Seminar für Computerlinguistik
Universität Zürich

Sommersemester 2002

Lizentiatsarbeit
bei
Prof. Dr. Michael Hess

Automatische E-Mail Beantwortung

Theoretische Grundlagen und Aspekte praktischer Anwendung

Björn Metzinger
Käppelistrasse 24
4600 Olten
b.metzinger@access.unizh.ch

Abgabedatum: 8. Mai 2002

Inhalt

Abbildungen **2**

Tabellen **2**

1 Einleitung und Überblick **3**
1.1 Warum automatische E-Mail Beantwortung? 3
1.2 Begriff und Einordnung 4
1.3 Datenlage 4
1.3.1 Dokumentation 4
1.3.2 Anwendungen 5

2 Modelle und Verfahren für Indexierung und Retrieval freier Texte **6**
2.1 Textretrieval im Überblick 6
2.1.1 Komponenten und Modelle im Textretrieval 6
2.1.2 Abhängigkeiten und Paradigmen 7
2.1.3 Evaluation von TR-Systemen 8
2.2 Anwendungstypen im Textretrieval 9
2.2.1 Document Retrieval 9
2.2.2 Informationsextraktion 9
2.2.3 Antwortextraktion 10
2.2.4 Fragebeantwortung 10
2.3 Indexierungsmodelle und -verfahren 11
2.3.1 Einwort-Indexierung 11
2.3.2 Linguistische Indexierung 12
2.4 Sprachanalyse in der linguistischen Indexierung 14
2.4.1 Lexikalische Ressourcen 14
2.4.2 Erkennung der Analyseeinheiten 14
2.4.3 Lexikalisch-Morphologische Analyse 14
2.4.4 Wortklassenkennzeichnung 15
2.4.5 Erkennung von Stichworten, Eigennamen und anderen Entitäten 15
2.4.6 Syntaktische Analyse 16
2.4.7 Disambiguierung und Referenz-Auflösung 17
2.5 Repräsentationsmodelle und Index-Generierung 18
2.5.1 Index-Listen und -Verzeichnisse 18
2.5.2 Freie Kennzeichnung 19
2.5.3 Hierarchische Repräsentation 20
2.5.4 Templates 22
2.5.5 Logische Formen 23
2.6 Retrievalmodelle und -verfahren 24

3 Grammatikmodellierung für die Sprachanalyse **25**
3.1 Grammatikmodellierung mit Regulären Ausdrücken 25
3.1.1 Stichwortsuche 25
3.1.2 Grammatikmodellierung und Fragebeantwortung mit Pattern-Matching 26
3.1.3 Kaskadierung regulärer Grammatiken (FST-Kaskaden) und Pattern-Matching 27
3.2 Regel- oder symbolorientierte Verfahren 29
3.2.1 Funktionale Grammatiken am Beispiel der Dependenz-Grammatik 29
3.2.2 Gewichtete und probabilistische Grammatiken 30
3.3 Stochastische Systeme und maschinelle Lernverfahren 31
3.4 Semantisch orientierte Analyse-Verfahren 32
3.5 Fazit: Welches Verfahren für welche Anwendung? 32

4 Anwendungsbeispiele für Textretrieval und E-Mail Beantwortung **33**
4.1 Wissenschaftliche Systeme 33
4.1.1 FASTUS – ein IE-System für englische Texte 33
4.1.2 Zweistufiger partieller Chart-Parser für deutsche Textkorpora 34
4.1.3 „Divide-and-Conquer“ Strategie für seichtes Parsen deutscher Texte 35
4.1.4 EXTRANS: Vollständige Syntaxanalyse und semantische Indexierung 37
4.2 Kommerzielle E-Mail Beantwortungssysteme 38
4.2.1 Kiwilogic Lingubot 39
4.2.1.1 Überblick 39
4.2.1.2 Grammatikmodellierung und Retrieval 39
4.2.1.3 Erweiterbarkeit 40
4.2.1.4 Evaluation 40
4.2.2 eGain Mail 40
4.2.2.1 Überblick 40
4.2.2.2 E-Mail Verarbeitung 41
4.2.2.3 Sprachverarbeitung, Indexierung und Retrieval 42
4.2.2.4 Evaluation 43
4.2.3 Firepond eService Performer Email Assistance 43
4.2.3.1 Übersicht 44
4.2.3.2 Formale und inhaltliche Repräsentation als DOM 45
4.2.3.3 Sprachverarbeitung 46
4.2.3.4 Regel-Lexikon und Ermittlung der intendierten Bedeutung 46

4.2.3.5 Stochastische Bedeutungszuweisung 47
4.2.3.6 System-Aktionen und Matching-Regeln 48
4.2.3.7 Evaluation 49
4.3 Fazit: Errungenschaften praxisorientierter Systeme 50

5 Strategien für die automatische E-Mail Beantwortung 51
5.1 Analyse 51
5.1.1 Ausgangslage: Warum werden bestimmte Fragen so schlecht beantwortet? 51
5.1.2 Ansatz: Erkennung und Behandlung unterschiedlicher Fragetypen 53
5.2 Strategien der Fragebeantwortung 55
5.2.1 Lexikalisch-orientierte Strategien 55
5.2.1.1 Entfernung inhaltsarmer Wörter und Konstrukte 55
5.2.1.2 Sukzessive Wortkategorie-Filterung 55
5.2.1.3 Zertrennung von Komposita 56
5.2.1.4 Lexikalische Frage- und Antwort-Indikatoren 56
5.2.1.5 Anfrage-Erweiterung durch Synonyme und Hypernyme 58
5.2.1.6 Such-Restriktionen 59
5.2.2 Syntaktische Beziehungen als Frage- und Antwort-Indikatoren 59
5.2.2.1 Wort-Distanz vs. Phrasenzugehörigkeit 59
5.2.2.2 Syntaktische Kategorien als funktionale Indikatoren 60
5.2.2.3 Lexikalisch-syntaktische Frage- und Antwortmuster und Relationen 60
5.2.3 Dynamische Gewichtung 62
5.2.3.1 Gewichtung von Treffer-Passagen 62
5.2.3.2 Gewichtung von Treffer-Termen 63
5.2.4 Meta-Suchstrategien 64
5.2.4.1 Strukturelle Einordnung von Suchstrategien 64
5.2.4.2 Gewichtung von Suchstrategien 64
5.3 Fazit 65
5.3.1 Zusammenfassung: Jedem Fragetyp seine Suchstrategie? 65
5.3.2 Einschränkende Faktoren 66

6 Ausblick 67

7 Referenzen 69
7.1 Literatur 69
7.2 Software 72

Abbildungen

Abbildung 1: Verfahrens-Modelle im Textretrieval und deren Realisation als Komponente 6
Abbildung 2: Architektur und Dependenzen der Komponenten eines Textretrieval Systems 7
Abbildung 3: Präzision und Ausbeute 8
Abbildung 4: Strukturelle Interaktion der Komponenten im syntaktischen Textretrieval 12
Abbildung 5: Dynamische Interaktion bei der Korpusindexierung (offline) 13
Abbildung 6: Dynamische Interaktion bei Anfrageindexierung und Retrieval (online) 13
Abbildung 7: Ebenen einer Finite-State-Kaskade 27
Abbildung 8: Übergangsrelationen auf Basis einer regulären Grammatik 27
Abbildung 9: Verarbeitungsstufen im Indexierungs- und Retrievalprozess in EXTRANS 37
Abbildung 10: Arbeitsweise von Lingubot 39
Abbildung 11: Benutzeroberfläche des Lingubot Creator 39
Abbildung 12: eGain Mail Benutzeroberfläche 41
Abbildung 13: eGain Global Settings 41
Abbildung 14: eGain Rule Administration 42
Abbildung 15: eGain Phrase Rule Editor 42
Abbildung 16: eGain Auto Reply Editor 43
Abbildung 17 : Benutzeroberfläche Email Assistance 44
Abbildung 18: Architektur-Übersicht Email Assistance 44
Abbildung 19 : Erweiterung des "Message Data" Objekts (M) durch Zufügen neuer Attribute (A) 45
Abbildung 20 : Text-Muster (•), Features (F) und Intents (I) eines benutzerdefinierten Regel-Lexikons 46
Abbildung 21 : Definition regulärer Text-Muster (T) 47
Abbildung 22 : Definition von Aktions-Regeln (R= Rule, I= Intent, A= Action) 49
Abbildung 23 : Definition von Custom-Regeln 49

Tabellen

Tabelle 1: TREC-Korpora mit heterogener Evaluationsgrundlage 8
Tabelle 2: Konzeptuelle Fragetypologie und Test-Ergebnisse 51
Tabelle 3: Frage- und Antwort-Indikatoren für verschiedene semantische Fragetypen 54
Tabelle 4: Gewichtung von Suchstrategien in verschiedenen Test-Szenarien 64
Tabelle 5: Suchstrategien für verschiedene Fragetypen 65

1 Einleitung und Überblick

1.1 Warum automatische E-Mail Beantwortung?

> „Natural language interpretation will transform the way people use the internet." [Kiwilogic 00]

Innert weniger Jahre hat sich das World Wide Web von einer primär wissenschaftlichen Kommunikationsplattform zum omnipräsenten Mehrkanal-Medium gemausert. Vornehmlich Grossfirmen, in zunehmendem Masse aber auch kleinere bis mittlere Unternehmen und öffentliche Institutionen sehen sich im Angesicht der heutigen De-facto-Notwendigkeit elektronischer Erreichbarkeit nicht nur mit steigenden Investitionsaufwendungen fachpersoneller und finanzieller Natur, sondern insbesondere auch mit einer bislang unkontrollierbaren regelrechten „E-Mail-Flut" von Kundenanfragen konfrontiert.

Obschon die kommunikationstechnischen Möglichkeiten des Internet eine Fülle von Interaktionskanälen zwischen Firma und Kunde vorgebracht haben – Internet Telephonie, Chat, Voice-over-IP, Video-Conferencing, Fax, dynamisches kundenspezifisches HTML – nimmt das E-Mail nach wie vor die Führung der elektronischen Zweiweg-Kommunikation ein. Viele Firmen können jedoch nicht erfolgreich damit umgehen (vgl. [Lambert 98]).

Derzeit werden elektronische Anfragen selbst bei renommierten Grossfirmen grösstenteils noch manuell gehandhabt, also von mehr oder minder spezialisiertem Personal gesichtet, weitergeleitet und – falls überhaupt – Tage oder Wochen später beantwortet. So blieben nach einer Branchenstudie bei Finanzfirmen 1998 denn auch um die 30 Prozent aller Kunden-E-Mails unbeantwortet (vgl. [Duvall 98]). Mit zunehmendem Mail-Aufkommen – für 2001 werden gegen 12 Mia. E-Mail-Meldungen weltweit geschätzt – vergrössert sich die Kluft zwischen Kundenerwartung und Firmenservice, insbesondere was Qualität und Geschwindigkeit von Reaktionen auf elektronische Anfragen anbelangt. Ohne eigenes Verschulden schaden Firmen ihrem Ruf und ihrer Profitabilität (vgl. [Lambert 98]):

> In today 's world, using e-mail to transact business is a fact of life. If your business can 't handle these transactions, you stand to lose one customer after another to competitors who can. And you 'll miss out on a growing group of potential customers who do business exclusively online. [Quintus 00]

Branchenkenner sehen in der Vermarktung einerseits und dem Einsatz anderseits von E-Mail-Verarbeitungssoftware grosse Wachstumschancen (vgl. [Duvall 98]). Die Interpretation natürlichsprachlicher Texte erlaubt eine einfachere und „natürliche" Kommunikation zwischen Firma und Kunden und nutzt das volle Potential von e-commerce – so die Hersteller (vgl. [Kiwilogic 00]). Allerdings findet sich in den einschlägig optimistischen Produktbeschreibungen kommerzieller Systeme kaum detaillierte Information über deren Funktionalität, Algorithmen und Trefferquoten.

Diese Lizentiatsarbeit möchte hier etwas „Licht ins Dunkel" bringen. Dabei wird das interdisziplinäre Gebiet der E-Mail Beantwortung einerseits theoretisch etwas beleuchtet, wobei vorderrangig dessen computerlinguistische Praxis-Relevanz fokussiert wird.

Anderseits soll die Arbeit als eine Art „Bestandesaufnahme" auch einen praktischen Zugang zur Problematik darstellen, indem die auf dem Markt befindlichen Produkte beschrieben, verglichen und getestet (falls möglich), sowie auf computerlinguistische Kriterien (Relevanz, Funktionalität, Robustheit, Performanz, Modularität) hin diskutiert werden.

In einem abschliessenden Teil wird versucht, aus den hieraus gewonnenen Erkenntnissen sowie den Erfahrungen eines eigens entwickelten Antwortextraktions-Systems Strategien und Verbesserungsvorschläge für die automatische E-Mail Beantwortung zu finden, welche die erfolgversprechendsten Ansätze der zuvor diskutierten Systeme kombinieren und erweitern.

1.2 Begriff und Einordnung

Die Idee zu einer maschinellen Beantwortung von E-Mail Texten entspricht einem durch jüngste Technologie-Entwicklungen geförderten Bedürfnis und ist dementsprechend noch schwach „erforscht". Diese interdisziplinäre Problemstellung fordert wissenschaftliche Anstrengungen auf den Gebieten der Informatik, der (Computer-) Linguistik und peripher sicherlich auch der (Tele-) Kommunikationstechnologie und der Publizistikwissenschaft.

Das Online-Computerlexikon „Webopedia" umschreibt den Begriff „E-Mail" wie folgt (vgl. [Webopedia 00]):

> Short for electronic mail, the transmission of messages over communications networks. [...] enabling users to send electronic mail anywhere in the world. Companies that are fully computerized make extensive use of e-mail because it is fast, flexible, and reliable. [...] In recent years, the use of e-mail has exploded. By some estimates, there are now 25 million e-mail users sending 15 billion messages per year.

Alle Ansätze und Konzepte, online übermittelte Texte (darunter E-Mail, HTML-Formulare u.a.m.) *maschinell automatisiert auszuwerten und entsprechende System-Reaktionen zu veranlassen* werden im folgenden zusammenfassend als *(automatische) E-Mail Beantwortung* bezeichnet.

Bei den zu beantwortenden Inhalten handelt es sich grundsätzlich um *„Real World" Texte* in ihrer vollen Komplexität und nicht um irgendwelche Ideal- oder Beispielgrammatiken, was sich unter anderem äussern kann in Unterschieden bezüglich (vgl. [Ganz-Blättler/Süss 96], [Carstensen et al. 01] sowie [Metzinger 00]):

- *Sprachtyp:* z.B. konfigurational vs. Free-word-order
- *Sprachcode bzw. -stil:* elaboriert vs. restringiert, Fach- vs. Umgangssprache
- *Lexikalisch-syntaktische Komplexität:* Schachtelungen, Abkürzungen, Komposita etc.
- *Grammatikalität / Wohlgeformtheit*
- *Kontextgehalt / -tiefe*

Es wäre daher (am besten anhand „echter" Beispiele) zu untersuchen, ob E-Mail Texte allgemein oder zumindest domänen- oder branchenspezifisch gewissen sprachlichen Restriktionen unterliegen, beispielsweise ein bestimmtes Vokabular oder gar bestimmte Syntax-Regeln verwenden. Derartige Einschränkungen würden nämlich die Entwicklung eines akkuraten, robusten Analysewerkzeugs erleichtern[1] (vgl. 4.1, 4.2 und 5.1.1).

Da elektronisch übermittelter Inhalte wie E-Mail Anfragen als „freie Texte" (Real-World-Texte) angesehen werden können, deren (potentielle) Antworten oft in domänenspezifischen, also zum Beispiel firmeninternen Dokumenten zu finden sind, ist der Gedanke naheliegend, deren Analyse und Beantwortung dem computerlinguistischen Anwendungsfeld der *Textzugriffssysteme*, also dem Bereich *Text-Retrieval* zuzuordnen, und im speziellen dessen Anwendungsbereichen der Antwortextraktion und Fragebeantwortung (vgl. 2.2).

1.3 Datenlage

Die Erschliessung (und Adaption) bestehender Lösungen im Bereich Antwortextraktion und Fragebeantwortung wird erschwert durch deren meist unzureichende Dokumentation sowie die beschränkte Zugänglichkeit der Implementation (als ausführbares Programm oder Quellcode):

1.3.1 Dokumentation

Die meisten der verfügbaren wissenschaftlichen Arbeiten verstehen sich nicht als umfassende technische Dokumentation des jeweiligen Systems, sondern eher als eine Art „Forschungsbericht" und gehen dementsprechend nur wenig auf (implementatorische) Details ein. Zu kommerziellen Systemen sind meist nur werbeträchtige Produktbeschreibungen verfügbar.

Die in dieser Arbeit eingeflossenen Dokumentationen sind im Literaturverzeichnis am Schluss aufgeführt (mehrheitlich mit Internetverweis → Referenzen).

[1] dies wurde im Rahmen dieser Arbeit nicht untersucht, wäre jedoch vor allem für praktische Anwendungen hilfreich

1.3.2 Anwendungen

Wissenschaftliche und nicht-kommerzielle Systeme sind teilweise als Quellcode frei verfügbar. Dessen Adaption gestaltet sich allerdings oft schwierig:

- Quellcode ist in unterschiedlichen Sprachen geschrieben
- Quellcode und ausführbare Programme sind meist für eine bestimmte Plattform implementiert bzw. kompiliert und oft nicht portabel
- Analyse-Programme wie Tagger, Parser etc. sind häufig „Stand-Alone“ Lösungen, welche kaum in andere Systeme eingebunden werden können
- nicht immer ist der komplette Quellcode verfügbar
- umfassende lexikalische Ressourcen wie etwa Thesauri sind kaum frei verfügbar

Software-Lösungen kommerzieller Anbieter sind zu wissenschaftlichen Testzwecken noch unzugänglicher; wochenlange „Verhandlungen“ scheiterten, erschwerende Faktoren schmälerten die Evaluationsausbeute:

- Programme sind oft nur als funktional eingeschränkte Demo-Versionen erhältlich
- Demo- oder Testversionen bereiten oft Installations- oder Laufzeitprobleme
- Zugang zu sogenannten „Test-Servern“ bestand nur für 1-2 Tage
- Anfragen blieben oft unbeantwortet
- Viele Firmen wurden aufgekauft, fusionierten oder verschwanden gänzlich

Eine kommentierte Auflistung ausgewählter (und zum Teil hier verwendeter) Implementationen im Bereich Textretrieval befindet sich am Schluss dieser Arbeit (→ Referenzen).

2 Modelle und Verfahren für Indexierung und Retrieval freier Texte

Aufgrund des marginalen Forschungsstandes bezüglich E-Mail Beantwortung im speziellen (vgl. 1.2 und 1.3) werden im folgenden verschiedene Modelle und Vorgehensweisen für die *Beantwortung freier Texte im allgemeinen* (= Textretrieval) eruiert, um daraus Schlüsse für eine spezifischere Anwendung auf E-Mail Texte zu ziehen.

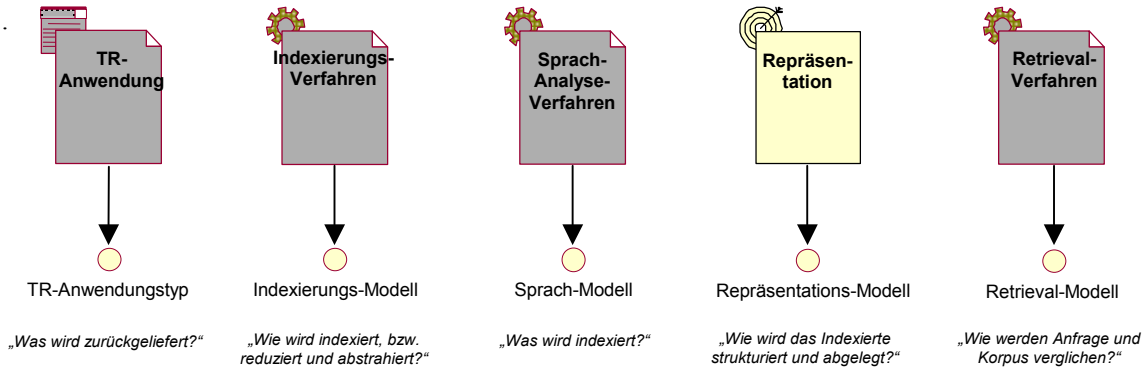
2.1 Textretrieval im Überblick

Die Aufgabe von Textzugriffs- oder Textretrievalsystemen[2] (TR-Systeme) ist das *Auffinden relevanter Dokumente, Textpassagen oder Phrasen aus einer bestimmten Suchkollektion nach vorgegebenen Kriterien* aus einer Benutzeranfrage (vgl. [Verdejo et al. 99]). Darunter fällt auch die Extraktion von potentiellen Antworten zu einer E-Mail, welche als Benutzeranfrage verstanden werden kann.

2.1.1 Komponenten und Modelle im Textretrieval

Der Prozess des Textzugriffes (= der Suche, des Retrievals) besteht in der Regel aus mehreren Komponenten[3], denen bestimmte technische und/oder linguistische *Modelle*[4] zugrunde liegen. Die Implementation einer Komponente ist damit die *Realisation* des entsprechenden Modells:

Abbildung 1: Verfahrens-Modelle im Textretrieval und deren Realisation als Komponente



Bestimmt das Modell „TR-Anwendungstyp“ beispielsweise, ob es sich um ein Document Retrieval- (DR) oder ein Fragebeantwortungs-System (QA) handelt, so ist die konkrete Umsetzung als entsprechende „TR-Anwendung“ die Realisation oder Instanziierung dieses Modells. TR-Systeme unterscheiden sich also bezüglich:

- *Anwendungstyp / Spezifität der Anwendung*: Welche Informationen werden in welcher Form zurückgeliefert (ganze Dokumente, einzelne Textpassagen oder ausformulierte Antworten? → 2.2)
- *Indexierungsverfahren*: Mittels welcher Verfahren wird indexiert? Wie hoch ist der Grad der Abstraktion und Reduktion vom Originaltext auf die interne Repräsentation? (→ 2.3)
- *Sprachmodellierung und -analyse* (als Teil des Indexierungsverfahrens): Welche sprachlichen Phänomene werden analysiert und nach welchen Formalismen? (→ 3.)
- *Repräsentation* (als Teil des Indexierungsverfahrens): Aus welchen der zuvor analysierten Sprachphänomene (und anderer Textelemente) werden in welcher Form „Indexterme“ generiert? (→ 2.5 und 5.1.1).
- *Retrieval / Suche*: Nach welchen Kriterien werden Suchterme aus der Frage mit Index-Termen der Dokumentenkollektion verglichen und wie werden Treffer klassiert? (→ 2.6)

Auf diese Fragen soll im folgenden (ab 2.2) genauer eingegangen werden.

[2] die Begriffe „Zugriff“, „Retrieval“, „Matching“ und „Suche“ werden im folgenden synonym verwendet
[3] der Begriff „Komponente“ umfasst hier im Gegensatz zu dessen Verständnis im Software-Engineering nicht nur prozedurale Bestandteile eines Systems, sondern auch deklarative (z.B. konkrete Grammatiken oder Repräsentationen)
[4] der Begriff „Modell“ wird hier nicht modelltheoretisch verwendet, also als Form der semantischen Repräsentation eines konkreten Inhalts, sondern ist an den objektorientierten Term „Schnittstelle“ („Interface“) angelehnt und meint eine abstrakte Beschreibung eines Sachverhaltes ohne konkreten Inhalt (wie z.B. eine Grammatiktheorie).

2.1.2 Abhängigkeiten und Paradigmen

Es ist einleuchtend, dass oben beschriebene Modelle nicht voneinander unabhängig sind. In der Tat bedingen sie einander in hohem Masse:

So hängt das gewählte *Indexierungsverfahren* (und damit auch dessen Teile Sprachanalyse und Repräsentation) in der Regel vom *Typ der Anwendung* ab, da spezifischere Antworten auch detailliertere Indexterme bedingen; ein Fragebeantwortungssystem beispielsweise wird kaum ohne umfassende sprachliche Analyse auskommen. Allerdings hat sich herausgestellt, dass selbst für das relativ unspezifische Wiederfinden ganzer Dokumente (= Document Retrieval) eine genaure Sprachanalyse mit entsprechender Repräsentation von Vorteil ist (vgl. [Verdejo et al. 99]). Dementsprechend ist natürlich auch das *Suchverfahren* (Retrieval) vom Anwendungstyp bestimmt.

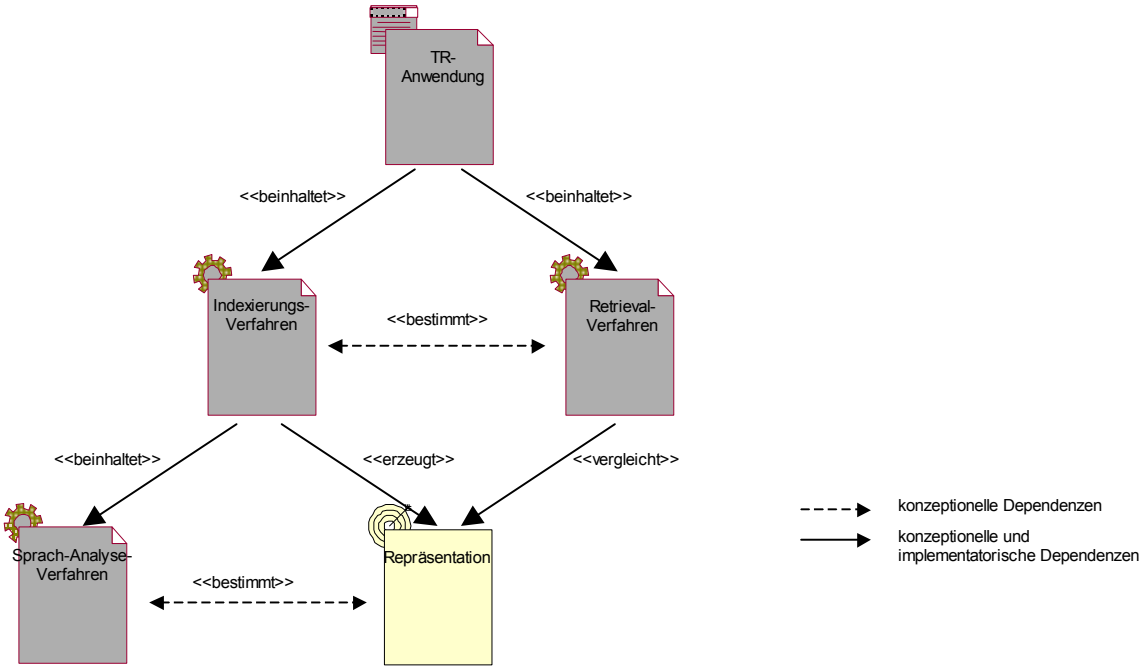
Noch stärkere, gar substantielle oder aber paradigmatische Abhängigkeiten bestehen zwischen *Indexierung* und *Retrieval*:

Eine reine Muster-Suche (oder gar eine eins-zu-eins Suche) ist prinzipiell ohne vorgängige Analyse direkt über dem Originaltext, d.h. *ohne Indexierung und Abstraktion* möglich. Dies bedingt einen entsprechend ausgefeilten Suchalgorithmus (zumindest Pattern-Matching Technologie), d.h. die Leistung des Systems liegt *ausschliesslich beim Retrieval.*
Dies ist zum Beispiel bei der Suchfunktion gängiger Textverarbeitungssoftware der Fall (z.B. Microsoft Word), wo allerdings keine eigentlichen Fragen beantwortet werden sollen; ansonsten müssten entsprechend aufwendige *Suchstrategien* eingebaut werden (vgl. 5.2), was ein hochgradig *prozedurales Retrieval* bedingt.

Im anderen Extremfall werden Dokumente und Anfrage soweit *abstrahiert*, dass das Suchverfahren in einem „simplen“ Muster-Abgleich, allenfalls mit automatischer Variablenbindung, besteht (z.B. Unifikation, vgl. [Mollá et al. 00b]). Das Retrieval wird also auf ein automatisierbares Minimum beschränkt und beinhaltet keinerlei Suchstrategien – sämtliche linguistischen und pragmatischen Überlegungen werden in die Indexierung verfrachtet.
Dieses „Retrieval-Paradigma“ unterstützen vor allem *deklarative, unifikationsbasierte Verfahren* wie etwa IE-Systeme mit Template-Unifikation (vgl. [Neumann 01]) oder bestimmte Antwortextraktions- / Fragebeantwortungs-Ansätze mit logischer Inferenz (z.B. LUIS, vgl. 2.2.3).

Der durch das Indexierungsverfahren festgelegte Abstraktionsgrad bestimmt natürlich wiederum die Repräsentationsform und so fort:

Abbildung 2: Architektur und Dependenzen der Komponenten eines Textretrieval Systems



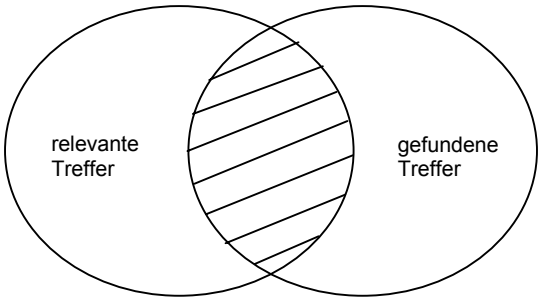
2.1.3 Evaluation von TR-Systemen

Als Evaluationskriterien für die „Güte“ eines TR-Systems werden die Masse *Präzision* (precision) und *Ausbeute* (recall) verwendet (teilweise kombiniert zum sog. *F-Mass* = gewichtetes geometrisches Mittel, vgl. [Neumann 01]):

- Präzision = Anzahl gefundener relevanter Treffer im Verhältnis zur Gesamtzahl gefundener Treffer
- Ausbeute = Anzahl gefundener relevanter Treffer im Verhältnis zur Gesamtzahl relevanter Treffer

Präzision und Ausbeute werden zusammen als *Performanz* bezeichnet. Das Retrieval-Verfahren gilt umso erfolgreicher, je höher die Performanz ist:

Abbildung 3: Präzision und Ausbeute



Diese häufig anzutreffenden Performanzkriterien sind allerdings meist sehr *domänenspezifisch*, da Themenbereich, Korpusgrösse und Stichprobe stark variieren:

Tabelle 1: TREC-Korpora mit heterogener Evaluationsgrundlage[5]

Korpus	Domäne	Dokumente	Anfragen
ADI	Informationswissenschaft	82	35
CACM	Computerwissenschaft	3200	64
CISI	Literaturwissenschaft	1460	76
CRAN	Aeronautik	1400	225
LISA	Literaturwissenschaft	6004	35
MED	Medizin	1033	30
NLM	Medizin	3078	155
NPL	Elektrik	11429	100
TIME	allgemeine Artikel	423	83

Zudem entscheiden oft *unterschiedliche, qualitative Klassifikationskriterien* über die Relevanz bzw. Korrektheit gefundener Treffer (wann ist eine Antwort „korrekt“?). Leider existieren bislang keine allgemein gültigen echt-quantitativen Evaluationskriterien für computerlinguistische Software, sodass Präzision und Ausbeute nach wie vor den Quasi-Benchmark für Textretrieval-Systeme darstellen.

[5] vgl. [Verdejo 99]

2.2 Anwendungstypen im Textretrieval

Textzugriffssysteme als lassen sich (mit zunehmender Spezifizierung) einteilen in (vgl. [Mollá et al. 00b]):

2.2.1 Document Retrieval

Per Definition liefert das Document Retrieval[6] (DR) Informationen über die *Existenz* von relevanten *Dokumenten* bezüglich bestimmter Anfragekriterien (vgl. [Verdejo et al. 99]).

DR-Systeme operieren typischerweise über *sehr grossen Dokument-Kollektionen / Korpora* (viele Mega- bis Gigabytes, evt. das „gesamte" Internet). Sie verfolgen traditionellerweise die Philosophie, dass eine tiefe linguistische Analyse sich nicht auszahlt und sind daher meist *keyword-basiert,* ignorieren also sämtliche morphologische und syntaktische Information sowie alle Funktionswörter ([ebenda]).

Als Form der Repräsentation kommen dementsprechend typischerweise einfache *Indexlisten bzw. -Verzeichnisse* zur Anwendung (vgl. 2.5.1).

Dediziertere Ansätze verwenden morphologische Normalisierung und teilweise auch seichte, partielle Satzanalyse, da ein gewisses Mass an Sprachwissen erfahrungsgemäss zu präziseren Suchresultaten führt (vgl. „Google"[7] sowie [Mollá et al. 00b]).

Traditionelle DR-Anwendungen können (technisch bedingt) die gesuchten Informationen nur sehr unspezifisch und oft nicht besonders präzise zurückliefern. Sie werden allerdings häufig als Komponente von komplexeren Textzugriffsverfahren eingesetzt, z.B. als „Vorfilter" zur *Eingrenzung der Suchkollektion* oder als *„Fall-back" Strategie* bei erfolgloser (linguistischer) Suche (vgl. [ebenda] sowie Kap. 4.1.4).

Beispiele: Internetsuchmaschinen; Teilkomponente komplexer TR-Systeme

2.2.2 Informationsextraktion

Systeme der Informationsextraktion (IE) operieren ebenfalls auf grösseren Datenmengen, finden und strukturieren jedoch gezielt bestimmte *domänenspezifische* Informationen aus freien Texten, bei gleichzeitigem „Überlesen" irrelevanter Information. Dabei werden aufgrund vordefinierter Lexikoneinträge oder Regeln jene Textpassagen identifiziert, die grundsätzlich in einen spezifischen *Themenbereich* (*Domäne*) fallen, und daraus wird eine sehr *begrenzte Menge hochspezifischer Daten extrahiert* (vgl. [Neumann 01]).

Diese Information wird in eine meist relativ einfache Zielrepräsentation mit *vordefinierten semantischen Rollen-Plätzen* eingepasst (die sog. *Templates*, „Schablonen", Datenbankmuster, vgl. 2.5.4).

Da nur bestimmte Teile des Textes relevant sind, betrachtet die IE linguistische „Nuancen" wie etwa die Intention des Autors als sekundär bis irrelevant (vgl. [Hobbs et al. 93]). Als Kompromiss zwischen theoretischen linguistischen Ansprüchen und pragmatischen Anforderungen werden daher meist *seichte syntaktische Textverarbeitungsmethoden* eingesetzt, vornehmlich *Pattern-Matching Technologie* und *Chunk-Parsing* unter Einbezug weniger und sehr genereller syntaktischer sowie domänenspezifischer Information (z.B. Eigennamen- und Verblisten, vgl. [Neumann et al. 00] sowie Kap. 4.1.3).

Diese *heuristisch* motivierte, „ingenieurmässige" Sichtweise auf Sprachverarbeitung ist präziser und spezifischer als DR, gegebenenfalls aber auch eingeschränkter. Insbesondere für die deutsche Sprache zeigt sich die weitgehend eingesetzte *bottom-up Strategie* als unzureichend robust hinsichtlich bestimmter syntaktischer Phänomene. Aktuelle IE-Systeme (meist fürs Englische) sind dennoch ziemlich erfolgreich in der Analyse grosser freier Textsammlungen (z.B. das Internet); vornehmlich, weil sie zu einem gewissen Grad bestimme, spezifische Textsorten *partiell „verstehen"* und die irrelevanten Teile ohne tiefe Analyse überlesen können (vgl. [Neumann et al. 00], [Neumann 01]).

[6] Der oft synonym verwendete Begriff „Information Retrieval" ist missverständlich und wird hier daher nicht eingeführt.
[7] Google-Suchmaschine: http://www.google.ch

Die Stärke des IE-Ansatzes liegt also neben dessen Robustheit in der Fähigkeit, *gezielt semantische Informationen zu erfassen* – dieser Grundgedanke führte denn auch zur Entwicklung semantisch orientierter Extraktionsmechanismen in Antwortextraktions- und Fragebeantwortungs-Systemen, insbesondere zur Erkennung von *Named Entities* (vgl. 2.4.5).

Beispiele: Extraktion bestimmter Entitäten in Kriegsnachrichten;
Teilkomponente von E-Mail Beantwortungssystemen (Extraktion von Produktnamen, Kunden-Anweisungen usw.)

2.2.3 Antwortextraktion

Im Gegensatz zu DR- und IE-Systemen versteht sich die Antwortextraktion (AE) als spezifischer Problem-Lösungs-Ansatz. Ein AE-System versucht, *exakt jene Textstellen* innerhalb einer uneditierten, text-basierten Dokumentenkollektion zu lokalisieren, welche *explizite Antworten* auf eine in natürlicher Sprache gestellte Benutzeranfrage enthalten (sog. *Antwort-Passagen)*. Dabei sollen all jene (und damit auch diskontinuierliche) Teile eines Textes lokalisiert werden, welche eine potentielle Antwort auf die gestellte Frage *enthalten* (vgl. [Mollá et al. 00a] und [Mollá et al. 00b]).

Zur Wiedergewinnung von Passagen werden indexierte Texte mittels *freier Kennzeichnung, hierarchischer (syntaktischer) Repräsentation* oder *logischer Formen* repräsentiert (vgl. 2.5).

Verglichen mit einem vollumfänglichen Fragebeantwortungssystem werden *keinerlei oder kaum Inferenzen* über dem Inhalt der Dokumente angestellt und nur minimales oder gar kein Situations- und Weltwissen konsultiert. Zudem kommen zumeist weniger dezidierte Suchstrategien zum Einsatz (vgl. 5.2). Antwortextraktion markiert die gefundenen Passagen, generiert also keine ausformulierten Antworten.

Beispiele: EXTRANS (vgl. 4.1.4); halbautomatische E-Mail Beantwortung

2.2.4 Fragebeantwortung

Fragebeantwortung („Question Answering“, QA[8]) ist die „ideale“ (und dementsprechend schwierigste) Lösung des Problems, möglichst präzise Informationen zu einer gestellten Frage zu finden (vgl. [Mollá et al. 00b]). Hierbei ist der *gesamte Text bedeutungsrelevant* – eventuell sogar mit allen Bedeutungsnuancen und Intentionen (und sollte daher „verstanden“ werden, vgl. [Hobbs et al. 93]).

Da die interne (meist semantische) Repräsentation (z.B. logische Formen) die volle Komplexität natürlicher Sprache abdecken muss (vgl. [Hobbs et al. 93]), besteht QA aus komplexen linguistischen und informatischen Teilaufgaben bis hin zu komplexen logischen Schlussfolgerungen über syntaktisch und semantisch erschlossenem Welt- und Situationswissen und der dynamischen *Erzeugung einer spezifischen Antwort*. QA funktioniert bis anhin dementsprechend nur innerhalb sehr enger Domänen (vgl. [Mollá et al. 00b]).

Beispiele: Frage-Antwort- bzw. Dialogsysteme, vollautomatische E-Mail Beantwortung.

[8] Oft auch mit Q&A abgekürzt, im folgenden aber mit QA

2.3 Indexierungsmodelle und -verfahren

Die Aufgabe von Textretrievalsystemen ist das Auffinden relevanter Dokumente, Textpassagen oder Phrasen nach vorgegebenen Kriterien aus einer Benutzeranfrage. Dazu wird eine Dokumentkollektion (das Korpus) in eine Kombination aus *Indextermen* überführt und die jeweilige Anfrage in eine Kombination aus *Suchtermen.* Diese Transformation in eine interne Repräsentation wird Indexierung genannt. Das Retrieval (also die Suche) entspricht dann dem Vergleich von Such- und Indextermen (zusammen auch einfach Terme genannt).

Terme können dabei sein: Einzelne Wörter (ggf. auf Wortstamm oder Lemma zurückgeführt), Mehrwort-Komposita (z.B. komplexe Eigennamen), Wortgruppen, Phrasen, spezielle Ausdrücke wie Datumsangaben oder irgendwelche „semantischen Einheiten".

Idealerweise sollte ein indexiertes Dokument als *effiziente Repräsentation* des für die jeweilige Anwendung *relevanten semantischen Inhalts* des Original-Dokuments (bzw. der Anfrage) fungieren. Das Indexierungsverfahren erstellt daher eine *Liste der bedeutungsrelevanten Indexterme*, auf deren Grundlage die Anfrage beantwortet wird (vgl. [Verdejo et al. 99], [Tan 98]).

Damit die relevanten Dokumente, Textstellen oder Phrasen zu einer Anfrage gefunden, also verglichen und „gematcht" werden zu können, müssen Anfrage und Dokumente in der *gleichen Art und Weise repräsentiert*, also gleich indexiert werden (vgl. Kap. 2.6). Dazu kommen verschiedene Verfahren mit unterschiedlichem linguistischen Bezug zur Anwendung:

2.3.1 Einwort-Indexierung

Das „klassische" Indexierungsverfahren durchläuft drei Phasen (vgl. [Verdejo et al. 99]):

1. *Entfernung „inhaltsloser" Ausdrücke („noise words")* wie Funktionswörter sowie domänenspezifisch besonders häufige Inhaltswörter mittels „Stop-Wort-Listen" und/oder regulärer Ausdrücke
2. *Reduktion auf eine Grundform* (Stemmatisierung, Lemmatisierung, Auflösung spezieller Konstruktionen wie Abkürzungen und Datumsangaben)
3. (Eventuell:) *Gewichtung der Indexterme* hinsichtlich ihrer Relevanz zur Anfrage gemäss der gemessenen *Term- und Dokumenthäufigkeit* (tf/idf bzw. idtf) und/oder linguistisch bzw. heuristisch motivierten Kriterien.

Das Resultat herkömmlicher Einwort-Indexierungsverfahren sind einfache Index-Listen bzw. -Verzeichnisse, wobei praktisch keine linguistische Information verwendet wird (vgl. 2.5.1). Insbesondere die Stemmatisierung erfolgt meist mittels simpler Algorithmen zum nicht-linguistischen Entfernen von Affixen, Plural, Flexionsendungen etc. Dies funktioniert recht gut für schwach flektierte Sprachen wie etwa englisch, ist für deutsch oder gar finnisch dagegen nicht besonders geeignet. Stemming-Algorithmen ohne morphologische Analyse können zudem keinerlei lexikalische Ambiguitäten auflösen (vgl. [Verdejo et al. 99]). Ausserdem werden sprachlich ausgedrückte Beziehungen zwischen den einzelnen Wörtern nicht erfasst.

Klassische Einwort-Indexierungsverfahren haben daher mehrere Nachteile:

- *Unbequeme Suche:* die Frage muss als Kombination von Suchtermen zu formuliert sein
- *Unpräzise Suche,* da keine linguistische Information verglichen werden kann
- *Aufwand*, die Dokumente zu indexieren (meist grosse Korpora)
- nur *ganze Dokumente* werden gefunden.

Nicht-linguistische Einwort-Indexierungsverfahren kommen daher selten auf zufriedenstellende Resultate bezüglich Präzision und Ausbeute. Bei der Informationsextraktion stellt sich beispielsweise das Problem, dass die zur Instanziierung der Templates notwendigen domänenspezifischen Rollenplatzhalter schwierig zu identifizieren sind, wenn unter anderem die Konstituentenstruktur nicht bekannt ist (vgl. [Neumann et al. 00] und 2.5.4).

Alternative: Einsatz *linguistischer* Analysen:

2.3.2 *Linguistische Indexierung*

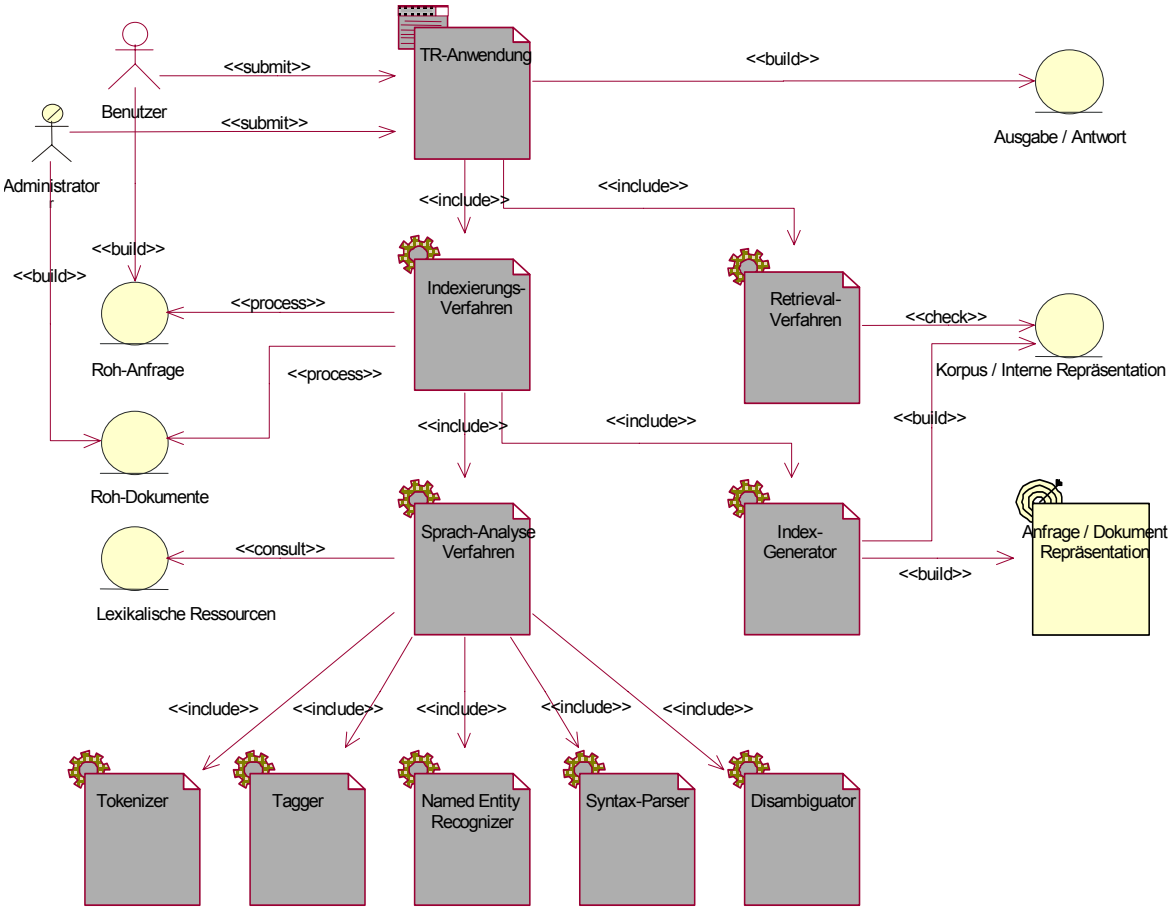
Im Gegensatz zur Indexierung einzelner Wörter können linguistische Verfahren syntaktische und semantische Beziehungen zwischen den Wörtern mitberücksichtigen (z.B. Phrasenindexierung, vgl. [Tan 98]).

Rein syntaktische Analysen vermögen allerdings in vielen Fällen Ambiguitäten nicht aufzulösen; syntaktisch „intakte“ NPs beispielsweise können aus Wörtern bestehen, die nicht zu einer semantischen Einheit gehören. An dieser Stelle kommen teilweise semantische Analysen zum Einsatz, welche allerdings in der Regel sehr aufwendig sind und grosse semantische Wissensbasen erfordern (vgl. [Tan 98] sowie 2.4.7).

Praktisch alle diskutierten Ansätze verfolgen die Schlüsselidee, den Indexierungsprozess in verschiedene Analyse-Stufen mit kleinen und gut definierbaren „Objekten“ zu unterteilen (vgl. [Hobbs et al. 93]). Dies bringt unter anderem Vorteile hinsichtlich der Skalierbarkeit und Portabilität auf verschiedene Sprachen und Domänen mit sich.

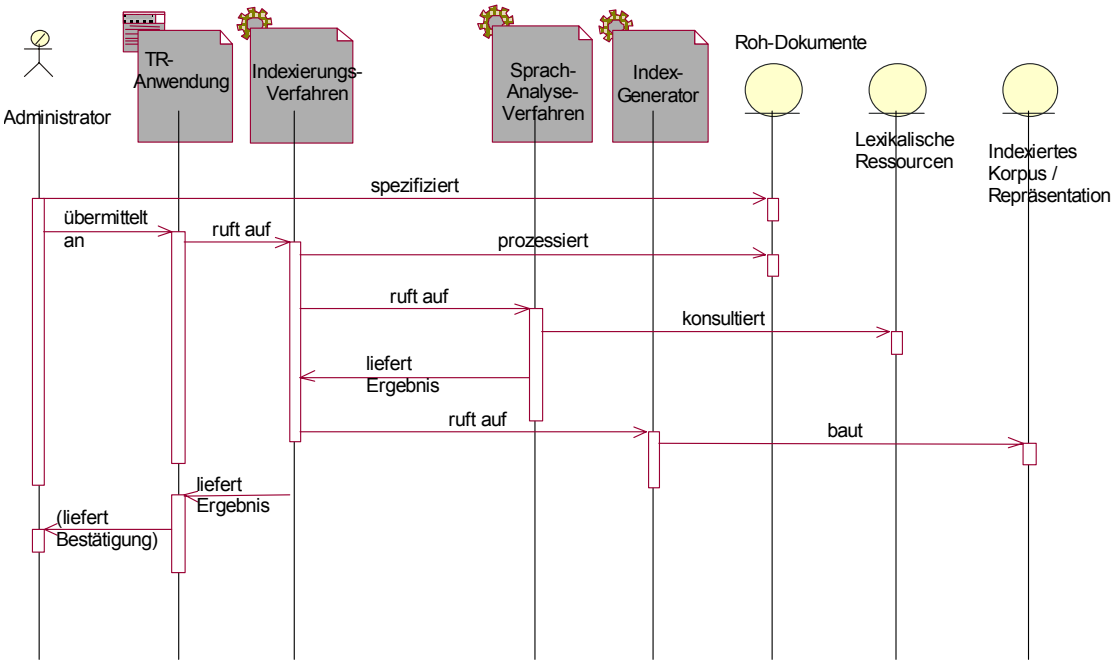
Die statischen Abhängigkeiten der verschiedenen Komponenten eines *syntaktischen Indexierungsverfahren* könnten beispielsweise folgende Struktur haben:

Abbildung 4: Strukturelle Interaktion der Komponenten im syntaktischen Textretrieval



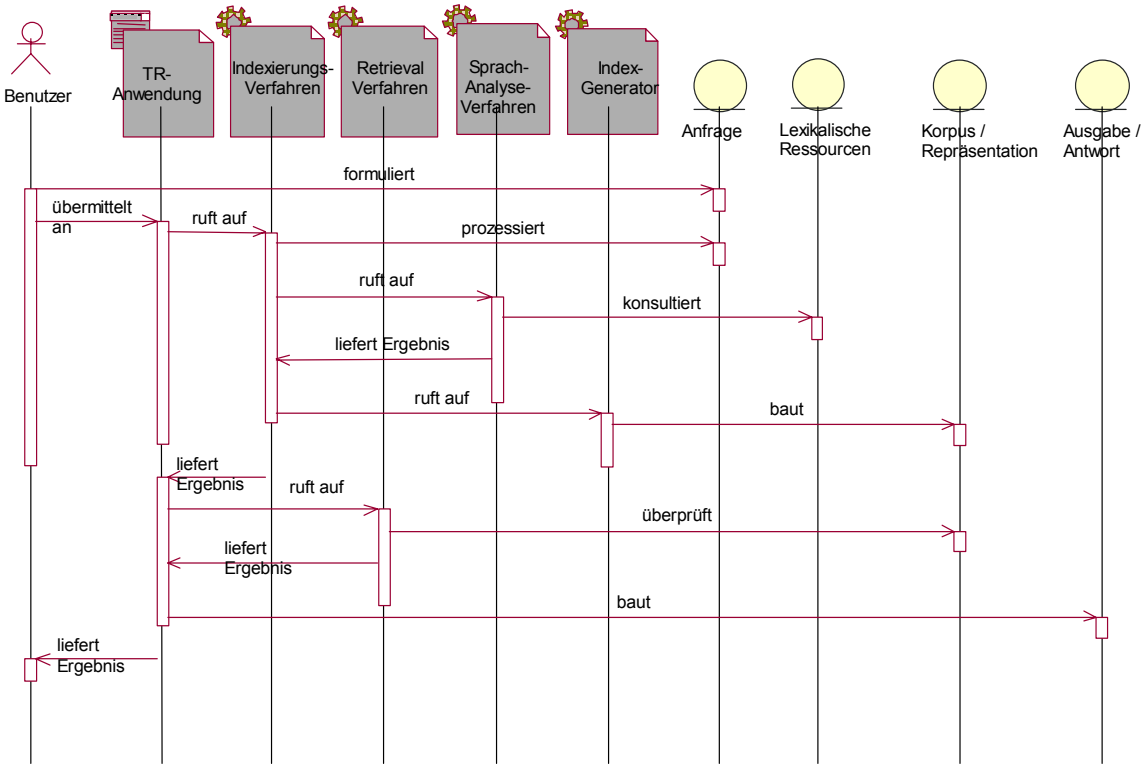
Der eigentliche Indexierungsprozess der Dokumentenkollektion (des Korpus) durchläuft die verschiedenen Stufen dann sequentiell nach ungefähr dem folgenden Schema:

Abbildung 5: Dynamische Interaktion bei der Korpusindexierung (offline)[9]



Das Endergebnis der meist offline prozessierten Dokumentenindexierung ist eine Wissensbasis bestehend aus den Indextermen der analysierten Texte (= das indexierte Korpus). Beim Retrieval wird die Benutzeranfrage grundsätzlich in der *gleichen Art und Weise* wie das Korpus indexiert und die so entstandenen Suchterme anschliessend mit den bestehenden Einträgen der Wissensbasis verglichen / gematcht / unifiziert:

Abbildung 6: Dynamische Interaktion bei Anfrageindexierung und Retrieval (online)



Für die Anwendung der automatischen E-Mail Beantwortung ist das linguistische Sprach-Analyse-Verfahren als Komponente der Indexierung von besonderer Bedeutung:

[9] Der „Administrator“ muss nicht zwingend menschlicher Natur sein;. Viele Internetsuchmaschinen indexieren selbständig fortlaufend neue Webseiten.

2.4 Sprachanalyse in der linguistischen Indexierung

Der linguistische Indexierungsprozess durchläuft normalerweise mehrere, meist mehr oder weniger sequentiell angeordnete (d.h. kaskadierte) Teilschritte. Die konkrete Struktur und Reihenfolge muss allerdings nicht derjenigen der folgenden Beschreibung entsprechen.

2.4.1 Lexikalische Ressourcen

Der Erfolg der linguistischen Analyse hängt auf verschiedenen Stufen stark von der Verfügbarkeit lexikalischer Ressourcen ab (vgl. [Verdejo et al. 99]), darunter:

- *Maschinenlesbare Wörterbücher (Machine Readable Dictionaries, MRD):* Wörter mit Zusatzinformation, meist alphabetisch strukturiert
- *Thesauri / Ontologien*: Wörter in semantischem Kontext strukturiert (z.B. Synonyme)
- *Lexikalische Wissensbasen*: Vollstrukturierte Lexika mit Morphosyntaktischer und / oder semantischer (und ggf. noch weiterer) Information
- *Domänenspezifische Wortlisten* / Lexika, z.B. für Eigennamen und Stopp-Wörter

Dabei ist weniger die Einbindung solcher Ressourcen problematisch, sondern vielmehr deren aufwendige (weil oft domänenspezifische und vorwiegend manuelle) *Erstellung*. Umfassende vorgefertigte elektronische Lexika und dergleichen sind dementsprechend oft ziemlich teuer.

Auf lexikalischen Ressourcen aufbauend operieren die einzelnen Analysestufen:

2.4.2 Erkennung der Analyseeinheiten

Der *Tokenizer* (oder *Tokenscanner*) identifiziert die durch Wortgrenzen, Interpunktionszeichen und typographische Merkmale definierte Textstruktur sowie spezielle Zeichenketten wie numerische Ausdrücke, Datums- und Zeitangaben, Abkürzungen, evt. auch Komposita. Solche Ausdrücke sind in den meisten Sprachen und Domänen sehr produktiv und morphosyntaktisch stark variabel (vgl. [Hobbs et al. 93]). Dazu kommen gerade in E-Mails computertechnische Syntaxgefüge wie Pfadnamen, komplexe Anweisungen mit Sonderzeichen und Neologismen.

Es werden meist „Mikrogrammatiken“ mit domänenneutralen und -spezifischen Regeln eingesetzt (implementiert als Reguläre Ausdrücke, vgl. [Neumann 01]). Dabei hat sich gezeigt, dass eine grosse Variabilität der Token-Klassen die Verarbeitung durch die nachfolgenden Stufen vereinfacht (vgl. [Neumann et al. 00]).

Bei bestimmten Ausgangsrepräsentationen, z.B. bei Mark-Up Dokumenten wie etwa HTML-Seiten und E-Mail Dateien, ist eine vorgängige Extraktion der eigentlichen Textstellen erforderlich, wobei allenfalls auch strukturelle Merkmale wie Überschriften, Absätze, Aufzählungen usw. in die Indexierung einbezogen werden können (vgl. [Metzinger 02]).

2.4.3 Lexikalisch-Morphologische Analyse

Um Wörter unabhängig von ihrer jeweiligen Flexionsform (z.B. Verbkonjugation) wiederzufinden, müssen sie auf irgendeine „Grundform“ zurückgeführt werden.

Für die englische Sprache existieren verschiedene „*Stemming*“-Verfahren, welche heuristisch und ohne linguistische Analyse den „Wortstamm“ zu ermitteln versuchen. Diese Algorithmen arbeiten einfach und sehr schnell, aber erwartungsgemäss nicht immer korrekt, und sind insbesondere für stark flektierte Sprachen wie deutsch nur schwer adaptierbar (vgl. [Verdejo et al. 99]).

Morphologische Analysen ermitteln nicht nur präzise den Wortstamm (→ *Lemmatisierung*), sondern können auch in hohem Masse zur *Disambiguierung* beitragen, beispielsweise durch:

- Morphologische *Normalisierung* (Aufsplitten Komposita, Auflösen von Hyphenkoordination wie „An- und Verkauf“ etc.)
- Auswahl der möglichen *Lesarten* von Komposita nach empirisch relevanten Selektionsregeln (vgl. [Tan 98])

Des weiteren hilft die deutsche *Gross- / Kleinschreibung* bei der Erkennung potentieller *Eigennamen* (vgl. 2.4.5). Ungeklärt ist allerdings, inwiefern dies bei E-Mail-Texten überhaupt eingehalten wird.

Ambivalent erscheint die oft vorgeschlagene automatische Behebung von *Tippfehlern* – welche gerade in E-Mails häufig anzutreffen sind – mittels *Spelling-Correction*. Dies erweist sich nicht in allen Fällen als hilfreich, da z.B. unbekannte Eigennamen zu bekannten Wortformen korrigiert werden (vgl. [Hobbs et al. 93]). Besser wäre daher das Verfolgen beider Alternativen und die Entscheidung auf höherer Ebene (vgl. Anwendungsbeispiele, Kapitel 4)

2.4.4 Wortklassenkennzeichnung

Der Zuweisung von Wortklassen / -kategorien („*Tagging*“, „Part-of-Speech“ (POS)) kommt grosse Bedeutung zu, da sie als *Schnittstelle* zwischen den lexikalischen Ressourcen und den darauf aufbauenden Syntax-Parsern fungiert. Dementsprechend spielen Präzision und Robustheit hier eine entscheidende Rolle; es hat sich gezeigt, dass die Fehlerrate gebräuchlicher Tagger von 3-5% oft zu unzuverlässigen Parsing-Ergebnissen führt (→ Fehler-Kumulation, vgl. [Tan 98] sowie [Wauschkun 96]).

Es kommen regelbasierte und stochastische Tagger zum Einsatz, welche die korrekte respektive wahrscheinlichste Wortklasse unter Einbezug des lexikalischen Umfelds ermitteln (eben „Part-of-Speech“). Diese sind im allgemeinen schneller als Syntax-Parser, aber offenbar häufig auch langsamer und fehleranfälliger als eine reine *FST-Analyse* (vgl. [Hobbs et al. 93] sowie 3.1.3).

Zur Reduktion des Ambiguitätspotentials eignen sich dedizierte *POS-Filter*, welche mittels *Kontextregeln* bestimmte Wortklassen ausschliessen. So haben im Deutschen beispielsweise Verben zu ca. 30% eine zusätzliche, nicht-verbale Lesart. Die heuristisch motivierten (recht einfachen) Disambiguierungsregeln eines POS-Filters für deutsche Verben könnten daher wie folgt lauten (vgl. [Neumann et al. 00]):

1. Ein gross geschriebenes potentielles Verb am Satzanfang, gefolgt von einem finiten Verb, schliesst die verbale Lesart des ersten Wortes aus (z.B. „*Unternehmen* sind an Gewinnmaximierung interessiert“)[10].
2. Adjektive und attributiv verwendete Partizipien sind in den meisten Fällen Teil einer NP. Daher: Ein solches Wort kann kein Verb sein, falls die vorherige Wortform ein Determinator oder die folgende einen Nomen ist.

Schwierigere Probleme beim POS-Tagging entstehen dagegen durch *Bindestriche* in Komposita, (Post-) *Modifikatoren* sowie bei der Erkennung des nominellen Kerns (sozusagen des „Haupt-Nomens“, vgl. [Tan 98]).

Einige Ansätze verfolgen das Paradigma der Theorie- und *Sprachunabhängigkeit* bereits auf lexikalischer Ebene. Demnach ist die exakte POS-Kategorie sprachabhängig und daher nicht determinierbar. Jede Sprache enthält jedoch bestimmte Elemente mit *fundamental nominalem, verbalem und prädikativem Charakter*. Die Wortklassen-Zuweisung erfolgt dann auf Grundlage dieser Meta-Kategorien (vgl. [Hobbs et al. 93]).

2.4.5 Erkennung von Stichworten, Eigennamen und anderen Entitäten

Namen von Personen, Firmen, Produkten und Lokalitäten stellen bei der Verarbeitung freier Texte immer ein Problem dar, da sie meist in den Lexikalischen Ressourcen nicht aufgeführt sind und in allen Sprachen sehr produktiv und mit grosser Variabilität verwendet werden (vgl. [Tan 98]).

Hier kommen *Named Entity Finder* zum Einsatz, welche anhand bestimmter Regeln (oft regulärer Grammatiken) versuchen, komplexe Ausdrücke oder unbekannte Eigennamen aufgrund ihrer internen Struktur (Morphosyntax) zu identifizieren[11] und gegebenenfalls auf eine Nennform zurückzuführen (zu normalisieren), welche anschliessend mit domänenspezifischen Lexika wie etwa Eigennamenlisten verglichen werden kann.

Bei der Erkennung solcher Entitäten gilt es allerdings einige Problemfälle zu beachten (vgl. [Neumann et al. 00], [Neumann 01]):

[10] Wiederum fragt sich, inwiefern auf die Gross-/Kleinschreibung in E-Mail Texten Verlass ist.
[11] dies geschieht oft mittels „Pattern Matching“

- *Optionale Designatoren*: Bestimmte Namenentitäten können mit oder ohne Designator auftreten (z.B. „Braun“ vs. „Braun AG“).
- *Lexikalische Ambiguität*: Eigennamen können oft auch anderen Wortkategorien zugeordnet werden (z.B. „Braun“ vs. „braun“).
- *Koreferenz*: Textuelle Bezüge zwischen Eigennamen (z.B. „Bundeskanzler Schröder“ vs. „G. Schröder“)

Das Problem der Designatoren kann durch Unterspezifikation teilweise entschärft werden, indem alle als solche erkannten Namen *ohne Designator* in ein (dynamisches) Lexikon assertiert und jeweils von dort gematcht werden, wobei das Pattern-Matching Verfahren alle möglichen Designatoren als optional dazugehörig betrachtet[12] (vgl. [Neumann et al. 00]).

Die anderen beiden Fälle können nicht alleine aufgrund ihrer internen Struktur analysiert werden, sondern verlangen nach zumindest syntaktischem Kontext. Ambiguitäten werden daher als solche *gekennzeichnet* und an nachfolgende Verarbeitungsstufen *weitergereicht* („containment of ambiguity“, vgl. [Neumann et al. 00], 3.1.3). Koreferenzen werden hier natürlich nicht als solche erkannt und müssen ebenfalls später aufgelöst werden (vgl. [Hobbs et al. 93]). Dies erfolgreich auch satzübergreifend zu bewältigen erfordert allerdings eine zumindest ansatzweise Diskurs-Analyse.

Als Alternative zu Namensentitäten-Erkennern werden alle *unbekannten Wörter* nur normalisiert (z.B. trunkiert) und als *Stichwörter* in eine Wissensbasis (Keyword-Lexikon) assertiert und später durch simple Stichwort-Suche (Keyword-Spotting) verarbeitet. Eine derartige nicht-linguistische Verarbeitung von Schlüsselwörtern ist im allgemeinen wesentlich einfacher und manchmal dennoch erstaunlich verlässlich, ermöglicht allerdings keine gezielte Beantwortung bestimmter Fragetypen (z.B. „Wer“ Fragen, welche eine Person als Antwort erwarten, vgl. 5.2). Andere Systeme ignorieren unbekannte Wörter robustheitshalber gänzlich und analysieren nur, was wirklich erkannt werden kann (vgl. [Verdejo et al. 99]).

Eine Sonderstellung nehmen sogenannte *Trigger Words* ein. Diese speziellen domänenspezifischen Schlüsselwörter induzieren eine bestimmte *Systemreaktion*. Denkbar wäre beispielsweise die direkte Weiterleitung eingehender E-Mails mit Triggerwort „UBS e-banking“ an die entsprechende (menschliche) Bearbeitungsinstanz. Es hat sich gezeigt, dass die Verwendung von Trigger Words deutliche Effizienzgewinne einbringt, jedoch leider auch die Ausbeute (vgl. 2.1.3) gefährdet (vgl. [Hobbs et al. 93]).

2.4.6 Syntaktische Analyse

Die Erkennung syntaktischer Gefüge erfolgt meist (jedoch nicht zwingend) auf Basis bereits erkannter Wortkategorien. Viele TR-Anwendungen bauen keine tiefe, linguistisch motivierte Konstituentenstruktur im Sinne verschachtelter *Phrasenkategorien* (vgl. [Borsley 97]) auf, sondern sammeln bestimmte Wortkategorien in pragmatisch relevanten *Wortgruppen* („*Chunks*“). Dies geschieht mittels flacher, fragmentarischer (d.h. partieller) Analyse, oft als *kaskadierte Finite-State-Transducers* (FST) implementiert. Dadurch wird eine explizite Trennung einzelner „phrasaler“ Ebenen erreicht, welche meist strikt *bottom-up* gesteuert durchlaufen werden, bis die oberste Satzebene erreicht wird (vgl. [Neumann 01], Kap. 4.1.3).

Ein Hauptproblem aller *regelbasierten* Parser ist die durch die Grammatikregeln unausweichliche *syntaktische Ambiguität*; zum Beispiel können Nominal-Phrasen manchmal wegen PP-Anschlussambiguität nicht korrekt und vollständig erkannt werden. Da jedoch jede Sprache (lexikalische) Elemente mit grundsätzlich nominalem, verbalem und prädikativem Charakter enthält, besitzen auch bestimmte *Wortgruppen* relativ klar definierbare syntaktische Strukturen, die sich mittels verhältnismässig einfacher Regeln ziemlich verlässlich und zum Teil sogar sprachübergreifend identifizieren lassen (vgl. [Hobbs et al. 93]), so z.B.:

- *Nominalgruppe* (Kopf-Nomen, Determinatoren und andere linksseitige Modifikatoren)
- *Verbalgruppe*[13] (Vollverb plus Hilfsverben, Modal- oder Spezialverben und intervenierende Adverbien).

[12] meist implementiert als „longest match“-Prinzip: Im Beispiel „Der Börsengang der Braun AG war erfolgreich“ würde das Pattern-Matching die Namens-Entiät „Braun AG“ identifizieren, wenn „Braun“ als möglicher Eigenname und „AG“ als potentieller Zusatz erkannt wird.
[13] auch Verbalkomplex genannt (vgl. [Wauschkun 96])

Gerade diese „Kategorien“ bilden bereits ein sehr brauchbares Fundament für eine domänenspezifische Analyse von E-Mail Texten: Im Gegensatz zu beispielsweise Zeitungsartikeln sind Benutzeranfragen oft sehr kurz und bestehen vorwiegend aus eher einfachen Nominalgruppen oder -phrasen, welche auch die relevante Information enthalten; diese syntaktischen Konstrukte können mit partieller Syntaxanalyse durchaus erkannt werden (vgl. [Tan 98])[14].

Die syntaktische Analyse freier Texte wird daher oft entwickelt nach dem pragmatischen Grundsatz: *Zuerst analysieren, was man wirklich analysieren kann* (vgl. [Hobbs et al. 93]).

Manche Implementationen schalten der Syntaxanalyse einen regelbasierten *Pruner* nach. Dieser *verwirft (domänenspezifisch) offensichtlich falsche Strukturen* des Parsers anhand eines *heuristischen* Regelsets und erhöht damit die Retrieval-Präzision (vgl. [Mollá et al. 00b]).

Die Konzepte der Grammatik-Modellierung zur syntaktischen Analyse in TR-Anwendungen werden in Kapitel 3 ausführlicher behandelt.

2.4.7 Disambiguierung und Referenz-Auflösung

Auf dieser Stufe sollen *möglichst alle* der zuvor erkannten und markierten sowie *bestimmte* der noch nicht erkannten Ambiguitäten und Referenz-Gefüge aufgelöst werden. Insbesondere:

- *Textuelle Referenzen*: Pronominale Anaphern, Referenzen zwischen Designatoren sowie gewisse Koordinationsellipsen können intrasententiell oft bereits anhand rein syntaktischer Information aufgelöst werden (vgl. [Mollá et al. 00b], [Neumann 01])
- *Anschlussambiguitäten* von Präpositionalphrasen lassen sich mittels trainiertem Domänen-Wissen auflösen (vgl. [Mollá et al. 00b]).

Die Auflösung dieser Phänomene geschieht meist im Anschluss an die syntaktische Analyse, da bei einem reinen bottom-up Verfahren erst hier phrasen- und gegebenenfalls auch satzübergreifende Beziehungen erfasst werden können.

Zur Verringerung der Fehler-Kumulation kann und sollte jedoch jede einzelne Verarbeitungsstufe zur Disambiguierung beitragen, indem Unsicherheiten zumindest markiert oder z.B. mit der jeweiligen statistischen Auftrittswahrscheinlichkeit weitergereicht werden, bis irgendwo später im Analyseprozess eine zuverlässigere Auflösung erfolgen kann.

[14] Da die Korpustexte allerdings gleich wie die Anfrage indexiert werden (sollten), sprachlich oft aber weit komplexer sind, gelten diese Erleichterungen dort wahrscheinlich nicht oder nur teilweise

2.5 Repräsentationsmodelle und Index-Generierung

Nach Abschluss der linguistischen Analyse muss das Ergebnis für den Vergleich von Dokumentensammlung und Anfrage in eine sinnvolle *Zielrepräsentation* umgewandelt, also die eigentlichen Indexterme generiert und strukturiert werden[15].

Die Reglementierung der Umwandlung der natürlichen Sprache in die Zielrepräsentation ist für das Auffinden der gesuchten Information der *performanz- und effizienz-kritische Teil* eines TR-Systems: Die Repräsentationsform bestimmt, welche Informationen grundsätzlich gefunden werden können und wie präzise respektive wie schnell. Textzugriffssysteme unterscheiden sich dementsprechend nicht nur darin, welche linguistischen Merkmale wie etwa Keywörter, Phrasen oder semantische Rollen überhaupt erfasst, sondern auch wie diese intern repräsentiert werden[16] (vgl. [Neumann 01]).

Die *linguistische Abdeckung* der Repräsentationsform hängt primär von den jeweiligen Anwendungs-Anforderungen ab; eine Internetsuchmaschine benötigt eventuell andere (und vermutlich auch weniger spezifische) Textelemente als ein Frageantwortsystem.

Die *typographische Repräsentationssyntax* ist dagegen eine eher darstellerische und implementatorische Angelegenheit, die es gemäss des verwendeten Suchverfahrens (z.B. logische Unifikation oder Template-Unifikation) sowie den zu repräsentierenden linguistischen Elementen zu wählen gilt. Beispiele sind Merkmalsstrukturen und XML-Syntax als typographische Varianten der Hierarchischen Repräsentationsform (vgl. 2.5.3) oder Horn-Klauseln als Instanziierung der logischen Repräsentation (vgl. 2.5.5).

Die Repräsentationsform an sich ist also grundsätzlich unabhängig von der intendierten Anwendung, in der Praxis jedoch von jener bestimmt. Mögliche Repräsentationsformen in TR-Anwendungen (vgl. Kapitel 4) sind beispielsweise[17]:

- (Einwort-) Index-Listen bzw. -Verzeichnisse
- Freie Kennzeichnung bestimmter Konstrukte und Entitäten im Fliesstext
- Hierarchische Repräsentation (z.B. komplexe Merkmalsstrukturen)
- Templates
- Logische Formen

Im einzelnen:

2.5.1 Index-Listen und -Verzeichnisse

In Anwendungen des Document Retrievals müssen oft riesige Korpora indexiert werden (bei Web-Suchmaschinen potentiell das gesamte Internet). *Index-Listen* sind eine effiziente Repräsentationsform für den Inhalt von ganzen Dokumenten. Die Text-Passage

```
Dr. Hermann Wirth, bisheriger Leiter der Musikhochschule München, verabschiedete sich heute aus
dem Amt. Der 65jährige tritt seinen wohlverdienten Ruhestand an. Als seine Nachfolgerin wurde
Sabine Klinger benannt. Ebenfalls neu besetzt wurde die Stelle des Musikdirektors. Annelie
Häfner folgt Christian Meindl nach.
```

würde bei der klassischen *Einwort-Indexierung* (vgl. 2.3.1) beispielsweise auf folgende Liste aus Indextermen abstrahiert:

```
Dr., Hermann, Wirth, bisherig, Leiter, Musikhochschule, München, verabschieden, heute, Amt,
65jährig, antreten, wohlverdient, Ruhestand, Nachfolgerin, Sabine, Klinger, benennen, neu,
besetzen, Stelle, Musikdirektor, Annelie, Häfner, nachfolgen, Christian, Meindl.
```

Einzelne *isolierte Wörter* werden also entweder normalisiert (hier bereits mit ausgereifter Lemmatisierung) oder aber gänzlich entfernt – weitere lexikalische Eigenschaften oder gar syntaktische oder semantische Beziehungen gibt es nicht[18]. Dafür sorgen heuristische

[15] Die Erzeugung einer internen Repräsentation aus der Analysestruktur wird teilweise als Post-Processing bezeichnet (vgl. [Hobbs et al. 93]).
[16] Indexierungs-Modell und Retrieval-Modell hängen natürlich voneinander ab (vgl. 2.1)
[17] Die Terminologie wurde selber gewählt, da in der Literatur keine allgemeine Namensgebung gefunden werden konnte
[18] theoretisch wäre auch die Repräsentation syntaktischer und semantischer Terme als Index-Liste denkbar; das macht in der Praxis allerdings kaum Sinn, da Index-Listen nur isolierte Terme ohne Beziehungen beinhalten

Methoden der *Zugriffs-Optimierung* für eine effiziente Suche (z.B. Hashing, also Umordnen der Indexreihenfolge nach heuristischen Relevanzkriterien).

Ein *Index-Verzeichnis* (oder kurz *Index*) fasst den Inhalt mehrerer Index-Listen (z.B. mehrerer Dokumente) in einer strukturierten Aufstellung zusammen, welche intern wiederum zugriffsoptimiert wird. Auf diese Weise können alle Vorkommen beliebiger Suchterme schnell und einfach ermittelt werden (vgl. [Verdejo et al. 99]).

Oft werden die einzelnen Terme eines Index-Verzeichnisses nach deren Auftrittshäufigkeit in einzelnen Dokumenten und/oder im gesamten Korpus gewichtet, sodass das Retrieval-Verfahren (vgl. 2.6) die gefundenen Treffer nach deren Übereinstimmung zur Frage priorisieren kann. Zur Gewichtung von Indextermen wird im allgemeinen das IDTF-Mass verwendet (inverse document term frequency, vgl. [Verdejo 99]).

Da Index-Listen und -Verzeichnisse keinerlei linguistische Beziehungen repräsentieren, eignen sie sich für Aufgaben der Antwortextraktion und Fragebeantwortung nur sehr begrenzt[19].

2.5.2 Freie Kennzeichnung

Als Indexterme werden hier die *relevanten Textstellen* wie Wörter, Phrasen, Passagen oder benannte Entitäten nach lexikalischen, syntaktischen oder semantischen Kriterien *direkt im Ausgangstext markiert*, ohne jenen auf eine völlig andere Darstellung zu abzubilden. Beispiel[20]:

```
Der <AP>neu gewählte</AP> <ROLE>Bundeskanzler</ROLE> <PERSON>Gerhard Schröder</PERSON>
hat <TEMP>gestern</TEMP> eine <NN>Brezel</NN> <VG>verschluckt</VG>
```

Dies hat einerseits den Vorteil, dass die Zielrepräsentation *inkrementell* erzeugt werden kann (solange keine kreuzenden Kanten erforderlich sind, sich bestimmte Indexterme also überschneiden). Anderseits können einzelne Terme einer bestimmten Kategorie (z.B. <PERSON>) schnell und einfach wiedergefunden und der ursprüngliche Text durch Filterung der Markierungen wiederhergestellt werden. Diese Repräsentationsform ist daher ein typisches Beispiel für den Einsatz von *Pattern-Matching Techniken*.

Nachteilig wirkt sich allerdings die *mangelnde Strukturierung* aus, welche Zusammengehörigkeiten von und Beziehungen zwischen Indextermen nicht berücksichtigt. Zwar wären mit der gleichen Technologie (mittels Kaskadierung) durchaus auch verschachtelte Konstruktionen denkbar:

```
Der <NP>
     <AP>neu gewählte</AP>
     <ROLE_PERSON>
          <TITLE>Bundeskanzler</TITLE>
          <NAME>Gerhard Schröder</NAME>
     </ROLE_PERSON>
</NP> hat...
```

Dies führt jedoch zu organisch wachsenden und komplexen Repräsentationen. Ausserdem bleiben viele *ungenutzte Wörter* in den Endrepräsentation erhalten, obschon sie lediglich zur Wiedergewinnung des ursprünglichen Textes dienen. Das ist unschön und erschwert ein effizientes Retrieval.

Zudem gestaltet sich auch eine Trennung verschiedener *linguistischer Ebenen* (Lexikon, Syntax, Semantik) schwierig; entsprechend kompliziert wird daher die Zuweisung *mehrerer linguistischen Eigenschaften* pro Wort, beispielsweise Genus *plus* Wortkategorie *plus* semantische Markierung:

```
<ROLE_PERSON>
     <TITLE><NN><M>Bundeskanzler</M></NN></TITLE>
     <NAME><NE><M>Gerhard Schröder</M><NE></NAME>
</ROLE_PERSON>
```

Verschiedene Markierungs-Kategorien (z.B. Genus) müssen also ziemlich unstrukturiert vermischt werden und sind im Endergebnis kaum als solche wiedererkennbar.

[19] das heisst nicht zwingend, dass bei deren *Erstellung* kein linguistischen Wissen verwertet wird
[20] die nachfolgenden Beispiele werden in SGML- bzw. XML-Syntax formuliert

Eine freie Kennzeichnung von Indexermen ist daher vor allem für flache Zielrepräsentationen mit entsprechend wenig differenzierter Sprach-Analyse geeignet.
Um die Vorteile verschachtelter Indexterme zum Beispiel für die Repräsentation komplexerer linguistischer Begebenheiten zu nutzen, empfiehlt sich die Abstraktion auf eine

2.5.3 Hierarchische Repräsentation

Im Gegensatz zur freien Indexierung werden hier *nur* die relevanten Wörter, Konstrukte und Entitäten als Indexterme repräsentiert und in eine *abstrahierte hierarchische Struktur* eingepasst. Das Endresultat ist damit viel homogener und auf die Darstellung bestimmter (im folgenden syntaktischer) *Repräsentationsebenen* fokussiert (vgl. [Metzinger 02]):

```
<sentence id=0>
     <phrase pos="NP" sem="">
          <subphrase pos="AP" sem="MODIFIER">
               <lextag pos="ADV" flex="neu" lem="neu"/>
               <lextag pos="A" flex="gewählte" lem="wählen"/>
          </subphrase>
          <subphrase pos="CNP" sem="ROLE_PERSON">
               <lextag pos="NN" flex="Bundes#kanzler" lem="Bund#kanzler" sem="TITLE"/>
               <lextag pos="NE" flex="Gerhard_Schröder" lem="Gerhard_Schröder" sem="NAME"/>
          </subphrase>
     </phrase>
</sentence>
```

Einzelne Wörter oder Entitäten werden also *einheitlich auf gleicher Repräsentationsebene* indexiert (hier als <lextag>), während ihre Abhängigkeiten und Verschachtelungen auf höhere Ebenen verfrachtet werden (z.B. <subphrase>, <phrase> usw.). Die einzelnen Markierungs-Kategorien und die linguistischen Ebenen sind klar getrennt und als solche erkennbar.

Das kann äquivalent auch als verschachtelte Merkmalsstruktur formuliert werden[21]:

```
[                                                                    ]
| sentence | id       0                                              |
|          | phrase   [ pos      NP                                   ]
|          |          [ sem      ---                                  ]
|          |          | subphras [ pos      AP                      ]
|          |          |          [ sem      MODIFIER                ]
|          |          |          | lextag   [ pos     ADV          ]
|          |          |          |          [ flex    neu          ]
|          |          |          |          [ lem     neu          ]
|          |          |          | lextag   [ pos     A            ]
|          |          |          |          [ flex    gewählte     ]
|          |          |          |          [ lem     wählen       ]
|          |          | subphras [ pos      CNP                     ]
|          |          |          [ sem      ROLE_PERSON             ]
|          |          |          | lextag   [ pos     NN           ]
|          |          |          |          [ flex    Bundes#kanzler ]
|          |          |          |          [ lem     Bund#kanzler ]
|          |          |          |          [ sem     TITLE        ]
|          |          |          | lextag   [ pos     NE           ]
|          |          |          |          [ flex    Gerhard_Schröder ]
|          |          |          |          [ lem     Gerhard_Schröder ]
|          |          |          |          [ sem     NAME         ]
|          |          [                                             ]
|          [                                                        ]
[                                                                    ]
```

[21] Markierungs-Elemente (Entitäten, also z.B. <phrase>) und deren Eigenschaften (Attribute, z.B. „pos“) werden in der Merkmalsstruktur gleichgesetzt. Das ist legitim, da die Markierungs-Kategorie (also etwa „phrase“) als solches eigentlich auch eine Eigenschaft des jeweiligen Elements ist. Mann könnte auch schreiben: <element name=“phrase“ pos=“NP“>

Die *vordefinierte Repräsentationsstruktur*[22] legt fest:

- die *Benennung* der einzelnen Elemente (Markierungen, Entitäten, Tags)
- deren erforderliche und optionale *Anzahl*
- deren erforderliche und optionale *Eigenschaften*
- deren erforderliche und optionale *Inhalte* (d.h. eingebettete Elemente) und damit die *maximale Einbettungstiefe*

Diese starke Reglementierung der Abbildung von Sprache auf die interne Darstellung sorgt unabhängig vom jeweiligen Sprachmaterial für sehr homogene Indexierungsbäume und hat den Vorteil, das oberflächliche (z.B. syntaktische) Varianten gleichen Inhalts teilweise auch gleich repräsentiert werden und damit auch bei variierter Fragestellung relativ einfach wiedergefunden werden können (ohne dass eine eigentliche funktionale Struktur ermittelt wird) [23].

Zudem ist diese Darstellung *mehrdimensional*, sodass einem bestimmten Wort verschiedene – linguistische als auch informatische – Eigenschaften, d.h. *Attribute* zugewiesen werden können[24], was ein differenziertes und vor allem *zielgerichtetes Retrieval* erlaubt. Bei der Verwendung von Pattern-Matching Algorithmen kann sogar eine unifikationsartige „Variablenbindung“ erzielt werden. Die Frage

```
Wer ist der neu gewählte Bundeskanzler
```

liesse sich nach entsprechender Abstraktion eine Repräsentation folgender Art bringen (vgl. [Metzinger 02][25]):

```
<sentence id=0>
     <phrase pos="NP" sem="">
          <subphrase pos="AP" sem="MODIFIER">
               <lextag pos="ADV" flex="neu" lem="neu"/>
               <lextag pos="A" flex="gewählte" lem="wählen"/>
          </subphrase>
          <subphrase pos="CNP" sem="ROLE_PERSON">
               <lextag pos="NN" flex="Bundes#kanzler" lem="Bund#kanzler" sem="TITLE"/>
               <lextag pos="NN|NE" flex="(.*)" lem="(.*)" sem="NAME"/>
          </subphrase>
     </phrase>
</sentence>
```

Hier kommt trotz unterschiedlicher Oberflächen-Syntax die gleiche Indexstruktur zustande, wobei das Fragepronomen „Wer“ durch einen „Universal-Matcher“ ersetzt wurde. Dieser besagt: *„Suche eine Markierung des Typs „lextag“ mit der Wortkategorie „NN“ oder „NE“ und dem semantischen Typ „Name“ sowie beliebigen Werten für die flektierte und die lemmatisierte Form“,* was in diesem Fall „*Gerhard Schröder*“ und damit die korrekte Antwort ergibt:

```
(Q)  <lextag   pos="NN|NE"    flex="(.*)"            lem="(.*)"               sem="NAME"/>
(A)  <lextag   pos="NE"       flex="Gerhard_Schröder" lem="Gerhard_Schröder"   sem="NAME"/>
```

Der hohe linguistische Detaillierungsgrad bei gleichzeitiger Strukturtreue spricht für eine Anwendung in den Bereichen Informations- und Antwortextraktion. Für reines Dokument Retrieval wäre diese Repräsentationsform überdimensioniert und eventuell zu langsam, wogegen für eine vollumfängliche Fragebeantwortung wahrscheinlich ein noch grösserer semantischer Bezug erforderlich wäre (z.B. textuelle Referenzen, vgl. 2.4.7).

Etwas unschön ist die Grösse und damit Unübersichtlichkeit der (an sich zwar gut strukturierten) Indexdateien aufgrund der mehrschichtigen linguistischen Eigenschaften. Will man wirklich nur die innerhalb einer begrenzten Domäne *wesentliche semantische Information* isoliert repräsentieren, so greift man daher besser zu *Templates*:

[22] die „Grammatik“ für die erforderliche Dokumentenstruktur wird im Falle der hier beschrieben XML-Dateien durch die sogenannte „Document Type Definition“ (DTD) festgelegt, welche jedem XML-Dokumente zugrunde liegt.
[23] das setzt natürlich eine für die jeweilige Anwendung sinnvolle und daher meist heuristisch motivierte Definition der Repräsentationsstruktur voraus
[24] diese Darstellungsform ähnelt also einer umfangreichen Merkmalsstruktur
[25] diese Repräsentation wurde gegenüber der tatsächlichen etwas „normalisiert“; die beschriebenen Vorzüge treffen aber auch für die tatsächliche Implementation zu

2.5.4 Templates

Ein Template ist eine interne Repräsentationsstruktur mit einer *bestimmten Menge vordefinierter Rollen-Plätze* („Argument Slots“). Es definiert als eine Art „Schablone“ die Zuweisung bestimmter lexikalischer, syntaktischer und semantischer Einheiten zu den zu „füllenden“ Rollenplätze (diese Einheiten werden dann „Argument-Filler“ genannt). Eine mit den eingepassten Spracheinheiten gefüllte Schablone wird *instanziiertes Template* genannt (vgl. [Neumann 01]). Für den Text:

```
Dr. Hermann Wirth, bisheriger Leiter der Musikhochschule München, verabschiedete sich heute aus
dem Amt. Der 65jährige tritt seinen wohlverdienten Ruhestand an. Als seine Nachfolgerin wurde
Sabine Klinger benannt. Ebenfalls neu besetzt wurde die Stelle des Musikdirektors. Annelie
Häfner folgt Christian Meindl nach.
```

ergeben sich bei der Extraktion von Informationen über Personalwechsel beispielsweise diese zwei instanziierten Templates:

PersonOut	Dr. Hermann Wirth
PersonIn	Sabine Klinger
Position	Leiter
Organization	Musikhochschule München
TimeOut	heute
TimeIn	

PersonOut	Christian Meindl
PersonIn	Annelie Häfner
Position	Musikdirektor
Organization	Musikhochschule München
TimeOut	
TimeIn	

Templates sind also typische Beispiele einer Repräsentationsstruktur aus *Attribut-Wert-Paaren* (Rollenplätze und deren zugewiesene Werte) und werden dementsprechend gerne als Merkmalsstruktur implementiert.

Das hat den Vorteil, das eigentliche Retrieval als *Merkmals-Unifikation* (auch Template-Unifikation genannt) realisieren zu können, und dessen Entwicklungs- und Laufzeit-Aufwand damit auf ein Minimum zu beschränken (deklaratives Retrieval-Paradigma, vgl. 2.1.2). Die Frage

```
Wer ist der Nachfolger von Dr. Hermann Wirth?
```

ergäbe das Template

PersonOut	Dr. Hermann Wirth
PersonIn	**X**
Position	
Organization	
TimeOut	
TimeIn	

welches durch Unifikation mit obigen Korpus-Templates zu folgender Variablenbindung und damit (zumindest in diesem Falle) zur korrekten Antwort führt:

```
X    =    Sabine Klinger
```

Hinsichtlich der Verwendung eines Unifikations-Mechanismus unterscheiden sich Templates gegenüber *logischen Formen* (siehe unten: 2.5.5) also nur in ihrer Syntax bzw. typographischen Darstellung. Allerdings sind im Gegensatz zu letzteren *keinerlei Schlussfolgerungen* über einem grösseren Kontext möglich; Frage- und Antwort-Templates werden lediglich rein mechanisch abgeglichen. Im Unterschied zu der ebenfalls als Merkmalsstruktur darstellbaren *hierarchischen Repräsentation* (vgl. 2.5.3) sind Anzahl und Benennung der einzelnen Argumente bei Templates vorgegeben.

Probleme bei der Template-Instanziierung bereiten insbesondere *Synonyme* und sinnverwandte Informationen. Diese ergeben verschiedene Template-Instanzen, welche anhand domänenspezifischer (ggf. semantischer) Information wie etwa *Thesauri* wieder verknüpft werden müssen. Auch Koreferenz-Phänomene führen oft zu Schwierigkeiten, etwa bei Verbalgruppen wie „attack“ vs. „plan to attack“ vs. „fail to attack“ (vgl. [Hobbs et al. 93]). Letzteres gilt natürlich auch für die oben beschriebene hierarchische Repräsentation.

Da Templates als Indexterme eine *statische Anzahl voneinander weitgehend unabhängiger linguistischer Elemente* repräsentieren, sind sie in ihrer Anwendung auf die *Erkennung domänenrelevanter Muster* und damit auf die Aufgaben der *Informationsextraktion* spezialisiert. Der Grundgedanke, *gezielt semantische Informationen zu extrahieren,* kommt jedoch auch in AE- und QA-Systemen zum Zuge, z.B. bei der Erkennung von *Named Entities* (vgl. 2.4.5).

Sollen Informationen dynamisch repräsentiert und in Relation zueinander gestellt werden, wie z.B. bei Anforderungen der Antwortextraktion oder der Fragebeantwortung, erschwert diese Einschränkung die Wiedergewinnung der relevanten Textstellen. Hier kommen mit Vorteil hierarchische Repräsentationen zum Einsatz, oder aber:

2.5.5 Logische Formen

Eine Möglichkeit, linguistische und insbesondere auch *semantische* Informationen dynamisch zu repräsentieren, sind *logische Formen*. Im EXTRANS-System beispielsweise werden aus (zuvor partiell disambiguierten) Dependenz-Strukturen der Dokumentenkollektion wie auch der Anfrage flache, „minimale logische Formen“ in Horn-Klausel-Logik (≈Prolog) erzeugt (vgl. [Mollá et al. 00b] und 4.1.4).

Die Software LUIS tut ähnliches, jedoch mit geringer linguistischer Verarbeitung. So wird der Korpustext

```
Suchen Sie einen Job oder gar eine Festanstellung? - Verschiedene Organisationen bieten
Jobbörsen an.
```

im Falle von LUIS zu folgender Prolog-Repräsentation (vgl. [Arnold et al. 01]):

```
ta_database(6601,5885,6636,'Arbeit_html',('V','such~e',1,9,9)).
ta_database(6636,6601,6654,'Arbeit_html',('S','fest#an|stell~ung',1,9,9)).
ta_database(6654,6636,6682,'Arbeit_html',('A','ver|schieden',1,9,10)).
ta_database(6682,6654,6689,'Arbeit_html',('V','biet~en',1,9,10)).
ta_database(6689,6682,7733,'Arbeit_html',('S','jobb#börse',1,9,10)).
```

Hier sind Position, Quelle, Wortart und morphologisch analysierte Flexionsform gespeichert (plus einige informatisch motivierte Variablen). Diese Informationen werden in der Dokumentenkollektion als *Axiomensystem* und in der Anfrage als *Theorem* behandelt, das zu *beweisen* dem Retrieval entspricht.

Es wird zwar auch hier nur eine beschränkte Menge linguistischer Informationstypen erfasst (im Falle EXTRANS sind es die *semantischen* Entitäten *Objekt*, *Ereignis* und *Eigenschaft,* vgl. [Mollá et al. 00b]). Im Gegensatz zu Templates können diese Informationstypen jedoch beliebig oft instanziiert und deren Instanzen vor allem auch in Beziehung zu einander gestellt werden.

Da letzteres Aufgabe des standardmässig implementierten *Theorembeweisers* (z.B. von Prolog) ist, fallen für das eigentliche Retrieval nur minimale Bemühungen an – die unifikationsbasierte Beweisstrategie sorgt zudem gleich für die korrekten Variablenbindungen.

Logische Formen können als einzige hinlänglich bekannte Repräsentationsform *direkt als Basis für Inferenzen* über den bereits in der Wissensbasis assertierten Einträgen zur *Erschliessung impliziten Wissens* dienen. Dementsprechend eignen sie sich besonders für die Verwendung in Antwortextraktions- und Fragebeantwortungssystemen. Die Generierung der entsprechenden semantisch motivierten logischen Endrepräsentation aus den ursprünglichen Texten bedarf allerdings (wie bei allen semantisch orientierten Verfahren) den Beizug diverser Analysestufen mit spezifischen Unzulänglichkeiten, was in der Praxis teilweise zu einer sehr fehleranfälligen Analyse Indexierung führt (vgl. 5.1 und [Mollá et al. 00b]).

2.6 Retrievalmodelle und -verfahren

Damit die relevanten Dokumente, Textstellen oder Phrasen zu einer Anfrage gefunden, also verglichen und „gematcht" werden zu können, werden Anfrage und Dokumente in der *gleichen Art und Weise repräsentiert* (also gleich analysiert bzw. indexiert). Das *Retrieval-Modell* (oder Zugriffs- oder Such-Modell) legt nun fest, in welcher Art und Weise die Index- und Suchterme miteinander verglichen und damit die relevanten Antworten gefunden werden (vgl. [Verdejo et al. 99]).

Wie zuvor erwähnt, hängt das Retrieval in hohem Masse von der Indexierung und damit der internen Repräsentation ab: Während bei der Verwendung logischer Formen das Retrieval-Modell weitgehend durch den Theorembeweiser bestimmt wird, erfordern andere, gegebenenfalls weniger abstrahierte Repräsentationsformen die Entwicklung eines applikations- oder zumindest repräsentationsspezifischen Suchverfahrens. Viele Suchmodelle bzw. -verfahren basieren auf denjenigen des „klassischen" Document Retrievals, welches beispielsweise in Internetsuchmaschinen zur Anwendung kommt (vgl. [Verdejo et al. 99]):

1. *Zeichenkette-Suchmodell* (String-Search-Model)
2. *Bool'sches Modell* (Boolean Model)
3. *Vektorraum-Modell* (Vector-Space Model)

ad 1: Jeder Suchterm der Anfrage wird separat und unbearbeitet im Korpus gesucht („Oder"-Suche). Je mehr Terme die Frage enthält, desto mehr „Treffer" werden zurückgeliefert.

ad 2: Als Suchkriterium werden die einzelnen Suchterme mittels logischer Operatoren AND, OR, NOT (ungewichtet) verknüpft. Beim *erweiterten* Bool'schen Modell werden diese *Verknüpfungen gewichtet*, um beispielsweise einem Suchresultat bestehend aus zwei von drei mit OR verknüpften Kriterien eine höhere Relevanz zuzuweisen als einem Resultat mit nur einem erfüllten Suchkriterium (d.h. einem gefundenen Suchterm).

ad 3: Indexierte Anfrage und Korpus werden gleichermassen als Vektoren in einem n-dimensionalen Raum repräsentiert (wobei n die Anzahl der Terme ist), und dann mittels eines bestimmten *Ähnlichkeits-Masses* verglichen. Die einzelnen Terme werden dabei als *unabhängig* voneinander auftretend betrachtet (was allerdings meist nicht der Realität entspricht).

Diese mechanischen, schwach bis gar nicht linguistischen Retrieval-Modelle haben dem Document Retrieval nicht zur erhofften Performanz verholfen und sind auch für spezifischere Anwendungen wie etwa Antwortextraktion oder Fragebeantwortung nur bedingt geeignet.

Verbesserungsansätze postulieren daher die Verwendung linguistischer bzw. heuristischer Information nicht nur während der Indexierung, sondern auch im Suchverfahren (vgl. [Verdejo et al. 99]), zum Beispiel:

- *Anfrage-Erweiterung (Query-Expansion):* Benutzung von Wildcards, Markup-Sprachen, Lexika und Thesauri (z.B. Synonymie); Umformen von Information der Anfrage.
- *Suchraum-Eingrenzung:* Präzisierung der Suchergebnisse durch zusätzliche Kriterien; (Benutzer-) Feedback oder stochastische Gewichtung.
- *Suchraum-Erweiterung* durch inkrementelle Relaxion der Suchkriterien bis hin zur reinen Keyword-Suche („Fall-back-Strategie", vgl. [Mollá et al. 00b]).
- *Treffer-Priorisierung*: Ausgabe der *Rangreihenfolge* geordnet nach Relevanz bzw. Ähnlichkeit; Extraktion der präzisesten *n* Antworten. Und schliesslich
- *Einsatz linguistisch und heuristisch motivierter Suchstrategien*

Auf den letzten Punkt soll im abschliessenden Teil dieser Arbeit eingegangen werden (→ Kapitel 5).

3 Grammatikmodellierung für die Sprachanalyse

Dieses Kapitel soll detaillierter auf real implementierte Sprachmodellierungs-Techniken eingehen und deren Vor- und Nachteile hinsichtlich der Aufgabe „E-Mail Beantwortung“ diskutieren.

Die wissenschaftliche Erkenntnis in diesem Bereich scheint derweil noch eingeschränkt zu sein, insbesondere da vor allem kommerzielle Hersteller die von Ihnen entwickelten und/oder verwendeten Algorithmen unter Beschluss zu halten pflegen. Die bestens bekannten „Produktinformationen“ enthalten lediglich vage Beschreibungen wie etwa in [Quintus 00]:

> The eContact eMail server connects to the eContact engine and receives inbound e-mail messages. ***An intelligent message processing engine analyzes*** incoming messages and ***composes personalized answers*** that are either dispatched automatically to customers or forwarded to the eContact engine

Im folgenden werden daher unterschiedliche Sprachanalyse-Verfahren für freie nicht-annotierte Texte vorgestellt, welche in allgemeinen Textretrievalsystemen zur Anwendung kommen und der Beantwortung von E-Mails als Grundlage dienen könnten.

3.1 Grammatikmodellierung mit Regulären Ausdrücken

Robustheit in der Verarbeitung grosser Korpora und Schnelligkeit sind wesentliche Ziele, die den Einsatz *endlicher Automaten* und die Beschreibung von Grammatiken mit Regulären Ausdrücken motivieren, also einer Kombination von Zeichenketten mit logischen und prozeduralen Operatoren, welche definitionsgemäss durch eine reguläre Grammatik erkannt werden kann (vgl. [Smith]). Beispiel:

```
(I+love+you)&!(don´t/not)
```

Finite State Recognizers erkennen bestimmte Muster und können Passagen übergehen, die den jeweils gesuchten Mustern nicht entsprechen (→ Pattern Matching[26]).
Finite State Transducers erzeugen darüber hinaus Markierungen. Anwendungen letzter sind Wortklassenkennzeichnung (Tagging, vgl. 2.4.4), Markierung von Nominal- und Verbal-Gruppen, Markierung der Köpfe phrasaler Einheiten, Filtern syntaktischer Funktionen.

3.1.1 Stichwortsuche

Die traditionelle Stichwortsuche extrahiert aus einer Anfrage einzelne Schlüsselwörter („Keywords“) wie etwa „Konto“ oder statische Wortketten wie „Informationen über Ihr Produkt“, welche als Indexliste an das Retrieval-Verfahren weitergereicht werden (vgl. 2.5.1). Neuere Verfahren ermöglichen zusätzlich eine *Anfrage-Erweiterung* durch Benutzung von Wildcards und einfacher, „lexikalischer“ Regulärer Ausdrücke (vgl. [Verdejo et al. 99]).

Den Vorteilen der Einfachheit und Effizienz steht der Mangel an Syntax- und Kontextinformation gegenüber. Während die Stichwortsuche im klassischen Textretrieval in erster Linie zum unspezifischen Wiederfinden ganzer Dokumente eingesetzt wird, erfüllt sie für den Bereich Antwortextraktion bzw. Fragebeantwortung und im speziellen für die Beantwortung von E-Mails dennoch wichtige Teilaufgaben:

- Auffinden relevanter Dokumente als Vorfilter der Antwortextraktion
- Auffinden relevanter Passagen anhand der lexikalischen Übereinstimmung
- Entscheidung über die adäquate Weiterverarbeitung anhand bestimmter „Trigger-Words“ → Weiterleiten, Ignorieren, Datenbankaktionen einleiten, Beantworten mittels Standard-Antwort oder weiteranalysieren (vgl. [Duvall 98]).

Alternativ kann manchmal auch je eine Regel pro Schlüsselbegriff spezifiziert werden (vgl. [eGain 00]). Das geht dann allerdings bereits in Richtung Grammatikmodellierung.

[26] Reguläre Ausdrücke („Regular Expressions“, kurz „RegEx“) sind beruhen ursprünglich auf regulären Grammatiken und damit auf Endlichen Automaten, welche die Entwicklung der sehr schnellen „Pattern-Matching“-Technologie ermöglichten.

Unterdessen haben allerdings zahlreiche Erweiterungen des RegEx-Konzepts (z.B. Speicherung bestimmter Teilausdrücke) zu einer Mächtigkeit geführt, die diejenige regulärer Grammatiken übersteigt, vgl. [Smith].

3.1.2 Grammatikmodellierung und Fragebeantwortung mit Pattern-Matching

Pattern-Matching Verfahren wurden zur Erweiterung der vormals rein lexikalischen Keyword-Suche entwickelt und entfalten die volle Mächtigkeit regulärer Ausdrücke durch deren Operatoren für Quantität, Optionalität, Negation, Eingrenzung und Speicherung[27].

```
(I+love+you)&!(don´t/not)
```

Diese Technologie erfreut sich in verschiedensten Anwendungen auch heute noch grosser Beliebtheit; dies vor allem aus Gründen der Einfachheit, Effizienz, Portabilität und Skalierbarkeit – und der Tatsache, dass es vor Jahrzehnten noch nichts besseres gab und heute dementsprechend viele Programmierer damit umgehen können. Insbesondere ermöglicht Pattern-Matching die einfache Verarbeitung regulärer Grammatiken ohne linguistischen Parser.

Die linguistische Abdeckung einer Pattern-Matching Grammatik liegt dementsprechend in deren *heuristischem und domänenspezifischem Regelsystem*. Dabei gilt es möglichst vielen der potentiellen Anfragen die entsprechenden *Erkennungs-Muster* (die „Grammatik-Regeln“) zuzuordnen und gegebenenfalls die entsprechenden Standard-Antworten bzw. Antwort-Fragmente zu formulieren. Hersteller sprechen von einem „kreativen Prozess“ ([Kiwilogic 00]).

Die Suche und Fragebeantwortung geschieht durch den Vergleich einer Eingabekette (der Anfrage) mit den vorgefertigten Regeln, wonach die entsprechende Antwort entweder direkt abgerufen oder aber über eine ebenfalls vorgefertigte Ausgabestruktur aus den Standard-Fragmenten synthetisiert wird[28] ([ebenda]).

Da ein reguläres Pattern alle denkbaren Schreibweisen des zu erkennenden Textes inklusive allfälliger Schreibfehler berücksichtigen sollte, resultieren mitunter sehr unübersichtliche, kaum eigenhändig zu erstellende geschweige denn zu administrierende Ausdrücke (vgl. [ebenda]):

```
(((((((how/hwo/hou/hows/howz/how's/hows'))+(much/muhc))/howmuch/hwomuch)&(money/cash/dough/brea
d/pesetas/peseta/dollar/dollars/bucks/buck/quid/cents/euro/euros/dm/{$}/usd/penny/quarter/dime/n
ickle))/((((how/hwo/hou/hows/howz/how's/hows')/(what/wot/waht/wat/waddya/whut)/(((what/wot/waht/
wat/waddya/whut)+is)/whats/what's/wats/wat's/wots/wot's/wahts/waht's/wotz))&(expensive/expencive
/much/muhc/cost/costs/(does+cost)/(do+cost)/cots/expense/expence/expensivness/expensiveness/pric
e/(retail+price)/rp))/(((((how/hwo/hou/hows/howz/how's/hows'))+(many/mayn/mani))/howmany/howmayn
/((((how/hwo/hou/hows/howz/how's/hows'))+(much/muhc))/howmuch/hwomuch))&((to+pay+(for/fro))/(mon
ey/cash/dough/bread/pesetas/peseta/dollar/dollars/bucks/buck/quid/cents/euro/euros/dm/{$}/usd/pe
nny/quarter/dime/nickle))))/(is+(affordable/expensive/dear)/((cost/price/expense)+(high/low/grea
t/small/big/affordable))&%?)/((can/may/might/possible/possibility)&(buy/purchase/acquire/aquire)
&(i/me))/((is/are)&(purchaseable/acquireable/(for+sale)/(on+the+market)/purchasable/acquirable))
)))&((((eye/eyye/eyes/eie/e)+(t/trek/treck/track/trak/treks/trecks/tracks/traks/thing/thingamagi
c/thingamagik/thingi/thingo/things/thingumy/thingummy/thinggy/thingy/gimmick/gimmic/gimmik/gimic
k/gimic/gimik))/eyething/eyethingi/eyethingo/eyethings/eyethingumy/eyethingummy/eyethinggy/eyeth
ingy/eyethingys/eyethingies/eyegimmick/eyegimmic/eyegimmik/eyegimick/eyegimic/eyegimik/eyetrek/e
yetreck/eyetrak/eyetrack/eytreck/eitrek/eytrek/eytrack/eytrak/eyetreks/eyettrecks/eyetrecks/eyet
raks/eyetracks/eytrecks/eytrecs/eytracs/eytracks/eitreks/eitrecs/unit/units/((the/them/this/thes
e/those)&glasses/shades/sunnies/sunglasses)/((it/one)&^eytrek)/{e-t})!{e.t.})
```

Derart aufgeblähte lexikalisierte Grammatiken sind das Resultat fehlender Abstraktions- und Modularisierungsmöglichkeiten von Pattern-Matching Grammatiken: Reguläre Ausdrücke enthalten nebst Operatoren nur die terminalen Wortformen, die in der Eingabe enthalten sind. Befriedigende Performanzresultate bei einer breiten linguistischen Abdeckung lassen sich daher nur mit entsprechend hohem Entwicklungsaufwand erzielen; spätere Verbesserungen erfordern gar einen überproportional zur Grösse des Regelsets wachsenden Erweiterungsaufwand.

Die Stärke kommerzieller Pattern-Matching Systeme liegt aus linguistischer Sicht damit nicht in der verwendeten Analyse-Technologie, sondern vielmehr in bestimmten „Zusatzdienstleistungen“ wie etwa:

[27] Der Begriff des „Pattern“ kann prinzipiell auf unterschiedliche Weise implementiert werden und theoretisch auch komplexe Analyse-Algorithmen umfassen. In kommerziellen textbasierten Systemen scheint die Verwendung Regulärer Ausdrücke jedoch vorzuherrschen, vgl. [Smith]

[28] In komplexeren Systemen ist diese Zuweisung zum Teil nicht-deterministisch und wird beispielsweise mittels stochastischer Methoden entschieden, vgl. Kapitel 4

- *Bereitstellung vorgefertigter Muster* für bestimmte Anwendungen (Lingubot: derzeit ca. 1'000 Standard-Patterns) bzw. kundenspezifische Entwicklung durch den Lieferanten
- Anbindung einer (graphischen) *Benutzerschnittstelle zur supervisorischen Entwicklung eigener Muster* auch ohne entsprechende Programmiererfahrung (vgl. Kap. 4.2.1).

Die Nachteile fehlender Abstraktion können allerdings auch umgangen werden:

3.1.3 Kaskadierung regulärer Grammatiken (FST-Kaskaden) und Pattern-Matching

Bei der kaskadierten FST-Analyse werden mehrere *Endliche Transduktoren* („*Finite State Transducers*“, FST) mit je einer eigenen zugrundeliegenden regulären Grammatik *hintereinandergeschaltet* (vgl. [Hobbs et al. 93]).

FSTs verfügen im Gegensatz zu endlichen Automaten zusätzlich über eine Relation zwischen möglichen Sequenzen von Eingabesymbolen und zugehörigen Ausgabestrukturen, d.h. liefern eine von der Eingabe abhängige Ausgabe zurück (vgl. [Neumann 01]):

Abbildung 7: Ebenen einer Finite-State-Kaskade

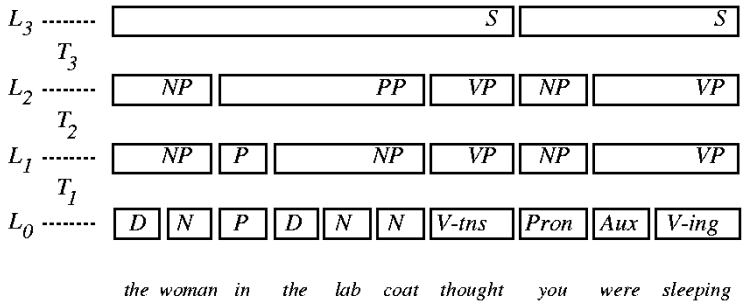


Abbildung 8: Übergangsrelationen auf Basis einer regulären Grammatik[29]

$$T_1: \left\{ \begin{array}{l} \mathrm{NP} \rightarrow \mathrm{D?\ N*\ N} \\ \mathrm{VP} \rightarrow \mathrm{V\text{-}tns \mid Aux\ V\text{-}ing} \end{array} \right\}$$

$$T_2: \{\mathrm{PP} \rightarrow \mathrm{P\ NP}\}$$

$$T_3: \{\mathrm{S\ PP*\ NP\ PP*\ VP\ PP*}\}$$

Die kaskadierte Verarbeitung regulärer Grammatiken propagiert bestimmte linguistisch und heuristisch motivierte *Basis-Prinzipien,* die insbesondere eine Umsetzung als FSTs und damit als schnelles Pattern-Matching ermöglichen (vgl. [Abney 96a], [Abney 96b]):

1. Inkrementelle Analyse
2. Keine Rekursion
3. Nur Lokale Entscheidungen
4. Easy-First-Parsing
5. Containment of ambiguity
6. Keine Unifikation

ad 1: Inkrementelle Analyse
Die syntaktische Analyse (Parsing) besteht aus einer Serie aus endlichen Transduktionen, einer sog. regulären Kaskade. Phrasen einer Ebene werden auf Phrasen der vorherigen Ebene aufgebaut (also wie beim (kontextfreien) bottom-up Parsing).

ad 2: Keine Rekursion
Phrasen enthalten nie Phrasen der gleichen oder einer höheren Abstraktions- und damit Verarbeitungsebene. Endrekursion wird durch Iteration ersetzt, welche eingebettete Teilphrasen destruktiv durch deren Kategoriesymbol ersetzt (vgl. Kap. 4.1.3).

[29] LHS = Ausgabe des FST;
diese Regeln ähneln also denjenigen in Phrasenstrukturgrammatiken (PSG, vgl. [Borsley 97])

Dies ist ein elementarer Unterschied zu kontextfreiem bottom-up Parsing, wo jede Regel in jedem beliebigen Parsing-Schritt und beliebig oft angewendet werden darf. Regulären Kaskaden unterliegt keine derart mächtige Analysetechnologie. Dafür werden verschiedene *Phrasentypen* eingeführt:

„*Chunks*" sind nicht-rekursive Kernphrasen (zum Beispiel Basis-Nominal-, -Verbal-, -Präpositional-, -Adjektiv- und -Adverbial-Phrasen). Sogenannte „*Simplex Clauses*" sind Phrasen, in denen eingebettete Strukturen destruktiv ersetzt worden sind.

Jede Transduktion ist dabei durch eine *Menge von Mustern* definiert. Ein Muster besteht aus einem Kategorie-Symbol und einem zugehörigen regulären Ausdruck[30] (ähnlichen wie bei PSG-Regeln, vgl. [Borsley 97]). Bei der inkrementellen Analyse gelten die folgenden *Regeln*:

1. Nur der *längstmögliche Treffer* erzeugt eine Output-Phrase der entsprechenden Kategorie für die nächste Ebene. z.B. „*Det N N → NP*" wird „*Det N → NP*" vorgezogen
2. Ist für das momentane Eingabesymbol der aktuellen Ebene kein Übergang möglich, und befindet sich der Automat in keinem Endzustand, so wird dieses einzelne Symbol unverändert zur nächsten Ebene weitergereicht (und. ggf. dort mit dem entsprechenden Muster erkannt, vgl. Abbildung 7).

ad 3: Lokale Entscheidungen - Keine globale Validierung bereits erkannter Phrasen
Bei traditionellen PSG-Parsern werden erkannte low-level Phrasen auf höherer Stufe oft wieder verworfen, weil sie nicht in die globale Struktur passen (zumeist aufgrund der unvermeidlichen Unvollständigkeit der zugrundeliegenden Grammatik). Diese Robustheitsverlusten gilt es zu vermeiden.

Die reguläre Grammatik einer FST-Kaskade wird denn auch nicht unbedingt als *linguistische Beschreibung* der zu analysierenden Sprache angesehen, sondern eher als *pragmatische* „Programmiersprache" des Transducers. Effizienz und Reliabilität (bzw. Robustheit) sind bei symbolorientierten Ansätzen in der Regel eine Funktion der zugrundeliegenden Grammatik; das Ziel besteht somit in der Entwicklung von Mustern, welche zuverlässige *Indikatoren für Teile der syntaktischen Struktur* sind.

ad 4: „Easy-First-Parsing"
Für die Analyse freier Texte ist Reliabilität das zentrale Anliegen. Einfache Aufgaben („Islands of certainty") sollten daher zuerst analysiert, schwierigere Aufgaben späteren Analysestufen zugeschoben werden. Dies erlaubt die Verwendung eines seichten deterministischen, und somit sehr effizienten Parsings.

ad 5: „Containment of ambiguity"
Auch unauflösbare Ambiguitäten werden den höheren Verarbeitungsstufen weitergereicht. Anschlussposition von Teilphrasen ist immer das Wurzelelement der enthaltenden Phrase.

ad 6: Keine (eigentliche) Unifikation
Unifikationsfehlschläge bei Merkmalsstrukturen und logischen Formen führen oft zur Verwerfung bereits erkannter Phrasen. Aus diesem Grunde werden die Merkmale der Satzstruktur intern als *Bit-Vektoren* repräsentiert; Merkmalsvergleiche (z.B. Kasus-Kongruenz) und -vererbung erfolgen dann über einfache Bit-Operationen. Dies ist implementatorisch etwas aufwendiger als die Verwendung vorgefertigter Unifikationsmechanismen, führt oftmals aber zu robusteren Resultaten (vgl. [Abney 96a], [Abney 96b]).

Obwohl die FST-Technologie nicht die volle Komplexität natürlicher Sprache zu modellieren vermag, hat sich dieses Vorgehen in den letzten Jahren vor allem in der Informationsextraktion stark verbreitet (vgl. [Abney 96b]), denn es ist laufzeit- und speichereffizient, robust, modularisierbar, gut erforscht und implementiert, und mit speziellen Verfahren prinzipiell *aus jeder kontextfreien Grammatiken ableitbar* – allerdings mit entsprechender Grammatik-Aufblähung, vgl. [Hobbs et al. 93]). Um letzteres zu verhindern und mehr Parsing-Flexibilität zu erreichen, ist eine zumindest kontextfreie Analyse erforderlich. Daher:

[30] Der entsprechende endliche Transducer für diese Ebene wird auch „*level recognizer*" genannt.
Die syntaktische Analyse als Pattern-Matching kann als Endlicher Automat ausgedrückt werden: Jeder Endzustand des Endlichen Automaten wird mit einem eindeutigen Muster assoziiert. Eine Phrase der entsprechenden Kategorie wurde erkannt, falls der Automat nach Abarbeitung des gesamten Musters in einem Endzustand ist (vgl. [Abney 96b])

3.2 Regel- oder symbolorientierte Verfahren

Im Gegensatz zu den meist lexikalisch-regulären Grammatiken im Pattern-Matching Verfahren werden beim *kontextfreien Parsing* ganze Sätze oder Satzfragmente, deren grammatikalische Struktur *eindeutig identifizierbar* ist, anhand vordefinierter Regeln analysiert (vgl. [Tan 98]). Die Verwendung von symbolischer *Syntaxregeln* (sog. „Constraints", vgl. [Skut et al. 97]) ermöglicht eine sehr akkurate Textanalyse, da syntaktische Teilstrukturen („Phrasen") unter Berücksichtigung derer grammatikalisch *möglichen* Abhängigkeiten (und eben nur jener) deduktiv zu einer Gesamtstruktur kombiniert werden. Einfaches Beispiel:

```
s → np, vp        np → det, n        vp → v, np
```

Der (kontextfreie) symbolorientierte Analyseprozess kann gewissermassen als eine linguistisch motivierte *Abstraktion* des lexikalischen Pattern-Matching Verfahrens betrachtet werden – mit dem Vorteil, das *vergleichsweise wenig Syntaxregeln* spezifiziert werden müssen (im Vergleich zu den oft hunderten Patterns) und sich demzufolge auch Erweiterungen der sprachlichen Komplexität recht einfach realisieren lassen (vgl. [Borsley 97].

Diese Erleichterungen werden allerdings häufig mit dem *Verlust der Theorieneutralität* und einer *mangelnden Robustheit* erkauft: Regelbasiertes Parsing erfordert die Formulierung einer zugrundeliegenden *Grammatiktheorie*, welche gerade in Hinblick auf die Analyse von Real-World-Texten wie etwa E-Mails in den meisten Fällen unzureichend hinsichtlich ihrer *linguistischen Abdeckung* sein wird: Für sich alleine genommen sind symbolorientierte Formalismen niemals in der Lage, alle *möglichen* Sprachphänomene abzudecken, und ergeben daher in vielen Fällen ein unvollständiges oder überhaupt kein Analyseresultat (vgl. [Borsley 97]).

Der Einsatz von kontextfreien Regeln (Constraints) als Möglichkeit zur *Eingrenzung* der zu erfassenden Sprachphänomene erscheint allerdings insbesondere im Zusammenhang mit *partikulären* (z.B. domänenspezifischen) Sprachanalyse-Systemen sinnvoll, wo ein bezüglich Sprachstil und Kontextbezug *bestimmbares Sprachverhalten* zu erwarten ist (vgl. Kap. 1.2).

Die Verwendung reiner Oberflächensyntax-Regeln, welche die *Konstituentenstruktur* und damit eine bestimmte Wortreihenfolge festlegen, haben sich allerdings für Aufgaben der Antwortextraktion bzw. Fragebeantwortung als unzureichend erwiesen, da sie *funktional verwandten Sätze unterschiedliche Analyse-Strukturen zuweisen*. Ein oberflächensyntax-basiertes Suchverfahren (z.B. Phrasen-Retrieval) findet dementsprechend nur Korpus-Sätze mit der Konstituentenstruktur der Anfrage (vgl. [Mollá et al. 00a]).

Sinnvoller wäre die Erkennung der *funktionalen Struktur*, also der wortstellungsunabhängigen Relationen zwischen den grammatisch funktionalen Kategorien wie etwa Subjekt, Objekt und Prädikat. In einer funktionalen Grammatik sollten verwandte Sätze verwandten oder gar identischen Repräsentationen zugewiesen werden. Somit würden verschieden formulierte Sätze mit dem selbem funktionalen Inhalt gefunden – welcher ein *Indikator für den semantischen Inhalt*, den eigentlichen Informationsgehalt des Satzes sein kann (vgl. 5.2.2.2).

Ein Retrieval über der funktionalen Struktur scheint für die Beantwortung von freien Texten wie etwa E-Mail-Anfragen also hilfreich zu sein. Daher:

3.2.1 Funktionale Grammatiken am Beispiel der Dependenz-Grammatik

Die Erkenntnis, dass bestimmte Wörter (die sog. „governors") andere bestimmte Wörter (oder Wortarten, sog. „dependents") *verlangen*, führte zur Entwicklung der Dependenz-Grammatik (DG). Bei dieser speziellen *Valenz-Grammatik* wird das Konzept der zu füllenden Valenzen von Verben auf Nomen, Adjektive und schliesslich alle Wortklassen ausgedehnt (vgl. [Mollá et al. 00a]).

Hauptmerkmale des „historischen Ansatz“ (von L. Tesnière, 1959, aus [Mollá et al. 00a]):

- *Sprachunabhängigkeit*
- *Unterteilung in Oberflächensyntax und Tiefensyntax* (→ Dependenz-Ebenen)
- *Funktionalismus:* Die *Dependenzstruktur* ist eine tiefen-syntaktische, sog. tektogrammatische oder funktionale Struktur (ähnlich f-Struktur in LFG, vgl. [Borsley 97])
- *Wortfolge* (also Oberflächensyntax, Konstituentenstruktur) *spielt eine sekundäre Rolle*; Dependenzen zwischen Wörtern dürfen sich auch überkreuzen: sog. *nicht-projektive* oder *diskontinuierliche* Strukturen.

Die (wortstellungsunabhängigen) *Abhängigkeiten zwischen einzelnen Satzelementen* spielen damit die zentrale Rolle und werden in der Dependenzstruktur des Satzes ausgedrückt. Da jede Dependenzstruktur eine kohärente hierarchische Struktur des ganzen Satzes ist, können jedoch auch die eigentlichen *Konstituenten* sehr einfach mittels rekursivem „Einsammeln“ aller Dependenzen extrahiert werden – sinnvoll beispielsweise für die Bestimmung des Satztyps (z.B. Indikativ vs. Interrogativ).

Die aus den Transformationsgrammatiken bekannten Konzepte wie Koindizierung und Bewegung (vgl. [Borsley 97]) werden in der DG nicht benötigt und es gibt keine leeren Kategorien; die DG kennt ferner *ausschliesslich lexikalische Knoten*.

Eine *nicht-projektive* funktionale Repräsentation hat den grossen Vorteil, dass verwandte Sätze eine identische Analyse erhalten, d.h. allesamt der gleichen Grundstruktur zugeordnet werden. Dies ist vor allem interessant für (vgl. [Mollá et al. 00a]):

- Strukturelle Varianten in stark flektierten Sprachen (also auch Deutsch)
- Aktiv und passiv
- Indikativ und Interrogativ
- tiefensyntaktische Abhängigkeiten auch über grosse Wortdistanz hinweg

Es existieren allerdings kaum vernünftig effiziente und ausbaubarere nicht-projektive DG-Parser, da Nicht-Projektivität ein „teures“, sprich *laufzeit- und speicheraufwendiges Konzept* ist, da hierfür unter anderem eine kontext-sensitive Sprachverarbeitung erforderlich ist.

Zwar zeigt sich, dass ein Grossteil des dependenz-orientierten Parsings bereits mit herkömmlichen kontextfreien Grammatiken gehandhabt werden kann. Einige etablierte Parsing-Verfahren geben als Kompromiss daher einige Prinzipien der klassischen DG auf und arbeiten pseudo- oder nicht-nicht-projektiv, dies dafür mit polynominaler Komplexität (Beispiel: Link Grammar, verwendet in EXTRANS, vgl. [Mollá et al. 00a] sowie 4.1.4).

Allerdings sind auch derartige funktionale Parser eher selten anzutreffen und weisen oft Mängel auf hinsichtlich Robustheit, Effizienz, Erweiterbarkeit und Modularität. Ein wirklich zuverlässiger Praxis-Einsatz funktionaler Grammatiken scheint daher trotz der theoretischen Vorteile zum jetzigen Zeitpunkt leider eher fraglich.

Ein weiteres Problem für die Verarbeitung realer Texte stellt der Determinismus aller traditioneller symbolorientierter Syntaxtheorien (und damit auch der DG) dar: Abgesehen von der praktischen Unmöglichkeit, alle sprachlichen Einzelheiten in Regeln festzuhalten, tendieren reine deterministische Regelgrammatiken dazu, bei grösser werdenden Satzlängen und Regelmengen zunehmend ambig zu werden, das heisst sie ordnen einer Wortfolge viele strukturell unterschiedliche Analysen zu (vgl. [Charniak 93]).

Innerhalb der symbolorientierten Ansätze scheint daher insbesondere für die Analyse freier Texte eine *heuristische Gewichtung* von Syntax-Regeln interessant:

3.2.2 Gewichtete und probabilistische Grammatiken

Probabilistische Grammatiken gewichteten die einzelnen Syntax-Regeln nach der Wahrscheinlichkeiten ihres Auftretens. Dies soll die Disambiguierung durch Auswahl der *wahrscheinlichsten Satzbeschreibung* ermöglichen, welche durch Verrechnung aller Gewichte der einzelnen Bestandteile (z.B. Phrasen) ermittelt wird. Zusätzlich erhält die Grammatik hierdurch eine gewisse Robustheit gegenüber sprachlichen Variationen und grammatikalischen

„Fehlern“. Die Wahrscheinlichkeiten werden meist *heuristisch* auf der Basis von annotierten Korpora gewonnen (vgl. [Charniak 93]). Beispiel:

```
VP → V          (0.6)
VP → V NP       (0.4)
```

Gewichtete Grammatiken verwenden im Gegensatz dazu *beliebige* heuristisch motivierte Gewichtungsfaktoren, deren Wert nicht der prozentualen Auftrittswahrscheinlichkeit bestimmter Regeln entsprechen muss (d.h. die Gesamtsumme aller Regeln eines Typs muss nicht 100% ergeben). Noch „statistischer“ arbeiten:

3.3 Stochastische Systeme und maschinelle Lernverfahren

Statistische Ansätze, welche die linguistischen Eigenschaften von Texten entweder gänzlich missachten oder aber inkrementell „erlernen“, haben im Textretrieval in den letzten Jahren erstaunliche Erfolge erzielt (vgl. [Verdejo et al. 99]).

Stochastische Systeme gehen das Problem der Grammatikmodellierung grundsätzlich von der anderen Seite als regelbasierte Verfahren an: Statt eine restriktive Benutzung der natürlichen Sprache *vorauszusetzen*, werden die im realen Sprachgebrauch auftretenden Abhängigkeiten *gelernt* (vgl. [Negra 00]). Dazu bedienen sich stochastische Systeme verschiedener statistischer Formalismen (z.B. Markov-Modelle, vgl. [Charniak 93]), um Auftrittswahrscheinlichkeiten und Kombinationen bestimmter Wörter und Wortfolgen *induktiv* in Relation zueinander zu bringen[31]. Allerdings resultiert daraus nicht unbedingt eine eigentliche Grammatik im Sinne von PSGs (vgl. [Borsley 97]), sondern vielmehr ein internes Modell der gelernten Abhängigkeiten.

Das Ziel *maschineller Regel-Lernverfahren* ist dagegen die automatische Induktion von Regeln für das Analysieren der Ausgangstexte oder das direkte Instanziieren von Templates (oder anderen Repräsentationsformen, vgl. 2.5). Dies erfordert ein eingehendes supervisiertes Training über einer grossen Menge realer Sprachbeispiele, welche schrittweise zu beispielsweise Templateinstanzen abstrahiert werden (vgl. [Neumann 01] und [Metzinger 00]).

Stochastische Systeme und maschinelle Lernverfahren versprechen daher besonders für *Real World-Anwendungen* und damit auch die Analyse von Korpora und E-Mail-Texten grosse Robustheitsgewinne, da sie bei entsprechendem Trainingsstand beinahe beliebige Texte zu analysieren vermögen. Allerdings benötigen die meisten aktuellen Verfahren noch eine recht umfangreiche Menge an bereits vor-annotiertem Trainingsmaterial (vgl. [Neumann 01]).

Beide Ansätze beruhen auf statistischen Verfahren, welche die relative Wichtigkeit bestimmter (nicht unbedingt linguistischer) Merkmale inkrementell präzisieren (durch Veränderung verschiedener Gewichtungsfaktoren) und daraus spezifische Muster zu erlernen versuchen. Zur Anwendung kommen vor allem Bayesische und Neuronale Netze. Diese eignen sich vor allem zu *Klassifikationszwecken* ganzer Entitäten wie zum Beispiel (linguistischer) Muster. Die Extraktion *spezifischer Bestandteile* der Analyseeinheiten (z.B. Informationsextraktion) gehört dagegen nicht zu ihren Stärken, was ihre Anwendbarkeit insbesondere für syntaktische Analysen stark einschränkt (vgl. [Brightware 99]).

Das grösste Problem dabei ist das „Black Box“-Verhalten solcher Systeme; für den menschlichen Supervisor ist es praktisch unmöglich, bestimmte (z.B. falsche) Schlussfolgerungen nachzuvollziehen und die entsprechenden Ansatzpunkte für Korrekturen zu lokalisieren (vgl. [Brightware 99]).

In der praktischen Anwendung zeigen bestehende Statistische Verfahren wie Bayesische und Neuronale Netze noch wesentliche Unzulänglichkeiten, die eine Verbesserung der verwendeten Formalismen, aber auch den *Einbezug anderer Analyseverfahren* nahe legen (z.B. Constraints zur Einschränkung der möglichen Syntaxstrukturen, vgl. [Metzinger 00]). Die Kombination constraintbasierter und statistikbasierter Verarbeitungsalgorithmen unter Beibehaltung derer jeweiligen Vorteile gehört dementsprechend zu den aktuellen Herausforderungen der maschinellen Sprachverarbeitung (vgl. [Negra 00]).

[31] gelegentlich als „Grammatikinduktion“ bezeichnet, vgl. Abbildung 1

3.4 Semantisch orientierte Analyse-Verfahren

Semantische Techniken bilden zu den zuvor erwähnten lexikalisch-syntaktischen Verfahren keine disjunkte Teilmenge, sondern können diese verschiedentlich *ergänzen*[32].

Im wesentlichen besteht der Vorteil aller Formalismen dieser Gattung in der Einschränkung der syntaktisch denkbaren Analysen durch Einbezug *zusätzlicher Informationen*. Ob diese aus einem inkrementell aufgebauten Weltmodell stammen, durch logische Inferenzschritte deduziert werden oder aus dem unmittelbaren syntaktischen Kontext erschlossen werden, ist wahrscheinlich den Applikationsanforderungen entsprechend zu entscheiden und müsste erst genauer untersucht werden.

Es zeigt sich, dass der „Königsweg" der semantischen Indexierung nicht ohne Schwierigkeiten begehbar ist. Die Transformation natürlicher Sprache beispielsweise in logische Formen bedingt eine umfassende Vor-Analyse der topologischen und/oder funktionalen Satzstruktur mit allen bekannten Unzulänglichkeiten und Fehlerquellen, die sich im Indexierungsprozess aufaddieren (vgl. Kap. 4.1.4).

Im Rahmen dieser Arbeit beschränkt sich die Diskussion semantischer Techniken auf deren *unterstützende* Funktionen, also sogenannte „seichte" Semantik wie etwa Named Entities, Synonymklassen sowie Indikatoren für semantische Frage- und Antwort-Kategorien (vgl. 5.2).

3.5 Fazit: Welches Verfahren für welche Anwendung?

Werden Keyword-Spotting- und Pattern-Matching-Verfahren als „Low End" Techniken bezeichnet, so verspricht die geschickte Kombination regelbasierter, statistischer und semantischer Ansätze die Entwicklung eines „High End" der maschinellen Sprachverarbeitung. Die Auswahl des adäquaten Analyseverfahrens ist daher abhängig von den Anforderungen der konkreten Anwendung – welche auch innerhalb des Bereichs automatische E-Mail Verarbeitung unterschiedlich aussehen kann (vgl. 4.2).

Statistische Ansätze der Sprachanalyse wie Neuronale und Bayesische Netzwerke werden im Anwendungsbereich der Fragebeantwortung vorzugsweise eingesetzt, wenn

- *Statische Antworten* genügen: Ganze *Textblöcke* werden klassifiziert und (als ganzes) beantwortet ohne Extraktion spezifischer Informationen
- *nur eine Frage* in der Anfrage enthalten ist: aus obigem Grunde
- *Unveränderliche Frage- und Antworttypen* dominieren: Alle statistischen Ansätze erfordern eine aufwendige *Trainingsphase* vor ihrem Einsatz; neue Anfrage- und Antworttypen bedingen erneutes Training.
- Klassifikation bestimmter (nicht-) linguistischer *Merkmale* bzw. *Muster* prioritär ist.

Linguistisch motivierte, symbolorientierte Analyse-Verfahren (ggf. unter Einbezug semantischer Techniken) eignen sich dagegen besonders für:

- *Dynamische* und *spezifische* Anfragebeantwortung: Extrahierte Informationen aus der Anfrage können in der Antwort verwendet werden (z.B. Kundenname) oder in Datenbankabfragen umgewandelt werden.
- Analyse und separierte Beantwortung *mehrerer Fragen* im Text
- *Einfache Administration* von (neuen) Frage- und Antworttypen.

Im Allgemeinen erfordert die *Entwicklung* statistischer Systeme etwas weniger Zeit, Innovation und sicherlich weniger linguistisches Know-How als regelgeleitete Ansätze; deren *Wartung* und *Verständnis* ist dagegen um Grössenordnungen schwieriger.

Unabhängig von der Wahl des technischen Ansatzes gilt es zu beachten, dass kein computergestütztes System die korrekte Antwort auf einen bestimmten Fragetyp ermitteln kann, ohne diese vom menschlichen Entwickler jemals eingetrichtert bekommen zu haben. Daher benötigen regelbasierte wie statistische Verfahren zumindest in der Entwicklungsphase gleichermassen menschliche Bemühungen.

[32] eine rein semantische Analyse wäre prinzipiell sicherlich denkbar, hat nach Kenntnis des Autors bislang aber keine wirklich brauchbare Inkarnation gefunden

4 Anwendungsbeispiele für Textretrieval und E-Mail Beantwortung

4.1 Wissenschaftliche Systeme

Zur E-Mail Beantwortung im speziellen sind bislang keine öffentlichen wissenschaftlichen Arbeiten bekannt. Im folgenden daher einige IE- und AE-Systeme, welche grundsätzlich vergleichbaren Aufgaben der Verarbeitung freier Texte gerecht werden[33].

Die oftmals ähnliche Systemarchitektur soll dabei jeweils nur skizziert werden, wogegen deren Schlüsselideen, Eigenheiten, spezielle Lösungsansätze und gewonnene Erkenntnisse besonders interessieren (leider sind meist keine Details wie etwa Syntaxregeln beschrieben).

4.1.1 FASTUS – ein IE-System für englische Texte

(vgl. [Hobbs et al. 93])

Die Schlüsselidee des FASTUS-Systems liegt in der Unterteilung des Indexierungsprozesses *in mehrere Stufen mit kleinen und gut definierbaren „Objekten“* . Das hat den Vorteil, dass schnelle, robuste *FST-Technologie* eingesetzt werden kann (vgl. 3.1.3), und ermöglicht eine strikte *Trennung* zwischen domänen- (evt. sogar sprach-) unabhängigen „puren“ linguistischen Informationen und spezifischem Sachwissen:

1. *Erkennung spezifischer und komplexer Wörter* wie Eigennamen und Komposita
2. *Erkennung von Basis-Phrasen*: Nominalgruppen, Verbalgruppen, Partikel
3. *Erkennung komplexer Phrasen*: Komplexe Nominal- und Verbalkonstruktionen werden anhand domänenneutraler syntaktischer Information analysiert, soweit verlässlich möglich (z.B. Nominalkonjunktionen, Massangaben)
4. *Domänen-Ereignisse*: Die Phrasen aus Stufe 3 werden in domänenspezifische Templates eingepasst (Template-Instanziierung)
5. *Struktur-Verknüpfung*: Die Informationen der vorangehenden Stufen (d.h. die einzelnen Template-Instanzen) werden miteinander verknüpft, sofern sie sich auf dasselbe Ereignis beziehen (Template-Unifikation, vgl. 2.5.4).

ad 4: Die Robustheit wird erhöht, indem alles, was bis zu diesem Zeitpunkt nicht sauber analysiert werden konnte, ignoriert wird.

Verschiedene nicht-deterministische (Sub-) Syntaxregeln (Reguläre Ausdrücke) *filtern* optionale Kategorien wie z.B. Relativsätze aus, irrelevante Adjunkte werden direkt überlesen. Gleichermassen werden auch bestimmte funktionale Relationen wie die Subjekt-Objekts-Beziehung erkannt. Die dafür verwendeten Regeln werden leider nicht beschrieben.

Aus den Korpus-Templates werden nach bestimmten Regeln offline zusätzliche Template-Varianten instanziiert, so dass dann bei der Online-Verarbeitung automatisch mehr Analysen zur Verfügung stehen, ohne dass diese einzeln spezifiziert werden müssen. Leider wird auch hierauf nicht näher eingegangen.

ad 5: Unifikation, also Vergleich und „Verschmelzung“ der instanziierten Templates geschieht über dem *gesamten Text,* d.h. Auflösung von Koreferenzen wird hier möglich. Dazu wird allerdings oft domänenspezifische, ggf. semantische Information benötigt (z.B. Thesauri).

Erkenntnisse des FASTUS-Projektes:

- jede Stufe der Kaskade entspricht einem linguistischen, sprachlich universellen Phänomentyp. Beispiel: Das auf Stufe 2 modellierte „basic clause level“ charakterisiert alle Sprachen (S-V-O etc.); hier wird also ein „linguistisches Universal“ abgedeckt
- viele IE-Aufgabe können einfacher bewältigt werden als erwartet; dies erfordert die Reduktion des linguistischen Wissens bei der Analyse auf das nötigste, genau richtige Mass an Syntaxanalyse.

[33] wie eingangs erwähnt, kann die E-Mail Beantwortung den TR-Anwendungstypen Antwortextraktion (AE), Fragebeantwortung (QA) sowie in gewisser Hinsicht auch Informationsextraktion (IE) zugeordnet werden, vgl. 2.2

4.1.2 Zweistufiger partieller Chart-Parser für deutsche Textkorpora

(vgl. [Wauschkun 96])

Das von [Wauschkun 96] beschriebene Verfahren verfolgt die Schlüsselidee, den syntaktischen Analyseprozess aus Gründen der Robustheit in zwei Stufen unterschiedlicher Granularität zu zerteilen:

1. *Grobanalyse*: Erkennung der Satz-Grundstruktur mit Fokus auf Verben
2. *Feinanalyse*: Partielle Erkennung von „Ergänzungsfolgen“, morphologische Analyse

Dazu wurde eine nicht-deterministische, symbolorientierte, partielle Syntax-Analyse mit stochastischer Gewichtung implementiert und anhand nichtrestringierter deutschsprachiger Texte getestet.

ad 1: Die *Grobstufe* ermittelt die grundlegende Satzstruktur. Diese erste Analysestufe erkennt „Teilsätze“ im Sinne der deutschen Grammatik, also Konstrukte mit Prädikat (Vollverb) und zugehörigen Ergänzungen. Dabei werden berücksichtigt:

- Verb (Vollverb, Hilfsverb, Modalverb, Passivverben u.a.)
- abtrennbares Verbpräfix
- Konjunktion
- Relativpronomen
- Interrogativ
- Infinitivpartikel „zu“
- und alle Interpunktionszeichen.

Die hier nicht untersuchten Teilbäume werden als „Ergänzungsfolgen“ gekennzeichnet und in der zweiten Stufe analysiert:

ad 2: Die *partielle Feinanalyse der Ergänzungsfolgen* analysiert Kategorien, die in der ersten Stufe unberücksichtigt blieben:

- Nominalphrasen
- Eigennamen
- Pronomen / Determinative
- Präpositionalphrasen
- Appositionen (z.B. recht häufige „nähere Angaben“ zu Personen und Firmen)
- Präpositionaladverbien, Adverbien und Adjektive

Diese Konstituenten enthalten häufig (v.a. lexikalisch) ambige Funktionswörter (z.B. „der“ → Determinativ vs. Demonstrativ). Die Überprüfung von *Kongruenzbedingungen* ist in dieser Stufe besonders daher wichtig (NPs, PPs).

Beim *Aufbau des Gesamtergebnisses* werden Ambiguitäten durch alternative Parsbäume (Grobstufe) bzw. Teilbäume (Feinstufe) repräsentiert und mit heuristischen, in Syntaxregeln deklarierten *Sicherheitsfaktoren* gegeneinander gewichtet; die zu bevorzugende Lesart wird in der weiteren Verarbeitung mittels dieser Gewichtung evaluiert.

Die *Implementation* geschieht mittels *Merkmalsstrukturen* mit morphosyntaktischen Merkmalen. Dadurch können unter anderem die zur Disambiguierung benötigten Kongruenzphänomene ausgedrückt werden.

Zu jeder Analysestufe existiert eine *umfangreiche Phrasenstrukturgrammatik* (Grobstruktur 366 Syntaxregeln (v.a. wegen Verbalkomplex-Erkennung benötigt), Feinstruktur: 249 Regeln).

Die *Evaluation* anhand eines Zeitungskorpus ergab für 56.5% der getesteten Sätze eine korrekte partielle Struktur, für 29.2% eine fast korrekte und für 14.3% gar keine Syntaxstruktur. Die durchschnittliche Parsingdauer je Satz betrug 1.3 Sekunden.

4.1.3 „Divide-and-Conquer" Strategie für seichtes Parsen deutscher Texte

(vgl. [Neumann et al. 00])

Die meisten IE-Parser analysieren bottom-up. Dies ist für englisch in Ordnung, für deutsch aber häufig unzureichend robust, weil der Parsererfolg von den aufeinander aufbauenden Parsestufen abhängt. Sätze mit sehr einfacher Grobstruktur werden daher oft nicht erkannt, wenn sie etwa komplexe NPs enthalten, welche bereits in einer früher Verarbeitungsphase nicht analysiert werden können. Beispiel:

```
„[Die vom Bundesgerichtshof und den Wettbewerbshütern als Verstoss gegen das Kartellverbot
gegeisselte zentrale TV-Vermarktung] ist gängige Praxis"
```

Die gerade für IE-Anwendungen relevanten domänenspezifischen „Argument-Fillers" (vgl. 2.5.4) sind jedoch schwierig zu identifizieren, wenn die Konstituentenstruktur nicht bekannt ist. Neumann et al.'s Schlüsselidee ist daher eine s*eichte, gemischte „Divide-and-conquer"* Parsing-Strategie aus mehreren Stufen:

1. *Vorverarbeitungsstufe (Preprocessor)*: Tokenizer, Wortartenerkennung, Morphologie-Analyse, POS-Filter, Named Entity Erkennung
2. *Domänenunabhängige Analyse der Topologischen Struktur* gemäss der linguistischen „Feld-Theorie" nach [Engel 88]
3. *Generelle und domänenspezifische Phrasenstrukturgrammatiken* (für NPs und PPs) werden auf die Inhalte der verschiedenen „Felder" der einzelnen Phrasen appliziert.

Diese Unterteilung verspricht Vorteile hinsichtlich Robustheit, Auflösbarkeit bestimmter Ambiguitäten sowie einen hohen Grad an Modularität.

ad 1: Die Vorverarbeitungsstufe besteht aus vier Modulen:

Der *Tokenizer* ermittelt die Analyse-Einheiten und kategorisiert detailliert spezielle Ausdrücken wie etwa Abkürzungen mittels regulärer Grammatiken (schon ähnlich wie POS-Tagging). Anschliessend erfolgt die Satztrennung durch Interpunktion.

Bei der *Morphologie-Analyse* werden *Komposita und Hyphenkoordinationen* aufgelöst, und allen erkannten Wörtern werden Stamm- und Flexionsinformationen, POS-Kategorie sowie eine Liste der möglichen Lesarten (für die spätere Disambiguierung) zugewiesen.

Das *POS-Filter* definiert *Regeln zur Disambiguierung*, welche zu einem FST kompiliert und durch die Regeln des Brill-Taggers ergänzt werden (vgl. 2.4.4).

Der *Named Entity Finder* ermittelt bestimmte Namenentitäten basierend auf einer regulären Grammatik. Zur Verringerung des *Koreferenzpotentials* werden diese stets ohne allfällige Designatoren in ein dynamisches Lexikon eingetragen (z.B. „Braun AG" als „Braun"). *Kategoriale Ambiguitäten* (z.B. „Braun" vs. „braun") können auf dieser Ebene nicht aufgelöst werden und müssen daher als solche gekennzeichnet und weitergereicht werden („containment of ambiguity", vgl. Kap. 3.1.3).

ad 2: Die *Analyse der Topologischen Struktur* erfolgt mittels in vier Stufen kaskadierten FS-Grammatiken gemäss der linguistischen *Feld-Theorie*:

Verb-Gruppen dienen dabei als Grundlage zur Feld-Segmentierung: Verbgruppen wie etwa

```
„hätte überredet werden müssen"
```

können in linke und rechte Verb-Teile aufgespaltet werden. Diese segmentieren den Satz in die Bereiche *Vorfeld, linker Verbteil, Mittelfeld, rechter Verbteil* und *Nachfeld* (vgl. [Engel 88]). Diese Verbgruppen-Analyse ist gemäss [Neumann et al. 00] recht komplex.

Nebensätze /Subphrasen werden auf gleiche Weise behandelt, wobei der linke Verbteil entweder leer ist oder ein Relativpronomen enthält, und die gesamte Verbgruppe in den rechten

Verbteil verfrachtet wird. Auf diese Weise wird die Phrasenfolge innerhalb eines Felds kaum eingeschränkt. Das ist natürlich ideal für die Verarbeitung deutschsprachiger Texte.

Schwierigkeiten bei der Analyse von Verbgruppen bereiten *Diskontinuitäten*. Diese benötigen zur Auflösung Kontext-Information, welche bis zu dieser Stufe noch nicht erschlossen wurde. Ein zweites Problem ist die massive *morphosyntaktische Ambiguität der (deutschen) Verben;* in diesem Falle kommen unterspezifizierte Chunk-Kategorien zur Anwendung, welche später in weiter gefasstem Kontext konkretisiert werden können.

Die *strukturelle Repräsentation* der Verbgruppen und Nebensätze erfolgt als Merkmalsstruktur (mit den Merkmalen Typ, Stamm u.a.). Auch *Basis-Phrasen* werden als Merkmalsstrukturen abgelegt.

Rekursive Strukturen (z.B. Relativsätze), welche in Regulären Grammatiken nicht als solche repräsentiert werden können, werden mittels *Iterationen* modelliert, welche die erkannten eingebetteten Strukturen mit deren Typ ersetzen (vgl. 3.1.3). So wird die strukturelle Komplexität erheblich reduziert. Diese *destruktive Phrasen-Kombination* wird wiederholt durchlaufen, bis keine Basis-Phrasen mehr reduziert werden können.

Anschliessend folgt der Aufbau der kompletten topologischen Satzstruktur auf Basis der erkannten Phrasen und der bis hierher noch nicht konsumierten Wortinformationen anhand regulärer Regeln:

```
CSent ::=...LVP...[RVP ]...
SSent ::=LVP...[RVP ]...
CoordS ::=CSent (, CSent)*Coord CSent |::=CSent (, SSent)*Coord SSent
AsyndSent ::=CSent , CSent
CmpCSent ::=CSent , SSent |CSent ,CSent
AsyndCond ::=SSent , Ssent
```

ad 3: Bei der *partiellen Phrasen-Erkennung* werden in erster Linie nicht-rekursive NPs und PPs mittels regulärer Grammatiken identifiziert. Dies erfordert nun keine kontextfreie Sprache mehr, da die kritischen Satzteile, also vor allem Verbalkonstrukte, schon in der vorangegangenen FST-Kaskade erschlagen wurden. Eine zusätzliche *Kongruenzprüfung* entfernt unplausible Phrasen.

Diese Analysestufe ist *domänenenunspezifisch* und kann wegen des hohen Grads an Modularität bei Bedarf prinzipiell durch eine spezifische ersetzt werden, ohne die Generalität der restlichen Komponenten zu gefährden.

Die Endausgabe des gesamten Parsing-Prozesses ist eine flache, *unterspezifizierte Dependenzstruktur* („UDS"). Hier werden nur die obersten Skopus- und Anschlusspositionen (Wurzel des Haupt- resp. Teilsatzes) von Modifikatoren (d.h. Komplemente oder Adjunkte enthaltende Phrasen, z.B. PPs) repräsentiert. Die Entscheidung über die exakte Anschlussposition wird somit dem domänenspezifischen Wissen allfällig nachfolgender Analyse-Komponenten überlassen.

Die *Evaluation* des Systems ergab ein F-Mass von 87.14% bei unannotiertem Text sowie eine mittlere Parsingdauer 0.57s pro Satz. Als Weiterführung wäre der Aufruf eines *„tiefen" Parsers* zur Ermittlung der exakten syntaktischen Struktur *bei Bedarf* denkbar (z.B. zu Disambiguierungszwecken und domänenspezifischer Weiterverarbeitung).

Wichtige Erkenntnisse des Projekts:

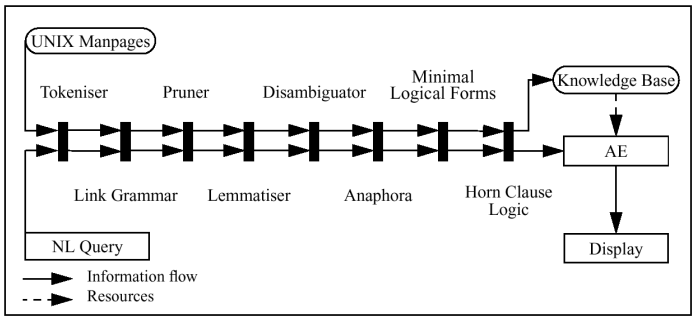
- Ein „*Unterbau*" (Preprocessor) leistet hilfreiche Vorarbeit
- Die *Trennung von genereller und spezifischer Information* scheint sich zu bewähren
- *Modularität* erleichtert inkrementelle Verbesserungen und (domänenspezifische) Weiterentwicklungen.

4.1.4 EXTRANS: Vollständige Syntaxanalyse und semantische Indexierung

(vgl. [Mollá et al. 00b])

Das Projekt EXTRANS[34] versteht sich als Versuch, den aktuellen State-of-the-art der NLP[35] für eine brauchbare *semantikorientierte Antwortextraktion* einzusetzen (am Beispiel des Unix-Benutzerhandbuchs, englisch). Dadurch gelangt dieser Ansatz in den Anforderungsbereich für eine (halb-) automatische Mail-Beantwortung. Dazu werden verschiedene Analysestufen kaskadiert:

Abbildung 9: Verarbeitungsstufen im Indexierungs- und Retrievalprozess in EXTRANS



Als erstes verwertet der *Tokenizer* sämtliche *Formatierungs-Information* der Originaltexte (Überschriften, Paragraphen etc.) und separiert die Analyseeinheiten anhand *heuristischer Regeln* (implementiert als Reguläre Ausdrücke).

Die anschliessende nicht-deterministische *vollständige Syntaxanalyse* (Link Grammar Parser*)* gibt alle möglichen Satzanalysen als *gerichtete Dependenzstrukturen* des Satzes aus. Die Kategorien unbekannter Wörter werden durch Evaluation des umliegenden syntaktischen Kontexts *abgeschätzt*. Der Parser ignoriert dabei sukzessive einzelne Wörter, bis eine gültige Dependenzstruktur gefunden wird; die ungebundenen Wörter sind jedoch nicht verloren, sondern können bei Bedarf im Retrieval als Keywörter genutzt werden.

Der nachgeschaltete *Pruner* verwirft alle offensichtlich falschen Strukturen des Syntax-Parsers anhand eines manuell erstellten *heuristischen Regelsets*. Danach werden die flektierten Wortformen auf ihre *morphologische Grundform reduziert* (Lemmatiser). Ein *Disambiguator* löst *Anschlussambiguitäten* (PPs) mittels *trainiertem Domänen-Wissen* auf; nicht auflösbare Mehrdeutigkeiten werden als Alternativen weiterverfolgt. *Anaphern* wie etwa Kopf-Adjunkt Beziehungen werden intrasententiell anhand rein syntaktischer Information aufgelöst; ungeklärt bleibt die Frage, wie sinnvoll bzw. nötig eine *inter*sententielle Referenzauflösung wäre.

Aus den zuvor (partiell disambiguierten) Dependenz-Strukturen der Dokumentenkollektion wie auch der Anfrage werden flache *minimale logische Formen* (MLF) in Horn-Klausel-Logik erzeugt, welche als semantische Entitäten die in den analysierten Sätzen enthaltenen *Objekte, Ereignisse* und *Eigenschaften* repräsentieren. Nicht auflösbare Ambiguitäten werden unterspezifiziert dargestellt. Die MLF werden zusätzlich mit *Synonymie-Informationen* für Nomen und Verben angereichert.

Grundidee der MLF ist es, genau denjenigen Teil der semantischen Information zu repräsentieren, der für die entsprechende Anwendung (hier: Antwortextraktion über Unix Manpages) relevant ist, um eine schnelle und robuste Verarbeitung zu ermöglichen.

EXTRANS versucht Anfragen online durch *Refutation* (Beweis durch Widerspruch) zu inferieren und damit zu „beantworten“ (nur seichte Inferenz). Das System findet *alle Beweise* (≈ Antwortpassagen) und berechnet auch die „semantische“ Übereinstimmung von Frage und potentieller Antwort als *priorisierte Ausgabe*. Dies wäre insbesondere auch für eine halbautomatische E-Mail Beantwortung denkbar.

[34] Stand April 2001; spätere Verbesserung wurden nicht berücksichtigt.
[35] NLP = Natural Language Processing = Verarbeitung natürlicher Sprache

Eine *Fall-back-Strategie* erhöht die Robustheit durch *inkrementelle Relaxion* der Beweis-Kriterien und Durchlaufen verschiedener Analysestrategien bis hin zur reinen Keyword-Suche. Als Schlüsselwörter assertiert werden dabei alle unbekannten und unanalysierbaren Wörter sowie diejenigen Ausdrücke, welche in den MLF die Objekte, Ereignisse und Eigenschaften ergeben; sehr häufige Wörter werden als schwach diskriminierend erachtet und daher ignoriert.

Nicht repräsentiert werden (bis anhin): Tempus, Aspekt, Modalität sowie tiefe semantische Bezüge wie etwa Intentionalität.

Die EXTRANS-Analyse ist vor allem wegen der vollständigen Syntaxanalyse und der logischen Beweisführung ziemlich zeit- und speicheraufwendig. Die Resultate sind zum Teil recht präzis und vollständig, allerdings *nur für bestimmte Fragetypen*. Insbesondere werden (noch) keine Strategien der effektiven Fragebeantwortung eingesetzt, sodass zum Beispiel Anfragen zu Kausalitäten wie *„Warum...?“* nicht gezielt beantwortet werden (vgl. 5.1).

4.2 Kommerzielle E-Mail Beantwortungssysteme

Die Spannweite der auf dem kommerziellen Markt befindlichen E-Mail Beantwortungssysteme reicht vom simplen „E-Mail Router“ bis zum kompletten „e-mail management system for intelligent customer communication“ (vgl. [eGain 00]).

Grundsätzlich lassen sich diese Systeme hinsichtlich der Art und des Grades menschlicher Interaktion in mehrere *applikationstechnische Segmente* und deren Derivate unterteilen (vgl. [Lambert 98])[36]:

- *Routing*: Empfang, Analyse und Weiterleitung an den richtigen menschlichen Bearbeiter[37]
- *Queuing*: Kategorisierung der eingehenden Mail-Anfragen in einem mehreren Bearbeitern zugänglichen *Pool*, aus dem letztere beliebige Mails selektieren und individuell beantworten (wird v.a. in Call Centers eingesetzt)[38]
- *Answering*: Analyse, Vergleich mit einer Wissensbasis, Entscheidung über automatische Beantwortbarkeit, Abruf oder Generierung einer kompletten Antwort oder Kombination mehrerer Antwort-Fragmente; ansonsten vermerken oder weiterleiten[39].

Die Grenzen sind jedoch zunehmend verschwimmend, sodass unterdessen verschiedene *praxisorientierte Kombinationen und Erweiterungen* der oben erwähnten Ansätze anzutreffen sind, so zum Beispiel[40]:

- *Routing mit Antwortvorschlägen*, wenn keine zuverlässige Beantwortung möglich ist
- *Automatische Auswahl des geeigneten menschlichen Bearbeiters* mittels Wissensbasen betreffend Spezialgebiet, Erfahrung und Erreichbarkeit ([eGain 00]).
- *Einbezug der bisherigen Kundenkommunikation* und damit Identifizierung der einträglichsten, treuesten, unangenehmsten Kunden (oder Mitarbeiter, vgl. [eGain 00])
- *Integration* in eine e-commerce Entwicklungsumgebung bzw. direkt in Web-Sites, z.B. als „virtual salesman“ (vgl. [Kiwilogic 00])
- Workflow-Automation

Im folgenden die getesteten Systeme:

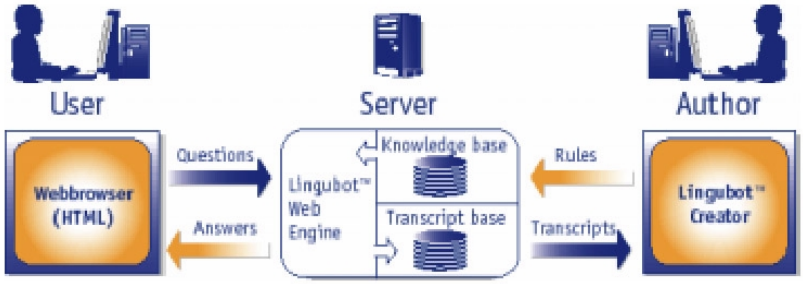
[36] *Linguistisch* betrachtet wäre sicherlich eine andere Einteilung adäquater, denn die verschiedenen Anwendungstypen beruhen teilweise auf den gleichen Analyse-Formalismen!
[37] Vertreter: Genesys (vormals Adante), Distributed Bits, Kana Communications
[38] Vertreter: Mustang Software
[39] Vertreter: HNC Mindwave (vm. Aptex), Firepond (vm. Brightware), Dialog, General Interactive, Transform Response
[40] Vertreter: eGain, Firepond, Avaya (vm. Quintus), Kiwilogic

4.2.1 Kiwilogic Lingubot

4.2.1.1 Überblick

Der Softwarehersteller Kiwilogic bietet für (Gross-) Firmen verschiedene Produkte zur automatischen Verarbeitung natürlichsprachlicher Kundenanfragen (in Englisch) an. Die Software „Lingubot“ erlaubt gemäss Herstellerangaben die Aufbereitung und Beantwortung von via E-Mail oder HTML-Formularen empfangenen Texten. Der supervisierende Administrator (der „Autor“) benötige dazu weder linguistische noch Programmiererfahrung.

Abbildung 10: Arbeitsweise von Lingubot



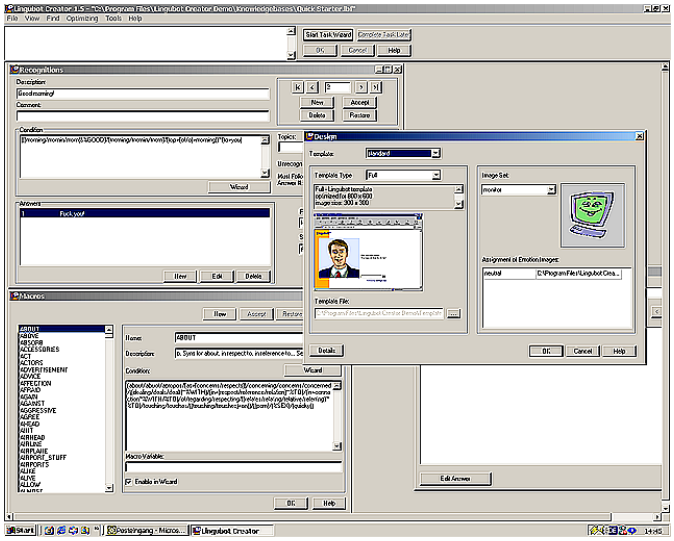
4.2.1.2 Grammatikmodellierung und Retrieval

Die frei erhältliche Demoversion gibt das Geheimnis des vermeintlichen Wunderwerks frei: Die Analyse des Anfragetextes erfolgt *ausschliesslich mittels Regulärer Ausdrücke*. Diese kann der Administrator entweder eigenhändig formulieren, den über tausend vordefinierten Muster entnehmen oder aber durch Eingabe des zu beantwortenden Fragetextes automatisch vom System generieren lassen:

Ein *interaktives Hilfsprogramm* („*Recognition Wizard*“) bittet den Autor, weniger wichtige Wörter der Frage – beispielsweise die Funktionswörter – zu markieren. Diese werden erst gar nicht in den regulären Ausdruck übernommen (was der Benutzer ohne Programmierkenntnisse aber nicht bemerkt).

Anschliessend wird ein Thesaurus-artiges *Lexikon* konsultiert und die Anfrage mit *Synonymen* zu den verbleibenden Wörtern angereichert („Query-Expansion, vgl. Kap. 2.6). Auch dieser Schritt erfolgt in Interaktion mit dem Administrator, welcher zur Auswahl einer *Synonymklasse* gebeten wird (Vermeidung lexikalischer Ambiguität). Eindeutige Wörter werden automatisch synonymisiert (z.B. „about“, „do“, „for“). Erfreulicherweise kann das Lexikon auch eigenhändig erweitert werden.

Abbildung 11: Benutzeroberfläche des Lingubot Creator



Nach Abschluss dieser „Offline-Analyse“ kann der Benutzer für die reguläre Repräsentation der nun generalisierten Frage eine oder auch mehrere *komplette natürlichsprachliche Antwort(en)* spezifizieren.

Der Rest ist selbsterklärend: Eine empfangene Kundenanfrage wird – falls möglich – mit einem der definierten regulären Muster „gematcht“ und durch dessen Ausgabetext beantwortet. Der linguistische Kern von Lingubot ist damit ein simpler endlicher Transducer auf Basis von Pattern-Matching-Technik.

Die Präzision des Systems ist dementsprechend von der Abdeckung durch die definierten Muster bestimmt und daher nur mit entsprechend hohem Administrationsaufwand zufriedenstellend. Die mitgelieferten *lexikalischen Ressourcen* (Thesaurus mit rund tausend Synonymen, Wissensbasis mit ebenso vielen regulären Mustern) stellen immerhin eine solide Grundlage dar.

4.2.1.3 Erweiterbarkeit

Es sind bei Lingubot „Skripts“ vorgesehen, welche die Einbindung prinzipiell beliebiger *externer Algorithmen* zur Erweiterung des Analyse-Formalismus ermöglichen sollen (wie das wirklich funktioniert, bleibt im Rahmen der getesteten Evaluationsversion leider unerwähnt).

Kiwilogic Lingubot scheint auch bestimmte *statistische Hilfsmittel* einzusetzen, welche dem Entwickler / Administrator Aufschluss geben über die durchschnittliche Dialoglänge, die Auftrittshäufigkeiten bestimmter Wörter und sogar die *Matching-Wahrscheinlichkeit*. Ob diese Informationen auch zum Suchprozess beitragen oder nur der Evaluation des Regel-Sets dienen, bleibt leider unerwähnt.

4.2.1.4 Evaluation

Die Investitionskosten für Lingubot werden vor allem durch das reichhaltige „Drumherum“ gerechtfertigt, also die graphische Benutzerschnittstelle mit „Recognition Wizard“ zur einfachen Generierung eigener Muster auch ohne entsprechende Programmiererfahrung, die vordefinierten Regeln und Lexikoneinträge, ein hilfreiches Tutorial und die automatische Einbindung in ein Web-Interface, aber keinesfalls durch dessen „linguistischen Kern“. Bemerkenswert ist allerdings die hohe Erkennungsrate im Verhältnis zur bescheidenen Analyse-Technologie mit regulären Ausdrücken.

4.2.2 eGain Mail

Der amerikanische Hersteller eGain bietet eine ganze Palette von Systemen für die (natürlichsprachliche) Kommunikation zwischen einem Unternehmen und dessen Kundschaft an. Den Kunden stehen verschiedene Kommunikationskanäle (z.B. Web, Telefon, Mail) offen. Die entsprechenden Systeme können integriert werden und ergänzen einander:

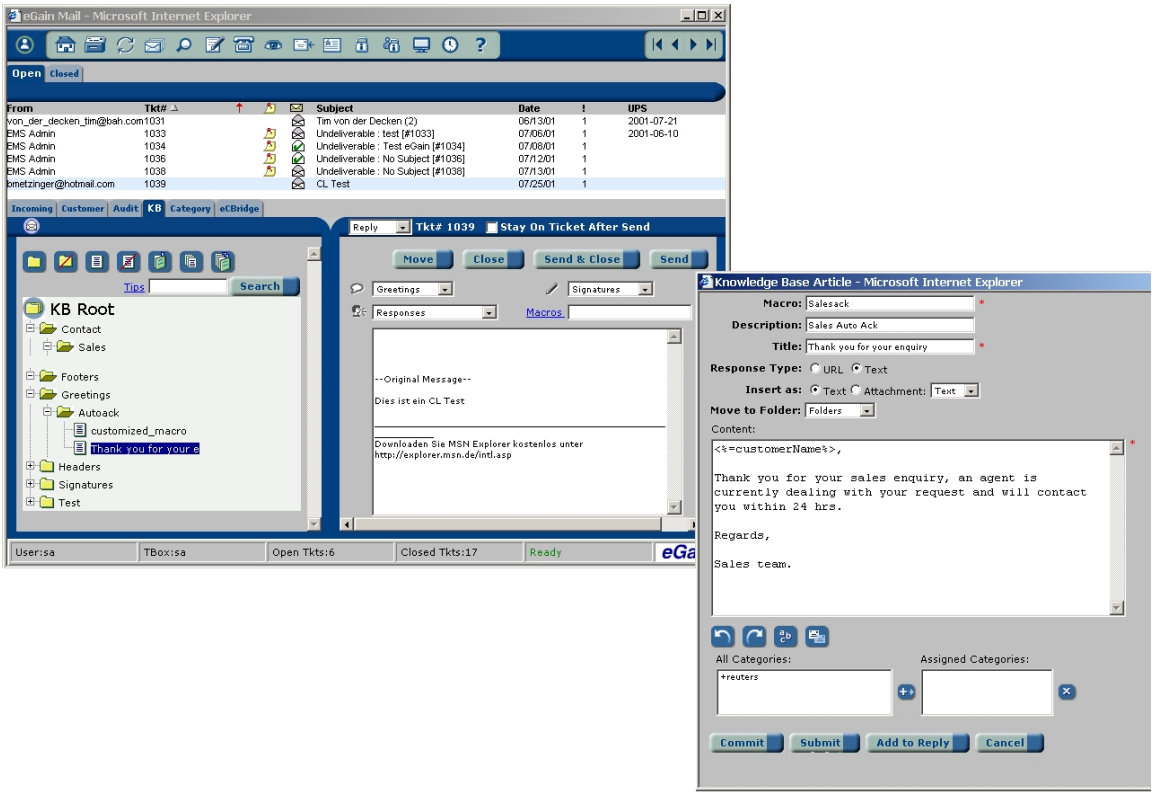
- *eGain Assistant*: Virtuelle Führung über das Web
- *eGain Inform*: Informations-Selbstbedienung über das Web
- *eGain Knowledge*: Virtuelle Beratung über das Web
- *eGain Live*: Live Beratung über das Web
- *eGain Voice*: Ergänzt eGain Live durch Voice-over-IP Technik
- *eGain Mail*: E-Mail Empfang und Verarbeitung
- *eGain Campaign*: Email Distribution
- *eGain Commerce Bridge*: Integration von Drittsystemen

Für die Auswertung stand Zugang zu einem Testserver mit *eGain Mail* zur Verfügung. Das linguistisch ausgereiftere eGain Knowledge (mit Morphologischer Analyse) konnte leider nicht getestet werden.

4.2.2.1 Überblick

eGain Mail ist ein komplettes Management-System zur Erfassung, Verarbeitung und Überwachung eingehender elektronischer Anfragen. Die Software ist auf eine benutzerfreundliche graphische Bedienung ausgerichtet und umfasst verschiedenste Werkzeuge und Ansichten (z.B. Benutzer-Management, Monitoring, Reporting).

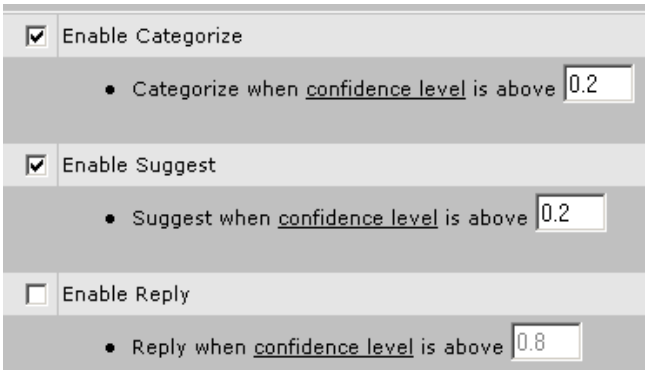
Abbildung 12: eGain Mail Benutzeroberfläche



4.2.2.2 E-Mail Verarbeitung

Eingehende Nachrichten werden wahlweise direkt manuell verarbeitet oder nach bestimmbaren Verarbeitungsregeln (sog. *Rules*) automatisch gefiltert, kategorisiert, weitergeleitet oder beantwortet. Global einstellbare Schwellwerte (*„Confidence-Levels“)* bestimmen dabei die für die automatische Verarbeitung erforderliche Übereinstimmung zwischen Anfrage und den internen Regelsets (siehe weiter unten).

Abbildung 13: eGain Global Settings



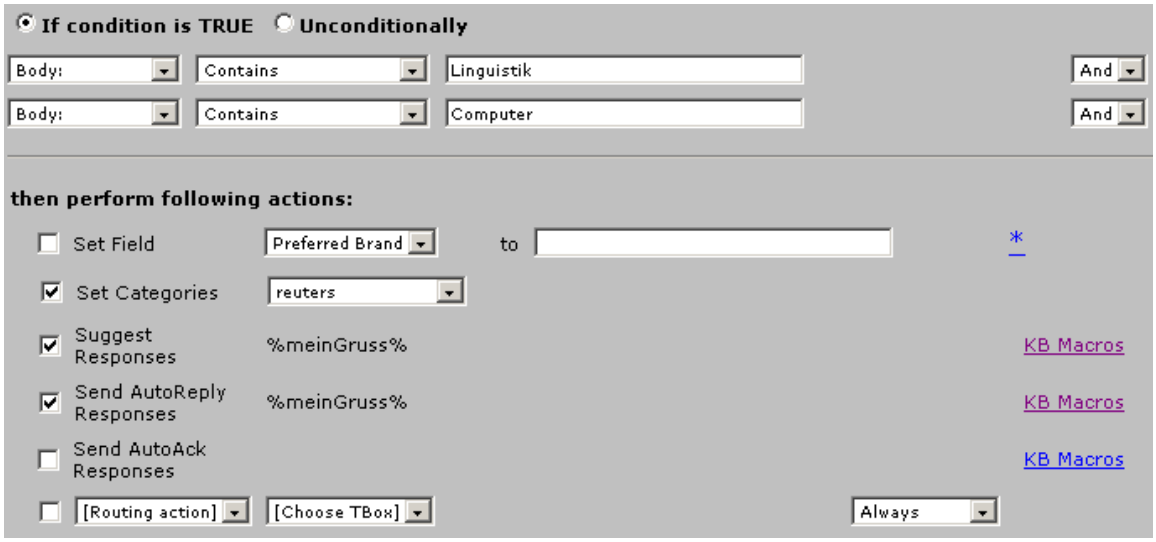
Interessanterweise wird bei eGain Mail zwischen vorgeschlagener (d.h. halbautomatischer) und autonom durchgeführter Mail-Beantwortung unterschieden, was eine angesichts der linguistisch bedingten Unsicherheiten bei der Erkennung freier Sprache sehr sinnvolle Abstufung des *Automationsgrads* erlaubt.

Die (zum Teil linguistischen) Verarbeitungsregeln stehen unabhängig von der Systemreaktion (Kategorisierung, Vorschlag, Beantwortung) zur Verfügung:

4.2.2.3 Sprachverarbeitung, Indexierung und Retrieval

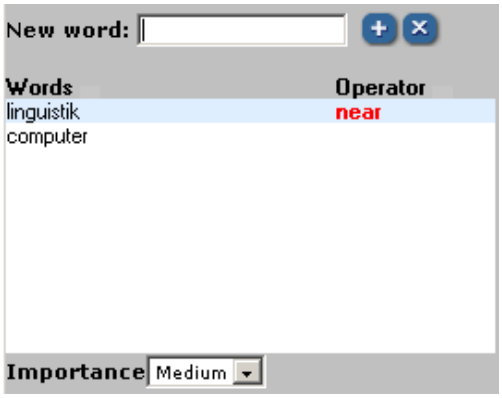
eGain Mail verwendet zwei „linguistische“ Analysemechanismen. Zum einen können einfache logisch verknüpfbare Keyword-Regeln direkt erstellt und zugewiesen werden:

Abbildung 14: eGain Rule Administration



Dediziertere Möglichkeiten bieten zum anderen die sogenannten *Phrase-Rules*, welche eine auf dem „Indexing-Service“ von Microsoft basierende Sprachanalyse vornehmen:

Abbildung 15: eGain Phrase Rule Editor



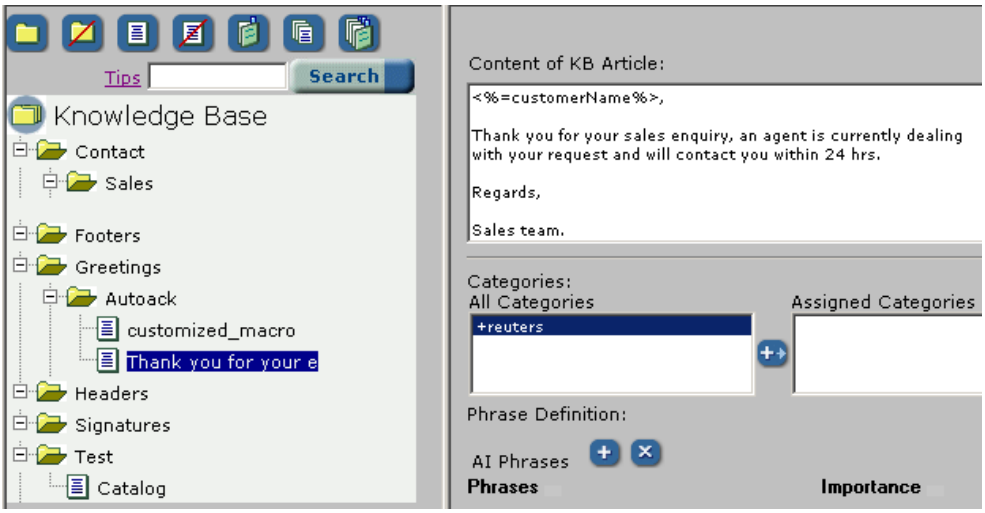
Der Microsoft Indexierungsservice bietet zur Verarbeitung natürlichsprachlicher Texte eine recht mächtige Absprachesprache (*Query Language*) mit ausgereiften Suchmöglichkeiten:

- *Trunkierung* (z.B. key* für key, keying, keyhole, keyboard usw.)
- *Lemmatisierung* (z.B. sink** für sink, sinking, sank und sunk)
- *Logische Verknüpfung* mit den Operatoren AND, OR, NOT
- *Approximative Verknüpfung* mit NEAR (Wortdistanz-Mass)
- *Volltextsuche* mit Noun- und (einfachem) *Noun-Phrase-Spotting*
- *Vektor-Suche* mit Komponentengewichtung (z.B. light[50], bulb[10], "light bulb"[400])
- Suche nach *Dokumenteneigenschaften* (Name, Grösse, Titel, Autor usw.)
- Relationale *Verknüpfung* von Dokumenteneigenschaften <,<=,=,>=,>
- Definition neuer Dokumenteneigenschaften für die Suche
- *Reguläre Ausdrücke*
- Unterstützung verschiedener *Objektsprachen* (En, ge, fr, es usw.).

Auf diese Weise indexierte E-Mail-Texte können unter Einbezug der gefundenen und extrahierten Sprachmerkmale wie z.B. Kundenname kategorisiert und weitergeleitet oder aber

mittels vordefinierter und *dynamisch assemblierter Antwort-Muster* aus der Wissensbasis automatisch beantwortet werden:

Abbildung 16: eGain Auto Reply Editor



4.2.2.4 Evaluation

Die Software vermittelt einen äussert professionellen Eindruck und scheint sehr gut auf die kommunikativen Bedürfnisse moderner (Gross-) Firmen ausgerichtet[41]. Der Grossteil linguistischer Verarbeitung ist zwar nicht in eigenem Hause entstanden, dafür aber sehr gut und vor allem auf breiter Ebene in die gesamte Applikation integriert.

Der Microsoft Indexierungsservice vereint die Vorteile verschiedener Textretrieval-Ansätze (Stichwortsuche, Stemming, Morphologische Analyse, Phrasenindexierung, Vektorsuche) und funktioniert ausserdem erstaunlich gut für restringierte und elaborierte freie Sprache. Komplexere Sprachphänomene tiefensyntaktischer und semantischer Natur, etwa textuelle Referenzen oder Bedeutungsintention werden angesichts der schwach-syntaktischen Textanalyse natürlich nicht angegangen; dafür arbeitet der Indexing-Service sehr robust und „praxisorientiert", d.h. ist für viele Anwendungen sicherlich mehr als genügend (offenbar bestätigt sich, dass Microsoft die grösste kommerzielle Computerlinguistik-Abteilung betreibt).

Wünschenswert wäre natürlich eine Programmierschnittstelle (API), um das Werkzeug spezifischen Anforderung entsprechend erweitern zu können. Dieser Möglichkeit wurde im Rahmen dieser Arbeit jedoch nicht nachgegangen.

4.2.3 Firepond eService Performer Email Assistance

Firepond (ehemals Brightware) vertreibt umfangreiche, derzeit vielleicht die ausgereiftesten Lösungen im Bereich Mail-Beantwortung und NLP-Web-Integration. Der Hersteller spricht von der „besten und teuersten" Variante im Konkurrenz-Vergleich, was nach Recherche des Autors zumindest bezüglich des zweiten Arguments zuzutreffen scheint[42]. Hinsichtlich der Verwendung von Sprachtechnologie wecken aus diesem Hause vor allem zwei Produkte das computerlinguistische Interesse:

- *Email Assistance*: Halb- und vollautomatische Mail-Verarbeitung (inbound)
- *Concierge*: Frage-Beantwortungs-Modul zur Integration in Web-Portale

Die Software „*Email Assistance*" vereint traditionelle Pattern-Matching Technik mit sub-syntaktischer und morphologischer Sprachverarbeitung sowie Data-Mining Ansätzen in benutzerfreundlicher und heuristisch motivierter Weise zu einem umfassenden Mail-Verarbeitungs-System. Die professionelle Ausrichtung des Herstellers, der die Position des

[41] und ebenso auf deren Budget: Die Kosten für eine eGain Mail Lizenz betragen rund USD 50'000.
[42] USD 190'000, oder 95'000 pro Jahr plus 50-75'000 für die Installation sowie rund 18% pro Jahr für den Unterhalt.

Marktleaders in diesem Bereichs anstrebt, zeigt sich In der Unterstützung unterschiedlichster Systemplattformen[43], Datenbankstandards[44] und Entwicklungsschnittstellen[45].

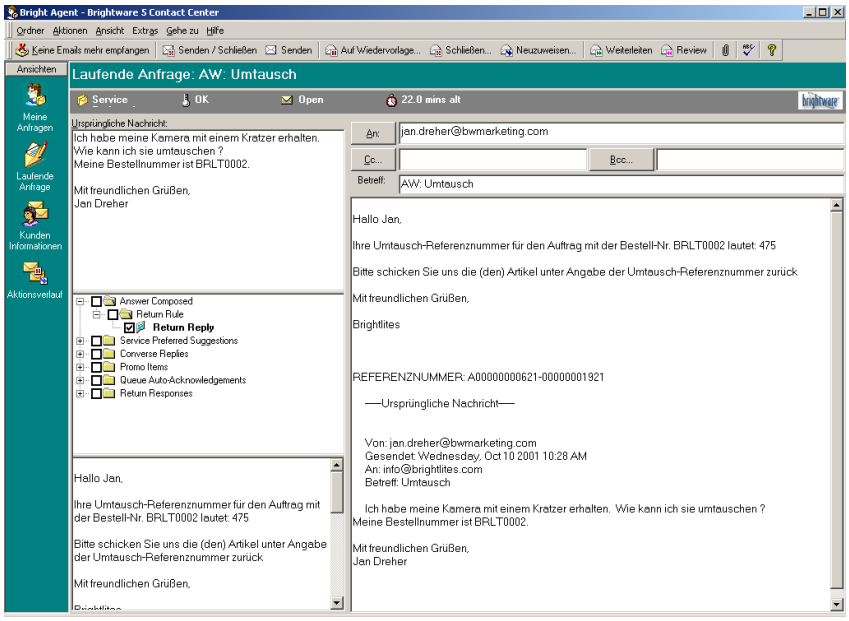
Das Produkt „*Concierge*“ baut auf einer etwas eingeschränkteren Sprachanalyse-Engine zur zeitkritischen Beantwortung natürlichsprachlicher Fragen innerhalb einer Web-Umgebung auf, verwendet die gewonnenen Informationen aber zusätzlich als Basis zum Document Retrieval. Zudem stellt die Software bei Unsicherheiten Gegenfragen.

Da beiden Lösungen im wesentlich die gleichen NLP-Technologien zugrunde liegen, wird im folgenden vor allem die linguistisch etwas ausgereiftere Software „Email Assistance“ betrachtet. Da zu dieser Software gegenüber den anderen getesteten Produkten mehr Detailinformation verfügbar war, gestaltet sich dieses Kapitel etwas umfassender als die vorherigen.

4.2.3.1 Übersicht

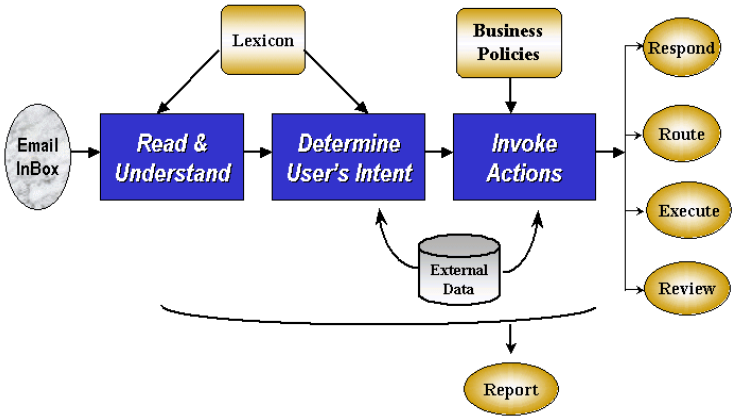
Wie alle getesteten Lösungen bietet Email Assistance eine intuitive Benutzerschnittstelle:

Abbildung 17 : Benutzeroberfläche Email Assistance



Die Betrachtung des groben Analyse-Verlaufs spiegelt das hinlänglich bekannte Verarbeitungsmodell der meisten TR-Verfahren wieder:

Abbildung 18: Architektur-Übersicht Email Assistance



[43] Unter anderem Windows 2000 / NT4, Linux, Solaris, HP-UX
[44] SQL, Oracle, DB2, Sybase
[45] HTML, XML, CORBA, ODBC, JDBC, OLE, API etc.

Der natürlichsprachliche Input wird anhand von sprachlichen (hier: lexikalischen und „externen“) Ressourcen analysiert und „interpretiert“, um anschliessend anhand bestimmter domänenspezifischer Vorgaben („Business Policies“) die entsprechende Systemreaktion auszulösen, was im Falle E-Mail-Verarbeitung eben bedeutet: Anfrage beantworten, weiterleiten, zur manuellen Bearbeitung senden oder gegebenenfalls direkt bestimmte Folgeraktionen einleiten.

Des weiteren wird – wie ebenfalls bei den meisten kommerziellen Lösungen – über all diese Schritte buchgeführt und für bestimmte Anfragen ein entsprechender Report generiert.

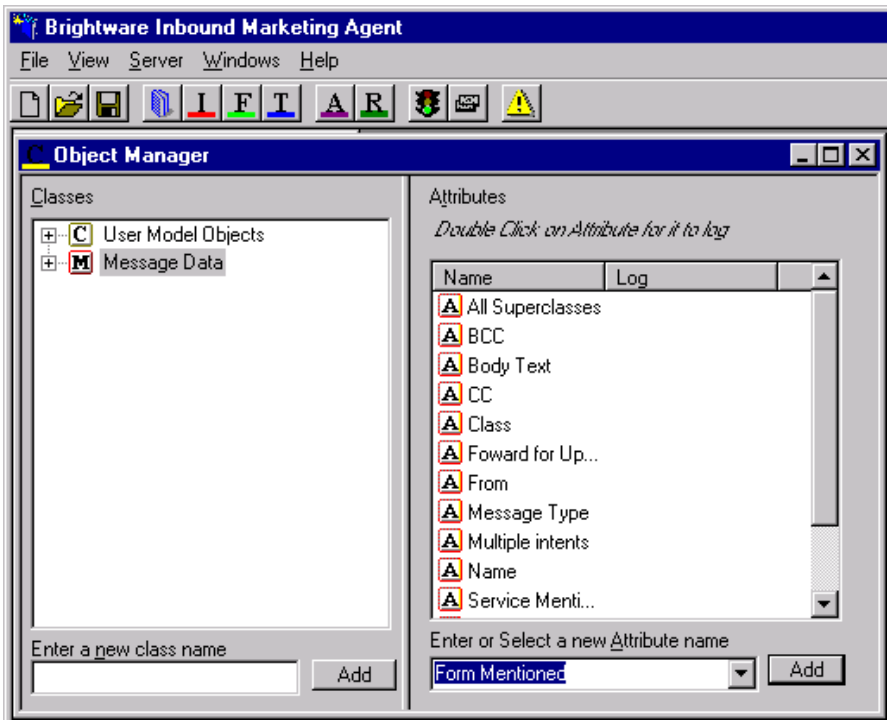
4.2.3.2 Formale und inhaltliche Repräsentation als DOM

Email Assistance verwendet zur internen Repräsentation der analysierten Anfragen ein an die Verwendung in objektorientierten Sprachen angelehntes „Document Object Model“ (DOM).

Standardmässig werden die *formalen* Bestandteile einer E-Mail (Sender, Empfänger, Subjekt und Textkörper) in einer Instanz des „Message Data Objects“ gekapselt und sind damit für anschliessende Aktionen (Systemreaktionen) verfügbar. Durch Definition von spezifischen Regeln wird das Standard-Datenmodell um die entsprechenden *inhaltlichen* Attribute (z.B. bestimmte Phrasen, Muster oder Features) erweitert. Darauf werden dann die gewünschten *Aktions-Regeln* (siehe weiter unten) appliziert.

Externe Daten werden auf die gleiche Weise in benutzerdefinierten Datenmodellen repräsentiert und im Bedarfsfall daraus zurückgewonnen, wobei die zugeordneten Regeln eben externe (Datenbank-) Zugriffroutinen auslösen. Dies wird vor allem bei vollautomatischer Mail-Beantwortung angewendet, wo zur Generierung der Antwort ein vordefiniertes *Template* (vgl. 2.5.4) dynamisch mit den passenden externen Daten instanziiert wird.

Abbildung 19 : Erweiterung des "Message Data" Objekts (M) durch Zufügen neuer Attribute (A)



4.2.3.3 Sprachverarbeitung

Die Sprachverarbeitung von *Email Assistance* beinhaltet die folgenden NLP-Komponenten:

- *Identifikation von Text-Mustern mittels Regulärer Ausdrücken*, welche sich mittels einer intuitiven Meta-Syntax[46] vereinfacht generieren lassen.
- *Subsyntaktische Analyse*, d.h. Auflösung von Abkürzungen („Dr." → „Doctor"), Kontraktionen („won't" → „will not") und Possessivformen („Robert's" → „Robert")
- *Spell Checking*: Verwendung eines (nicht näher spezifizierten) kommerziellen Programms; statistische Auswertung der wahrscheinlichsten Form; Verwendung von Original- und korrigierter Form in der weiteren Verarbeitung.
- *Morphologie-Analyse*: Verwendung eines (nicht näher spezifizierten) kommerziellen Programms; Verwendung von Original- und lemmatisierter Form in der weiteren Verarbeitung unter Einbezug von Ausnahmen zur Vermeidung ungewollter Lemmatisierung (benutzerdefinierbares Literal-Lexikon).

Es fällt auf, dass - wie bei den zuvor besprochenen Ansätzen - keinerlei syntaktische und/ oder semantische Analyse im Sinne einer „traditionellen" Sprachverarbeitung mittels Parser oder wenigstens Chunker zum Einsatz kommt.

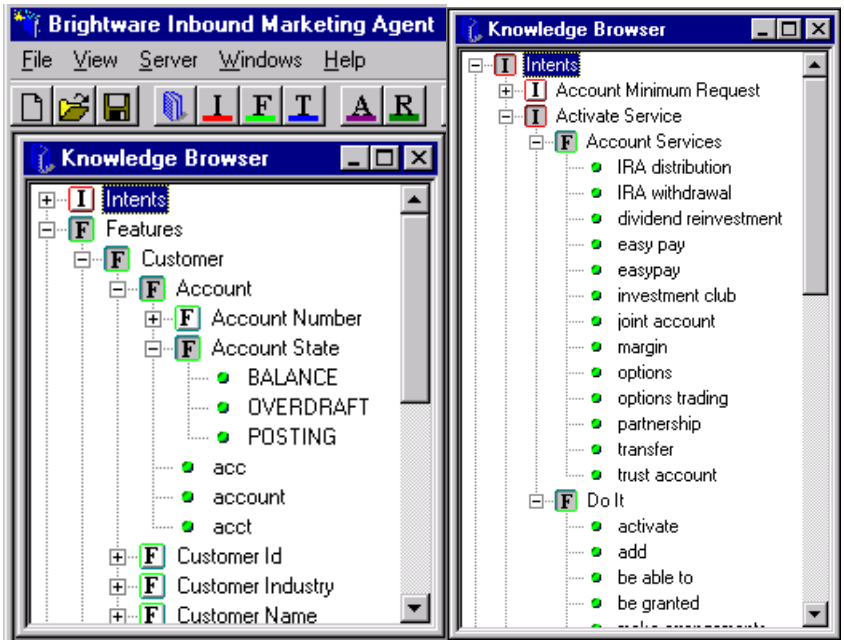
Die selbstauferlegten hohen Erwartungen an das Leistungspotential der Software müssen daher einmal mehr durch eine extensive Nutzung von „Low-Level" Technologien und deren geschickte Kombination erzielt werden. In diesem Fall werden die obigen Technologien *sequentiell auf den Eingabetext appliziert*, unter Beibehaltung der jeweiligen Originalform (vgl. „Containment of Ambiguity"-Prinzip, vgl. 3.1.3).

4.2.3.4 Regel-Lexikon und Ermittlung der intendierten Bedeutung

Eine auf obige Weise analysierte Mail-Anfrage wird im folgenden mit einem vom Administrator erstellten Regel-Lexikon („Wissensbank") auf Übereinstimmung getestet, wobei es die *domänenspezifisch bedeutungsvollen Wörter und Phrasen* eindeutig (d.h. deterministisch) zu identifizieren gilt (→ IE-Ansatz, vgl. 2.2.2).

Email Assistance bedient sich dazu einer Art „Semantics-by-Syntax"-Analyse: Durch kaskadierte und logisch verknüpfte *reguläre Text-Muster* und *literale Phrasen* des Mail-Textes entsteht inkrementell eine hierarchische Regelstruktur, in deren Wurzelknoten die *Intention(en)* der Anfrage mündet(-en):

Abbildung 20 : Text-Muster (•), Features (F) und Intents (I) eines benutzerdefinierten Regel-Lexikons

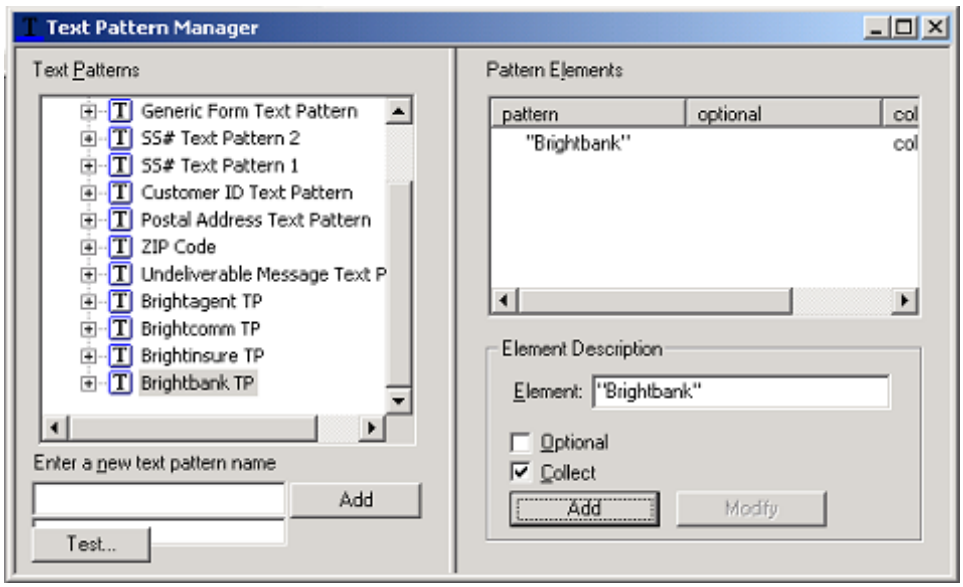


[46] der zugrundeliegenden Entwicklungsumgebung („Art*Enterprise")

Literale Phrasen und reguläre Text-Muster (die Punkte in obiger Abbildung) sind die *terminalen Bausteine* des Lexikons, welche anhand der vorgängigen linguistischen Analyse gefunden wurden (vgl. 4.2.3.3).

Phrasen und Text-Pattern sind Anfrage-unabhängig wiederverwertbar und werden in einer eigenen Benutzeransicht domänenspezifisch erstellt und in die interne Repräsentationssprache aus eigenem Hause („Art*Enterprise“) übersetzt.

Abbildung 21 : Definition regulärer Text-Muster (T)



Features sind Gruppen von Terminalen ähnlicher Bedeutung („F“ in Abbildung 20)*:* Jedes gefundene Terminal wird *deterministisch* genau einem Feature, sozusagen einem „Bedeutungsbaustein“, zugeordnet. Diese Zuweisung erfolgt auf der Basis des domänenspezifisch vordefinierten Regel-Lexikons (vgl. Abbildung 20).

Die einzelnen Elemente eines Features sind dabei sinnigerweise als *logische Disjunktion* verknüpft, da eine Feature ja eben alle *möglichen* Phrasen und Muster einer bestimmten Bedeutungseinheit enthält. Features können auch verschachtelt werden, was eine *generelle als auch spezifische Bedeutungszuweisung* ermöglicht (spezifische als Bestandteil allgemeinerer Features).

Intents sind Gruppen von Features, welche die Gesamt-Intention der Anfrage beinhalten („I“)*:* Im Gegensatz zur deterministischen Zuordnung von Terminalen können Features als Bedeutungsbaustein in mehreren Intents auftauchen. Das macht intuitiv durchaus Sinn, da verschiedene Teil-Bedeutungen in ihrer Kombination zu verschiedenen Gesamt-Intentionen führen können. Features sind innerhalb eines Intents nicht logisch (kausal) verknüpft, sondern *stochastisch*, d.h. die Erfüllung einzelner Features führt zur Wahrscheinlichkeit der Gesamt-Bedeutung.

4.2.3.5 Stochastische Bedeutungszuweisung

Eingehende Mail-Texte durchlaufen also kaskadierte Verarbeitungsstufen, um schliesslich einer (oder mehrerer) Intention(en) mit entsprechender Wahrscheinlichkeit zugeordnet zu werden Dies ist insofern bemerkenswert, als dass hier aus Analyse-Ergebnissen auf linguistisch tiefem Level (sub-syntaktisch) schliesslich ganze Bedeutungen (→ Semantik) zu ermitteln versucht werden:

Wird ein Terminal in der Anfrage gefunden, so wird es als „benutzt“ markiert und das enthaltende Feature ermittelt, welches seinerseits als „gefunden“ etikettiert wird. Ist dieses Feature Bestandteil weiterer Features, so werden alle darüber liegenden Eltern ebenfalls als „gefunden“ bewertet, was es ermöglicht, sehr spezifische Phrasen als Teil allgemeinerer Bedeutungen zu benutzen.

Die *Wahrscheinlichkeit der Gesamtbedeutung* wird wie schon erwähnt aus der Anzahl *zutreffender Teil-Intentionen* (den Features) ermittelt. Dabei entscheiden zwei (einstellbare) Schwellwerte gegen oben und unten über „Sicherheit“ und „Unsicherheit“ der Intention, dazwischenliegende Intents erhalten den Status „vorgeschlagen“.

Diese stochastische Einstufung gefundener Bedeutungen dient massgeblich für die weitere Verarbeitung der Anfrage: „Unsichere“ Intentionen werden ohne Umweg an die manuelle Bearbeitungsinstanz geschickt, während darüberliegenden Resultaten ein oder mehrere Antwortvorschläge zugewiesen werden. Als mit Sicherheit zutreffend taxierte Bedeutungen können auf Wunsch vollautomatisch beantwortet werden, sofern pro Anfrage genau eine Intention mit diesem Status identifiziert wurde (vgl. „Confidence level“ bei eGain Mail, 4.2.2.1).

Bei mehreren „sicheren“ Intents wird durch einen *Diskriminierungsfaktor* entschieden, ob ein bestimmter Intent deutlich zutreffender ist als die anderen (d.h. genügend Diskriminierungskraft besitzt). Ist dies der Fall, wird dieser weiterverwendet, andernfalls wird die Anfrage als „*Multiple Intents*“ gekennzeichnet und mit Antwort-Vorschlägen zu den einzelnen Bedeutungen zum manuellen Bearbeitung weitergeleitet.

4.2.3.6 System-Aktionen und Matching-Regeln

Die Entscheidung darüber, was mit einer in bestimmten, genügend wahrscheinlichen Intents klassifizierten Mail-Anfrage angestellt werden soll (d.h. welche System-Reaktionen in welchen Fällen ausgelöst werden), wird in sog. *Aktions-Regeln* (*Action Rules*) deklariert. Email Assistance unterstützt dabei die folgende *Aktionen* (wovon jedem Intent *zumindest eine zugeordnet* sein sollte):

- *Automatische Beantwortung* an den ursprünglichen Sender: Der dynamisch generierte Antworttext richtet sich an die ermittelte Intention und kann Textpassagen aus Original-Text und/oder externen Datenquellen enthalten (eingebettete Variablen).
- *Weiterleitung* an einen bestimmten (meist firmeninternen) Empfänger: Der generierte Text enthält die Begründung der Weiterleitung statt einer Beantwortung; die Originalnachricht wird beigefügt.
- *Weiterleitung zur manuellen Bearbeitung*: Unsichere und multiple Intentionen sowie spezielle Anfragen (z.B. leere oder überlange Texte) werden – ggf. mit mehreren Antwortvorschlägen – dem menschlichen Bearbeiter zugesandt.
- *Ausführen externer Prozesse,* z.B. Datenbank-Zugriffe und beliebige API-Funktionen.

Aktions-Regeln assoziieren Intentionen mit diesen Handlungsanweisen. Diese logischen Verknüpfungsregeln erhalten die ermittelte Intention als Antezedens und die zugehörige Systemreaktion als Kosequens (hier in Meta-Syntax notiert: IF <intent> THEN <action>):

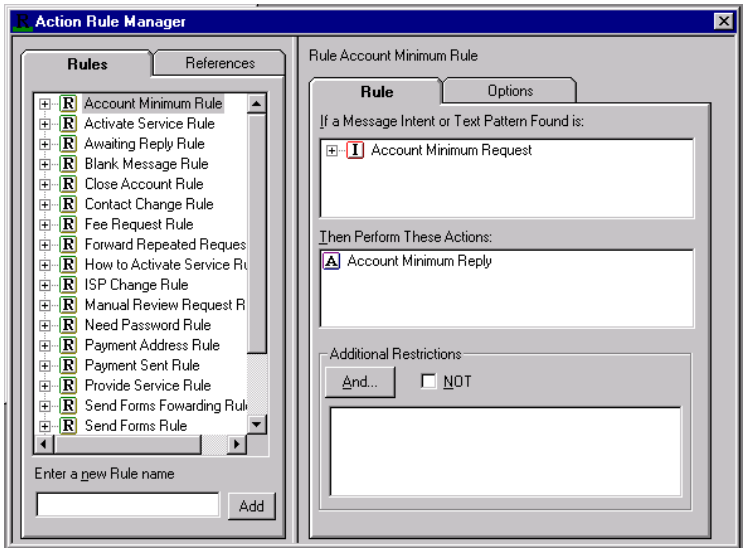
```
IF  Intent="Mortgage Information Request"
THEN  "General Info Response"

IF  Intent = "Loan Request Intent"  AND  Database Lookup = "Preferred Customer"
THEN  Send to High Priority Loan Queue

IF  Intent="Mortgage Rate Request"  AND   Proposed Signup Date < 60 days
THEN  Guaranteed Interest Rate Response
```

Was in der graphischen Repräsentation dann folgendermassen aussieht:

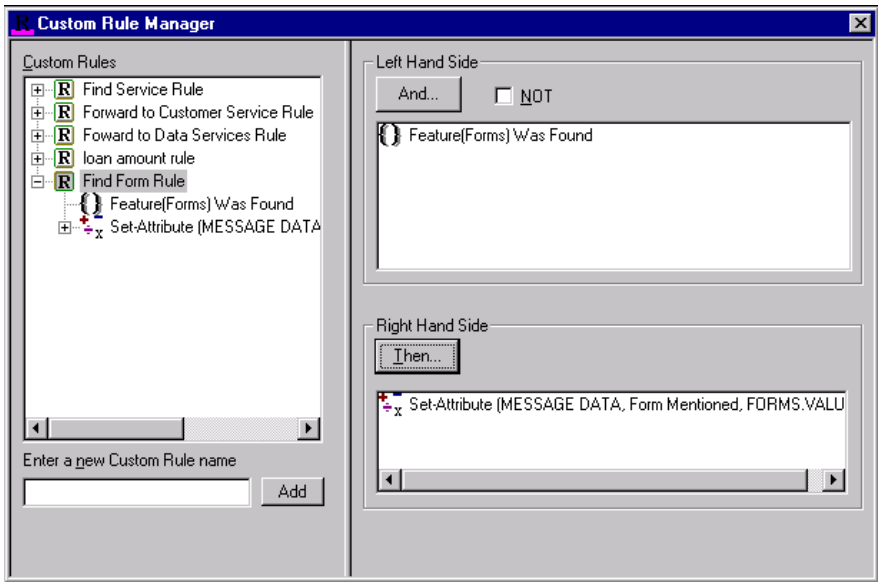
Abbildung 22 : Definition von Aktions-Regeln (R= Rule, I= Intent, A= Action)



Auf dieser obersten Matching-Ebene können zusätzliche Randbedingungen („*additional restrictions*“) formuliert werden. Dies können einerseits bestimmte Attribute aus der Original-Nachricht (DOM, vgl. 4.2.3.2) oder aus externen Datenquellen sein, anderseits aber auch zusätzliche Features oder Text-Pattern, was eine sehr differenzierte Suche ermöglicht.

Matching-Regeln können aber nicht nur auf der Ebene der Intents deklariert werden (als Aktions-Regeln), sondern auch schon auf der tieferen Ebene der Patterns und Features. Damit lassen sich einzelne Bestandteile des Anfragetextes in „klassischer Information Extraction Manier“ auf bestimmte Antwortmuster abbilden:

Abbildung 23 : Definition von Custom-Regeln



4.2.3.7 Evaluation

Firepond eService Performer *Email Assistance* macht als Gesamtlösung für manuelle und (halb-) automatische Mail-Verarbeitung einen professionellen und marktgerechten Eindruck. Der grosse Funktionsumfang erschliesst sich auch unerfahrenen Anwendern durch intuitive und anpassbare Benutzerführung im vertrauten Microsoft „Outlook“ Stil.

Das weitgehend transparente Analyse-Konzept überzeugt durch die Anlehnung an die objektorientierte Informationsrepräsentation (DOM) und die verständliche modulare Verarbeitungsstrategie auf Basis regulären Pattern-Matchings. Diese traditionelle „Low Level“-

Technologie wird durch Anreicherung mit dedizierter linguistischer Information, namentlich Morphologie und subsyntaktische Analyse (Auflösung von Abkürzungen etc.), in ihrer Mächtigkeit stark erweitert und bietet damit weitreichende Einsatzmöglichkeiten in der Sprachverarbeitung (und wäre sicherlich auch in anderen computerlinguistischen Bereichen interessant). Email Assistance verfolgt wie die anderen getesteten Lösungen die – scheinbar bewährte – Analysestrategie des „*easy-first-parsing*“ (vgl. 3.1.3).

Der „Semantics-by-Syntax“-Ansatz mutet aus computerlinguistischer Sicht insofern etwas überraschend an, als dass diese Idee der vielvertretenen Annahme widerspricht, semantische Information könne nur mittels komplexer linguistischer Analysen ermittelt werden. Allerdings wird hier auch keine eigentliche semantische Repräsentation einer Anfrage erzeugt, sondern lediglich versucht, die intendierte *Gesamt*-Bedeutung anhand der *statistischen Übereinstimmung vordefinierter Bedeutungsmuster* zu finden.

Einige Stichproben-Tests ergaben erstaunlich treffende Analyse-Ergebnisse – vorausgesetzt, das System ist ausreichend auf die jeweilige Domäne angepasst, d.h. es wurden zuvor die *zu erwartenden Textmuster und literale Phrasen* spezifiziert und kombiniert und die *Regeln für die entsprechenden System-Reaktionen* kreiert. Dieser anfängliche manuelle Trainingsaufwand muss bei der Auswertung einkalkuliert werden; schliesslich lassen sich mit entsprechend umfangreicher sprachlicher Abdeckung wie bei allen statistischen und regelbasierten Verfahren theoretisch beliebig hohe Trefferquoten erzielen. Nach Kenntnis von Firepond erreichen manche Unternehmen in sehr spezifischen, d.h. engen Domänen einen Durchsatz von 40 Prozent vollautomatischer Mail-Beantwortung, was allerdings nicht nachprüfbar ist.

4.3 Fazit: Errungenschaften praxisorientierter Systeme

Ein Vergleich der beschriebenen Systeme gestaltet sich aus Gründen der *fehlenden gemeinsamen Evaluationsbasis* als ziemlich schwierig (vgl. 1.3). Tendenzielle Gemeinsamkeiten und Unterschiede lassen sich dennoch erkennen.

Entgegen den Erwartungen sind die technologischen Differenzen zwischen den prototypischen, oft paradigmatisch anmutenden wissenschaftlichen Ansätzen und den durchwegs praxisorientierten kommerziellen Lösungen scheinbar gar nicht so gross. Die Erkenntnisse:

- *Mehrstufige. inkrementelle, modulare Analyse* hat sich bewährt
- *Reguläre Ausdrücke* machen einen Grossteil der Analyse aus
- *Stichwortsuche* wird als Retrieval-Grundlage oder Fall-Back-Strategie verwendet
- *Morphologische Normalisierung* spielt eine grosse Rolle
- *Seichte, partielle Syntaxanalyse* ersetzt ein vollständiges, unsicheres Parsing
- *Seichte Semantik*: Synonymklassen, Named Entities, semantische „Indikatoren“; meist keine tiefe semantische Analyse, keine komplexe Referenzauflösung
- *Partielle, dezentrale Disambiguierung* auf mehreren Analysestufen
- *Heuristik und Stochastik* sind in der Praxis zentral; die Definition domänenspezifischer Lexika, Regelsets, Gewichtungen und Sicherheitsfaktoren ist wichtige Vorarbeit
- *Dynamische, partielle Antwortgenerierung oder -extraktion* mittels Antwort-Fragmenten, -Passagen oder -Templates, angereichert mit spezifischen (externen) Daten
- *Praxisorientierte Zusatzfunktionen* wie etwa unterschiedliche System-Reaktionen und eine benutzerfreundliche Bedienung bringen nicht zu unterschätzenden Mehrwert

Die Kombination obiger Erkenntnisse verspricht also gute Resultate – wenngleich nur wenige konkrete Testresultate zu den einzelnen Systemen vorliegen und deren tatsächliche Praxistauglichkeit bestätigen. Es ist daher zu klären,

- ob diese heuristisch motivierten Strategien für eine weitreichende Abdeckung *unterschiedlicher Fragentypen* ausreichen, beziehungsweise
- was für eine möglichst automatisierte Beantwortung *darüber hinaus erforderlich* wäre, und
- was der aktuelle Stand der Technik und des computerlinguistischen Wissens dazu beitragen kann.

Diesen Fragen soll in Hinblick auf die Entwicklung und Verbesserung heuristischer *Suchstrategien für die E-Mail Beantwortung* im folgenden Kapitel nachgegangen werden.

5 Strategien für die automatische E-Mail Beantwortung

Nach der Erarbeitung der theoretischen Grundlagen von Textretrieval-Systemen und deren Sprachverarbeitungsverfahren (Kapitel 2 resp. 3) und der Analyse einiger wissenschaftlicher und kommerzieller Anwendungsbeispiele (Kapitel 4) soll im folgenden versucht werden, die aus diesen Betrachtungen gewonnen Erkenntnisse zur Entwicklung einiger Lösungsansätze für Systeme der automatischen E-Mail Beantwortung umzusetzen.

Als Ansatzpunkt dient die Evaluation zweier universitärer Systeme, die noch Schwierigkeiten bei der Beantwortung bestimmter Fragen zeigen, wobei deren Problemstellen lokalisiert und daraus Verbesserungsvorschläge eruiert werden. Dabei interessieren in erster Linie *praxis-orientierte lexikalische, syntaktische und seicht-semantische Sprachverarbeitungs-Strategien*.

5.1 Analyse

Die Verarbeitung von E-Mail-Anfragen kann den Problembereichen *Antwortextraktion* (AE) und *Fragebeantwortung* (QA) zugeordnet werden. Viele der in einem E-Mail-Text zu erwartenden Anfragetypen sind – auf Satzebene betrachtet – vergleichbar mit den möglichen Fragen an ein Antwortextraktionssystem, wobei der E-Mail-Text als Benutzeranfrage und eine bestimmte Dokumentensammlung als Menge potentieller Antworten verstanden wird[47].

5.1.1 Ausgangslage: Warum werden bestimmte Fragen so schlecht beantwortet?

Wissenschaftliche Ansätze wie die an der Uni Zürich entwickelten Systeme EXTRANS und UIS / LUIS verwenden unterschiedliche komplexe linguistische Analyseverfahren, funktionieren aber dennoch schlecht hinsichtlich der Beantwortung bestimmter Fragetypen, z.B. instrumentaler („Wie...“) oder kausaler („Warum...“), aber teilweise auch vermeintlich „trivialer“ Anfragen, wie die folgende Übersicht aus deren System-Evaluation zeigt (vgl. [Colombo/Nespeca 01]):

Tabelle 2: Konzeptuelle Fragetypologie und Test-Ergebnisse

Fragetyp[48]	**Beispiel**	**Reaktion / Trefferquote**[49]		**Erwünschte Reaktion**
		EXTRANS	**LUIS**	
Causal Antecedent	Warum ist die letzte CL-Vorlesung ausgefallen?	k.A.[50]	1 Antwort 0 relevante	Begründung
Goal Orientation	Wozu / Warum / Zu welchem Zweck wird der grüne Seminarschein benötigt?	k.A.	53 Antw. 2 relevante	Erklärung
Enablement	Welche Bescheinigungen werden zur Anmeldung zur Lizentiatsprüfung benötigt?	k.A.	1 Antwort 0 relevante	Aufzählung
Causal Consequent	Was passiert, wenn alle CL-Dozenten Grippe haben?	k.A.	2 Antworten 0 relevante	Voraussage/ Schlussfolge
Verification	Wird die Akzessprüfung zur Seminarteilnahme benötigt?	2 Antworten	3 Antworten 0 relevante	Ja / Nein, evt. Erklärung
Disjunctive	Müssen die Übungen in ein oder (in) zwei Wochen abgegeben werden?	k.A.	2 Antworten 0 relevante	Entscheidung
Instrumental / Procedural	Wie funktioniert EXTRANS?	16 Antw.	93 Antw. 4 relevante	Beschreibung
Concept Completion	Wann findet die nächste Akzessprüfung statt? Wo finde ich das Büro des Dekans? Wer wählt den Dekan?	-	k.A.	Konkreter Term (Datum, Ort, Person etc.)
Expectational	Warum macht Prof. XYZ so langweilige Vorlesungen?	k.A.	53 Antw. 2 relevante	Argumentation, Rechtfertigung
Judgemental	Welche Vorlesungen sollte ich zur Vertiefung meiner linguistischen Kenntnisse besuchen? Wie beurteilen Sie die missliche Lage der Publizistikstudenten?	17 Antw.	5 Antworten 3 relevante	Erklärung, Begründung, Argumentation

[47] wobei ein E-Mail Text im Gegensatz zur Anfrage bei manchen AE- und QA-Systemen aus mehreren Sätzen bestehen kann; mehr dazu in 5.1.2

[48] meint in etwa „Intention der Frage“; Typologie nach [Lehnert 81]

[49] Die System-Reaktionen wurden mit domänenspezifischen Fragen des jeweiligen Anfragetyps getestet und beziehen sich daher nicht auf die hier verwendeten Beispiel-Fragen, sondern eben auf deren Anfragetyp.

[50] Stand April 2001; spätere Verbesserung wurden nicht berücksichtigt.

k.A. = keine Antwort bei direkter Suche, Synonym/Hyponym-Suche und Approximations-Suche (letztes nur EXTRANS)

Quantification	Wie viele Studenten werden für die CL zugelassen? Wie viel kostet die Einschreibung?	k.A.	5 Antworten 2 relevante	Konkreter Term (Zahl oder Mass)
Feature Specification	Welche Analyseverfahren verwendet EXTRANS?	k.A.	21 Antw. 0 relevante	Konkreter Term/ Aufzählung
Request / Command	Gibt es / Wo finde ich detaillierte Informationen zum Studienangebot? Bitte senden Sie mir detaillierte Informationen zum Studienangebot. Exmatrikulieren Sie mich per sofort!	k.A.	1 Antwort 0 relevante	→ weiterleiten
Insult	Ihre Uni ist die lahmste, die ich kenne!	-	-	→ ignorieren

Es stellt sich die Frage, welche Probleme der tiefen Trefferquote dieser an sich recht komplexen NLP-Systeme zugrunde liegen. Vermutlich sind die heutigen Analyseverfahren für ein vollständiges Textverstehen immer noch unzulänglich. Die beiden AE-Systeme der Universität Zürich unterliegen möglicherweise aber noch einigen *grundsätzlichen* Problemen:

1. Zu wenig Technik bzw. linguistisches Wissen
2. schlecht genutzte oder schlecht nutzbare Technik trotz linguistischen Wissens
3. zu wenig oder irrelevante Korpus-Dokumente
4. keine Verwendung spezifischer Strategien der Fragebeantwortung

ad 1: LUIS erhält von Partnerkomponenten[51] weit mehr linguistische Information, als es tatsächlich nutzt. Insbesondere fehlt jegliche Behandlung *syntaktischer* Information (es kommt lediglich ein Wortdistanz-Mass zur Anwendung, vgl. [Arnold et al. 01]).

ad 2: EXTRANS versucht sich quasi auf dem „computerlinguistischen Königsweg" (semantische Indexierung und logische Inferenz); die Verarbeitung über viele heterogene Analyse-Stufen mit je eigenen Mängeln führt jedoch zu einer kontinuierlichen *Fehleraufsummierung*. Insbesondere die Übersetzung in eine *semantische Repräsentation* scheint aufgrund verschiedener, zum Teil unauflösbarer *Ambiguitäten* ein ernsthaftes Problem zu sein: Ein Satz eines Dokuments kann unauflösbare Mehrdeutigkeiten enthalten, und auch die Benutzeranfrage an sich kann ambig sein (vgl. [Mollá et al. 00a] und [Mollá et al. 00b]).

Da die eigentliche Suche durch logische Schlussfolgerungen über die semantische Repräsentation von Anfrage und Korpus geschieht, wirken sich derartige linguistische Mehrdeutigkeiten wie auch alle Fehler der einzelnen Verarbeitungsstufen unmittelbar auf das Suchergebnis aus.

ad 3: Die äusserst wenigen Fundstellen potentieller Antwortpassagen auch bei stark gelockerten Suchkriterien (v.a. bei EXTRANS) lassen vermuten, dass das System die Antwort gar nicht finden *kann*, da keine entsprechenden Passagen in der Dokumentenkollektion vorhanden sind (oder diese nicht indexiert wurden).

ad 4: Antwortextraktion ist keine Fragebeantwortung. Sowohl LUIS als auch EXTRANS suchen Antwortpassagen nach bestimmten strukturellen und inhaltlichen Kriterien, jedoch mehrheitlich ohne Einsatz gezielter, heuristisch motivierter Suchstrategien wie etwa die Behandlung von verschiedenen *Fragetypen*.

Der Einsatz *dezidierter QA-Strategien* erscheint – neben der Optimierung des Technikeinsatzes (durch Verwendung und Verbesserung *bewährter* AE-Strategien) und der Zurverfügungstellung umfassender, domänenspezifisch relevanter Dokumentkollektionen – jedoch gerade hinsichtlich der Adaption auf die E-Mail-Beantwortung besonders erfolgversprechend und soll im folgenden daher etwas genauer betrachtet werden.

[51] z.B. Prepocessor, Gertwol, Lextagger, vgl. [Arnold et al. 01]

5.1.2 Ansatz: Erkennung und Behandlung unterschiedlicher Fragetypen

Eine korrekte Erkennung des *Anfragetyps* wäre vermutlich hilfreich zur Entscheidung über die adäquate Verarbeitungsstrategie und die zu findende Information. Dazu beitragen könnten einerseits die *Fragepronomina* und anderseits d*ie topologische und funktional-syntaktische Struktur* (z.B. Verb-Subjekt-Objekt Relation und Dependenzen im allgemeinen).

Ein Grundgedanke vieler QA-Systeme (auch in anderen Anwendungsdomänen) ist denn auch die Erkennung von Frage- und entsprechenden Antworttypen durch *Abbildung bestimmter sprachlicher Merkmale auf eine vordefinierte Menge semantischer Kategorien*[52] (vgl. [Hovy 00], [Moldovan], [Wu 00]).

Verschiedene Ansätze setzen sich voneinander vor allem durch die unterschiedliche Nutzung und Interpretation linguistischer Informationen ab: Die meisten Systeme identifizieren verschiedene Fragetypen anhand des *einleitenden Interrogativpronomens* (plus allfällige Anhängsel, wie in „Wie hoch...“) sowie allenfalls mittels bestimmter lexikalischer Elemente (Wörter) oder einfacher subsyntaktischer Muster für bestimmte Named Entities ([ebenda]).

Allerdings deckt eine (fast) *ausschliessliche* Fragepronomen-Behandlung noch längst nicht alle denkbaren Fälle ab, denn:

1. Kaum irgendein Fragepronomen ist absolut diskriminierend, d.h. entscheidet eindeutig über die Zugehörigkeit zu einem bestimmten Fragetyp:

   „Was ist ein Testat?“
   „Was ist mit Professor Meier los?“

2. Viele (direkte) Fragen kommen ohne W-Fragepronomen aus (z.B. Verb oder Präposition am Satzanfang)

   „Müssen die Übungen in ein oder zwei Wochen abgegeben werden?“
   „In welchem Semester sollte ich mich für die Nebenfächer entscheiden?“

3. Anfragen können auch über indirekte Fragen formuliert werden:

   „Ich würde gerne wissen, ob/ was / inwiefern ...“
   „Ich würde gerne mehr über ... erfahren.“

4. Eine E-Mail-Anfrage muss weder direkte noch indirekte Fragen enthalten (bzw. nur intentionale); Beispiele sind Problembeschreibung oder Anweisungen:

   „Seit der Installation des Internet-Explorers läuft mein Modem nicht mehr.“
   „Bitte lösen Sie unverzüglich mein Konto auf!“

5. alle obigen Beispiele sind Ein-Satz-Anfragen; die semantische Information eines E-Mail-Textes kann aber auch auf mehrere Sätze verteilt sein (→ textuelle Referenzen) [53].

Dies führt zur Erkenntnis, dass für die Identifikation und Beantwortung verschiedener semantischer Fragetypen[54] noch *zusätzliche* linguistische Merkmale wie etwa syntaktisches und/oder semantisches Sprachwissen erforderlich sind, etwa:

(Q) „**Warum** ist die letzte Vorlesung ausgefallen?“
(A) „Ausfall der Vorlesung am 3. Dezember **wegen Krankheit**.“

Bestimmte sprachliche Elemente dienen also als *Indikatoren für die erfrage Information.*

Das Prinzip der Abbildung von Fragepronomen auf lexikalische und subsyntaktische Muster kann daher erweitert werden auf die *Abbildung bestimmter Frage-Indikatoren auf bestimmte Antwort-Indikatoren*:

[52] Es stellt sich natürlich die Frage, welche Anfragetypen überhaupt und wie häufig in E-Mails vorkommen (→Heuristik). Dies wurde hier jedoch nicht untersucht.
[53] Die folgenden Überlegungen beziehen sich einfachheitshalber vornehmlich auf Ein-Satz-Fragen, sind jedoch vielfach auch auf grössere Texte verallgemeinerbar. Textuelle Referenzen erfordern allerdings eine gewisse Diskurs-Analyse.
[54] verwirrenderweise auch „question type“, „question class“, „answer category“, „asking point“, „QA-Token“, „focus“, „target“ oder wiederum „Named Entity“ genannt, vgl. 7. Referenzen

Tabelle 3: Frage- und Antwort-Indikatoren für verschiedene semantische Fragetypen[55]

Frage- / Antwort-Typ	**Frage-Indikatoren**	**Antwort-Indikatoren**			
	Lexikalische Elemente	**Lexikalische Elemente**	**Wortkategorie** [56]	**Subsyntax / RegEx**	**Syntaktische Kategorie**
TIME | DATE	(Seit, ab, von, ...) Wann		CARD, ORD	Formate für Zeitangaben	PP
DURATION	(Von) Wann (bis...)		CARD, ORD	Formate für Zeitspannen	PP
REASON	Warum, weshalb, weswegen, wieso	darum. deshalb, deswegen, wegen, weil	APPR, KOUS, PAV		PP
?	Was	?	?		?
PERSON | ORG | SPEC	Welcher (NN)	Bezeichnung v. Personen, Organisationen, Berufen etc → Lexikon	NN, NE	Formate für Designatoren	NP
PERSON | ORG | ROLE	Wer (Von, zu , mit) wem		NE (, NN)		NP
PERSON | ORG | ROLE	Wessen		NE (, NN)		NP
?	Wie	?	?		?
QUANTITY | MONEY	Wie|viel(e) Wie teuer etc.	Mengendesignatoren, Wärungsangaben	CARD, ORD	Formate für Mengen etc.	NP
AGE	Wie alt etc.	Jahr(e), Alter etc.			
LENGTH	Wie lang etc.	Lang, Länge, Grösse etc-		Massformate	
LOCATION	Wo	In ... , Lexikon	NE, NN		PP (, NP)
	Wo|bei	(da) bei	APPR, KOUS, PAV		PP
	Wo|durch	(da) durch	APPR, KOUS, PAV		PP
(REASON)	Wo|für	(da) für	APPR, KOUS, PAV		PP
	Wo|gegen	(da) gegen	APPR, KOUS, PAV		PP
(LOCATION)	Wo|her	(da) her	APPR, KOUS, PAV		PP
(LOCATION)	Wo|hin	(da) hin	APPR, KOUS, PAV		PP
	Wo|mit	(da) mit	APPR, KOUS, PAV		PP
	Wo|ran	(dar) an	APPR, KOUS, PAV		PP
	Wo|runter	(dar) unter	APPR, KOUS, PAV		PP
(REASON)	Wo|zu	(da) zu	APPR, KOUS, PAV		PP

Der Begriff des semantischen *Fragetyps* wird demnach im Gegensatz zu ersterer Typologie (Tabelle 2 nach [Lehnert 81]) nicht über die *Intention* der Frage definiert, sondern über bestimmte *heuristisch motivierte Zusammengehörigkeiten bezüglich deren Beantwortung*; die Erkennung und Behandlung dieser Fragetypen auf Basis verschiedener Indikatoren ermöglicht (und erfordert) damit die Entwicklung gezielter *Suchstrategien* (siehe unten).

Für die Erarbeitung solcher Suchstrategien für die Anwendung der E-Mail-Beantwortung erscheinen dabei – teilweise im Gegensatz zu den für LUIS und EXTRANS erhobenen Anforderungen – folgende Charakteristika besonders relevant:

- *Präzision* und *Robustheit*[57] sind zentral (v.a. für eine vollautomatische Beantwortung).
- *Skalierbarkeit,* d.h. Anpassung an spezifische Domänen und evt. unterschiedliche Sprachen und Plattformen, ist ebenfalls wichtig.
- *Analysegeschwindigkeit* ist dagegen nicht zentral, da „offline“ analysiert werden kann[58].

[55] diese Auflistung ist längst nicht vollständig. Weitere Kategorien und Indikatoren finden sich (meist für die englische Sprache) bei [Ferret et al. 00], [Hovy et al. 00], [Lehnert 81], [Litkowski], [Moldovan et al.], [Prager et al. 00], [Srihari/Li 99], [Wu 00] sowie im Q&A Roadmap Paper von [Burger et al.]
[56] POS-Wortkategorien nach STTS
[57] also die Fähigkeit, auch syntaktisch nicht vollständig abgedeckte Konstruktionen analysieren zu können, vgl. [Wauschkun 96]
[58] Erkenntnisse zur Effizienz werden hier nicht diskutiert, finden sich aber bei [Metzinger 02]

Im folgenden werden verschiedene *lexikalische, syntaktische sowie seicht-semantische Strategien und Heuristiken der Fragebeantwortung* aufgezeigt, welche über die „klassische" Antwortextraktion von LUIS und EXTRANS hinausgehen und im Rahmen eines eigens durchgeführten Programmierprojekts auf die deutsche Sprache adaptiert und auch getestet wurden[59].

5.2 Strategien der Fragebeantwortung

Strategien zur *gezielten Wiederfindung bestimmter Informationen* werden im folgenden auch als *Suchstrategien*bezeichnet[60]:

5.2.1 Lexikalisch-orientierte Strategien

5.2.1.1 Entfernung inhaltsarmer Wörter und Konstrukte

Viele Verben und Adjektive, aber auch bestimmte Nomen haben nur *geringen Informationsgehalt*, führen jedoch zu einer höheren Anfrage-Spezifität.

Stoppwortlisten sind die gängige Methode zur Reduktion von Informationsarmut und „ungewollter" Spezifität, welche den Recall verringert und der Antwort-Präzision nicht zuträglich ist (vgl. [Arnold et al. 01]). Oft wäre es aber sinnvoller, potentielle Stoppwörter oder ganze Konstrukte *nur in bestimmten (syntaktischen) Kontexten herauszufiltern*, in denen sie keinen wesentlichen Informationsgehalt aufweisen.

Anhand einiger Beispiele wurde getestet, inwiefern die Entfernung bestimmter lexikalisch-syntaktischer Gefüge zu genaueren Resultaten führt. Die letztlich implementierten Muster sehen wie folgt aus:

```
Es   geben|existieren|dasein        →
Bsp: Es gibt / hat / ist da / existiert/-en

geben|existieren|dasein  es         →
Bsp.:Gibt es im Irchel eine Mensa?

Wie  heissen|lauten                 →
Bsp.:Wie lautet die Adresse des Instituts für Informatik?
```

Diese recht einfache Suchstrategie führte mehrheitlich zu Performance-Gewinnen. Allerdings hat sich die *Identifikation* wirklich nutzloser Information als eher schwierig herausgestellt, sodass selbst diese trivialen „Tilgungs-Regeln" teilweise allzu unspezifische Anfragen und damit unpräzise Antworten erzeugen. Daher wäre vielleicht eine dynamische Filterung in Dependenz zur Frage-Spezifität sinnvoll (vgl. 5.2.3).

5.2.1.2 Sukzessive Wortkategorie-Filterung

Bestimmte Wortkategorien, z.B. Nomen, sind wie oben erwähnt entscheidender für den Sucherfolg als andere, und sollten daher höher gewichtet werden (vgl. 5.2.3.2). Andere Begriffe und Konstrukte können dagegen mangels Informationsgehalt von vorn herein ignoriert und aus der Frage entfernt werden (vgl 5.2.1.1).

Wie sich in der Untersuchung herausgestellt hat, ist es in manchen Fällen jedoch zusätzlich sinnvoll, Wörter bestimmter *Kategorien* in der Frage sukzessive auszufiltern, um den Suchraum im Retrieval-Prozess zu erweitern (oder dadurch überhaupt Suchresultate zu erhalten, wenn zuvor keine Antworten gefunden wurden).

Eine *sukzessive Filterung* hat sich daher in der folgenden Abfolge bewährt:

[59] die durchgeführte Untersuchung zeigt allerdings gewisse methodische Schwierigkeiten, v.a. eine sehr kleine Stichprobe (Methodische Details siehe [Metzinger 02]).

[60] Der Begriff „Suchstrategie" suggeriert den Bezug zum Retrieval-Teil der Fragebeantwortung, umfasst aber auch entsprechende konzeptuelle und implementatorische Anpassungen des Indexierungsverfahrens

1. Modale Verben
2. Adjektive
3. Vollverben

Noch sinnvoller ist unter Umständen eine dynamische Gewichtung der einzelnen Terme (vgl. 5.2.3.2).

5.2.1.3 Zertrennung von Komposita

Ähnlich der Filterung bestimmter Wortkategorien kann der Suchraum – sukzessive oder standardmässig – auch durch Zertrennung von Wortkomposita in separate Suchterme erweitert werden[61].

Komposita-Splitting hat sich im Test als sehr komplexe Suchstrategie erwiesen, deren Optimierung genauere Studien voraussetzen würde:

Während „einfache" Komposita wie etwa „Vorlesungsverzeichnis" und „Prüfungsdaten" meist als „atomarer" Term betrachtet und unbearbeitet gematcht werden konnten, müssten komplexe Konstrukte wie „Socrates/Erasmus-Mobilitätsstelle" dagegen auf ähnliche, aber anders zusammengesetzte Terme abgebildet werden, also z.B. „Uni-Mobil-Stellen". Dies hat teilweise, leider aber nicht immer funktioniert.

Eine erfolgreiche Behandlung solcher Fälle erfordert daher vermutlich die Entwicklung „intelligenter", heuristischer Strategien des Zerteilens komplexer Komposita.

5.2.1.4 Lexikalische Frage- und Antwort-Indikatoren

Wie schon erwähnt, nutzen viele Systeme „Named Entities" (NE) zur Erkennung bestimmter *Bedeutungsträger* wie etwa Personennamen, geographische Ziele und Zeitangaben. Derartige Entitäten werden zur Beantwortung bestimmter Fragetypen herangezogen, wie z.B. „Wer", „Wo" und „Wann" (z.B. in [Wu 00], [Ferret 00]).

Diese Strategie der Fragebeantwortung erscheint besonders effektiv, da erfahrungsgemäss die Mehrheit aller Fragen irgendeine Form von Named Entity als Antwort verlangen (gemäss [Srihari/Li 99] sind es über 80%!). Allerdings führen NEs nur im Zusammenspiel mit anderen Suchkriterien, etwa einer gleichzeitigen Stichwortsuche, zu guten Ergebnissen ([ebenda]).

So vielversprechend der Einsatz von NEs zu sein scheint, so zweischneidig gestaltet sich deren korrekte *Identifikation* (welche zweifelsohne massgeblich für den Sucherfolg verantwortlich ist): Während bestimmte Entitäten wie Zeit- und Datumsangaben sowie Telefonnummern bereits mit grosser Treffsicherheit mittels *regulärer Ausdrücke* erkannt werden können, erfordern andere Fragen den Beizug grosser *Lexika*, um beispielsweise Personen- und Städtenamen der korrekten semantischen Kategorie zuzuordnen (vgl. [Srihari/Li 99]).

Es wird nun jedoch angenommen, dass bereits *einfache lexikalische Muster* als „semantische Indikatoren" auf bestimmte Fragepronomen abgebildet werden und damit *ohne grosses Wörterbuch* zur Verbesserung des Suchverfahrens beitragen können. Dieser eher unkonventionelle Ansatz wird in der Literatur nur sehr vereinzelt angesprochen (z.B. bei [Litkowski] oder [Srihari/Li 99]).

In der gemachten Untersuchung wurden Zeit- und Ortsangaben sowie finanzielle Ausdrücke getestet, also potentielle Antworten auf „Wann", „Wo", „Wie teuer" und ähnliches. Dazu kamen drei verschiedene Typen heuristischer Regeln zur Anwendung:

[61] diese sollten dann allerdings schwächer gewichtet werden, da sich durch das Zerteilen die Anzahl der Suchterme und damit die Spezifität der Frage erhöht und sich damit die zu erwartende lexikalische Übereinstimmung reduziert

```
1.   (Q,A)
     Geld|Währung|Kosten|Gebühr|Spesen|Preis|Gewinn|Entschädigung|Taxe|Steuer|staatliche_A
     bgabe|Rechnung |(Ver|Ab)rechnung|billig|(preis)günstig|teuer →      $1  <MONEY>


2.   (Q) Wann                  →   [NE="TIME|TIMESPAN|DATE|DATESPAN|TEMP"exp="_ANY"]

     (A) HRS:MIN:SEC           →   [NE="TIME"flex="$&"exp="$1;$2;$3"]

     (A) DAY_NR. MONTH YEAR    →   [NE="DATE"flex="$&"exp="_ANY;$1;$2;$3"]

     (A) jahr(e)|(sommer|winter|ausland|studien)(-)semester|(aufnahme|zwischen|vor|
     (ab)schluss|diplom|lizentiats)(-)  prüfung(en))|sobald)      →      $1 <TEMP>


3.   (Q) Wo                                                      →      _ANY     <LOC>

     (A) in|im|an|am|bei|beim|Adresse|Anschrift|befinden|finden  →      $1       <LOC>
```

Hier werden also einerseits bestimmte *Nomen, Verben und Adjektive als semantische Indikatoren* in Frage (Q) und Antwort-Korpus (A) gekennzeichnet, um im Suchprozess als zusätzliches *Vergleichs-Kriterium* zu fungieren[62].

ad 1: Finanzielle Ausdrücke werden nur mittels lexikalischer Ausdrücke und nicht durch bestimmte Fragepronomina gekennzeichnet, da die zu erwartenden Fragen höchstwahrscheinlich auch per se schon die entsprechenden Indikatoren enthalten (z.B. „Wie *teuer*" enthält sowieso die Kennzeichnung „<MONEY>").

Anderseits werden bestimmte *Fragepronomina* („Wann" und „Wo") direkt auf den entsprechenden Indikator abgebildet[63]:

ad 2: Das Fragewort „Wann" wird auf eine mittels regulärer Ausdrücke identifizierte temporale Named Entity[64] abgebildet und erhält daher kein zusätzliches semantisches Tag.

Grundsätzlich spielt es in diesem Ansatz keine Rolle, ob ein bestimmtes Fragepronomen auf eine NE oder einen semantischen Indikator abgebildet wird; massgebend ist die zugewiesene *semantische Kategorie*.

ad 3: „Wo" dagegen wird nicht durch eine Named-Entity ersetzt, weil hierzu wie oben erwähnt ein Lexikon mit entsprechenden lokalen und geographischen Ausdrücken erforderlich wäre. Hier sollen vielmehr mittels semantischer Kennzeichnung bestimmter lexikalischer Indikatoren (darunter auch Präpositionen) potentielle Antwortpassagen identifizieren werden.

Dieser hybride Ansatz geht also einen Mittelweg zwischen der Behandlung der beiden anderen Fragetypen, welche auch mit herkömmlichen QA-Theorien erschlagen würden (wie z.B. bei [Ferret 00]).

Im Gegensatz zur Verwendung grosser Named-Entity-Lexika propagiert diese Suchstrategie die *gleichwertige Nutzung unterschiedlicher Wortkategorien* als semantische Indikatoren, z.B. Nomen, Verben und besonders auch Präpositionen, wie z.B.:

```
in|in-d|an|an-d|bei|bei-d|Adresse|Anschrift|befinden|finden   → $1 <LOC>
```

Wie die Untersuchung gezeigt hat, kommt der *Gewichtung* solcher semantischer Indikatoren gegenüber anderen Suchstrategien eine zentrale Rolle zu (vgl. Tabelle 4). Insbesondere scheint dieses Suchkriterium nur im *Zusammenspiel mit lexikalischen und syntaktischen Übereinstimmungs-Faktoren* zu Verbesserungen zu führen, also z.B. als Eingrenzung und Präzisierung potentieller Suchergebnisse (sukzessive Suchraum-Eingrenzung, vgl. 5.2.4.1).

[62] die Variable $1 enthält den gefundenen Teilausdruck; das Lemma wird also gekennzeichnet und nicht etwa ersetzt
[63] deren Lemma wird durch den Universal-Matcher „_ANY" ersetzt und damit mit *irgendeinem Wort* im Korpus „unifiziert", vgl. Regeln 2 und 3
[64] die verwendeten „echten" NEs werden hier nicht im Detail beschrieben, siehe dazu [Metzinger 02]

Des weiteren sind die lexikalischen und syntaktischen Regeln für Frage- und Antwort-Indikatoren *domänen- und sprachformat-spezifisch* und massgeblich für den Sucherfolg verantwortlich. Deshalb wirken sich Mehrdeutigkeiten besonders gravierend aus:

```
am [ASVZ-Schalter] → <LOC>          vs.      am [Donnerstag] → <TEMP>
```

Potentiell mehrdeutige Indikatoren können allerdings mit zwei Strategien behandelt werden:

1. *Unterspezifizierung*: Zuordnung zu allen *möglichen* semantischen Klassen:

   ```
   am → <LOC|TEMP>                    (matcht Sätze mit <LOC> und/oder <TEMP>)
   ```

2. *Kontext-abhängige Zuweisung* mittels Regeln für das lexikalische und/oder syntaktische Umfeld:

   ```
   am [NN-TEMP(Donnerstag)] → <TEMP>  (matcht Sätze mit <TEMP>, nur im Kontext eines
                                       temporalen Nomen NN-TEMP)
   ```

Letztere Strategie erfordert allerdings wiederum den Beizug bestimmter Lexika mit den disambiguierenden Kontext-Wörtern wie zum Beispiel die Auflistung aller potentiell temporaler Nomen, weshalb hier die erste Lösung vorgezogen wird (wie z.B. auch bei [Srihari/Li 99]).

Grundsätzlich scheint das Prinzip der Kennzeichnung von Fragetypen und potentiellen Antworten durch lexikalische und subsyntaktische Muster sehr vielversprechend zu sein. Insbesondere lassen sich auf diese Weise auch gewisse „Sorgenkinder“ der Fragebeantwortung mittels Abbildung auf einfache Indikatormuster zumindest ansatzweise erschlagen, so z.B. kausale Fragen:

```
(Q) Warum|Weshalb|Weswegen|Wieso|Aus welchem Grund(e)   →   _ANY   <REASON>

(A) darum|deshalb|deswegen|wegen|weil|Grund(e)|         →   $1     <REASON>
```

Die Entwicklung solcher heuristischer Regeln bedingt sicherlich weiterführende Forschung[65]. Zur Abstraktion wäre zudem ein *Synonymklassen-Lexikon* erforderlich.

5.2.1.5 Anfrage-Erweiterung durch Synonyme und Hypernyme

LUIS und EXTRANS erweitern die Anfrage bei erfolgloser Suche durch gleichbedeutende und zum Teil auch übergeordnete Begriffe zu den Fragetermen. Die Suchpräzision dieser Strategie variiert bei erhöhter Ausbeute jedoch stark, weil *lexikalische Mehrdeutigkeiten* zu falscher Synonymisierung führen; echte Verbesserungen zeigten sich selten (vgl. [Colombo/Nespeca 01] bzw. Tabelle 3).

Auch in der eigenen Untersuchung hat sich die Realisierung einer erfolgreichen Anfrage-Erweiterung mittels Synonymen und Hypernymen als schwierig erwiesen: Moderate Recall-Gewinne wurden meist mit markanten Präzisionsverlusten erkauft (v.a. wegen besagter lexikalischer Ambiguität).

Dabei hat sich allerdings gezeigt, dass die Beachtung *syntaktischer Abhängigkeiten* diese durch lexikalische Mehrdeutigkeiten entstehenden Ungenauigkeiten zum Teil aufzuheben vermag (vgl. 5.2.2.1).

Der Sucherfolg bei Synonymerweiterung ist zudem abhängig von der *Wortkategorie*: Nomen sind lexikalisch oftmals weniger ambig als Verben und ermöglichen daher eine präzisere Synonymsuche. Dasselbe gilt – wenn auch weniger offensichtlich – für Adjektive im Vergleich zu Verben. Die Synonymisierung von *Komposita* ist aufgrund derer höherer lexikalischer Spezifität und damit geringerer Ambiguität im allgemeinen erfolgversprechender (vgl. [Metzinger 02]).

Grundsätzlich scheint der Präzisionsverlust massgeblich von der *Spezifität* bzw. der *Granularität* des verwendeten Synonym-Lexikons abhängig zu sein; je unspezifischer und breiter gefächert die Synonymklassen, desto ungenauer das Resultat. Sinnvoll wäre daher eine

[65] ein reales System, welches diesen Ansatz punktuell einbezieht, wird bei [Litkowski] beschrieben; bei [Srihari/Li 99] wurden kausale Fragen mit dem Ergebnis getestet, dass die Verwendung lexikalischer Indikatoren zumindest den Suchraum auf eine überschaubare Menge potentieller Antworten einschränken kann.

konzeptionelle Erweiterung auf eine Ontologie *semantischer Synonymklassen*, gegebenenfalls sogar über die Wortkategorie hinaus (vgl. [Srihari/Li 99]):

```
„Erfordernisse|verlangen|brauchen|notwendig“ → MUST
```

sowie eine gegenüber dem Original-Lexem *reduzierte Gewichtung der Synonyme und Hypernyme*.

5.2.1.6 Such-Restriktionen

In einigen neueren Ansätzen werden bestimmte Wörter, Wortkategorien oder Entitäten zur (sukzessiven) *Eingrenzung des Suchraums* einer Anfrage herangezogen.

[Litkowski] fordert beispielsweise, dass eine in der Frage enthaltene *Jahreszahl* entweder wieder auf eine Jahreszahl, oder zumindest auf den semantischen Antworttyp „temporal“ abgebildet wird. Diese Einschränkung kommt zudem nur in bestimmten Kontexten zur Anwendung, z.B. in einer „Wo“-Frage.

Hier wird also deterministisch über die Eignung potentieller Antworten entschieden; Text-Passagen ohne die erforderlichen Terme scheiden von vornherein aus (man spricht deswegen manchmal auch von „constraints“ (vgl. [Metzinger 02]), im Gegensatz zur dynamischen Gewichtung (vgl. 5.2.3)).

Im Test bestätigte sich diese Suchstrategie für gewisse Entitäten, welche den *semantischen Kontext restringieren*, durch erhöhte Präzisionswerte – teilweise allerdings zu Lasten der Ausbeute, da einige Passagen durch diese strenge Bewertungsmassnahme „zu Unrecht“ ausgefiltert wurden. Auch dieser Ansatz erfordert also genauere Studien.

Die zuvor beschriebenen vornehmlich *lexikalisch-orientierten* Retrieval-Strategien können auch mit nachgeschalteten *syntaktischen* Methoden kombiniert werden:

5.2.2 Syntaktische Beziehungen als Frage- und Antwort-Indikatoren

Wenn keine semantische Verarbeitung vorgenommen wird, stellt sich die Frage, inwiefern eine (partielle oder vollständige) Syntaxanalyse zur Erkennung von Anfragetyp und potentieller Antwort beitragen kann – beziehungsweise, ob syntaktisches Wissen überhaupt zur Lösung der beschriebenen Probleme genügt. Die gemachten Untersuchungen ergaben deutliche Gewinne durch Einbezug syntaktischer Kriterien:

5.2.2.1 Wort-Distanz vs. Phrasenzugehörigkeit

Viele vornehmlich lexikalisch orientierte Systeme (z.B. Suchmaschinen, LUIS, vgl. [Arnold et al. 01]) gewichten gefundene Terme nach deren *Wortdistanz* im jeweiligen Dokument (z.T. in Vergleich zu derjenigen in der Frage). Manchmal kommen auch einfache *Regeln* wie *„finde möglichst viele Suchwörter auf der gleichen Zeile / im gleichen Abschnitt“* zur Anwendung ([Srihari/Li 99]).

Es ist einleuchtend, dass derartige Kriterien allenfalls ein „Anzeichen“, aber keinesfalls ein echtes Mass für Beziehungen zwischen und Zusammengehörigkeiten von Wörtern sein können. Dies zeigt sich besonders deutlich bei diskontinuierlichen Phrasen mit grosser Wortdistanz:

```
(Q) Warum war die Kanzlei während der Semesterferien geschlossen ?
(Q) Was muss ist für die Beantragung eines Urlaubsemesters mitbringen ?
(Q) Zähle mir die Namen aller Dozenten in Computerlinguistik auf.
```

Es hat sich gezeigt, dass die Berücksichtigung syntaktischer Relationen im Sinne von *Phrasenzugehörigkeiten* zu präziseren Sucherfolgen führt als ein reines Wort-Distanz-Mass (Vergleich: LUIS). Das bedeutet: zwei (oder mehr) Wörter in einer Phrase der Frage finden sich mit recht hoher Wahrscheinlichkeit auch in der gleichen Phrase der Antwort wieder (dies impliziert natürlich auch die Zugehörigkeit von Wörtern zu Sätzen und ganzen Textpassagen).

Allerdings hat diese Strategie in einigen Fällen zu *syntaktischen Mehrdeutigkeiten* geführt, welche sich im Sucherfolg negativ niederschlagen können; vornehmlich sind dies Anschlussambiguitäten von Präpositionalphrasen:

```
(Q) Kann ich Adressen von Studierenden erfahren?
(A) Aus Datenschutzgründen kann die Universität Adressen von Studierenden nicht weitergeben.
```

Derartige Fälle sprechen für die gleichzeitige Beachtung von Phrasenzugehörigkeit *und* Wort-Distanz. Hierzu, als auch zur spezifischen Eignung des Einbezugs von diskontinuierlichen Phrasen, liegen derzeit allerdings noch kaum Ergebnisse vor.

Des weiteren können unterschiedliche *syntaktische Variationen* desselben Inhalts (z.B. Passivsätze) mit dieser Strategie natürlich noch nicht abgedeckt werden – hierzu wäre klassischerweise eine Umwandlung in eine funktionale oder semantische Repräsentationsform erforderlich (z.B. Logik). Wie ein ähnliches Resultat aber auch mit rein lexikalisch-syntaktischen Mitteln erzielt werden kann, wird in den nächsten zwei Kapiteln beschrieben:

5.2.2.2 Syntaktische Kategorien als funktionale Indikatoren

Die Übereinstimmung der syntaktischen Kategorie (NP, PP etc.) zwischen Frage und Antwort scheint ein wichtiges Suchkriterium zu sein:

Wenn keine Analyse der funktionalen Satzstruktur gemacht wird, also keine tiefensyntaktischen Funktionen wie Subjekt und Objekt identifiziert werden, so können doch zumindest *potentielle Bedeutungsträger* aus Frage und potentieller Antwort in Übereinstimmung gebracht werden, etwa *Handlungsträger* (nämlich NPs) oder *Modifikatoren* (zum Beispiel PPs); eine NP ist mit grosser Wahrscheinlichkeit Subjekt oder Objekt, eine PP kann etwa Prä- oder Postmodifikator sein.

Der Phrasentyp scheint damit ein relevanter *Indikator für tiefensyntaktische bzw. funktionale Beziehungen* zu sein.

Gleichermassen wäre auch eine Abbildung gewisser Fragepronomen bzw. -typen auf bestimmte syntaktische Kategorien denkbar (vgl. Tabelle 3):

```
(Q) Warum ist die letzte Vorlesung ausgefallen?
(A) Ausfall der Vorlesung am 3. Dezember wegen Krankheit.
```

Regeln für derartige Zuweisungen werden in der Literatur bislang kaum erwähnt und wurden auch in der gemachten Untersuchung nicht weiterfolgt, erscheinen aber als ein fruchtbarer Ansatzpunkt für innovative Erweiterungen.

Das Prinzip der Suche nach funktionalen Bedeutungsträgern kann aber noch erweitert werden:

5.2.2.3 Lexikalisch-syntaktische Frage- und Antwortmuster und Relationen

Einige neuere Ansätze (z.B. [Hovy 00], [Wu 00]) identifizieren verschiedene Fragetypen nicht nur anhand des einleitenden Interrogativpronomens, sondern versuchen lexikalisch-syntaktische Fragemuster („Question Templates“, vgl. [Hovy 00]) entsprechenden Antwortmustern („Answer Templates“) zuzuweisen, zum Beispiel in folgender Art:

```
(Q) Was ist [NP:Testat]
(A) [NP:Testat] nennt man ...
(A) [NP:Testat], eine Form der Bescheinung ...
(A) ..., auch als [NP:Testat] bezeichnet.
```

Derartige Abhängigkeiten können zu syntaktisch-semantischen *Relationen abstrahiert* und diese dann miteinander gematcht werden. Verschiedene syntaktische Variationen, etwa Aktiv- und Passiv-Konstruktionen, werden somit derselben „semantischen“ Repräsentation zugeordnet, ohne dass die syntaktischen Abhängigkeiten verloren gehen (ähnlich einer funktionalen bzw. Dependenz-Struktur (vgl. 3.2.1), aber mit weniger „Maschinerie“).

Die verschiedenen syntaktischen Varianten des obigen Beispiels könnten daher als binäre „Is-A“-Relationen formuliert werden (vgl. semantisch-logische Ansätze):

```
(Q) is_a(„[NP:Testat]“, X)
(A) is_a(„[NP:Testat]“, „....“)
```

Was in einer hierarchischen Repräsentation (vgl. 2.5.3) dann so aussehen könnte:

```
(Q) Was ist ein Testat?

<phrase pos="ISA_R">
     <subphrase pos="NP">
          <lextag lem="_" pos="_"/>
     </subphrase>
     <subphrase pos="NP">
          <lextag lem="testat" pos="NN"/>
     </subphrase>
</phrase>



(A) Ein Testat ist eine Bescheinigung

<phrase pos="ISA_R">
     <subphrase pos="NP">
          <lextag lem="testat" pos="NN"/>
     </subphrase>
     <subphrase pos="NP">
          <lextag lem="bescheinigung" pos="NN"/>
     </subphrase>
</phrase>
```

Solche „Is-A“-Relationen wurden mittels folgender lexikalisch-syntaktischer Muster untersucht:

```
sein|darstellen|repräsentieren|bilden|gelten|zählen|ähneln|gleichstehen|gleichkommen|entsprechen
|übereinstimmen|bedeuten|sagen|heißen|figurieren|fungieren     → Is_A_V

NP   PP*  Is_A_V    PP*  NP   → IS_A_Relation

Bsp.:     (Q) Was ist ein Testat?
          (A) Ein Testat (an der Universität) ist (in der Regel) eine Bestätigung.


Is_A_V    PP*  NP   PP*  NP   → IS_A_Relation

Bsp.:     (Q) Ist ein Testat ist eine Bestätigung?
          (A) [Im Normalfall] ist ein Testat eine Bestätigung.
```

Äquivalente Regeln liessen sich für „Has-A“- und für Vollverb-Relationen definieren.

Für die Verwendung *binärer* Relationen spricht auch die Erkenntnis, dass bei kopularen Verben, also Verben mit zwei Valenzen („sein“, „schreiben“ etc.) wie in

```
Was ist ein Testat?
```

für das Subjekt der Frage häufig das Objekt des Korpussatzes als Antwort verlangt wird und umgekehrt, was eine sehr gezielte Fragebeantwortung ermöglicht (vgl. [Litkowski]).

Die Implementation obiger Regeln zeigte, dass viele der ansonsten nur mit funktionaler bzw. Semantikanalyse zu erschlagenden Fragen dieses Typs (also zum Beispiel Definitionen) auf diese Weise mit weit geringerem Aufwand präzise beantwortet werden können[66].

In dieser Suchstrategie steckt vermutlich noch grosses Entwicklungspotential. Beispielsweise kann der Definition von Relationen die Erkennung bestimmter Named Entities vorgeschaltet werden, welche dann als abstrahierte semantische *Rollenträger* in die Regeln eingehen, wie beispielsweise bei [Hovy 00]:

[66] Für andere Fragetypen ergab sich allerdings eine leichte Reduktion der Trefferquote

Fragen-Beispiele	**Question Templates**
Who was Johnny Mathis' high school track coach?	who be <entity>'s <role>
Who was Lincoln's Secretary of State?	
Who was President of Turkmenistan in 1994?	who be <role> of <entity>
Who is the composer of Eugene Onegin?	
Who is the CEO of General Electric?	
Potentielle Antworten	**Answer Templates**
Lou Vasquez, track coach of…and Johnny Mathis	<person>, <role> of <entity>
Signed Saparmurad Turkmenbachy [Niyazov], president of Turkmenistan	<person> <role-title*> of <entity>
...Turkmenistan's President Saparmurad Niyazov…	<entity>'s <role> <person>
...in Tchaikovsky's Eugene Onegin...	<person>'s <entity>
Mr. Jack Welch, GE chairman...	<role-title> <person>...<entity> <role>
...Chairman John Welch said ...GE's	<subj>|<psv obj> of related role-verb

Bei [Srihari/Li 99] wird ein ähnlicher Ansatz verfolgt, wobei semantische Rollen der Art *„ist Aspekt/Merkmal von [Entität / Ereignis]"* spezifiziert werden:

Typ-I-Fragen: "ask about X's Y" (X ist eine Entiät, Y ist Aspekt/Merkmal bzw. Relation), z.B.:
What is X's address, email and telephone number?
(or even: What is X1's X2's address, email and telephone number?)
What is X's profession?
Who (or Which organization) is X's employer?
Who is X's wife?
Where is the company X located?
What products does the company X have?

Typ-II-Fragen: "ask about when, where, who/whom, what of an event", z.B.:
When did something happen?
Where did something happen?
Who did that?
Whom did X meet?
What did X give to Y?
To whom did X give Y?

Solche Regeln ermöglichen die Entwicklung sehr differenzierter und vor allem auch *kaskadierbarer* Suchmechanismen auf Basis nicht-semantischer (d.h. lexikalisch-syntaktischer) Sprachanalyse. Darüber hinaus sind obige Beispiele teilweise sogar domänenunabhängig (v.a. obige Typ-II-Fragen) und damit sehr universell einsetzbar.

5.2.3 Dynamische Gewichtung

Neben der „festen Verdrahtung" von sprachlichen Elemente mit entsprechenden Suchstrategien kann auch eine dynamische Zuweisung bestimmter *Gewichtungsfaktoren* („Matching-Gewichte"), welche während der Verarbeitung miteinander verrechnet werden und schlussendlich als *Kriterium für die Treffer-Präzision* über die Relevanz einer potentiellen Antwort entscheiden, den Retrievalerfolg erhöhen.

Wie sich herausgestellt hat, ist eine derartige dynamische Gewichtung allerdings nur im Zusammenspiel mit einer *Sortierung und Filterung der besten Antworten* nach deren Gewichtungssumme / Gesamtpräzision sinnvoll, da ansonsten kaum entschieden werden kann, wie hoch der Schwellwert für das erforderliche Gesamtgewicht einer potentiellen Antworten sein muss, um sie als Treffer zu akzeptieren.

5.2.3.1 Gewichtung von Treffer-Passagen

Potentielle Antwortpassagen können hinsichtlich ihrer Relevanz priorisiert werden, und zwar

1. In Dependenz zur Anzahl Suchterme
2. in Dependenz zur Anzahl Dokumente

ad 1: Je mehr indexierte Frageterme (= Suchterme) gefunden werden sollen, also je *spezifischer* eine Frage ist, desto schwieriger ist es, Antwortpassagen mit zufriedenstellender Übereinstimmung zur Frage zu finden.

Es hat sich daher ein dynamischer Schwellwert für die erforderliche Minimal-Präzision bewährt, welcher in Abhängigkeit zur Anzahl Suchterme variiert wird. Je höher dieser Wert, desto präziser muss eine Passage auf die Frage passen, um als Antwort akzeptiert zu werden (vgl. [Metzinger 02]).

ad 2: Nicht nur die Anzahl Suchterme, sondern auch die gefundenen und im Korpus gesamthaft verfügbaren Dokumente bestimmen die erwartbare und damit zu fordernde Übereinstimmung zwischen Frage und jeweiliger Antwort-Passage.

Je mehr Dokumente potentielle Antworten enthalten, desto „strenger" dürfen die einzelnen Passagen auf ihre Frage-Übereinstimmung hin bewertet werden, um noch genügend viele korrekte Antworten zu erhalten (davon abgesehen ist wie eingangs festgestellt natürlich auch die Auswahl der Korpus-Dokumente entscheidend).

5.2.3.2 Gewichtung von Treffer-Termen

Auch gefundene Wörter und Phrasen haben unterschiedliches „Antwort-Potential" und können daher gewichtet werden:

1. In Dependenz zur Term-Häufigkeit
2. in Dependenz zur Wort- und Phrasenkategorie

ad 1: In der Regel sind die gebräuchlichsten Wörter einerseits auch die ambigsten und anderseits die informationsärmsten, also unwichtigsten. Seltenere Wörter werden im allgemeinen viel spezifischer eingesetzt und besitzen damit auch weit grössere Diskriminierungskraft (vgl. [Verdejo et al. 99]).

Wird ein erfahrungsgemäss seltener Suchterm in der Dokumentkollektion gefunden, so empfiehlt sich die Zuschreibung eines entsprechend hohen Treffergewichts. Unter Umständen entscheidet damit ein einzelner Term bereits über die Treffer-Tauglichkeit der entsprechenden Antwort-Passage. Die Term-Gewichtung erfolgt traditionell in einer Document Retrieval Vorstufe des TR-Verfahren durch Berechnung der Dokument-Term-Häufigkeit (IDTF, siehe [Verdejo et al. 99]).

ad 2: Nebst der häufigkeitsbasierten Term-Gewichtung und der kompletten Entfernung bestimmter Stoppwörter gewichten manche Systeme auch bestimmte *Wortkategorien* nach heuristischen Relevanzkriterien. Häufig werden z.B. Eigennamen höher gewertet als andere Nomen (vgl. [Steinmann 01]).

Im Kontext der gemachten Untersuchung hat sich für die lexikalischen Kategorien die folgende Gewichtungshierarchie (mit absteigender Relevanz) bewährt:

1. Eigennamen
2. normale Nomen
3. Vollverben
4. Adjektive
5. Modale Verben

Darüber hinaus scheinen aber auch *syntaktische bzw. phrasale Kategorien* für den Sucherfolg von unterschiedlicher Wichtigkeit zu sein. Hier hat sich folgende Priorisierung als nützlich erwiesen:

1. Nominalphrasen
2. Präpositionalphrasen
3. Verbalgruppen

Gewichtungen anderer Wort- und Phrasenkategorien wurden hier nicht untersucht und sind auch in der Literatur kaum erwähnt.

5.2.4 Meta-Suchstrategien

Einzelne Suchstrategien lassen sich auf verschiedenen Stufen des Retrievalprozesses einsetzen und auch unterschiedlich gewichten.

Die Entscheidung darüber, in welcher Verarbeitungs-Stufe eine bestimmte Lösungsstrategie zum Einsatz kommt und wie viel Gewicht ihr zuzuordnen ist, schlägt sich oft unmittelbar im Suchergebnis nieder und kann daher – auf einer höheren Ebene des Software-Entwurfs betrachtet – schon als eigene Suchstrategie angesehen werden (eben als Meta-Suchstrategie)[67].

5.2.4.1 Strukturelle Einordnung von Suchstrategien

Die Reihenfolge und Optionalität von Retrieval-Teilschritten trägt unter Umständen zum Sucherfolg bei. Lösungsstrategien können auf verschiedenen Stufen eingesetzt werden:

1. Standardmässige Implementation
2. Sukzessive Suchraum-Erweiterung
3. Sukzessive Suchraum-Eingrenzung

In welchem Verarbeitungsschritt eine bestimmte Suchstrategie zum Einsatz kommt, wird unterschiedlich gehandhabt. Beispielsweise eignet sich die eine *Anfrage-Erweiterung durch Synonyme* offenbar gut zur Suchraum-Erweiterung (vgl. [Mollá et al. 00a]), ist grundsätzlich aber auch als standardmässige Implementation denkbar[68].

5.2.4.2 Gewichtung von Suchstrategien

Nebst der „festen Verdrahtung“ von Verarbeitungsschritten in der Retrieval-Abfolge können Suchstrategien unterschiedlicher Wichtigkeit auch gegeneinander priorisiert werden, was in eigenen Tests mit verschiedenen (statischen) Gewichtungsszenarien zu markanten Auswirkungen auf das Suchergebnis führte:

Tabelle 4: Gewichtung von Suchstrategien in verschiedenen Test-Szenarien

	Szenario 1	Szenario 2	Szenario 3	Szenario 4	Szenario 5
Treffer-Schwellwert insgesamt	niedrig	hoch	mittel	mittel	mittel
Syntaktische Kategorien	hoch	hoch	hoch	hoch	**niedrig**
Semantische Indikatoren	**keine**	hoch	**niedrig**	**niedrig**	hoch
Standardmässige Anfrage-Erweiterung durch Synonyme	ja	ja	ja	**nein**	ja
Präzision	***0.31***	***0.23***	***0.30***	***0.46***	***0.18***

Diese Ergebnisse sind natürlich sehr domänenspezifisch und daher kaum verallgemeinerbar, zeigen aber dennoch die Wichtigkeit, sich auf der Ebene des Software-Entwurfs über die Priorisierung der in der Suche eingesetzten Komponenten Gedanken zu machen.

Zudem handelt es sich bei den hier aufgeführten Präzisionsangaben um Durchschnittswerte über einem bestimmten Test-Set von Fragen. Im Detail betrachtet, fallen die Resultate etwas differenzierter aus, d.h. einzelne Fragen werden in anderen Gewichtungsszenarien als dem besten mit mehr Erfolg beantwortet. Es stellt sich daher die Frage, ob eine *dynamische Gewichtung* schon auf Ebene der Meta-Suchstrategie zu präziseren Antworten führen würde, also die vorgängige *Analyse der Frage* zu Schlussfolgerungen für die adäquate Priorisierung der einzelnen Suchstrategien führen könnte. Allerdings dürfe es recht schwierig sein, hierfür heuristische Kriterien zu finden, sodass zur Zeit das grössere Entwicklungspotential in der Optimierung der statischen Suchstrategie-Gewichtung zu liegen scheint.

[67] Die zuvor beschriebenen Ansätze wurden aus Gründen einer gewissen Theorie-Neutralität nicht einer bestimmten Stufe zugeordnet

[68] dies hat sich nach eigenen Untersuchungen aufgrund von lexikalischen Mehrdeutigkeiten allerdings als recht schwierig erwiesen (vgl. [Metzinger 02]).

5.3 Fazit

5.3.1 Zusammenfassung: Jedem Fragetyp seine Suchstrategie?

Die beschriebenen Strategien eignen sich unterschiedlich gut zur Beantwortung bestimmter Fragetypen – umgekehrt benötigen verschiedene Fragen auch ein unterschiedliches Mass an Technikeinsatz für eine halbwegs präzise Antwortsuche. Wie Literaturrecherchen und (eigene) Erfahrungen gezeigt haben, genügt bei einigen Anfragen bereits eine gute Stichwortsuche, während in anderen Fällen selbst komplexe syntaktische oder gar semantische Verfahren nicht zum Ziel führen.

Die folgende empirisch motivierte Aufstellung versucht die erforderlichen computerlinguistischen Mittel für die Beantwortung bestimmter Fragen gemäss deren intendierter Bedeutung einzuordnen (vgl. eingangs beschriebene Typologie in Tabelle 2):

Tabelle 5: Suchstrategien für verschiedene Fragetypen

Fragetyp / Intension[69]	**Beispiel**	**Erwünschte Reaktion**	**Erforderliche Technik / Suchstrategien**			
			Stichwort-Suche[70]	**Syntax**[71]	**Seichte Semantik**[72]	**Tiefe Semantik**[73]
Causal Antecedent	Warum ist die letzte CL-Vorlesung ausgefallen?	Begründung	✔	✔	✔ lexikal.-Indikatoren	
Goal Orientation	Wozu wird der grüne Seminarschein benötigt?	Erklärung	✔	✔	✔ lexikal.-Indikatoren	
Enablement	Welche Bescheinigungen werden zur Lizentiatsprüfung benötigt?	Aufzählung	✔	(✔)		
Causal Consequent	Was passiert, wenn alle CL-Dozenten Grippe haben?	Voraussage/ Inferenz	✔	✔	✔	✔
Verification	Wird die Akzessprüfung zur Seminarteilnahme benötigt?	Ja/ Nein, evt. Erklärung	✔	(✔)		
Disjunctive	Müssen die Übungen in ein oder (in) zwei Wochen abgegeben werden?	Entscheidung	✔	(✔)		
Instrumental / Procedural	Wie funktioniert EXTRANS?	Beschreibung	✔	✔	✔	✔
Concept Completion	Wann findet die nächste Akzessprüfung statt? Wo finde ich das Büro des Dekans? Wer wählt den Dekan?	Konkreter Term (Datum, Ort, Person etc.)	✔	✔	✔ Named Entities	
Expectational	Warum macht Prof. XYZ so langweilige Vorlesungen?	Argumentation, Rechtfertigung	✔	✔	✔	✔
Judgemental	Wie beurteilen Sie die missliche Lage der Publizistikstudenten?	Begründung, Argumentation	✔	✔	✔	(✔)
Quantification	Wie viele Studenten werden für die CL zugelassen? Wie viel kostet die Einschreibung?	Konkreter Term (Zahl oder Mass)	✔	✔	✔ NEs, lexikal. Indikatoren	
Feature Specification	Welche Analyseverfahren verwendet EXTRANS?	Konkreter Term/ Aufzählung	✔	(✔)		
Request / Command	Exmatrikulieren Sie mich per sofort!	→ weiterleiten	✔	✔	(✔)	(✔)
Insult	Ihre Uni ist die lahmste, die ich kenne!	→ ignorieren	?	?	?	?

[69] Typologie nach [Lehnert 81]
[70] inklusive Filterung, Lemmatisierung und ggf. Synonymisierung (vgl. 5.2.1)
[71] Phrasenzugehörigkeit, Übereinstimmung der syntaktischen Kategorie, evt. syntaktische Relationen (vgl. 5.2.2)
[72] Lexikalische Frage- und Antwort-Indikatoren, Named Entities (vgl. 5.2.1.4)
[73] z.B.: mittels logischer Inferenz (vgl. 2.5.5 und 4.1.4)

5.3.2 Einschränkende Faktoren

Die Eignung der meisten hier vorgestellten Suchstrategien wird in der Praxis durch einige Feststellungen geschmälert (vgl. [Metzinger 02]):

1. Suchstrategien sind textsorten-spezifisch
2. Die lexikalische Übereinstimmung ist entscheidend
3. Suchstrategien wirken teilweise antagonistisch

ad 1:
Die meisten der untersuchten Strategien, vornehmlich natürlich die lexikalisch-orientierten, haben eine sehr textsorten-spezifische Eignung; für das Auffinden potentieller Antworten in *FAQ-Dateien*[74] beispielsweise führen teilweise andere Techniken und Heuristiken zum Erfolg als für *allgemein formulierte, heterogene Fakten-Texte*.

Dies bedeutet, dass Suchstrategien auf die verwendeten Korpustexte manuell angepasst oder aber trainiert werden sollten.

ad 2:
Je näher die lexikalische Formulierung einer Frage an derer der Korpustexte ist, desto höher ist im allgemeinen die zu erwartende Trefferquote. Im „Idealfall" entspricht auch der in der Frage verwendete Sprachstil demjenigen des Korpus; im allgemeinen dürften die zu beantwortenden Fragen aber einen einfacheren Sprachstil als die Korpus-Texte verwenden, was natürlich zur Reduktion der lexikalischen Übereinstimmung und damit der Trefferquote führt.

Die lexikalische Übereinstimmung scheint für den Sucherfolg wichtiger als irgendeine andere Suchstrategie zu sein und insbesondere sogar entscheidender als die Spezifität der Frage (also die Anzahl der Suchterme) und als deren syntaktische Abhängigkeiten; Fragen mit unzureichender lexikalischer Übereinstimmung und/oder niedriger Spezifität wurden in fast allen Tests schlecht beantwortet, was dieses Kriterium zur *Grundlage vieler Suchstrategien* macht.

Eine gute *Stichwortsuche* (Keyword Spotting) ist und bleibt damit ein entscheidender Bestandteil eines jeden Retrieval-Verfahrens; dementsprechend kommt auch der Entwicklung eines *domänenspezifischen Synonym-Lexikons* fundamentale Bedeutung zu.

ad 3:
Als weitere zentrale Erkenntnis zeigt sich, dass verschiedene *punktuelle Lösungsansätze zu spezifischen Teilproblemen* der Sprachverarbeitung nicht immer ohne weiteres miteinander vereinbar sind, da sie im Zusammenspiel zu *konkurrenzierenden Abhängigkeiten* führen.

Dies manifestiert sich innerhalb eines Systems beispielsweise in der Auswirkung verschieden gewichteter Suchstrategien, welche – kaum nachvollziehbar – gleichzeitige Performance-Gewinne und -Verluste bewirken: Vergleicht man verschiedene Test-Szenarien mit ähnlichen durchschnittlichen Performance-Werten, so sind es durchaus nicht immer dieselben Fragen, welche gleich gut beantwortet werden. Und oft zeigt selbst die erfolgreichste Konstellation verschiedener Suchstrategien stellenweise Einbrüche gegenüber anderen, ansonsten schwächer ausfallenden Kombinationen, da es sich eben „nur" um den besten Kompromiss zwischen mehreren teilweise gegenläufigen Lösungsansätzen handelt (vgl. Tabelle 4 und [Metzinger 02]).

Das „Beste aus allen Welten" scheint also schwer zu finden zu sein. Konvergenz und Zusammenspiel von unterschiedlichen Lösungsansätzen wird die Computerlinguistik der kommenden Jahren daher sicherlich vermehrt beschäftigen.

[74] FAQ = Frequently Asked Questions = Häufig gestellte Fragen

6 Ausblick

Was leisten die heutigen E-Mail Beantwortungssysteme? Für den Kunden ist wohl das wichtigste Kriterium: *„Wie nützlich ist die Antwort für mich?“* und vor allem *„löst sie mein Problem?“* Es stellt sich die Frage, inwiefern diese Ansprüche durch die heutigen Technologien für eine (halb-) automatische E-Mail-Beantwortung erfüllt werden können.

Die *Evaluation* kommerzieller Software in diesem Bereich gestaltet sich allerdings nicht einfach; die Mehrzahl der Anfragen blieb unbeantwortet, viele Programme waren nicht installierbar oder konnten nur während kurzer Zeit online „getestet“ werden. Vor allem existieren aber bislang keine weitreichend anerkannten *Evaluationsstandards,* -raster oder -systeme für computerlinguistische Software; die hinlänglich bekannten Kriterien für Performanz und Effizienz sind selten allgemeingültiger Natur und lassen sich allenfalls als „Eckpunkte für eine vage Definition einer gewissen linguistischen Abdeckung“ verstehen. Das macht die Evaluation von CL-Software im allgemeinen schwierig.

Auch die mannigfaltigen wissenschaftlichen Ansätze lassen sich selten auf einen gemeinsamen Nenner bringen, der eine erfolgversprechende Kombination partikulärer Strategien erlauben würde: Verschiedene Forschungszweige, Schulen (v.a. Universitäten) und Firmen erforschen zumeist spezifische Detail-Probleme und finden dort im besten Fall gute Verbesserungsansätze. Diese sind oft aber nicht verallgemeinerbar, da domänenspezifisch, unzugänglich (hinter verschlossenen Türen), und vor allem funktionieren sie oft nicht im Zusammenspiel mit Lösungen zu anderen Teil-Problemen, da sich verschiedene Lösungsstrategien konkurrenzieren bzw. antagonistisch verhalten. Hier zeichnet sich daher ein Forschungsfeld in der Computerlinguistik der kommenden Jahre ab: Es hat sich – zumindest im akademischen Umfeld – scheinbar noch fast niemand mit dem Problemen beschäftigt, die durch die Kombination partikulärer Lösungsansätze entstehen (möglicherweise besteht die computerlinguistische Forschung aus zu vielen „Eigenbrödlern“). Konvergenz der verschiedenen Schulen und Ansätze ist daher wünschenswert und wahrscheinlich unerlässlich.

Umso mehr ist die Leistung moderner Fragebeantwortungssysteme bereits erstaunlich. Beschränkungen, die noch vor einigen Jahren manche theoretisch wünschenswerte Lösung in der Praxis verunmöglichten, etwa eine Analyse mittels grosser, heuristischer Grammatiken umfangreicher Korpora und komplexer, auch kontext-sensitiver Algorithmen werden angesichts der rasant steigenden Prozessorleistungen moderner Computersysteme zunehmend unwichtig und weichen neuen Möglichkeiten für die automatische Sprachverarbeitung. Dennoch erfreuen sich gerade linguistische Low-Level Technologien besonderer Beleibtheit; der Stellenwert von Einfachheit und insbesondere Heuristik ist gross. Vor allem im „Real World“ Einsatz kommt es insbesondere bei den symbolorientierten Ansätzen immer mehr auf das Know-How und die Arbeit eines menschlichen "Knowledge-Engineers" an, der Ontologien, semantische Modelle, Grammatikregeln, Korpora und dergleichen entwickelt, aufbereitet oder zugänglich macht. Vielleicht hat sich hier bereits eine neues Berufsfeld in der Computerlinguistik geöffnet.

Die Stärke kommerzieller CL-Software liegt denn auch zunehmend in der Benutzerfreundlichkeit für ein verständliches und effizientes Knowledge-Engineering sowie in der werkmässigen Bereitstellung (bzw. auf Anfrage Aufbereitung) benutzbarer Wissensbasen. Damit avancieren die Software-Hersteller zunehmend zu Dienstleistungsanbieten, die für die Entwicklung und Realisation domänenspezifischer Korpora und/oder Wissensbasen verantwortlich sind und diese Arbeit dem Endverbraucher – im Beispiel E-Mail Beantwortung ist das meistens eine grössere Firma – abnehmen.

Des weiteren scheint auch ein zweiter CL-Strang weiterhin sehr aktiv zu sein: Stochastische Systeme, statistische Algorithmen, Neuronale Netze versprechen vor allem dort Abhilfe zu leisten, wo regelbasierte Systeme trotz all ihren Heuristiken kapitulieren. Hier sind interdisziplinäre Avancen festzustellen, da gerade Strategien wie die Neuronalen Netze für die verschiedensten (auch nicht-linguistischen) Anwendungen ähnlich funktionieren und ähnliche Probleme aufwerfen, deren Lösungen wiederum oftmals auf alle Anwendungen (und damit eben auch die CL) adaptiert werden können (Beispiel: Muster-Erkennung). Zu klären ist daher nach wie vor die Frage der Konvergenz beider CL-Stränge, also das Zusammenspiel zwischen regelbasierten und stochastischen Ansätze mit ihren Stärken und Schwächen.

Der maschinellen Verarbeitung elektronischer Anfragen eröffnen sich also noch etliche Fragen. Der heutige Wissenstand erlaubt sicherlich bereits eine robuste *halbautomatische Fragebeantwortung*. Ein gewisses Mass an menschlicher Supervision und „Real-World-Training“ scheint zur Zeit aber (noch) unabdingbar. Die Marktentwicklung und wissenschaftliche Zukunft der E-Mail Beantwortung wird sich wahrscheinlich in Richtung Kombination mit anderen Kommunikationskanälen und Technologien bewegen und allmählich von der halb- zur vollautomatischen Verarbeitung natürlicher Sprache führen.

Bis dahin gibt es noch viel zu tun.

Björn Metzinger

Mai 2002

7 Referenzen

7.1 Literatur

[Abney 96a]
Abney, Steven (1996): Cascaded Finite-Sate Parsing. Folien zum Vortrag am Xerox Research Centre, Grenoble. In: http://www.sfs.nphil.uni-tuebingen.de/~abney/96a.ps.gz

[Abney 96b]
Abney, Steven (1996): Partial Parsing via Finite-State Cascades. Proceedings of the ESSLLI ,96 Robust Parsing Workshop. In: http://www.sfs.nphil.uni-tuebingen.de/~abney/96h.ps.gz

[Arnold et al. 01]
Arnold, Toni / Clematide, Simon / Nespeca, Roberto / Roth, Jeannette / Volk, Martin (2001): LUIS - Ein natürlichsprachliches, universitäres Informationssystem. In: Proc. of Symposium Unternehmen Hochschule 2001; http://www.ifi.unizh.ch/CL/CLpublications/luis.pdf

[Borsley 97]
Borsley, Robert D. (1997): Syntax-Theorie. Ein zusammengefasster Zugang. Deutsche Bearbeitung von Peter Suchsland. Tübingen.

[Brightware 99]
Brightware, Inc. (1999): Selecting an Effective AutoAnswer System. A comparison of information extraction, neural networks and other statistically-based autoanswer systems.

[Burger et al.]
Burger, John et al. (o.J.): Issues, Tasks and Program Structures to Roadmap Research in Question & Answering (Q&A). In:
http://www.ifi.unizh.ch/CL/hess/classes/seminare/semrep/stuff/trec8/qa-papers/qa.Roadmap-paper_v2.doc

[Carstensen et al. 01]
Carstensen, K.-U. et al. (Hg.) (2001): Computerlinguistik und Sprachtechnologie. Eine Einführung. Heidelberg / Berlin.

[Charniak 93]
Charniak, Eugene (1993): Statistical language learning. Bradford.

[Clarke et al.]
Clarke, C. L. A. et al. (o.J.): Question Answering by Passage Selection (MultiText Experiments for TREC). Computer Science, University of Waterloo, Canada. In:
http://plg.uwaterloo.ca/~claclark/trec9.ps

[Colombo / Nespeca 01]
Colombo, Franco / Nespeca, Roberto (2001): Extrans / LUIS. Handout (Kurz-Beschreibung und Evaluation) zum Vortrag im Seminar Antwort-Extraktion. Abteilung Computerlinguistik, Institut für Informatik der Universität Zürich.

[Duvall 98]
Duvall, Mel (1998): Egain Automates E-Mail Replies. In: http://www.zdnet.com/intweek/daily/980810k.html.

[eGain 99a]
eGain™ Communications Corp (1999): eGain Communications Announces eGain EMS 2.0 -- Setting New Standards for E-Commerce Customer Service. Newsletter. In:
http://www.egain.com/pages/Level2.asp?sectionID=6&pageID=217.

[eGain 99b]
eGain™ Communications Corp (1999): eGain Communications Launches Major Operations In The UK: Helps Companies Turn E-Commerce Customer Service Problems Into Business Opportunities. Newsletter. In: http://www.egain.com/pages/Level2.asp?sectionID=6&pageID=156.

[eGain]
eGain™ Communications Corp (o.J.): eGain Mail Overview. Produktbeschreibung (html). In:
http://www.egain.com/pages/Level2.asp?sectionID=2&pageID=527.

[eGain 00]
eGain™ Communications Corp (2000): eGain Mail. Produktbeschreibung (pdf). In: http://www.egain.com/docs/products/egainmail.pdf.

[Ferret et al. 00]
Ferret, Olivier et al (2000): QALC- the Question Answering program of the Language and Cognition group at LIMSI-CNRS. Orsay Cedex, Frankreich. In: http://www.ifi.unizh.ch/CL/hess/classes/seminare/semrep/stuff/trec8/qa-proc/LimsiLC-proceedings-paper.pdf

[Ganz-Blättler/ Süss 96]
Ganz-Blättler, Ursula/ Süss, Daniel (1996): Medien und Aussagen. In: Bonfadelli, Heinz/ Hättenschwiler, Walter (Hg.): Einführung in die Publizistikwissenschaft (2., erweiterte Auflage). Zürich, S. 39-58.

[Hobbs et al. 93]
Hobbs, Jerry R. et al (1993): Fastus: A Cascaded Finite-State Transducer for Extracting Information from Natural-Language Text. Paper. Artificial Intelligence Center, SRI International, Kalifornien. In: http://www.ifi.unizh.ch/cl/hess/classes/seminare/semrep/docs/fastus.pdf

[Hovy et al. 00]
Hovy, Eduard et al. (2000): Question Answering in Webclopedia. Information Sciences Institute, University of Southern California, Marina del Rey. In: http://www.ifi.unizh.ch/CL/hess/classes/seminare/semrep/stuff/trec9/qa/proceedings/webclopedia.pdf

[Kiwilogic 00]
Kiwilogic.com, Inc. (2000): Kiwilogic´s Lingubot™ technology: A toolkit for making websites understand natural language. White Paper. In: http://www.kiwilogic.com/htm/download/white_paper.pdf

[Lambert 98]
Lambert, Patrick (1998): Do these startups have the answer to too much e-mail? Artikel. Business Week Online. In: http://www.businessweek.com/bwdaily/dnflash/july1998/nf80713a.htm.

[Lehnert 81]
Lehnert, Wendy G. (1981): A computational theory of human question answering. In: Joshi, A. / Webber, B. / Sag, I. (Hg) (1981): Elements of Discourse Understanding. Cambridge.

[Litkowski]
Litkowski, Kenneth C. (o.J.): Question-Answering Using Semantic Relation Triples. CL Research, Damascus. In: http://www.ifi.unizh.ch/CL/hess/classes/seminare/semrep/stuff/trec8/qa-proc/clresearch.pdf

[Metzinger 00]
Metzinger, Björn (2000): Syntaxannotation. Theoretische Grundlagen und praktische Anwendung am Beispiel „Nebenläufige grammatische Verarbeitung“. Seminararbeit. Institut für Informatik der Universität Zürich, Abteilung Computerlinguistik, Zürich. In: http://www.ifi.unizh.ch/cl/hess/classes/seminare/theorien/arbeiten/negra.pdf

[Metzinger 02]
Metzinger, Björn (2002): Indexierung und Retrieval für korpusbasierte E-Mail-Beantwortung. Technische Dokumentation / Programmierprojekt in Computerlinguistik. Institut für Informatik der Universität Zürich, Abteilung Computerlinguistik, Zürich.

[Moldovan et al.]
Moldovan, Dan et al. (o.J.): LASSO: A Tool for Surfing the Answer Net. Department of Computer Science and Engineering, Southern Methodist University, Dallas. In: http://www.ifi.unizh.ch/CL/hess/classes/seminare/semrep/stuff/trec8/qa-proc/smu.pdf

[Mollá et. al 00a]
Mollá, Diego / Schneider, Gerold / Schwitter, Rolf / Hess, Michael (2000): Answer extraction using dependency grammar in extrans. Abteilung Computerlinguistik, Institut für Informatik der Universität Zürich. In: http://www.ifi.unizh.ch/cl/hess/classes/seminare/semrep/stuff/DGZurich.pdf

[Mollá et al. 00b]
Mollá, Diego / Schwitter, Rolf / Hess, Michael (2000): Extrans, an answer extraction system. Abteilung Computerlinguistik, Institut für Informatik der Universität Zürich. In: http://www.ifi.unizh.ch/CL/hess/classes/seminare/semrep/stuff/clzurich.pdf

[Negra 00]
Universität des Saarlands (Hg) (2000): Nebenläufige grammatische Verarbeitung. In: http://www.coli.uni-sb.de/info/projects/negra.html

[Neumann 01]
Neumann, Günter (2001): Informationsextraktion. Paper. DFKI GmbH. In: Klabunde et al (Hg.): Computerlinguistik und Sprachtechnologie - Eine Einführung. Heidelberg. http://www.ifi.unizh.ch/cl/hess/classes/seminare/semrep/docs/ie.pdf

[Neumann et al. 00]
Neumann, Günter / Braun, Christian / Piskorski, Jakub (2000): A Divide-and-Conquer Strategy for Shallow Parsing of German Free Texts. In: http://www.ifi.unizh.ch/cl/hess/classes/seminare/semrep/docs/final-acl2000.pdf

[Prager et al. 00]
Prager, John et al (2000): Question-Answering by Predictive Annotation. Michigan und New York. In: http://www1.acm.org/pubs/articles/proceedings/ir/345508/p184-prager/p184-prager.pdf

[Quintus 00]
Quintus Corporation (2000): eContact eMail. Produktbeschreibung (pdf). In: http://www.quintus.com/pdfs/email.pdf

[Schwitter et al. 00]
Schwitter, Rolf / Mollá, Diego / Fournier, Rachel / Hess, Michael (2000): Answer Extraction. Towars better Evaluations of NLP Systems. Abteilung Computerlinguistik, Institut für Informatik der Universität Zürich. In: http://www.ifi.unizh.ch/CL/schwitter/rc_final.pdf

[Skut et al. 97]
Skut, Wojciech/ Krenn, Brigitte/ Brants, Thorsten/ Uszkoreit, Hans (1997): An Annotation Scheme for Free Order Languages. Paper. Universität des Saarlands, Computational Linguistics, Saarbrücken.

[Smith]
Smith, John B. (o.J.): Perl Basics: Regular Expressions. Online Lehrgang. Department of Computer Science, University of North Carolina, Chapel Hill. In: http://www.cs.unc.edu/~jbs/resources/perl/perl-basics/regular_expressions.html

[Srihari/Li 99]
Srihari, Rohini / Li, Wie (1999): Information Extraction Supported Question Answering. Cymfony Inc., Williamsville. In: http://trec.nist.gov/pubs/trec8/papers/cymfony.pdf

[Steinmann 01]
Steinmann, Cornelia (2001): Question Answering. Strategien in TREC-8. Seminararbeit. Institut für Informatik der Universität Zürich, Abteilung Computerlinguistik, Zürich.

[Tan 98]
Tan, Huey Kein (1998): Phrasenindexierung für Passagenretrieval in deutschsprachigen Texten. Diplomarbeit. Institut für Informatik der Universität Zürich.

[Verdejo et al. 99]
Verdejo, Felisa / Gonzalo, Julio / Peñas, Anselmo (1999): Information Retrieval & Natural Language Processing. Online-Kurs. In: http://rayuela.ieec.uned.es/~ircourse/

[Wauschkun 96]
Wauschkun, Oliver (1996): Ein Werkzeug zur partiellen syntaktischen Analyse deutscher Textkorpora. Resultate der 3. KONVENS Konferenz, Bielefeld, 1996. Paper. Institut für Informatik, Universität Stuttgart. In: http://www.informatik.uni-stuttgart.de/ifi/is/Forschung/Papiere/oliver/konvens96.ps

[Webopedia]
Webopedia. Online Lexikon für Computerfachbegriffe. In: http://webopedia.internet.com (April 2001)

[Wu 00]
Wu, Lide (2000): FDU at TREC-9: CLIR, Filtering and QA tasks. Fudan University, Shanghai, China. In: http://trec.nist.gov/pubs/trec9/papers/FduT9Report.pdf

7.2 Software

Im folgenden einige (zumindest für Forschungszwecke) frei verfügbare Analyse-Programme und Java-Klassen für diverse TR-Aufgaben. Nicht alle Tools wurden ausgiebig getestet.

Allgemein:

- Diverse Links zu Taggern, Parsern, Lexika, Grammatiken und linguistischer Literatur. In: http://www.med.uni-muenchen.de/mki/instruct/members/Matthias/ling.html
- Linkliste zu statistisch- und korpusbasierten NLP-Tools und lexikalischen Ressourcen sowie zu linguistischer Literatur. In: http://www-nlp.stanford.edu/links/statnlp.html

Demo-Versionen komerzieller NLP-Evaluationssoftware:

- Kiwilogic Lingubot Creator: Web-Interface-Produktionspaket zur Einbindung von NLP direkt in Internet-Seiten (oder zur Beantwortung von E-Mails). Basiert auf regulären Ausdrücken. In: http://www.kiwilogic.com/htm/download/download_creator.htm

Indexierung:

- Swish-E: Einfaches Web-Seiten-Indexierungs-Paket für Keyword-Suche in Java; enthält eine Perl-basierte Benutzerschnittstelle zur manuellen Indexierung. In: http://sunsite.berkeley.edu/SWISH-E/

Logische Programmierung:

- Links zu diversen Emulationen (!!) unterschiedlicher Sprachen in Java (darunter z.B. „Java Internet Prolog“). In: http://grunge.cs.tu-berlin.de/~tolk/vmlanguages.html#logic

Morphologieanalyse und Tagging:

- QTag: Java-basierter, sprachunabhängiger stochastischer Tagger; Ausgabe in XML; wird mit englischem (leider nicht sehr grossem) Lexikon ausgeliefert. In: http://www.english.bham.ac.uk/staff/oliver/software/tagger/
- Winbrill: Der Brill-Tagger für Windows als Kommandozeilen- und GUI-Version. Mit französischem und englischem Lexikon und Regelset. In: http://jupiter.inalf.cnrs.fr/cgi-bin/mep.exe?HTML=mep_winbrill.txt?CRITERE=ENGLISH

Parser für NLP:

- Stefy: HPSG-Parser in Java; weitgehende Abdeckung der HPSG-Prinzipien; Grammatikmodellierung in Feature- oder „DCG“-Syntax; leider ohne lexikalische Ressourcen und eher schlecht dokumentiert. In: http://vlado.uwaterloo.ca/~vkeselj/stefy.html

Reguläre Ausdrücke:

- Savarese ORO-Matcher und Text-Tools: Sehr gutes Java-Archiv für Perl-5-kompatible Reguläre Ausdrücke und diverse Text-Processing-Tools. In: http://www.savarese.org/oro/downloads/index.html
- Programmiersprache Perl; Referenz für Reguläre Ausdrücke. In: www.activeperl.com

XML-Tools:

- SXP – Silfide XML Parser, in Java. In: http://www.loria.fr/projets/XSilfide/EN/sxp/
- XML tools: Umfangreiche (!) Link-Sammlung vieler Parser, Editoren, Suchmaschinen, Datenbanksysteme und mehr für XML. In: http://www.garshol.priv.no/download/xmltools/cat_ix.html#SC_XML