF-8 sind am besten geeignet.

128 An dieser Stelle wird der Begriff *Algorithmus*, nach dem Kapitel 3.3 benannt wurde und der dort näher erläutert wird, durch ein Beispiel erklärt. Im Verlauf der Arbeit wird hierauf häufig referenziert, da der Tokenizer als Fundament der Datenerzeugung fungiert und sich über ihn die Realisationsmöglichkeiten grundsätzlicher Berechnungen definieren.

identifiziert.[129] Der Buchstabe wird in eine neu erzeugte Zeichenkette *Zi*[130] gespeichert.

2. Dann wird das nächste Zeichen gelesen. Ist auch dieses ein Buchstabe, wird es an die Zeichenkette *Zi* angehangen und das Programm wird bei 2. fortgesetzt. Wird aber ein Zeichen gelesen, dass nicht als Buchstabe identifiziert wird, geht das Programm zu 3. vor.
3. Ist die Länge der Zeichenkette *Zi* kleiner zwei, wird das Programm bei 1. fortgesetzt. Ist es aber größer eins, ist ein Wort identifiziert. Ist das gelesene Zeichen ein Leerzeichen, fängt der Tokenizer wieder bei 1. an, wobei die Variable i um eins erhöht wird, sodass beim zweiten Mal nicht eine Zeichenkette Z1, sondern Z2 erzeugt wird. Ist das gelesene Zeichen kein Leerzeichen, geht das Programm zu 4. vor.
4. In die Datenbank werden nur Wörter mit Kookkurrenzen persistiert - es müssen also mindestens zwei Zeichenketten *Zi* vorhanden sein. Diese werden in die Datenbank gespeichert. Ist der Zeichenstrom noch nicht zu Ende gelesen, wird wieder bei 1. angefangen, wobei nun die Variable i wieder auf 1 gesetzt wird. Ansonsten bricht der Tokenizer ab. Ist ein weiteres Dokument vorhanden, so wird der Tokenizer mit dem neuen Dokument initialisiert. Der Tokenizer fängt wieder bei 1. an, bis keine weiteren Dokumente vorhanden sind.

In der Klasse *org.selfip.pascal.paradigmen.Creation* werden die vom Tokenizer zurückgelieferten Strings in Kleinbuchstaben umgewandelt. Damit macht die Groß- und Kleinschreibung keinen Unterschied. Dies ist aus folgenden Gründen beabsichtigt:

- viele Texte aus dem Bereich E-Mail und Chatkonversation unterscheiden nur sporadisch zwischen Groß- und Kleinschreibung
- in vielen Sprachen wird die Kleinschreibung verwendet (u.a. Französisch, Englisch, Russisch), es sei denn die Worte stehen am Satzanfang oder es handelt sich um Eigennamen[131]

Ein Effekt besteht in der höheren „Fehlertoleranz“ in Bezug auf die Schreibweise, da es keinen Unterschied macht, ob Worte groß oder klein geschrieben sind. Zudem ergibt sich ein geringerer Speicherplatzbedarf bei gleichzeitig größerer Wahrscheinlichkeit von

129 Dabei wurde auf Genauigkeit der Definition von Buchstaben verzichtet. Vor allem werden alle Zeichen, deren Bytewert größer als 255 ist, als Buchstaben definiert. Die Notwendigkeit liegt in der einfacheren Verarbeitung von UTF-8, dessen Verwendung sonst etlicher sprachspezifischer Regeln bedurft hätte.

130 Anfangs hat die Variable i den Wert 1.

131 Die so erzeugten Invarianten machen für PaGe im Allgemeinen keinen Sinn. Allerdings enstehen doch einige zusätzliche Ambiguitäten: im Englischen z.B. würden das entstehende signifiant *bill* sowohl eine Person bezeichnen können als auch durch *Rechnung* übersetzbar sein. Zur theoretischen Auflösung solcher Ambiguitäten siehe Kapitel 2.2.2.

übereinstimmenden Kotexten.

Die Persistierung der Daten

Die tokenisierten Strings werden in das Zeichenformat ISO-8859-1 gewandelt, durch einen weiteren Algorithmus aufbereitet und in die MySQL-Datenbank persitiert. Jedem String wird dabei eine eineindeutige ID zugewiesen, die mit ihrer Stringentsprechung in einer Tabelle des Namens *words* gespeichert wird, damit später aus den Daten der Kotexttabelle *cotexts*, in der nämlich lediglich die IDs gespeichert sind, die ursprünglichen Strings rekonstruierbar sind. Auf diese Weise kann Speicherplatz gespart werden, da für die Speicherung eines derart symbolisieten Strings in der *cotexts* Tabelle stets drei Bytes aufgebracht werden müssen, auch etwa dann, wenn der String an sich aus z.B. 30 Bytes bestünde.

Die Verarbeitung verschiedener Zeichensätze erwies sich als nicht ganz so trivial, wie sich auf den ersten Blick vermuten ließe. In der jetzigen Version wird ein Umweg genommen, um die Zeichensatzunabhängigkeit zu gewährleisten. Dieser Umweg besteht in der Umwandlung eines tokenisierten UTF-8-Strings zu einem ISO-8859-String. Der folgende Pseudocode zeigt dies:

```
String isoWord = ( new String ( tokenizer.getWord().getBytes
("UTF8"), "ISO-8859-1"));
```

Das so erzeugte Wort kann in eine ISO-8859 verwendende Standarddatenbank eingespeichert werden. Später können die aus der Datenbank gelesen Strings wieder in UTF-8 gewandelt werden:

```
String utf8Word = ( new String ( wordAusDatenbank.getBytes
("ISO-8859-1"), "UTF8"));
```

Parameter, die vom Client an den Server via HTTP gesendet werden, können durch folgende Anweisung im HTML in UTF-8 codiert:

```
<FORM ACTION="Server" encoding="UTF-8">
```

Vor einer weiteren Datenverarbeitung bedarf es wiederum einer Codierwandlung dieses Strings. Der Server wandelt die Zeichenkette durch die schon bekannte Funktion:

```
String isoWord = ( new String (wordVonUTF8_Formular.getBytes
("UTF8"), "ISO-8859-1"));
```

Der Vorteil dieser Codierwandlungen ist die Zeichenformatunabhängigkeit gegenüber der

Persistenzebene: statt die Datenbank auf ein anderes als das voreingestellte Format zu konfigurieren, werden einfach die verwendeten Daten an den Zeichensatz der Datenbank angepasst. Diese Vorgehensweise hat sich während der praktischen Entwicklung vom PaGe als die robusteste erwiesen.[132] Allerdings ergibt sich aus diesem Ansatz eine Einschränkung: Die Namen der Tabellen und Datenbanken müssen im ISO-8859-1 Format erstellt werden. Es wäre intuitiver, wenn aus den Datenbanknamen statt z.B. *kyrilischesKorpus* корпусКирилицы zur Auswahl stünde. Es wäre möglich, die Namen in der Tabelle *paradigm_data.base* zu speichern und mit einer ID zu verknüpfen, die ISO-8859-1 konform ist – auf diese Weise könnten extern die Namen der Datenbanken in beliebigem Zeichensatz repräsentiert werden. PaGe der Version 0.3 bietet dies jedoch nicht.

3.2.2. Datenbank

Nach der Installation und Konfiguration (dem Verteilen und Einrichten von Zugriffsrechten auf Datenbanken) wird lediglich der MySQL-Server gestartet. Die Kommunikation zum MySQL-Server geschieht mittels JDBC[133] unter Java. Anfragen an die Datenbank werden als SQL-Statements vorgenommen und aus Java heraus ausgeführt. Der Programmierer von PaGe muss sich also mit SQL, der *Structured Query Language*, vertraut machen. Folgende Zeilen in *org.selfip.pascal.paradigmen.mysql.Mysql* autorisieren den Clienten bei dem MySQL Server:

```
this.connection = DriverManager.getConnection(
"jdbc:mysql://localhost/" + database ,"username", "passwort");
```

Username und *passwort* müssen den MySQL-Einstellungen entsprechend angepasst werden.

SQL ist eine deklarative Datenbanksprache für Relationale Datenbanken. Ein Dutzend Befehle reichen aus, um Datenbanken und Tabellen zu erstellen, Daten zu persitieren, zu manipulieren und Zählfunktionen für statistische Untersuchungen über die Dateneinträge laufen zu lassen.

Nach dem Einlesen eines Korpus sollten die Daten eigentlich nicht mehr verändert

132 So gibt es etwa zur Einstellung der MySQL-Datenbank über eine ältere JDBC-Version einen *Hack*, mit dem der Zeichensatz bei Etablierung einer Verbindung konfiguriert werden kann. Wird allerdings später eine neuere MySQL-Version verwendet, führt dies zu Problemen.

133 *Java Database Connectivity*: Der Datenbanktreiber stellt das Interface zur Datenbak dar.

werden.[134] Innerhalb von PaGe, d.h. über den Client, ist aber die Funktionalität des Löschens von Datenbanken erreichbar.[135] Sollte sich darüber hinaus die Notwendigkeit einer späteren Datenbankmanipulation ergeben, so lässt sich dies nur über das direkte Ansprechen des MySQL-Servers erreichen. Dazu wird Zugriff auf den Server benötigt. Dies geschieht normalerweise lokal, kann aber auch von außerhalb der Maschine des Datenbankservers erfolgen. Notwendig dazu ist das Vorhandensein z.B. eines SSH-Servers[136], auf den sich ein Benutzer über das Internet mittels SSH-Clients einloggen kann und über den dann der Zugriff auf den Datenbankserver (durch entsprechende Authentifizierung) möglich ist. Die Möglichkeit der *Administration aus der Ferne* kann bei Bedarf eingerichtet werden.

3.2.3. Client

Folgende Schaubilder stellen die HTML-Seiten, wie sie sich dem Benutzer von PaGe präsentieren, in schematischer Form dar. Die Verknüpfungen der Seiten untereinander ist druch Linien symbolisiert:

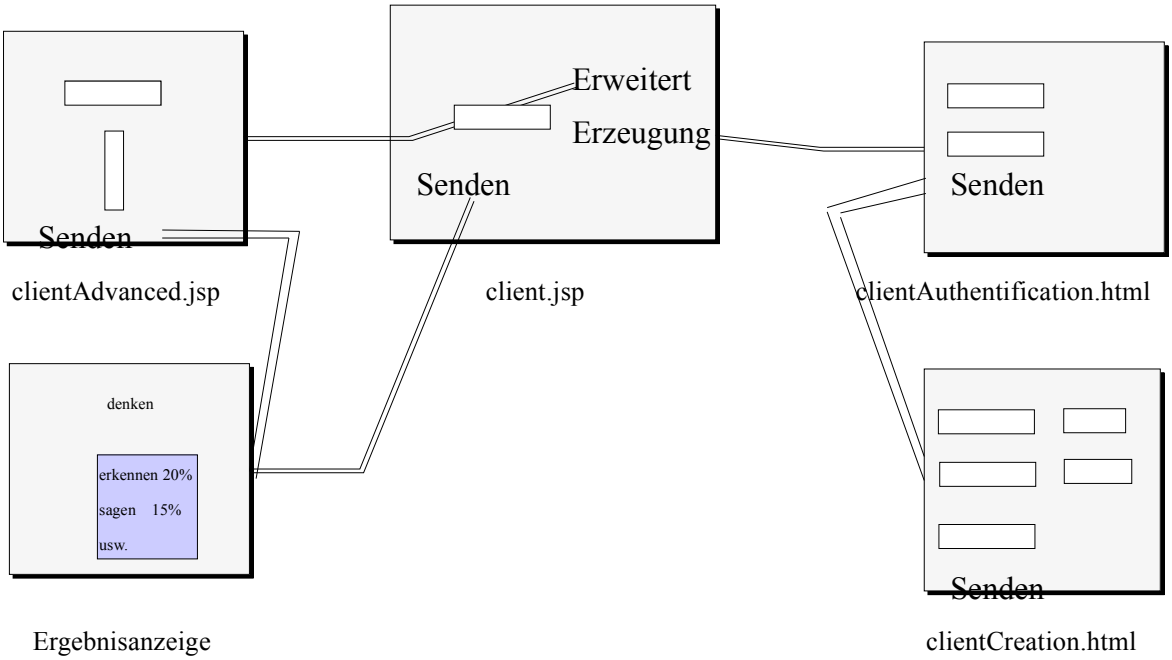


Abbildung 8: Die HTML-Seiten für den Client

Die weißen horizontalen Rechtecke stellen Texteingabefelder dar. Der dort eingegebene Text wird an den Server geleitet, wenn der Benutzer den durch das Wort „Senden“

134 Tatsächlich würde die Möglichkeit der dynamischen Veränderung spätestens dann benötigt, wenn eine Disambiguierungsfunktionalität etabliert werden sollte: einzelne Datenbankeinträge müssten durch eine Indexierung am Wort über ihre verschiedenen Lesarten kenntlich gemacht werden.

135 Da diese Löschvorgänge aber nicht rückgängig gemacht werden können, sollten nur wenige Personen die Möglichkeit dazu haben.

136 Eine Bezugsquelle für die systemunabhängige, freie Software ist http://www.openssh.com/

symbolisierten Knopf anklickt. Die Textelemente „Erweitert“ und „Erzeugung“ innerhalb des *client.jsp* sind mit Hyperlinks verknüpft. Die Doppellinien verbinden die Elemente mit den durch sie ausgelösten Weiterleitungen auf die entsprechenden HTML-Seiten.
Wird die URL http://pascal.selfip.org/Paradigmenbildung/ aufgerufen, so erfolgt eine automatische Weiterleitung auf ./en/client.jsp[137] und zeigt damit die englischsprachige Version von PaGe an. Durch Anklicken der Länderflaggen können zusätzlich die Versionen Deutsch und Russisch ausgewählt werden. Die Seiten und ihre Funktionen im einzelnen:

- **client.jsp**

  Von der Startseite *client.jsp* können alle weiteren Funktionen des PaGe erreicht werden. Außerdem zeigt sie an, wie oft die Seiten insgesamt aufgerufen wurden und gibt Auskunft über die Daten, die der Browser des Benutzers über diesen preisgibt. Die Aktion *Senden* erzeugt auf der Basis des Wortes aus dem Textfeld die Ergebnisanzeige mit den berechneten Wörtern, die innerhalb des Wortparadigmas liegen. Die Berechnungen beruhen auf der durch die in der statischen Klassenvariable *Mysql.databaseDefault* festgelegten Datenbank.

- **clientAdvanced.jsp**

  Hier lassen sich weitere Datenbanken auswählen, z.B. *russkij*, die auf Grundlage eines kyrillischen Korpus erstellt wurden.

  Die Aktion *Senden* erzeugt auf Grundlage des Wortes aus dem Textfeld und der ausgewählten Datenbank die Ergebnisanzeige mit den berechneten Wörtern, die innerhalb des Wortparadigmas liegen.

- **Ergebnisanzeige**

  Die berechneten Wörter werden nach dem Grad der Ähnlichkeit sortiert in einer Tabelle angezeigt. In der linken Spalte befindet sich das signifiant, in der rechten Spalte befindet sich ein prozentualer Ähnlichkeitswert. Ein Wert von 100 bedeutet eine hundertprozentige Übereinstimmung mit dem übergebenen Wort, ein Wert von 0 dagegen überhaupt keine Übereinstimmung. Superordinierende Hyponyme haben einen Wert größer 100. Es werden nur die 29 ähnlichsten Wörter in der Ergebnisdatenbank

137 Das Punkt-Symbol '.' vor dem Slash ist eine Unix-Schreibweise und bedeutet: *dies ist eine relative Ortsangabe* (im Gegensatz zu einer *absoluten* Ortsangabe). Ausgangspunkt ist das aktuelle Verzeichnis, also die vorher beschriebene URL. Die absolute Pfadangabe für dieses Beispiel lautet also: http://pascal.selfip.org/Paradigmenbildung/en/client.jsp

gespeichert.

- **clientAuthentification.html**

  Um eine Datenbank erstellen zu können, muss sich der Benutzer vorher authentifizieren. In Version 0.3 geschieht die Übermittlung aller Daten „im Klartext“ über das Internet, da der Apache-Server ohne *Secure Sockets Layer* (SSL) Unterstützung konfiguriert wurde. Die Daten können daher relativ einfach mitgelesen werden, z.B. mittels des Packet Sniffer *Snort*[138]. Da keine sensiblen Daten geschützt werden müssen, und da der „worst case“ darin bestünde, den Rechner eine Nacht lang zum Erstellen eines sinnlosen Korpus zu bewegen, besteht momentan kein Handlungsbedarf für die Konfiguration eines SSL-Servers oder für die Verlagerung der Datenerstellungssteuerung mittels SSH.

- **clientCreation.html**

  Das Interface zur Steuerung des Einlesens von Korpora ist am wenigsten elaboriert. Bis auf die Möglichkeit der Benennung der Datenbank und der Aktivierung des Einleseprozesses sind die vorhandenen Steuerungselemente (noch) funktionslos.

Die Webseiten mit der Endung *jsp* sind dynamischer Art, das heißt, dass der generierte Inhalt, anders als bei Verwendung reiner HTML-Seiten, nicht von vornherein feststeht. In die Seite *client.jsp* ist (in ihrem nicht-umgesetzten Originalzustand) folgende Anweisung zwischen dem HTML eingebettet:

```
<%= org.selfip.pascal.servlet.utilities.Counter.getHits() %>
```

Die Anweisung wird vom Servlet Container *Tomcat* interpretiert und (im somit generierten HTML) durch den Rückgabewert (die Methode getHits() der Klasse *Counter* gibt die Anzahl der Seitenbesuche zurück) substituiert.
Ähnliches geschieht durch den Aufruf der Methode

```
<%= org.selfip.pascal.servlet.utilities.Counter.getClientInfos
(request)%>
```

Der Methode *getClientInfos()* der Java-Bean *Counter* wird der Parameter *request* , der vom Browser des Benutzers mit Inhalt gefüllt ist, übergeben. Als Rückgabewert wird ein

138 Bezugsquelle unter http://www.snort.org/ . Für Mac OS X : http://www.apple.com/downloads/macosx/networking_security/henwen.html

String erzeugt, dessen Inhalt aus Informationen besteht, die der Browser über sich preisgibt. Diese beinhalten unter anderem die IP[139] und das Betriebssystem der Maschine, sowie den Namen des HTML-Browsers.[140] Die Daten werden lokal gespeichert.

Die *clientAdvanced.jsp* sorgt für die dynamische Mitteilung der von PaGe eingesetzten Datenbanknamen. In der Datenbank *paradigm_data* werden Metadaten über die Persistenzebene von PaGe vorgehalten, z.B. stehen in der Tabelle *base* alle Datenbanknamen, die von PaGe erzeugt wurden, und in der Tabelle *user* befinden sich die Authorisierungsdaten. Die Einträge der Tabelle werden durch folgenden Aufruf ausgelesen:

```
<%= org.selfip.pascal.paradigmen.mysql.DatabaseBase.getDatabases
(response) %>
```

Der zurückgegebene String besteht aus formatiertem HTML-Code und wird zwischen den beiden *select*-Tags der HTML-Seite eingebettet. Wenn z.B. nur eine Datenbank mit dem Namen *Kant* vorhanden wäre, sähe dies so aus:

```
<select NAME="databaseSource" size="15">
<option selected> Kant </option>
</select>
```

Der Benutzer kann nun die Datenbank, auf deren Grundlage das im Textfeld eingegebenen Wort einem Wortparadigma zugeordnet wird, auswählen.

Sollte PaGe anders als erwartet bedient werden oder sollten Fehler z.B. im Zusammenhang mit der Datenbanknutzung auftauchen, so ruft der PaGe-Server entsprechende HTML-Seiten aus dem Ordner ./error auf.

In der vom Servlet generierten *Ergebnisanzeige* werden paradigmatisch ähnliche Worte in Tabellenform präsentiert. Die Ergebnisse der Berechnungen sollten nicht nur Wahrscheinlichkeitswerte der Zugehörigkeit eines Wortes zu einem Wortparadigma sein, sondern zusätzlich eine Visualisierung erfahren (ähnlich der Abbildungen 5 und 6), da die bildhafte Darstellung anschaulicher ist. PaGe bietet aber in der vorliegenden Version 0.3

139 IP ist ein Akronym von *Internet Protocol* .

140 Sofern der Benutzer keinen Anonymisierer verwendet hat, lassen sich hierdurch Benutzer lokalisieren, da jeder Rechner prinzipiell eine eineindeutige IP hat (auch wenn viele Adressen dynamisch vergeben werden und sich hinter Subnetzen befinden, so ist die Zuordnung unter Berücksichtigung der Zeit, d.h. durch Aufzeichnungen und Vergleichen der Daten, möglich).

diese Art der Ausgabe nicht.

3.3. Algorithmen und Performanz

Wie noch zu zeigen ist, ist die Wahrscheinlichkeit eines Auftretens gleicher Kookkurrenzen mit hoher Kotextbreite äußerst gering. Die qualitative Aussagekraft jedoch steigt non-linear mit der Zunahme der Kotextbreite. Aus diesen beiden Tatsachen erklärt sich das Ziel vorliegender Arbeit, das in der Verarbeitung eines Korpus von mehr als 100 Millionen Worten bestand.[141] Für die Realisierung bedeutet das die Entwicklung äußerst performanter Algorithmen, bzw. der Einsatz geeigneter Speicherstrategien (wie etwa Datenbanken). Die im Laufe der Entwicklung von PaGe entworfenen Algorithmen werden in diesem Kapitel dargestellt.
Algorithmen und Daten lassen sich nicht separat betrachten. Algorithmen arbeiten auf Daten. Rolshoven (2002:3) stellt fest:

> Die Objektorientierung betont den Primat der Struktur, speziell der Datenstruktur und sieht den Prozess oder Algorithmus (in Gestalt von Methoden) als Komponente der Struktur dieser untergeordnet.

Die Datenstrukturierung determiniert den Algorithmus, dessen Aufbau und auch dessen Performanz. Zudem hängt der Output eines Algorithmus nicht nur vom Aufbau des Algorithmus selbst und vom Aufbau der Datenstruktur ab - auch die Datenstruktur*inhalte* (kurz: die Daten), die er verarbeitet, bedingen die Ausgabe. Die Datenstruktur und die Datengrundlage des PaGe-Algorithmus wirkt also bestimmend auf die Ergebnisse. Da zudem die Datenstrukturierung der Korpora und die Datengrundlage durch Selektion geeigneter Korpora für den Anwender leichter zu manipulieren ist als die Manipulation am Code des Algorithmus wird im Folgenden die Datenextraktion[142] und der Aufbau der generierten Datenstruktur näher beleuchtet.
PaGe arbeit relativ[143] sprachunabhängig. Da die eingegebenen Daten nicht zwingend auf linguistischen Worten beruhen, sondern beliebige Textstrings sind, können diese Strings als Referenzen für z.B. höhere Syntagmen wie Phrasentypen oder auch gänzlich andere Datenstrukturen wie Bilder eingesetzt werden. So können für diese Umgebungen

141 Das Korpus *gutenbrg1* enthält zwar 12 Millionen Worte, zum Arbeiten empfiehlt sich aber das Korpus namens *deutsch* mit lediglich 400.000 Worten.

142 Siehe hierzu auch 3.2.1: Der Algorithmus des Tokenizers

143 Vergleiche dazu Kapitel 2.2.3 undKapitel 3.2.1

ebenfalls *Paradigmen* berechnet werden. Es handelt sich dann nicht mehr um Wort- sondern um Bild- oder Verhaltensparadigmen usw. Die Einschränkungen hinsichtlich des Korpusaufbaus wurden in Kapitel 3.2 durch die Darstellung des Tokenizeralgorithmus behandelt. Ein weiteres Limit ist die Beschränkung der Anzahl der verschiedenen Referenzen[144] auf ein Maximum von ungefähr 8 Millionen distinkter Einheiten und deren Länge auf jeweils 255 Bytes.

Anders als der SOG-Algorithmus steht in PaGe von vornherein eine einzelner String im Fokus der Aufmerksamkeit. Deswegen werden, im Gegensatz zu SOG, die Phrasen „eine Zeitung“ und „ein Buch“ nicht als austauschbare Elemente interpretiert – es sei denn die Stopwortliste des Tokenizers enthielt zum Zeitpunkt der Datengewinnung die Worte „*ein*“ und „*eine*“.[145] Eine weitere Variante läge in der Beschränkung der distributionellen Analyse auf nur eine, im Deutschen der rechten, Seite. Da die Genus anzeigenden Artikel im Deutschen meist direkt vor dem Nomen stehen, wäre damit ihre differenzierende Eigenschaft im Hinblick auf die Paradigmenbildung wirkungslos.[146] Durch diesen „Trick“ würden die Worte allerdings geschlechtslos gemacht, was sie in Wirklichkeit aber nicht sind, und sie würden als substituierbare Einheiten innerhalb eines Syntagmas eingestuft, was zu ungrammtischen Phrasen führt (siehe dazu das Beispiel um die Abbildung 2 in Kapitel 2.2.2.).

Nur wenn der zu indizierende Text einer Präprozessierung unterläge, dessen Ergebnis Stringreferenzen auf größere linguistische Einheiten lieferte, könnten im Ergebnis auch größere syntagmatische Einheiten in paradigmatische Relation gesetzt werden.[147]

PaGe arbeitet mit einem Kotextfenster fester Breite, wie z.B. auch HAL (vgl. Burgess et al. 1998:6, zit. nach Schwiebert 2004). Eine große Fensterbreite ist wünschenswert, da die Wahrscheinlichkeit einer starken paradigmatischen Beziehung mit zunehmender Größe des gemeinsamen Kotexts der Worte steigt. Allerdings zeigen Untersuchungen, dass die Wahrscheinlichkeit, einen größeren gemeinsamen Kotext zu finden, ziemlich gering ist.[148]

144 Ein Wort, dass polysem ist, also mehrere Bedeutungen hat, trägt mehr als nur eine Referenz. Für einen Datenbankeintrag könnte dieser Sachverhalt durch eine Nummernindizierung an dem Wort kenntlich gemacht werden.

145 Aufgrund der Performanzoptimierung lassen sich Stopworte nicht nachträglich verwenden.

146 Genustragende Pronomina erscheinen oft nach einem Interpunktionszeichen. Sie würden also wegen der Arbeitsweise des Tokenizers (siehe Kapitel 3.2.1) nicht als Kotext indiziert und hätten folglich keine Auswirkungen.

147 Das Einbetten von Algorithmen in Prozesskettensysteme könnte dies ohne größeren Aufwand möglich machen. Das Institut *Sprachliche Informationsverarbeitung* an der Universität zu Köln entwickelt dazu TESLA.

148 So fehlten z.B. schon 14.7% der Tripel (also Worte mitsamt dem Kotext der Breite von zwei), die aus neuen englischen Texten gewonnen und mit einem 366 Millionen Wörter umfassenden Korpus verglichen wurden. Siehe dazu die IBM-Studie (Peter Brown, Stephen dellaPietra, Vincent dellaPietra u.a. 1992)

Dies bestätigt sich auch durch eigene Untersuchungen.[149] PaGe versucht zuerst qualitativ hochwertige (also möglichst breite) Kotexte zu finden und bricht weitere Berechnungen ab, wenn ein im Programm gesetzter Schwellwert erreicht ist, bzw. verringert die Kotextbreite, wenn der Schwellwert nicht erreicht werden konnte. Dies bedeutet zwar, dass nicht alle Kotexte zur Evaluierung der Distributionsverhältnisse Verwendung finden, ist aber notwendig für ein „vernünftiges“ Arbeiten mit dem Programm, da sich eine Kotextanalyse möglicherweise über Tage hinziehen könnte.

3.3.1. Aussagekräftige Kotexte

Der Algorithmus speichert nur *echten Kotext*[150], d.h. keine Leerstellen. Diese tauchen immer dann auf, wenn eine unmittelbare Kotextgrenze[151] überschritten wird. Beispielsweise sind dem Satz [152]:

> Sie müssen im Lexikon vielmehr als nicht weiter analysierbare Entitäten eingetragen werden, als Glieder der entsprechenden Distributionsklasse, und sind nicht von der Grammatik zu generieren. (Lyons 1971:237)

folgende *echte Kotexte* inhärent (das kursive Wort markiert das Zentrum des Kotextes):

Kotexte einer beidseitigen Tiefe von fünf[153]:

5a) "Sie müssen im Lexikon vielmehr *als* nicht weiter analysierbare Entitäten eingetragen"

5b) "müssen im Lexikon vielmehr als *nicht* weiter analysierbare Entitäten eingetragen werden"

Kotexte einer beidseitigen Tiefe von vier:

4a) "Sie müssen im Lexikon *vielmehr* als nicht weiter analysierbare"

4b) "müssen im Lexikon vielmehr *als* nicht weiter analysierbare Entitäten"

4c) "im Lexikon vielmehr als *nicht* weiter analysierbare Entitäten eingetragen"

4d) "Lexikon vielmehr als nicht *weiter* analysierbare Entitäten eingetragen werden"

---

149 Siehe dazu Kapitel 4

150 Die Phrase *echter Kotext* hat keine generelle Bedeutung, sondern ist ein spezieller Begriff, der innerhalb dieser Arbeit Verwendung findet.

151 Eine *Kotextgrenze* ist ein beliebiges, zwischen zwei Strings stehendes Interpunktionszeichen. Die exakte Definition findet sich im Kapitel 3.2. sowie im beiliegenden Quelltext des Tokenizers.

152 Die Definitionen des Begriffs *Satz* sind mannigfaltig. In dieser Arbeit wird er im herkömmlichen Sinn und in seiner wohl gröbsten Form verwendet, nämlich als alles, was zwischen den Punktuationszeichen Punkt(.), Fragezeichen(?) oder Ausrufezeichen(!) vorkommt, ausgenommen aber z.B. Abkürzungen, die mit einem Punkt markiert sind. Eine weitere Definition ist z.B. der in dieser Arbeit verwendete Begriff der *unmittelbaren Kotextgrenze*: diese umfasst alles, was zwischen *beliebigen* Satzzeichen vorkommt, und engt damit den vorher definierten Satzbegriff stärker ein . Kommatas[,], Doppelpunkte[:] und Trennstriche[-] werden also ebenfalls als *Satz*grenzen interpretiert.

153 Die Kotextbreite (oder: das „Fenster“) hat also insgesamt eine Größe von 10.

Kotexte einer beidseitigen Tiefe von drei:

3a) "Sie müssen im *Lexikon* vielmehr als nicht"

3b) "müssen im Lexikon *vielmehr* als nicht weiter"

3c) "im Lexikon vielmehr *als* nicht weiter analysierbare"

3d) "Lexikon vielmehr als *nicht* weiter analysierbare Entitäten"

3e) "vielmehr als nicht *weiter* analysierbare Entitäten eingetragen"

3f) "als nicht weiter *analysierbare* Entitäten eingetragen werden"

3g) "und sind nicht *von* der Grammatik zu"

3h) "sind nicht von *der* Grammatik zu generieren"

und so weiter. Dieses Beispiel wurde recht ausführlich dargestellt, um die im Folgenden dargelegten Probleme zu veranschaulichen.

Die Kotextsammlung ist eine Aussage über die kursiv dargestellten Okkurrenzen. Der PaGe-Algorithmus zur Berechnung der Ähnlichkeit von Wörtern auf Grundlage dieser Kotexte sieht vor, gleiche Kotexte anderer Worte zu finden. Nun ist es zwar intuitiv sehr wünschenswert eine hohe Kotexttiefe zu haben, doch lässt sich erahnen dass z.B. der linke Kotext aus 5b) [*müssen im Lexikon vielmehr als*] kein sehr guter Kandidat ist, um überhaupt wenigstens noch einmal irgendwo anders verwendet worden zu sein. Es scheint zudem auch kein besonders guter Kotext zum bestimmen der Okkurrenz [*nicht*] (was immer das auch für ein Kotext wäre).

Interessanter sind die Kotexte mit geringerer Tiefe:

Der rechte Kotext des Beispiels 3g) [*der Grammatik zu*] enthält ein Dativobjekt [*der Grammatik*], das als Parade-Ergänzung zu der Okkurrenz [*von*] gelten kann. Hiermit lassen sich wahrscheinlich gute Vergleiche finden – auch wenn das Deutsche über die alleinstehende Nominalphrase [*der Grammatik*] nicht nur die Interpretation eines Dativobjektes, sondern auch die eines Genetivobjektes zulässt, d.h. der Kotext ist in seiner oberflächlichen Form grammatisch-kategorial ambig.

Die Strategie, nur unmittelbare, nicht durch Satzzeichen begrenzte Kotexte zu speichern, reduziert die bezüglich der Bestimmung von Paradigmen eher mindere Qualität der zu verspeichernden Daten. Festzuhalten bleibt, dass eine große Menge an zur Bestimmung von Paradigmen belanglosen Daten verspeichert wird. Unter anderem durch diese Tatsache lässt sich der Algorithmus über den Begriff *brute force* charakterisieren.

3.3.2. Performanz

Um mittels rein quantitativer Mittel Aussagen über Qualität zu treffen, ist es hilfreich möglichst viele Daten zu haben. Es gilt die Annahme:

Je größer die Datengrundlage desto genauere Aussagen sind berechenbar.

Bei zunehmender Datenmenge ergeben sich Performanzprobleme, da ihre Verarbeitung Rechenleistung und Speicher benötigt. Jede größere Programmierleistung ist mit folgenden beiden Grenzen konfrontiert:

- *annehmbare Prozessierungsdauer*
- *gegebener Datenspeicherplatz*

Die a*nnehmbare Prozesssierungsdauer* ist von den individuellen Bedürfnissen des Anwenders abhängig und von Fall zu Fall verschieden. Der g*egebene Datenspeicherplatz* ist eine statische Größe und wird durch die gegebene Hardware bestimmt[154].

Für diese Arbeit wird angenommen, dass eine Höchstgrenze der *annehmbaren Prozesssierungsdauer* zwei Minuten nicht überschreiten sollte, d.h. dass der Benutzer des Programms auf eine Anfrage nicht länger als zwei Minuten für die Berechnung der Ausgabe wartet.[155]

Die Berechnungen werden dabei für das erste Mal *on demand* ausgeführt. Das Ergebnis wird daraufhin in einer Datenbank gespeichert. Sollte eine Anfrage mehr als einmal gestellt werden, wird lediglich das Ergebnis aus der Datenbank gelesen und nicht etwa die Berechnung ein zweites Mal durchgeführt.

Eine andere Möglichkeit wäre, alle möglichen Berechnungen durchzuführen, *bevor* eine Abfrage getätigt würde. Das hätte zur Folge dass eine Abfrage in kürzester Zeit beantwortet wäre (nämlich in der Zeit die das Verschicken der Anfrage über das Internet, das Auslesen eines Datums aus der Datenbank und die Rücksendung des Datums benötigt usw., also typischerweise wenige Sekunden). Allerdings benötigt das Berechnen jedes einzelnen Wortes möglicherweise die *annehmbare Prozesssierungsdauer:*

Komplette_Berechnungsdauer = (∑ verschiedene_Worte) * a*nnehmbare_Prozesssierungsdauer*

Selbst wenn die *annehmbare Prozesssierungsdauer* eine Höchstgrenze darstellt und davon ausgegangen werden kann, dass für viele Berechnungen diese Grenze bei weitem nicht erreicht wird, und sich also die durchschnittliche Dauer auf vielleicht ein Zehntel

154 Gemeint ist die Hardware auf seiten des Servers, nicht des Clients.

155 Dieser Wert sollte übr den Client einstellbar sein. In der Verion 0.3 ist dies noch nicht möglich. So hängt es vom PaGe-Entwickler ab, wie hoch die Grenze eingestellt wird.

reduziert, so würde die komplette Berechnungsdauer bei z.B. einer Million verschiedener Worte

1.000.000 * 12 sek. = 139 Tage

dauern. Aus diesem Grund und der Annahme, dass nicht von jedem Wort eine Wortparadigmemzuordnung gefordert wird, wurde die zuerst beschriebene Weise der zuletzt beschriebenen vorgezogen.

PaGe unter der URL http://pascal.selfip.org/Paradigmenbildung/ ist so konfiguriert, dass Berechnungen ab einer beidseitigen Kotextbreite von fünf beginnen. Die maximale *annehmbare Prozesssierungsdauer* darf eine Stunde nicht überschreiten und die maximal gefundene Kotextanzahl liegt unter 30.000. Diese Parameter sind Teil der Klasse QueryEngine:

```
final static byte searchDepth = 5 ;
private final static int dataMax = 30000 ;
private final static int timeMax = 3600 ;
```

Diese Parameter extern zugänglich zu machen ist nicht sehr aufwendig. Allerdings bietet die beschriebene Umsetzung größere Kontrolle des Entwicklers gegenüber der Erstellung der Berechnungen. So erlaubt obige Konfiguration eine ausführliche Ermittlung von Paradigmen. Dies ist zwar für die erstmalige Abfrage eines Wortes von Nachteil, da die annehmbare Prozessierungsdauer bei weitem Überschritten werden kann.[156] Alle mehrmalige Abfragen eines derart berechneten Wortes haben aber den Vorteil, dass ein feineres Ergebnis dargestellt wird.

Die Höchstgrenze des *gegebenen Datenspeicherplatz* ist in erster Linie nicht vom RAM abhängig, sondern von der Festplattenkapazität des Servers, da die Daten auf einer Festplatte ausgelagert sind.[157] Momentan stehen 20 GB zur Verfügung. Dieser Wert kann leicht erhöht werden, wobei zu beachten ist, dass die *Prozesssierungsdauer* in Abhängigkeit zu der Menge der zu verarbeitenden Daten steht. Ein gleichbleibender Wert für *timeMax* kann bei einer signifikanten Erhöhung der Datengrundlage dazu führen, dass möglicherweise ein immer geringer werdender Teil der Kotexte überhaupt Berücksichtigung findet. Dies kann zu schlechteren Ergebnissen führen.

156 Zudem führen Berechnungen von mehr als 90 Sekunden Dauer dazu, dass das generierte HTML nicht mehr dargestellt wird. Der Benutzer muss dann die Seite ein weiteres Mal aufrufen – allerdings wird ihm das vorher nicht mitgeteilt. Hier wäre die Implementation einer Fortschrittsanzeige hilfreich.

157 Sie sind dort in einer MySQL Datenbank persistiert.

3.3.3. Algorithmuswahl

Zur Berechnung von Paradigmen lassen sich viele verschiedene Methoden verwirklichen, selbst wenn diesen der gleiche Grundgedanke, nämlicher der Kotextvergleich, zugrunde liegt. Während der Enstehung von PaGe sind einige Ansätze verfolgt worden, die sich aber aus der im vorigen Abschnitt erwähnten *annehmbaren Prozesssierungsdauer* als untauglich oder nur bedingt einsetzbar erwiesen. Im Folgenden werden die zugrundliegenden Ideen dieser Algorithmen dargelegt. Für PaGe der Version 0.3 kommt lediglich der in Kapitel 3.3.3.3 dargestellte Algorithmus, der auf die in 3.3.3.1 ff. Algorithmen basiert, zur Verwendung.

3.3.3.1. Beschreibung des *komplexen* Algorithmus

Der erste Versuch der Algorithmusimplementierung ist zugleich der komplexeste und flexibelste. Er scheitert jedoch am Zeitkriterium. Die Datenstruktur eines Datensatzes für den Satz [*du liest gerade den text*] stellt sich wie folgt dar:

l_5	***l_4***	***l_3***	***l_2***	***l_1***	***centre***	***r_1***	***r_2***	***r_3***	***r_4***	***r_5***	***cnt***
0	0	0	0	0	du	liest	gerade	den	text	0	1
0	0	0	0	du	liest	gerade	den	text	0	0	1
0	0	0	du	liest	gerade	den	text	0	0	0	1
0	0	du	liest	gerade	den	text	0	0	0	0	1
0	du	liest	gerade	den	text	0	0	0	0	0	1

Tabelle 1

Der Vorteil gegenüber den in den Kapiteln 3.3.3.2 ff. dargestellten Strukturen liegt im einfachen Zugriff auf die einzelnen Elemente des Kotextes. So lässt sich schnell ermitteln, dass für das Wort [*liest*] der erste linke Kotexteintrag [*du*], der erste rechte Kotext [*gerade*] und die ersten beiden rechten [*gerade den*] lauten. Diese Datenstruktur hat aber einen entscheidenden Nachteil: zur Ermittlung eines gleichen Kotextes mit einer Breite größer eins müssen mehrere Spalten durch MySQL verglichen werden. Dies bremst die Datenbank stark aus. Ab Kapitel 3.3.3.2. werden deshalb andere Datenstrukturstrategien erläutert.

Der Ähnlichkeitsalgorithmus zum Berechnen von Paradigmen vergleicht die Ko-Kotexte mit seinen originären Kotexten. Folglich kann aus den beiden folgenden Datensätzen

1. ins *Haus* gehe ich
2. in den *Park* gehe ich

geschlossen werden, dass

3. in den *Hausflur* laufend

zu 1. eine mittelbare Ähnlichkeit aufweist, da der rechte Kotext von 1. und 2. sowie der linke Kotext von 2. und 3. gleich ist.
Für den komplexeren Algorithmus berechnet sich der Aufwand wie folgt:

k= ∑ Kotext, $_r$ = rechtsstehend, $_l$ = linksstehend, _n = Verschiedene Kotexte

(k_r _3(k_l _2(k_r _1))) = Aufwand

Hätte z.B. das Wort [*Haus*] im Korpus 50 Umgebungsworte auf der rechten Seite[158] :

(k_r _1) = 50

und jedes einzelne dieser 50 Worte durchschnittlich 40 Worte auf der linken Seite:

(k_l _2) = 40

und diese wiederum durchschnittlich 45 auf der rechten Seite:

(k_r _3) = 45

so wäre der Aufwand zur Berechnung im Verhältnis:

Aufwand = 50 * 40 * 45 = 90.000

Es ist ersichtlich, dass sich dieser Algorithmus für größere Datenmengen nicht eignet, sondern nur für die Auswertung einzelner kleinerer Dokumente tauglich ist. Die weitere Elaborierung des Algorithmus wurde deshalb eingestellt.

3.3.3.2. Beschreibung des *redundanten* Algorithmus

Wie bereits in Kapitel 3.3 einleitend geschrieben, determiniert die Datenstrukturierung den Algorithmus, dessen Aufbau und auch dessen Performanz. Der Algorithmus, der die Daten aus dem Korpus in die Datenbank strukturiert, ist nicht minder wichtig wie der Algorithmus, der später auf diesen Daten operiert. Im Folgenden wird ein datenstrukturierender Algorithmus erläutert, der die Daten aus dem Korpus aufbereitet

158 *Links* und *rechts* erscheinen auf den ersten Blick seltsame Angaben, doch sind sie im Gegensatz zu Begriffen wie *Nachfolger* und *Vorgänger* auch auf z.B. arabische Texte anwendbar (im Arabischen wird von rechts nach links geschrieben respektive gelesen)

und in die Datenbank verspeichert. Die dort abgelegten Daten sind so strukturiert, dass auf sie möglichst schnell zugegriffen werden kann.

Der Einsatz des *redundanten* Algorithmus ist unbedingt notwendig, wenn die Datenmenge größer ist. Redundanz ist eine zweckmäßige Vorkehrung gegen Störungen (Franke 1982:192). Die Störung liegt hier in dem durch die Komplexität des Zugriffs auf die gewünschten Informationen bedingten Zeitaufwand. Der Zeitaufwand wird durch folgenden Aufbau verringert:

In der ersten Spalte steht die Okkurrenz (respektive deren ID) und in der zweiten Spalte steht der gesamte Kotext. Dabei wurde der linke und der rechte Kotext der Okkurrenz folgendermaßen in einen einzigen String codiert:

Da sich 201 diskrete Zeichen des Typs ISO-8859-1 in MySQL-Tabellen speichern lassen und $201^3 = 8.120.601$ (und dieser Wert kleiner als die vermutete Anzahl verschiedener Worte im Korpus) ist, lässt sich die ID jedes Wortes in einen String der Länge drei umrechnen. Die so erzeugten „Triplets“ werden aneinandergekettet, wobei ein String der Länge 30 entsteht (für die beidseitige Kotextbreite von fünf). Eine solche Variante der Verspeicherung der Daten ist der Verspeicherung in einer Tabelle mit mehreren (hier also 10) Spalten vorzuziehen, da die Suchalgorithmen von MySQL schneller einen eineindeutigen String der Länge 30 auffinden können, als alle Zahlen von 10 Spalten zu vergleichen.[159]

Die gewonnene Schnelligkeit im Auffinden von Kotexten wird dabei mit einem größerem Gebrauch an Speicherplatz bezahlt, denn es müssen *für alle* verschiedene Kotext*breiten* eigene Datensätze angelegt werden. Wie in Kapitel 3.3.2 gezeigt wurde, ist die Grenze der *Annehmbaren Prozessierungsdauer* viel eher erreicht als das Speicherlimit – somit ist eine Algorithmusoptimierung erzielt worden.

Der besseren Übersicht wegen sind Daten unterschiedlicher Kotextbreiten in unterschiedlichen Tabellen gespeichert. Bei einer Kotextbreite von 10 werden also maximal 100 Datensätze benötigt, um alle Kombinationen festhalten zu können. Beispielsweise wird das Vorkommen des Textes [*du liest den text*] folgendermaßen verspeichert - die kursiven Worte werden in eine Spalte *wort*, die nicht kursiven Strings aneinandergekettet in einer Spalte *leftnright* gespeichert :

Zeilennummer	***Text***	***Tabellenname***
1	*du* liest	01
2	*du* liest den	02

159 Mehr zum sogenannten Hashcode siehe unter http://de.wikipedia.org/wiki/Hash

Zeilennummer	*Text*	*Tabellenname*
3	*du* liest den text	03
4	du *liest*	10
5	du *liest* den	11
6	du *liest* den text	12
7	*liest* den	01
8	*liest* den text	02
9	du liest *den*	20
10	du liest *den* text	21
11	liest *den*	10
12	liest *den* text	11
13	*den* text	01
14	du liest den t*ext*	30
15	liest den t*ext*	20
16	den t*ext*	10

Tabelle 2

Die Summe der Einträge ist das Quadrat der Elementanzahl des Textes. Bei einer Kotextbreite von 10 ist das Maximum der permutierten Einträge also 100.

3.3.3.3. Beschreibung des *vereinfachten redundanten* Algorithmus

Der im Folgenden dargestellte Algorithmus (und nur dieser) kommt in PaGe der Version 0.3 zum Einsatz.

Der *vereinfachte redundante Algorithmus* beruht auf einer Synthese der Vereinfachung des *komplexen* Algorithmus, der Idee der Codierung größerer Textbausteine in einen einzelnen String sowie der Reduktion der permutierbaren Kotexte. Es werden nicht alle möglichen Kotextbreitekombinationen persistiert, sondern nur eine Auswahl. Diese Auswahl kommt dadurch Zustande, dass der Text gleichsam wie in einer Laufschrift von rechts nach links gelesen wird:

du liest den text
du *liest* den text
du liest *den* text
du liest den t*ext*

Somit wird nur *die längste*[160] Zeichenkette der möglichen Kotexten berücksichtigt – im Falle der hier angenommenen maximalen Breite von 10 also auch nur 10 Einträge. Die Anzahl der maximalen Einträge ist demnach gleich der Größe der maximalen Kotextbreite.
Ähnlich wie beim komplexen Algorithmus wird der linke Kotext und der rechte Kotext in verschiedenen Spalten gespeichert. Ein Klartexttabelleneintrag sieht damit folgendermaßen aus:

leftwidth	***left***	***centre***	***right***	***rightwidth***	***leftnright***
0		*du*	liest den text	3	! ! ! ! ! liest den text
1	du	*liest*	den text	2	! ! ! ! du den text
2	du liest	*den*	text	1	! ! ! du liest text
3	du liest den	*text*	-	0	! ! du liest den

Tabelle 3

Die Spalten mit dem Suffix *width* ermöglichen eine einfache Suche der gewünschten Breite des Kotexts.[161] Die führenden Ausrufezeichen in der Spalte *leftnright* codieren die Stelligkeit des Textes, der auf diese Weise später in seiner Reihenfolge richtig rekonstruiert werden kann. Durch folgende SQL Anfrage kann der Kotext für den Eintrag [*text*] gefunden werden:

```
SELECT leftnright FROM tabelle3 WHERE centre='text' AND leftwidth=3 AND rightwidth=0 ;
```

Auf die in Kapitel 4.3.2 erwähnte Codierung der Worte zu Triplets ist hier der Übersichtlichkeit wegen verzichtet worden. Die Codierung der führenden Leerstellen besteht nicht aus Triplets sondern aus einem einzigen Byte. Mögliche Optimierung für den Speicherbedarf wäre die Angabe eines einzelnen Integers.

Die bereits erwähnte Bevorzugung qualitativ hochwertiger Kotexte, also jener möglichst hoher Breite, ist in der Klasse *QueryEngine* umgesetzt. Als Hilfsklasse kommt *org.selfip.pascal.mathUtils.LinkSort* zum Einsatz. Hier wird lediglich ein geordnetes zweidimensionales Byte-Array erzeugt und sortiert. Für eine beidseitige Kotextbreite der Größe zwei wird folgendes Array zurückgegeben:

160 Und damit die potentiell „qualitativste“ Information

161 Implizit ließe sich die Breite des Kotexts aus der Länge des Stringeintrags aus den Kotextspalten folgern, doch würde dies eine unnötige Rechenbelastung für den Datenbankserver bedeuten.

{[2,2],[2,1],[1,2],[2,0],[1,1],[0,2],[1,0],[0,1]}

Das erste Byte innerhalb eines Feldes stellt den *leftwidth* Wert dar, das zweite Byte den *rightwidth* Wert. Die Sortierung ist in soweit nicht willkürlich, als zuerst die Kotexte mit großer Breite verglichen werden: die Summe des ersten Feldes , also 2 und 2, ergibt 4. Die des letzten Feldes ergibt 1. Dazwischen liegen die möglichen Abstufungen. Willkürlich und linguistisch nicht fundiert, ist die Reihenfolge der Felder gleicher Summe. Die Bevorzugung des linken Kotextes in [2,0] gegenüber dem rechten in [0,2] ist zufällig.

In *QueryEngine* wird das vom Client gesendete Wort daraufhin überprüft, ob es bereits in der Ergebnistabelle *calcs* vorhanden ist. Wenn es keinen Eintrag gibt, werden zuerst alle Kotexte mit beidseitiger Tiefe von 2 des Wortes [*text*] gesucht und gespeichert. Es wird überprüft, ob eine ausreichende Menge Kotexte für die Paradigmenberechnung vorhanden ist. Wenn nicht, wird aller Kotext einer linken Tiefe von 2 und einer rechten Tiefe von 1 akkumuliert et cetera.

Die erhaltenen Kotexte werden auf die Datenbanktabelleneinträge „gemappt“, das heißt es wird überprüft, ob Worte mit identischen Kotexten existieren. Die Anzahl der übereinstimmenden Kotexte wird gespeichert – diese Größe ist wichtig für die Evaluation der Relationen. Wie diese Evaluation durchgeführt wird, ist im folgenden Kapitel erläutert.

3.3.3.4. Berechnug der Relationen

Im Folgenden werden die nötigen Berechnungen theoretisiert. Im Anschluss daran werden die tatsächlich in PaGe umgesetzten Berechnungen dargelegt.

Die Beschränkung in der Berechnung (und Darstellung) der Relationen auf Worthäufigkeit sowie auf Sub- und Superordination ist dem Prinzip der möglichst starken Vereinfachung geschuldet. Die Schnittmengendarstellung zwischen zwei Worten sollte die wichtigste paradigmatische Eigenschaft wiedergeben.[162]

Wie kann es sein, dass das Wort *Roman* aus Abbildung 3 dem Wort *Buch* subordiniert ist, obwohl es mehr differente Okkurrenzen als dieses besitzt? Es müsste eine quantitative

162 Momentan ist nur eine Berechnung wie durch Abbildung 5 repräsentiert möglich, nicht etwa eine Berechnung, die direkt im Vergleich mehrerer Worte mündet (etwa wie Abbildung 6). Eine Hypothese an dieser Stelle ist, dass sich diese Berechnungen im Nachhinein auf Grundlage der Zweier-Beziehungen eruieren ließe, wodurch sich ein n-dimensionaler, und zwar vorher nicht festgelegter(!), Vektorraum ergäbe. Dazu müssten die ermittelten Worte untereinander in Verhältnis gesetzt werden.

Untersuchung der Kookkurrenzen selbst vorgenommen werden, aus der folgt, dass die Kotexte von *Buch* allgemeiner sind als die von *Roman. Buch* wäre dann eine Superordination, oder Hyperonym, von *Roman*.

Ein hypothethisches Beispiel soll zeigen, wie ermittelt werden könnte, dass das *Buch* allgemeiner ist als *Roman.* Die Summe der differenten Okkurrenzen der Kookkurrenzen sind für *Roman* deutlich höher als die von *Buch.* Zurückzuführen ist dies auf die häufige Verwendung eines Eigennamens im Kotext von *Roman,* während der Kotext von *Buch* allgemeiner ist und sich damit häufiger wiederholt.

Aus Performanzgründen ist es sinnvoll, die Analyse auf die breitesten Kotexte zu beschränken. Je *breiter ein Kotext, desto spezieller.* Es werden folglich nur die Kotexte größter Breite von *Roman* und *Buch* sortiert und einer Okkurrenzanalyse unterzogen. Allerdings könnte es dabei zu einer Verschiebung kommen, falls des einen Wortes Kotext schlimmstenfalls gar keine maximale Breite hat, während des anderen Wortes Kotext stets die maximale Breite ausfüllt. In der Praxis ist dies aber unwahrscheinlich, obwohl hier sicherlich empirische Untersuchungen notwendig sind.

In PaGe 0.3 berechnet sich die Ähnlichkeit eines Wortes wie folgt:

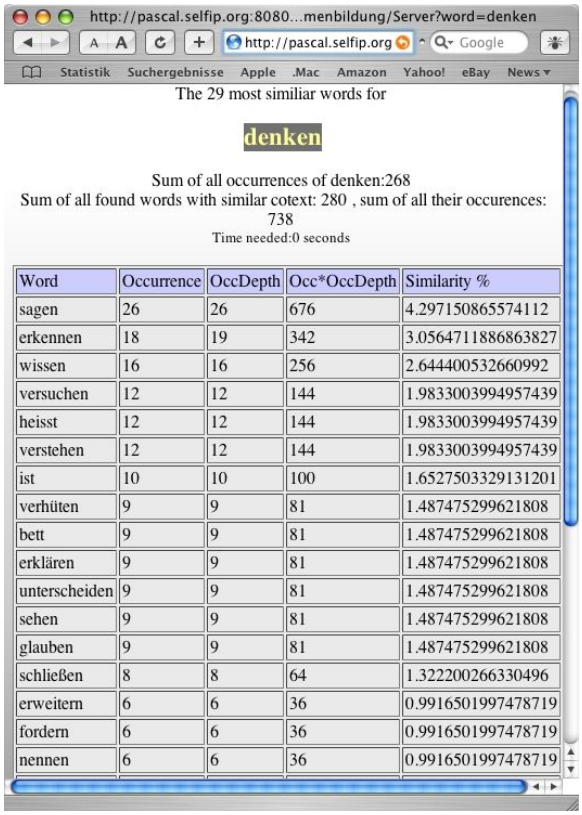
In der Bean *org.selfip.pascal.paradigmen.ParadigmBean* wird ein Wort zwischengespeichert, wenn es mindestens einen gleichen Kotext aufzuweisen hat. In der Variablen *existence* wird die Anzahl der Kotexte abgelegt. In *existenceDepth* sind die Kotextbreiten summiert. Ein Wert errechnet sich aus der Addition von *leftwidth* und *rightwidth*. Bei einer Kotexttiefe von 10 sind also Werte zwischen 1 und 10 möglich.[163]

Die Attribute *existence* und *existenceDepth* sind die einzigen Werte der Worte, die in die Ergebnistabelle *calcs* verspeichert werden. Der Ähnlichkeitswert berechnet sich dann wie folgt: Zuerst wird *existence* und *existenceDepth* vom abgefragten Wort multipliziert. Vom Ergebnis wird die zweite Wurzel gezogen. Dasselbe wird mit einem in Relation zu setzendem Wort durchgeführt. Der prozentuale Ähnlichkeitswert berechnet sich durch:

$$\frac{100 * \text{vergleichendesWort.existence} * \text{vergleichendesWort} * \text{existenceDepth}}{\text{abgefragtesWort.existence} * \text{abgefragtesWort.existenceDepth}}$$

Die Formel taugt nur für relative Beziehungen der zu vergleichenden Worte untereinander und ist weit davon entfernt, eine Schnittmenge der Worte zu bilden, die mit der Intuition des Benutzers in etwa übereinstimmen würde. Die Berechnung ist also noch nicht adäquat

163 Besser wäre eine höhere Gewichtung von breiteren Kotexten, etwa durch die Summierung von (leftwidth+rightwidth) * (leftwidth+rightwidth). Als Ergebnis sind natürliche Zahlen zwischen 1 und 100 möglich (bei einer Kotextbreite von 10). Die Gewichtung ist nicht-linear.



http://pascal.selfip.org:8080...menbildung/Server?word=denken

Statistik Suchergebnisse Apple .Mac Amazon Yahoo! eBay News

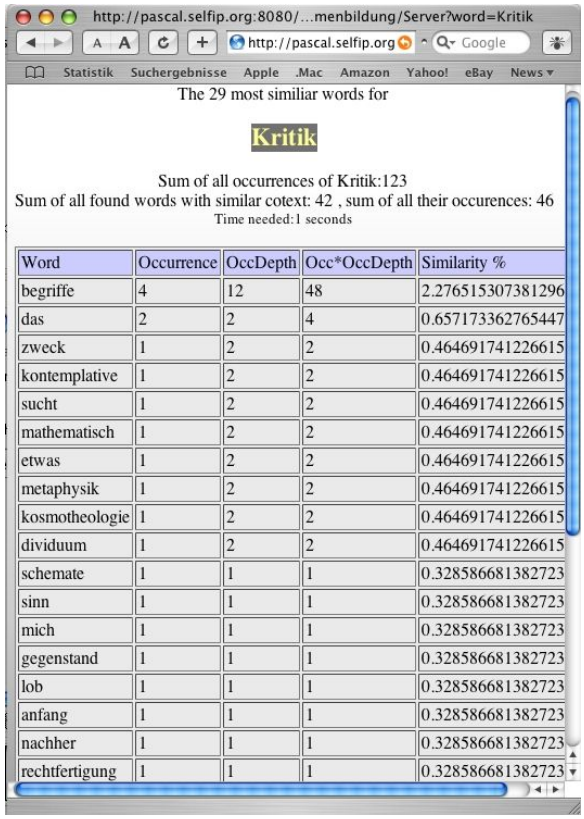
The 29 most similiar words for

denken

Sum of all occurrences of denken:268
Sum of all found words with similar cotext: 280 , sum of all their occurences: 738
Time needed:0 seconds

Word	Occurrence	OccDepth	Occ*OccDepth	Similarity %
sagen	26	26	676	4.297150865574112
erkennen	18	19	342	3.0564711886863827
wissen	16	16	256	2.644400532660992
versuchen	12	12	144	1.9833003994957439
heisst	12	12	144	1.9833003994957439
verstehen	12	12	144	1.9833003994957439
ist	10	10	100	1.6527503329131201
verhüten	9	9	81	1.487475299621808
bett	9	9	81	1.487475299621808
erklären	9	9	81	1.487475299621808
unterscheiden	9	9	81	1.487475299621808
sehen	9	9	81	1.487475299621808
glauben	9	9	81	1.487475299621808
schließen	8	8	64	1.322200266330496
erweitern	6	6	36	0.9916501997478719
fordern	6	6	36	0.9916501997478719
nennen	6	6	36	0.9916501997478719

Abbildung 8: denken



http://pascal.selfip.org:8080/...menbildung/Server?word=Kritik

Statistik Suchergebnisse Apple .Mac Amazon Yahoo! eBay News

The 29 most similiar words for

Kritik

Sum of all occurrences of Kritik:123
Sum of all found words with similar cotext: 42 , sum of all their occurences: 46
Time needed:1 seconds

Word	Occurrence	OccDepth	Occ*OccDepth	Similarity %
begriffe	4	12	48	2.276515307381296
das	2	2	4	0.657173362765447
zweck	1	2	2	0.464691741226615
kontemplative	1	2	2	0.464691741226615
sucht	1	2	2	0.464691741226615
mathematisch	1	2	2	0.464691741226615
etwas	1	2	2	0.464691741226615
metaphysik	1	2	2	0.464691741226615
kosmotheologie	1	2	2	0.464691741226615
dividuum	1	2	2	0.464691741226615
schemate	1	1	1	0.328586681382723
sinn	1	1	1	0.328586681382723
mich	1	1	1	0.328586681382723
gegenstand	1	1	1	0.328586681382723
lob	1	1	1	0.328586681382723
anfang	1	1	1	0.328586681382723
nachher	1	1	1	0.328586681382723
rechtfertigung	1	1	1	0.328586681382723

Abbildung 9: Kritik

und bedarf weiterer Evaluierung. Abbildung 8 stellt das Ergebnis des default eingestellten Wortes *denken* dar. Es wurde auf Grundlage des voreingestellten Korpus *deutsch* aus dem Gutenbergprojekt berechnet:[164] Die Werte in der Spalte *Okkurrenzen* weichen nicht oder wenig von den Werten in der Spalte *OccDepth* ab. Dies zeigt, dass die gefundenen, übereinstimmenden Kotexte eine Breite von lediglich *eins* aufweisen. Für den ersten Eintrag von Abbildung 9 ergibt sich diversifizierteres Bild: hier liegt die durchschnittliche Kotextbreite bei 3.

4. Zusammenfassung der Ergebnisse und Ausblick

»Each word when used in a new context is a new word«

Firth

Die bereits in vorigen Kapiteln erwähnten Planungen und Ideen betreffen vor allem die

164 Das Korpus besteht aus „Immanuel Kant: Kritik der reinen Vernunft“, „Georg Wilhelm Friedrich Hegel: Rede zum Schuljahresabschluß am 29. September 1809“, „Jakob Michael Reinhold Lenz: Der Landprediger“, „Friedrich Nietzsche: Menschliches, Allzumenschliches.Ein Buch für freie Geister“, „Friedrich Nietzsche: Also sprach Zarathustra.Ein Buch für Alle und Keinen“, „William Shakespeare: Macbeth. Übersetzt von Dorothea Tieck“. Das Korpus besteht aus insgsamt 27.764 verschiedenen Worten. Die Anzahl der Datenbankeinträge ist 386.363 .

Verbesserung des Algorithmus und die Darstellung der Ergebnisse. An dieser Stelle werden diese Bemerkungen noch einmal aufgegriffen und erweitert. Zudem werden weitere mögliche Einsatzmöglichkeiten von PaGe dargelegt.

Die Verbesserungen, die das Webinterface betreffen, beziehen sich vor allem auf die bereits erwähnte, wünschenswerte Implementierung einer Fortschrittsanzeige für den Benutzer.[165] Aber auch die Korpusauswahl und weitere Einstellungen sind nicht optimal umgesetzt: Ursprünglich sollten Korpora über eine beliebige URL ansprechbar und auslesbar gemacht werden. Diese Funktion steht aber zur Zeit nicht zur Verfügung – es können also nur lokal vorhandene Korpora eingesetzt werden. Die Auswahl dieser Korpora geschieht im Code, der dann neu übersetzt werden muss. Auch die maximale Obergrenze der Berechnungszeit für ein Paradigma lässt sich nur durch Quellcodemanipulation erreichen. Über die Parametrisierung, die mit Hilfe der Web-Steuerung erfolgen würde, wäre diese Barriere überwunden, ähnlich wie dies durch die Implementation in *clientAdvanced.jsp* erfolgt ist: Dort kann mittels des Browsers eine andere als die Stand-arddatenbank, die in *client.jsp* zur Bestimmung von Wortparadigmen verwendet wird, ausgewählt werden.
Die größte Herausforderung liegt in der Optimierung der Datenbankabfragealgorithmen. Für das größte geteste Korpus, eine englische Textsammlung mit 94.526 verschiedenen Worten und ca. 12 Mio Kotexten, benötigt die folgende Abfrage, die nur ein einziges Ergebnis liefert, 30 sec. Rechenzeit auf einem 1GHz IBook:

```
pstmtGetLeftnrightcotextFromCentre.setString(1, wordId);
pstmtGetLeftnrightcotextFromCentre.setByte(2, leftwidth);
pstmtGetLeftnrightcotextFromCentre.setByte(3, rightwidth);
resultLeftnright=pstmtGetLeftnrightcotextFromCentre.executeQuery();
```

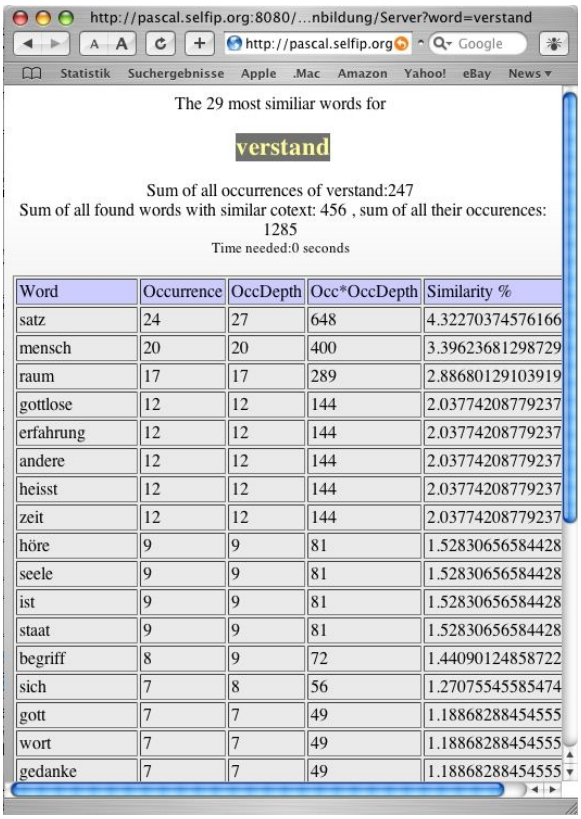
Eine mögliche Erklärung besteht darin, dass MySQL parallel arbeitet, also die Ergebnisse aller Spalten sammelt, um dann eine Schnittmenge zu bilden. Dies ist natürlich nicht im Sinne des Programmierers, da die Anzahl der Ergebnisse der Abfragen der Spalten mit dem Suffix *width* ziemlich groß ist, nämlich die Anzahl aller Spalten dividiert durch die Hälfte der maximalen Kotextbreite. Gedacht war an ein sequentielles Abarbeiten, also erst der Ermittlung aller Zeilen in denen die WordId vorkommt, und *danach* deren Selektion auf Grundlage des Kriteriums von *leftwidth*, gefolgt von der Einschränkung von *rigthwidth*. Sollte MySQL Abfragen parallel bearbeiten, könnte diesem Umsatnd so begegnet werden, dass, wie in Kapitel 3.3.3.2 dargelegt, für jede Kotextbreite eine eigene

165 Für den Entwickler werden Terminalausgaben erzeugt.

Tabelle angelegt wird, die dann direkt selektiert werden kann, ohne dass erst alle Spalteneinträge durchsucht werden müssten.
Auch folgende Abfrage dauert bei dem etwas größerem Korpus zu lange:

```
pstmtGetCentreFromLeftnrightcotext.setString(1, cotext);
resultSet = pstmtGetCentreFromLeftnrightcotext.executeQuery();
```

Möglicherweise ist die Codierung durch Triplets doch nicht optimal, da zuviele Hashkollisionen entstanden sein könnten. Hier sind weitere Tests durchzuführen.[166]



Word	Occurrence	OccDepth	Occ*OccDepth	Similarity %
satz	24	27	648	4.32270374576166
mensch	20	20	400	3.39623681298729
raum	17	17	289	2.88680129103919
gottlose	12	12	144	2.03774208779237
erfahrung	12	12	144	2.03774208779237
andere	12	12	144	2.03774208779237
heisst	12	12	144	2.03774208779237
zeit	12	12	144	2.03774208779237
höre	9	9	81	1.52830656584428
seele	9	9	81	1.52830656584428
ist	9	9	81	1.52830656584428
staat	9	9	81	1.52830656584428
begriff	8	9	72	1.44090124858722
sich	7	8	56	1.27075545585474
gott	7	7	49	1.18868288454555
wort	7	7	49	1.18868288454555
gedanke	7	7	49	1.18868288454555

Abbildung 10: Verstand

Die Berechnung der Verhältnisse von *verstand* (siehe Abbildung 10) haben ca. 10 Minuten in Anspruch genommen. Bei allen Beispielen zeigt sich, dass selten zwei Worte einen gemeinsamen größeren Kotext teilen. Das Einlesen, Umkodieren und Verspeichern des erwähnten Standardkorpus *deutsch* hat indes nur vier Minuten gedauert, für dieselbe Prozedur benötigte das etwa vierzigfach größere, englische Korpus *gutenbrg1* ca. 2 Stunden. Die Einleseprozedur verläuft also in linearer Zeit.

Neben den technischen Implikationen kann festgestellt werden, dass die Ergebnisse nicht immer dem Erwartungswert, der sich aus der theoretischen Grundlage ableitet, entsprechen. Eine mögliche Ursache liegt wohl in dem geringen Umfang des eingesetzten Korpus von nur etwa 400.000 Worten (gegenüber den anvisierten 100 Millionen). Auf diese Weise kommen zu wenig aussagekräftige, identische größere Kotexte vor. Hinzu kommt, dass der Auswertungsalgo-rithmus nicht sehr elaboriert ist, und vor allem die Berechnung der Ähnlichkeit weiter ausgearbeitet werden muss. Natürlich zeigen sich für die Worte interessante Ergebnisse, doch sind die als paradigmatisch bewerteten Worte teilweise sogar weit davon entfernt in einem bloß

166 MySQL erzeugt nicht automatisch einen Index. Die Erklärung für die schlechte Performanz geht auf das Fehlen dieses Index zurück. Folgende Zeile im MySQL-Interpreter erzeugt einen Binärbaum als Index für die aktuell benutzte Datenbank:
CREATE INDEX index_cotexts_leftnright ON cotexts(leftnright);
Der Geschwindigkeitszuwachs bewegt sich dadurch mindestens um den Faktor 100.

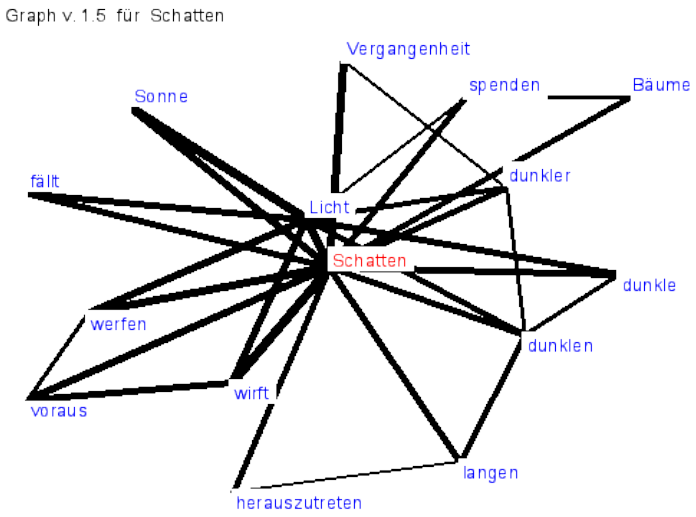


Abbildung 11: „Schatten“, Leipziger Wortschatzprojekt

lexikalischen Paradigma zu stehen: So rechnet PaGe etwa das Wort [*gottlos*] als zum Wortparadigma [*verstand*] zugehörig, und bewertet [*bett*] als in paradigmatischen Verhältnis zu [*denken*] sich befindend. Insgesamt sind die Aussagen eher mit denen des Leipziger Wortschatzprojektes zu vergleichen, wie die Grafik aus Abbildung 11 aus dem Projekt für das Wort [S*chatten*] zeigt. Es scheint sich eher ein semantisches Netzwerk zu bilden als eine bloße Hierarchie paradigmatischer Elemente. Es zeigt sich jedenfalls, dass sich PaGe als ein Instrument zur Erzeugung semantischen Wissens (im Sinne des Transkriptionsbegriffes von Jäger) einsetzen lässt.

Wenn die *paradigmatischsten*, also *modellhaftesten* Kotexte, ermittelt werden, könnten diese als Informationen zur syntaktische Subkategorisierung ins Lexikon aufgenommen werden, ähnlich wie es das Leipziger Wortschatzprojekt ausgeführt hat.

Lexikalische Bedeutung ist immer in Abhhängigkeit zu ihrem Kotext zu verstehen. Um diese Tatsache deutlicher werden zu lassen, ließen sich bipolare Kotexte anzeigen, einmal das modellhafteste und einmal ein ungewöhnlicheres Beispiel.[167]

Das selbstlernende System PaGe lässt sich auch auf anderer Ebene als der des ganzen signifiants anwenden. Paradigmatische Beziehungen bestehen auf allen syntagmatischen Ebenen. Sind z.B. alle Morpheme erkannt und durch entsprechende Analysewerkzeuge segmentiert, so ließen sich diese präprozessierten Daten dem PaGe als Ausgangsmaterial für paradigmatische Analysen übergeben. Inwiefern diese Zuordnungen Sinn machen, oder ob sie (wie Wiese 1999 anmerkt) eher zweifelhafter Natur sein würden, müsste eruiert werden.

PaGe besitzt sprachgenerierende Eigenschaften: Worte, die ein hohes Maß an Ähnlichkeit aufweisen, sind prinzipiell gegeneinander austauschbar. Doch kann sich, wie bereits erwähnt, auf die Adäquatheit nicht verlassen werden.

Den diese Arbeit einleitenden Fragestellungen, die Generierung von Taxonomien, Prototypen und Familienähnlichkeiten betreffend, kann mit Hilfe des PaGe unterstützend begegnet werden.

167 Das ungewöhnlichere Beispiel definiert sich aus seiner seltenen Erscheinung. Dies kann aber nicht das alleinige Kriterium sein. Weitere müssten erarbeitet werden.

5. Literaturverzeichnis

Bandholtz, Thomas. 2002. „Intelligente Agenten Taxonomie und Topic Maps – vom Stichwortverzeichnis zur Wissensnavigation.“ In: *XML Magazine & Web Service*. Nr. 2. Unter: http://www.xmlmagazin.de/itr/online_artikel/psecom,id,331,nodeid,69.html (letzter Zugriff: 21.02.2006)

Benden, Christoph. (im Entstehen). *Computergestützte unsupervisierte Morphemanalyse*. (Arbeitstitel) Dissertation, Universität zu Köln.

Bloomfield, Leonard. 1965 [1933]. *Language*. 8. Aufl., London: Allen and Unwin.

Brown, Peter/ Stephen dellaPietra/ Vincent dellaPietra. 1992. „An Estimate of the Upper Bound of the Entropy of English.“ In: Computational Linguistics. Vol. 18, S. 31-40.

Buyko, Ekaterina. 2005. „Numerische Repräsentation von Textkorpora für Wissensextraktion.“ Unter: www.spinfo.uni-koeln.de/forschung/papers/ebuyko_gldv.pdf (letzter Zugriff: 25.02.2006)

Cavanagh, Luke. 2002. „INSIDER PERSPECTIVE.“ In: *The Bulletin: Seybold News & Views On Electronic Publishing.* Vol. 7, No. 35 Unter: http://www.clearforest.com/whatsnew/IntheNewsArticles/SeyboldNews.html (letzter Zugriff: 21.02.2006)

Chandler, Daniel. 2003. *Semiotics for Beginners*. Unter: http://www.aber.ac.uk/media/Documents/S4B/sem05.html (letzter Zugriff: 28.02.2006)

Elger, Christian E./ Angela D. Friederici/ Christof Koch u.a.. 2004. „Das Manifest. Elf führende Neurowissenschaftler über die Gegenwart und Zukunft der Hirnforschung. Was wissen und können Hirnforscher heute?“ In: *Gehirn&Geist*, Heft 6. Unter: http://www.gehirnundgeist.de/blatt/det_gg_manifest (letzter Zugriff: 28.02.2006)

Franke, Herbert W. 1982. *Die geheime Nachricht. Methoden und Technik der Kryptologie. Die Geschichte um den unknackbaren Code*. Frankfurt/M.: Umschau.

Firth, John R. 1969. [1957]. *Papers in Linguistics 1934-1951*. London/ New York/ Toronto: Oxford University Press.

Hall, Marty/ Larry Brown. 2004. *Core. Servlets und JavaServer Pages*. 2. Aufl., München: Markt+Technik.

Harris, Zellig S. 1979. *Mathematical Structures of Language*. New York: Robert E. Krieger Publishing Company Huntington.

Hofstadter, Douglas R. 2001 [1991]. *Gödel, Escher, Bach: ein Endloses Geflochtenes Band.* 8. Aufl., München: dtv.

Geckeler, Horst. 1971. *Strukturelle Semantik und Wortfeldtheorie*. 2. Aufl., München: Fink.

Goll, Joachim/ Cornelia Weiß/ Frank Müller. 2001 [2000]. *Java als erste Programmiersprache. Vom Einsteiger zum Profi*. 3. Aufl., Stuttgart/Leipzig/Wiesbaden: Teubner.

Jäger, Ludwig. 2004. „Die Verfahren der Medien: Transkribieren – Adressieren – Lokalisieren". In: Jürgen Fohrmann/ Erhard Schüttpelz (Hrsg.): *Die Kommunikation der Medien*. Tübingen: Niemeyer, S. 69-80.

Johnson-Laird, Philip. 1993 [1988]. *The Computer and The Mind. An introduction to Cognitive Science*. 6. Aufl., London: Fontana: Press.

Kofler, Michael. 2001. *Linux. Installation, Konfiguration, Anwendung*. 5. erweiterte Aufl., München: Addisson-Wesley.

Kuester, Martin. 2001. „Signifikant (frz. s*ignifiant*) und Signifikat (frz. *Signifié*)." In: Ansgar Nünning (Hrsg.): Metzler Lexikon. Literatur- und Kulturtheorie. Ansätze-Personen-Grundbegriffe. 2. überarb. Aufl., Stuttgart, Weimar: Metzler, S. 584.

Lakoff, George. 1990 [1987]. *Women, Fire and Dangerous Things. What Categories Reveal about the Mind*. Paperback edition. Chicago, London: The University of Chicago Press.

Lenerz, Jürgen. 1999/2000 . *Einführung in die Semantik*. Unter: http://janeden.org/pdf/2182.pdf (letzter Zugriff: 26.02.2006)

Luft, Alfred L./ Rudolf Kötter. 1994. *Informatik - eine moderne Wissenstechnik*. Mannheim/ Leipzig/ Wien/ Zürich: BI.

Luhmann, Niklas. 1993 [1981] „Veränderung im System gesellschaftlicher

Kommunikation und die Massenmedien.“ In: *Soziologische Aufklärung 3. Soziales System, Gesellschaft und Organisation*. 3. Aufl., Opladen: VS Verlag für Sozialwissenschaften, S. 309-320.

Lyons, John. 1975 [1968]. *Einführung in die moderne Linguistik*. 4. Aufl., München: C.H. Beck.

Manning, Christopher D./ Schütze, Hinrich. 1999. *Foundations of statistical natural language processing*. Massachusetts Institute of Technology: The MIT Press.

Mehler, Alexander. 2004. „Konnotative Textbedeutungen. Zur Modellierung struktureller Aspekte der Bedeutungen von Texten.“ In: Reinhard Köhler (Hrsg.): *Korpuslinguistische Untersuchungen zur quantitativen und systemtheoretischen Linguistik,* S. 223-350. Unter: http://ubt.opus.hbz-nrw.de/volltexte/2004/279/ (Dokument 10.pdf) (Letzter Zugriff: 21.02.2006)

Putnam, Hilary. 1999. *Repräsentation und Realität.* Frankfurt/M.: Suhrkamp.

Reese, George. 2003. *MySQL*. Köln: O'Reillys.

Regionales Rechenzentrum für Niedersachen (Hrsg.). 2002. *SQL Grundlagen und Datenbankdesign*. Universität Hannover.

Rolshoven, Jürgen. 2002. *Zum Transfer in der maschinellen Übersetzung*. Unter: http://www.spinfo.uni-koeln.de/lehre/HS_Rolshoven/papers/Transfer.rtf (letzter Zugriff: 21.02.2006)

Sasse, Hans-Jürgen. 2003. „Vom Strukturalismus zur generativen Grammatik.“ In : Universität zu Köln, Institut für Linguistik, Abteilung Allgemeine Sprachwissenschaft (Hrsg.): *Theorien und Modelle. Materialien zum Proseminar.* Köln, S. 69-77.

Saussure, Ferdinand de. 1967 [1916]. *Grundfragen der allgemeinen Sprachwissenschaft*. 2. Aufl., Berlin: Walter de Gruyter & Co.

Schwiebert, Stephan. 2004. *Entwurf eines agentengestützten Systems zur Paradigmenbildung*. Unter: www.spinfo.uni-koeln.de/forschung/papers/MA_SOG_web.pdf (letzter Zugriff: 21.02.06)

Spitzer, Manfred. 2000. *Geist im Netz. Modelle für Lernen, Denken und Handeln*. Heidelberg/Berlin: Spektrum.

Stichweh, Rudolf. 1997. „Inklusion/Exklusion, funktionale Differenzierung und Weltgesellschaft.“ In: *Soziale Systeme. Zeitschrift für Soziologie*. Jg. 3. Heft 1, S. 123-136.

Thimm, Katja. 2003. „Jeden Tag ein neues Universum.“ In: *Der Spiegel*. Jg. 47, Nr. 43, S. 98-210.

Ullenboom, Christian. 2004. *Java ist auch eine Insel. Programmieren für die Java 2. Plattform in der Version 5*. 4. erweiterte Aufl., Bonn: Galileo Computing**.** Unter: http://www.galileocomputing.de/757?GPP=opjiV (letzter Zugriff: 26.02.2006)

Vater, Heinz. 1996 [1994]. *Einführung in die Sprachwissenschaft*. 2. Aufl., München: Fink.

Weizenbaum, Joseph. 2001. *Computermacht und Gesellschaft*. Frankfurt/M.: Suhrkamp.

Wiese, Bernd. 1999. „Unterspezifizierte Paradigmen. Form und Funktion in der pronominalen Deklination.“ In: *Linguistik online* 4, 3/99. Unter: http://www.linguistik-online.de/3_99/wiese.html (letzter Zugriff: 25.02.2006)

Wirth, Niklaus. 1983 [1975]. *Algorithmen und Datenstrukturen*. 3. überarb. Aufl., Stuttgart: Teubner.

Websites

http://de.wikipedia.org/wiki/Anwendungsprogramm (letzter Zugriff: 26.02.2006)

http://de.wikipedia.org/wiki/Colossus (letzter Zugriff: 21.02.2006)

http://de.wikipedia.org/wiki/Dokumentenmanagement (letzter Zugriff: 21.02.2006)

http://de.wikipedia.org/wiki/Hash (letzter Zugriff: 28.02.2006)

http://de.wikipedia.org/wiki/Hauptseite (letzter Zugriff: 21.02.2006)

http://de.wikipedia.org/wiki/Port_%28Protokoll%29 (letzter Zugriff: 26.02.2006)

http://de.wikipedia.org/wiki/Server (letzter Zugriff: 26.02.2006)

http://de.wikipedia.org/wiki/Synonym (letzter Zugriff: 08.01.2006)

http://de.wikipedia.org/wiki/Turing-Test (letzter Zugriff: 21.02.2006)

http://en.wikipedia.org/wiki/Model-view-controller (letzter Zugriff: 26.02.2006)

http://en.wikipedia.org/wiki/Three-tier_(computing) (letzter Zugriff: 26.02.2006)

http://fedoraproject.org letzter (Zugriff: 26.02.2006)

http://forums.mysql.com/ (letzter Zugrif: 26.02.2006)

http://freie-software.bpb.de/ (letzter Zugriff: 26.02.2006)

http://java.sun.com/j2se/ (letzter Zugriff: 26.02.2006)

http://java.sun.com/j2ee/1.4/docs/tutorial/doc/ (letzter Zugriff: 26.02.2006)

http://wiki.apache.org/tomcat/ (letzter Zugriff: 26.02.2006)

http://wortschatz.uni-leipzig.de/ (letzter Zugriff: 21.02.2006)

http://www.apple.com/ (letzter Zugriff: 26.02.2006)

http://www.apple.com/downloads/macosx/networking_security/henwen.html
(letzter Zugriff: 26.02.2006)

http://www.cyc.com (letzter Zugriff: 26.02.2006)

http://www.gnu.org/licenses/gpl.html (letzter Zugriff: 26.02.2006)

http://www.gnu.org/philosophy/free-sw.de.html (letzter Zugriff: 26.02.2006)

http://www.gutenberg.org/ (letzter Zugriff: 21.02.2006)

http://www.knowledgebusiness.com/knowledgebusiness/ (letzter Zugriff: 21.02.2006)

http://www.mpi.nl/world/ (letzter Zugriff: 26.02.2006)

http://www.mysql.de/ (letzter Zugriff: 26.02.2006)

http://www.openssh.com/ (letzter Zugriff: 26.02.2006)

http://www.snort.org/ (letzter Zugriff: 26.02.2006)

http://www.torsten-horn.de/techdocs/webanwendungen.htm (letzter Zugriff: 26.02.2006)

http://www.ttt.org/theory/barker.html (letzter Zugriff: 26.02.2006)

http://www.w3.org/Protocols/ (letzter Zugriff: 26.02.2006)

http://www.unicode.org (letzter Zugriff: 26.02.2006)

Erklärung

Hiermit versichere ich, dass ich diese Magisterarbeit selbständig verfasst und keine andere als die angegebenen Quellen und Hilfsmittel benutzt habe. Die Stellen meiner Arbeit, die dem Wortlaut oder dem

Sinn nach anderen Werken entnommen sind, habe ich in jedem Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht. Dasselbe gilt sinngemäß für Tabellen, Karten und Abbildungen.

_____

Unterschrift