Computerlinguistische Hausarbeiten

Fabian Steeg

I think that it's extraordinarily important that we in computer science keep fun in computing. When it started out, it was an awful lot of fun. Of course, the paying customers got shafted every now and then, and after a while we began to take their complaints seriously. We began to feel as if we really were responsible for the successful, error-free perfect use of these machines. I don't think we are. I think we're responsible for stretching them, setting them off in new directions, and keeping fun in the house.

– Alan J. Perlis
(1922-1990, Recipient of the first Turing Award in 1966)

Vorwort

Dies ist eine Zusammenstellung der Hausarbeiten von Fabian Steeg in den Fächern *Sprachliche Informationsverarbeitung* und *Allgemeine Sprachwissenschaft*, erstellt zwischen 2002 und 2007 im Rahmen eines Magisterstudiums an der Philosophischen Fakultät der Universität zu Köln.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Teil I.

Grundstudium

Kapitel 1

Computerlinguistische Grundlagen

Informationsextraktion, Sprachliche Informationsverarbeitung, Proseminar *Computerlinguistische Grundlagen* bei Jürgen Hermes MA, Wintersemester 2002–2003 und Sommersemester 2003.

1.1. Grundlegendes zur Informationsextraktion

1.1.1. Was ist Informationsextraktion?

Informationsextraktion (IE) kann aus zwei verschiedenen Perspektiven betrachtet werden. Einerseits als das Erkennen von bestimmten Informationen – so bezeichnet etwa Grishman (2003) IE als "the automatic identification of selected types of entities, relations, or events in free text" –, andererseits als das Entfernen der Informationen, die nicht gesucht werden. Letztere Sichtweise drückt etwa eine Definition von Cardie (1997) aus: "An IE system takes as input a text and 'summarizes' the text with respect to a prespecified topic or domain of interest". In diesem Sinne könnte man Informationsextraktion auch als gezielte Textzusammenfassung bezeichnen (Euler 2001b,a, 2002 und Abschnitt 1.4).Informationsextraktionssysteme sind also immer zumindest auf ein spezielles Fachgebiet, meist sogar auf bestimmte Interessengebiete (Szenarios) innerhalb eines allgemeineren Fachgebietes (Domäne) ausgerichtet. So wäre etwa in der Domäne 'Wirtschaftsnachrichten' ein mögliches Szenario 'Personalwechsel in einer Managementposition'. Eine weitergehende Einschränkung macht Neumann, wenn er schreibt, daß das Ziel der IE "die Konstruktion von Systemen" sei, "die gezielt domänenspezifische Informationen aus freien Texten aufspüren *und strukturieren* können [...]" (Neumann 2001, Hervorhebung von mir). In diesem Zusammenhang ist zu beachten, daß eine solche Einschränkung Konsequenzen für die technische Realisierung eines Informationsextraktionssystems hat. Ziel dieser Arbeit ist es, einen Überblick über Informationsextraktion als das zu verschaffen, was alle Definitionen gemeinsam haben, nämlich die Gewinnung von spezifizierten Informationen aus Texten sowie die wissenschaftliche Disziplin, die sich mit dieser Aufgabe beschäftigt.

1.1.2. Abgrenzung von Nachbargebieten

Abzugrenzen ist das eigenständige Forschungsgebiet der Informationsextraktion von verwandten Gebieten: Textzusammenfassung hat eine umfassende Zusammenfassung[1] des Inhaltes eines Textes zum Ziel. Textkategorisierung bedeutet das selbstständige Gruppieren von Texten, Textklassifikation das Einordnen von Texten in vorgegebene Gruppen. Mit Information Retrieval kann die Suche nach Dokumenten in einer Dokumentenmenge (Volltextsuche) oder auch – entsprechend der wörtlichen Bedeutung – die allgemeiner formulierte Aufgabe des Abrufs von Informationen gemeint sein (vgl. Strube 2001). Data Mining bezeichnet ganz allgemein den "Prozeß, Muster in Daten zu erkennen" (Witten & Frank 2000, 3).

1.1.3. Anwendungsmöglichkeiten

Generell sind zwei Arten der Anwendung von Informationsextraktion denkbar: Zum einen können die extrahierten Daten sofort für einen menschlichen Betrachter gedacht sein. In diesen Anwendungsbereich fällt etwa das von Euler (2001b) zu Testzwecken entwickelte System, das aus E-Mails extrahierte Informationen als SMS weiterleitet, oder ein System, das in einer Suchmaschine zu den Treffern extrahierte Informationen anzeigt, etwa die angebotenen Positionen in Stellenanzeigen. Zum anderen können die Daten für die maschinelle Weiterverarbeitung gedacht sein, sei es zur Speicherung in Datenbanken, zur Textkategorisierung oder -klassifikation oder als Ausgangspunkt für eine umfassende Textzusammenfassung. Bestehen die gesuchten Informationen aus mehreren Einzelinformationen, bestimmt das Anwendungsgebiet gewisse Ansprüche an das Informationsextraktionssystem. So müssen zu einer maschinellen Weiterverarbeitung die Informationen strukturiert vorliegen, während für eine Weiterverarbeitung direkt durch den Menschen auch ein unstrukturiertes Ergebnis genügen kann. Wenn die gesuchten Informationen nicht aus weiteren Einzelinformationen bestehen, wie bei der Erkennung von Eigennamen (s. Abschnitt 1.2), ist eine solche Unterscheidung überflüssig.

1.1.4. Evaluationskriterien

Zur Bewertung (Evaluation) von Informationsextraktionssystemen werden die im Information Retrieval gebräuchlichen Kriterien Vollständigkeit (Recall) und Präzision (Precision) bzw. das aus diesen Werten ermittelte F-Maß verwendet. Bei einer Extraktion von Informationen der Art 'a' aus einer Menge von Informationen {aaaab} mit dem Ergebnis {aab} läge die Vollständigkeit bei $R = \frac{2}{4}$, da von den vier Informationen vom Typ 'a' in der Ausgangsmenge zwei extrahiert wurden, die Präzision würde $P = \frac{2}{3}$ betragen, da von drei Elementen in der Ergebnismenge zwei der gewünschten Art von Information entsprechen. Ein weiteres Kriterium zur Bewertung der Güte des Extraktes ist der Anteil der unerwünschten Informationen (Fall-out), hier $\frac{1}{3}$, da eine von drei extrahierten Informationen nicht vom gesuchten Typ ist. Das F-Maß erlaubt eine einheitliche Betrachtung der Gütekriterien Vollständigkeit und Präzision: $F = \frac{2RP}{R+P}$

1 Die umfassende automatische Textzusammenfassung ist insofern problematisch, als daß auch menschliche Leser bei der Aufgabe, das Wichtigste eines Textes zusammenzufassen, nie völlige Übereinstimmung erzielen werden, wenn nicht spezifiziert wurde, *inwiefern* die Informationen wichtig sein sollen.

1.2. Eigennamenerkennung und -klassifikation

1.2.1. Ziele

Eine vergleichsweise einfache Aufgabe für ein Informationsextraktionssystem ist die Eigennamenerkennung und -klassifikation. Das gewünschte Resultat einer solchen Erkennung und Klassifikation könnte etwa so[2] aussehen:

```
<Name Typ=Person> Stefan Ernst </Name>, Mitarbeiter der <Name Typ=Firma>
IBM </Name>, fuhr nach <Name Typ=Ort> Köln </Name>.
```

Ein solches Resultat eignet sich auch als erste Stufe der Extraktion eines Ereignisses (vgl. Abschnitt 1.3.3) und wird etwa im System FASTUS so verwendet (s. Appelt *et al.* 1993 und Abschnitt 1.3.5).

1.2.2. Vorgehensweise

Am Beginn der Informationsextraktion steht hier die Regel, die das Muster der zu extrahierenden Information beschreibt. Ein einfacher Ansatz zur Identifizierung von Eigennamen wäre etwa, alle Wörter mit großem Anfangsbuchstaben als Namen zu bewerten (Appelt & Israel 1999).

Zu einer auf die so erfolgte Identifizierung aufbauenden Kategorisierung – etwa in die Kategorie 'Firma' – könnte die Übereinstimmung einer Wortgruppe mit dem durch folgenden regulären Ausdruck beschriebenen Muster überprüft werden:

```
name + (''ag''|''gmbh''|''gbr''|''& söhne''|''& co'')
```

Auf diese Weise würde ein oder mehrere Namen, gefolgt von einer der genannten Zeichenketten als Name vom Typ 'Firma' erkannt werden. Dieses Beispiel deckt natürlich nicht alle Formen von Firmennamen ab, es soll lediglich zur Verdeutlichung einer grundsätzlichen Herangehensweise dienen. Als weiterer Ausbau könnte etwa eine Liste geläufiger Abkürzungen von Firmennamen eingebaut werden, um Kurzformen wie 'IBM' ebenfalls abzudecken, sowie ein Mechanismus, der dafür sorgt, daß Koreferenzen – etwa von 'IBM' und 'Big Blue' – erkannt werden. Eine ausführlichere Darstellung solcher grundsätzlicher Überlegungen zur Eigennamenerkennung und -klassifikation gibt Grishman (2003).

1.3. Informationsextraktion mit strukturierter Ausgabe

1.3.1. Ziele

Die Entwicklung auf dem noch recht jungen Forschungsgebiet der Informationsextraktion wurde maßgeblich durch die 'Message Understanding Conferences' (MUC) vorangetrieben. Die sieben MUC wurden von 1987 bis 1997 von der 'Defense Advanced Research Projects Agency' (DARPA) – der zentralen Forschungs- und Entwicklungseinrichtung des US-amerikanischen Verteidigungsministeriums – veranstaltet. Vorgegebene Szenarios waren Nachrichten über nautische Operationen (MUC-1 1987 und MUC-2 1989), über terroristische Aktivitäten (MUC-3 1991 und MUC-4

2 Hier und in nachfolgenden Beispielen in der 'Standard Generalized Markup Language' (SGML)

1992), Joint Ventures und Mikroelektronik (MUC-5 1993), Personalwechsel in der Wirtschaft (MUC-6 1995), sowie über Raumfahrzeuge und Raketenstarts (MUC-7 1997) (Appelt & Israel 1999). Da zur gemeinsamen Evaluation ein standardisiertes Ausgabeformat notwendig war, verwendete man ab der zweiten MUC eine gemeinsame Ausgabeschablone (Template), weshalb nahezu alle[3] Informationsextraktionssysteme eine strukturierte Ausgabe der extrahierten Informationen leisten. Ein zu analysierender Text mit gesuchten Informationen sowie die daraus erstellte Schablone könnten etwa wie folgt aussehen:

> *Der Präsident nimmt es in diesen Januartagen auf sich, die Nation auf Krieg einzustimmen. Diesmal ist er in die weite Ödnis seines geliebten Texas gereist, zum 3. Gepanzerten Korps, das sich selbst 'America's Hammer' nennt.*
>
> (DIE ZEIT, 16. Januar 2003)

Domäne	Politische Nachrichten
Szenario	Besuch
Besucher	Der Präsident
Besuchter	3. Gepanzertes Korps *America's Hammer*
Zeit	Januar
Ort	Texas

1.3.2. Vorgehensweise

Bei heutigen System werden meist Regeln zum Füllen eines einzelnen Feldes der Ausgabeschablone verwendet[4], allerdings wird auch an Systemen gearbeitet, die ganze Schablonen füllen (Neumann 2001). Die Regel entscheidet, welche Textpassagen extrahiert werden und – wenn die Regel eine komplette Schablone füllt – auf welches Feld der Schablone sie gehören. Bevor Regeln zum Füllen der Felder einer Schablone festgelegt werden können, muß zunächst das Format (d.h. die Felder der Ausgabeschablone) bekannt sein, schließlich sollen die Regeln festlegen, welches Feld mit welchem Teil des zu untersuchenden Textes gefüllt werden soll. Das Vorgehen gliedert sich also in drei Schritte: Am Anfang steht die Festlegung der Ausgabeschablone, dann müssen entsprechende Regeln formalisiert werden, die schließlich zum Füllen der Schablone mit Werten aus analysiertem Text verwendet werden. Zu den benötigten Regeln kann man auf verschiedene Arten gelangen.

1.3.3. Regelerstellung

Manuelle Regelerstellung

Die Muster der gesuchten Informationen können manuell entdeckt und formalisiert werden. Ein System, das für das Szenario "Besuch" etwa *Der Präsident besucht die Truppen* als gesuchte Information erkennt und daraus die Ausgabeschablone

3 Eine Ausnahme hierzu bildet Euler (2001b,a, 2002), s. dazu Abschnitt 1.4.

4 Dies entspricht dem Vorgehen bei einer Eigennamenerkennung (s. Abschnitt 1.2), denn jede Regel beschreibt eine gesuchte Art von Information.

Besucher	Der Präsident
Besuchter	die Truppen

erstellt, könnte dies im einfachsten Fall mithilfe einer Regel, wie sie durch den regulären Ausdruck `name ''besucht'' name` repräsentiert wird. Zur Identifikation eines Namens könnte zunächst eine Eigennamenerkennung (s. Abschnitt 1.2) durchgeführt werden. Diese Regel beschreibt allerdings nur einen winzigen Ausschnitt aus der Menge von Möglichkeiten, einen Besuch zu beschreiben – so werden hier nicht einmal verschiedene Formen des Verbs *besuchen* berücksichtigt. Das weitere Vorgehen bestünde nun prinzipiell in einer sukzessiven Verbesserung der Regel durch wiederholtes Testen und Evaluation der Ergebnisse.

Die Entdeckung der Muster ist in diesem Beispiel für jeden Entwickler eines Informationsextraktionssystems möglich, werden die Szenarios allerdings fachlich komplexer, muß der Entwickler Zugang zum benötigten Fachwissen bekommen, etwa durch Lexika oder Berater.

Automatische Regelerstellung aus annotiertem Text

Die Regeln können automatisch aus mit Anmerkungen versehenem (annotiertem) Text erstellt werden. Dabei werden dem System die gewünschte Ausgabeschablone sowie Text, in dem Wörter entsprechend den Werten der Ausgabeschablone markiert wurden, vorgegeben:

Besucher	
Besuchter	

```
Der <Besucher> Präsident </Besucher> besucht die <Besuchter> Truppen
</Besuchter>.
```

Die Ausgangsfrage für das automatische Lernen einer Regel zum Erkennen eines solchen Szenarios lautet nun: Welches Muster findet man an den Textstellen, die die relevanten Informationen (Besucher und Besuchter) ausmachen, an dem später in nicht-annotierten Texten das Szenario wiedererkannt werden kann? Der Umfang an Daten, in denen nach einem solchen Muster gesucht werden kann hängt von der Tiefe der in einem nächsten Schritt zu leistenden Analyse der Textpassage ab. Angenommen, das System verfügt über Möglichkeiten zur Analyse auf morphologischer, syntaktischer und semantischer Ebene, wäre etwa folgendes Analyseergebnis des Satzes denkbar:

```
<S>
 <NP Kasus=Nominativ>
  <Det> Der </Det>
  <N Numerus=Singular Typ=Person> Präsident </N>
 </NP>
 <VP>
  <V Numerus=Singular Stamm=''besuch''> besucht </V>
  <NP Kasus=Akkusativ>
   <Det> die </Det>
   <N Typ=Person> Truppen </N>
  </NP>
 </VP>
</S>
```

Modelliert man die Konstituenten des Satzes als Objekte mit Zugriffmöglichkeiten auf die zugehörigen Daten, wäre folgende (in Java formulierte) Funktion zum Extrahieren der gesuchten Information denkbar:

```
void extrahiereInformation(Satz satz, Schablone ergebnis){
  if(satz.np.istNominativ()
  && satz.vp.v.hatStamm(``besuch'')
  && satz.vp.np.istAkkusativ()
  && satz.vp.np.n.istPerson()){
    ergebnis.bestimmeAlsBesucher(satz.np);
    ergebnis.bestimmeAlsBesuchter(satz.vp.np);
  }
}
```

Die Funktion überprüft, ob der zu analysierende Satz den Kriterien aus dem annotierten Korpus entspricht, d.h. ob die Regel hier angewendet werden kann. Ist dies der Fall, werden die entsprechenden Konstituenten des Satzes den entsprechenden Feldern der Schablone zugeordnet. Wirkliche Beschreibungen von Vorgängen wie einem Besuch, etwa das Beispiel aus Abschnitt 1.3.1 (*Der Präsident nimmt es in diesen Januartagen auf sich, die Nation auf Krieg einzustimmen. Diesmal ist er in die weite Ödnis seines geliebten Texas gereist, zum 3. Gepanzerten Korps, das sich selbst 'America's Hammer' nennt.),* sind ungleich komplexer und schwieriger zu modellieren (vgl. Abschnitt 1.3.4).

Automatische Regelerstellung aus nicht-annotiertem Text

Bei Ansätzen, automatisch aus nicht-annotiertem Text Regeln zu erstellen (s. Grishman *et al.* 2000), werden dem System zwei oder drei Regeln vorgegeben. Nun sucht das System in einem Trainingskorpus nach Dokumenten, in denen Muster, die diesen vorgegebenen Regeln entsprechen, besonders häufig vorkommen[5]. In der Annahme, daß diese Dokumente insgesamt relevante Informationen enthalten, werden weitere in diesen Dokumenten enthaltenen Muster extrahiert und in Form von Regeln vom System verwendet.

Vor- und Nachteile von automatischer und manueller Regelerstellung

Sowohl die manuelle als auch die automatische Regelerstellung haben Vor- und Nachteile: Die manuelle Erstellung ist aufgrund der sukzessiven Anpassung der Regeln an das Szenario sehr zeit- und arbeitsaufwändig – dafür bietet sich die Möglichkeit, ein sehr gut angepaßtes System zu entwickeln und damit ein hohes F-Maß zu erreichen. Außerdem lassen sich Regeln für Szenarios entwickeln, für die es keine Korpora gibt, oder diese (etwa aus Kostengründen) nicht zur Verfügung stehen. Allerdings sind solche Systeme auf das Szenario beschränkt, für das sie entwickelt wurden und eine Anpassung an veränderte Erfordernisse kann unmöglich sein: Appelt & Israel (1999) nennen etwa ein System zur Eigennamenerkennung, das so realisiert wurde, daß es Wörter in Großschreibung als Eigennamen einstuft. Hier kann eine Implementierung von Texten in Kleinschrift eine komplette Neuentwicklung erfordern, während ein System, das seine Regeln automatisch lernt, einfach

5 Dies ist eine Volltextsuche. Hier zeigt sich, wie eng die Informationsextraktion mit ihren Nachbargebieten verbunden ist, ob wie hier diese als Hilfsmittel einsetzend oder als Vorverarbeitung für diese, wie in Abschnitt 1.1.3 erwähnt.

einen Korpus in Kleinschrift bekommen könnte. Automatische Regellernsysteme haben also den Vorteil, auf neue Domänen oder Szenarios portierbar zu sein, allerdings kann der Annotierungsaufwand je nach gewähltem Verfahren sehr hoch sein, etwa wenn nicht länger nur Namen von Politikern sondern auch von politischen Institutionen von einem System zur Namensklassifikation als zum Typ 'politisch' zugehörig erkannt werden sollen, und so im kompletten Trainingskorpus die Anmerkungen erweitert werden müßten. Der Aufwand kann je nach neuem Einsatzgebiet bei manueller Regelerstellung viel geringer sein, da hier die Regel direkt bearbeitet werden kann. Zur Qualität der Ergebnisse bemerkt Grishman (2003), daß automatische Lernverfahren bei der Verarbeitung von Texten mit regelmäßiger, sich wiederholender Struktur gute Ergebnisse liefern, in Bereichen mit größerer linguistischer Variation aber hinter Systemen mit manuell erstellten Regeln zurückbleiben.

1.3.4. Regelbasierte Extraktion

Die grundsätzliche Vorgehensweise bei der eigentlichen Extraktion besteht darin, die zu bewertende Textstelle linguistisch so tief zu analysieren, daß die verfügbare Regel anwendbar ist. Um den Satz *Der Präsident besucht die Truppen* mit der in Abschnitt 1.3.3 beschriebenen Regel als relevant erkennen zu können, müßte die linguistische Analyse etwa die Erkennung von Phrasen, eine Stammformenreduktion und die Ermittlung des Numerus umfassen. Zudem müßte ein Mechanismus zur Ermittlung der semantischen Kategorien von *Präsident* und *Truppen* gegeben sein, etwa durch Zugriff auf ein Lexikon oder eine vorherige Eigennamenerkennung und -klassifikation. Handelt es sich dagegen um ein reales Beispiel – wie *Der Präsident nimmt es in diesen Januartagen auf sich, die Nation auf Krieg einzustimmen. Diesmal ist er in die weite Ödnis seines geliebten Texas gereist, zum 3. Gepanzerten Korps, das sich selbst 'America's Hammer' nennt.* – ist jedoch weiteres nötig. Hier sind vor allem zwei Probleme zu nennen: Zum einen die Auflösung anaphorischer Ausdrücke[6] – hier etwa pronominale Koreferenzen (*Der Präsident, er*) und Eigennamen-Koreferenzen (*3. Gepanzertes Korps, America's Hammer*) – zum anderen das Zusammenführen partieller Instanzen, wie sie hier bei einer Analyse einzelner Sätze entstehen würden:

Szenario	
Besucher	Der Präsident
Besuchter	
Zeit	Januar
Ort	

Szenario	Besuch
Besucher	er
Besuchter	3. Gepanzertes Korps America's Hammer
Zeit	
Ort	Texas

Nach der Analyse des ersten Satzes wäre die Information unvollständig, während die Schablone für den zweiten Satz im Feld 'Besucher' lediglich einen referenzierenden Wert hat. Dies sind nur einige der Schwierigkeiten auf dem Gebiet der Informationsextraktion, weitere ergeben sich etwa aus morphologischen Herausforderungen (z.B. die Zerlegung von Komposita, wie sie hier nötig wäre, um aus *Januartagen* im ersten Satz den Zeitpunkt 'Januar' zu erkennen) und auf dem Gebiet der Pragmatik, z.B. lediglich indirekt geäußerte Informationen zu erkennen – etwa daß im Bereich 'Terminabsprachen' *Mir ist etwas dazwischen gekommen* eine Absage bedeutet.

6 Die Auflösung anaphorischer Ausdrücke ist ein Thema für sich, s. hierzu Mitkov (2003).

1.3.5. Beispiele und Evaluation

Gemeinsamkeiten der Systeme

Die im Rahmen der MUC entwickelten und evaluierten Systemen haben neben der Ausgabeschablone auch die grundsätzliche Anforderung gemeinsam, große Textmengen schnell verarbeiten zu können. Systeme, die hohe F-Werte liefern, aber lange für die Analyse brauchen, waren im Rahmen der Domänen sämtlicher MUC (verschiedenen Arten von Nachrichten) unbrauchbar. Da selbst im Rahmen der heutigen, unvollständigen Kenntnisse der Funktionsweise natürlicher Sprache eine komplette linguistische Analyse hier zu aufwändig wäre, gehen Informationsextraktionssysteme üblicherweise einen Kompromiß bezüglich der Komplexität der Analyse zugunsten der Verarbeitungsgeschwindigkeit ein. So hat etwa die Verwendung von endlichen Automaten hier eine wahre "Renaissance" (Neumann 2001) erfahren. So verwenden etwa die Systeme FASTUS[7] und ANNIE[8] kaskadierte endliche Automaten zur Analyse. Viele Systeme haben einen modularen Aufbau, bei dem einzelne, domänenunabhängige linguistische Analyseschritte voneinander – und diese von den domänenspezifischen Modulen – getrennt sind. FASTUS etwa bestand 1993 aus Modulen zur Erkennung von komplexen Wörtern und Eigennamen, von einfachen und komplexen Phrasen, zur Erkennung der gesuchten Ereignisse und zum Zusammenführen von partiellen Ergebnissen (Appelt *et al.* 1993). Im Gegensatz dazu verfügt ANNIE etwa über Module zum Zerlegen des Textes in Wörter (Tokenizer), zum Zurückführen von Wörtern auf ihre Grundformen (Lemmatisierer) und zur Wortartenbestimmung mittels eines Part-of-Speech-Taggers sowie eines semantischen Taggers.[9] Dies zeigt andeutungsweise, wie stark der Umfang der linguistischen Analyse von System zu System variiert.

Evaluation

Im Bereich der Eigennamenerkennung wurden im Rahmen der MUC Ergebnisse von F-Werten bis zu 90% erreicht. Bei den evaluierten Szenarios, die Ereignisse beschreiben, wurden dagegen nie Werte jenseits der 60% erreicht (Appelt & Israel 1999; Grishman 2003). Dies könnte darauf hindeuten, daß eine seichte Sprachanalyse, wie sie aus oben beschriebenen Gründen verwendet wurde, nicht ausreicht, um solch komplexe sprachliche Konstrukte zu erkennen, allerdings schreibt Grishman, daß auch die Verwendung von "more powerful analysis techniques – what could be described as a 'deep understanding' model" (Grishman 2003) lediglich an ausgewählten Beispielen gute Leitungen bringt, nicht aber bei der allgemeinen Anwendung.

1.4. Informationsextraktion durch gezielte Zusammenfassung

1.4.1. Ziele

Ist eine strukturierte Ausgabe nicht erforderlich, ist ein anderer Ansatz möglich: Die Informationsextraktion durch gezielte Zusammenfassung (s. Euler 2001a). Hierbei würde aus dem Beispiel in

7 Eine veränderte Abkürzung für 'Finite State Automaton Text Understanding System', entwickelt von SRI International.
8 Eine Komponente der 'General Architecture for Text Engineering' (GATE), die an der University of Sheffield entwickelt wurde.
9 Für weitere Informationen zu den Modulen von ANNIE siehe Cunningham *et al.* (2003).

Abschnitt 1.3.1 statt der Schablone ein gekürzter Text, etwa in folgender Form: "Präsident [...] Texas gereist [...] zum 3. Gepanzerten Korps [...]". Denkbar wäre auch, daß lediglich die zwei Sätze, aus denen der Beispieltext besteht, aus dem kompletten Artikel extrahiert werden. Eine solche Ausgabe, versehen mit Hinweisen über Art und Umfang der Auslassungen, kann für bestimmte Bereiche (s. Abschnitte 1.1.3 und für eine konkrete Anwendung 1.4.5) ausreichend sein.

1.4.2. Die semantische Richtschnur: Wortlisten

Ausgangspunkt für die Informationsextraktion durch gezielte Zusammenfassung stellen Wortlisten dar, in denen allen Wörtern ein Gewicht entsprechend Ihrer Bedeutung für das Szenario, das erkannt werden soll, zugeordnet ist. Solche Wortlisten können etwa aus lexikalisch-semantischen Netzen gewonnen werden oder – ähnlich den Regeln bei der Extraktion mit strukturierter Ausgabe – automatisch aus annotiertem Text. Eine solche Liste stellt die semantische Richtschnur dar, die von der Auswahl der relevanten Sätze bis zu einer eventuellen Kürzung dieser eine "gewisse semantische Orientierung" (Euler 2001a) bietet.

Zum automatischen Erstellen von Wortlisten müssen im Lernkorpus relevante Informationen gekennzeichnet werden, allerdings reicht es hier bereits, ganze Sätze oder sogar Dokumente[10] zu kennzeichnen, da die Struktur der zu extrahierenden Informationen – etwa welches Wort bei einem Besuch den Besucher und welches den Besuchten bezeichnet – für das Vorgehen des Systems nicht relevant ist. Wollte man etwa Informationen über Raketenstarts (das Szenario von MUC-7) extrahieren, könnte ein Ausschnitt aus dem annotierten Korpus so aussehen:

> *Ein weiterer Fehlstart hätte möglicherweise sogar das Ende für die Fertigung großer Trägerraketen in Westeuropa bedeuten können. <Raketenstart>Die insgesamt 15. Ariane-5 verließ die Startrampe 3A in Kourou um 0:52 MEZ und verschwand Sekunden später in einer dichten Wolkenschicht. </Raketenstart>Eine gute halbe Stunde später hatte die Ariane-5 ihren Zielorbit erreicht...*
>
> (Aus einer Meldung auf `http://www.vfr.de/`)

Anhand eines in dieser Art gekennzeichneten Korpus erstellt das System eine Rangliste der wichtigsten[11] Wörter für das Szenario. Euler (2001a) hat vor der Gewichtung der einzelnen Wörter eine Stammformenreduktion durchgeführt, ohne diese habe er "keine guten Ergebnisse erzielt". An dieser Stelle wäre auch eine weitergehende linguistische Vorverarbeitung denkbar, etwa die schon in Abschnitt 1.3.4 erwähnte Zerlegung von Komposita. Die am höchsten bewerteten Wörter im Korpus (bei Euler 10%) bilden nun die für das Szenario relevante Wortliste.

1.4.3. Extraktion durch gezielte Zusammenfassung

Ermittlung relevanter Sätze

Bei der eigentlichen Extraktion bekommt jedes Wort im zu untersuchenden Text den Wert, der ihm in der Liste zugeordnet ist, bzw. keinen Wert, wenn es nicht in der Liste steht. Im nächsten Schritt

10 Zu Ergebnissen mit Korpora, in denen Sätze oder ganze Dokumente gekennzeichnet wurden siehe Abschnitt 1.4.5.

11 Euler erreichte die besten Ergebnisse über die Worthäufigkeit. Vergleichsverfahren waren Information Gain, G^2-Statistik und SVM-Gewichte (s. Euler 2001a).

werden die Werte der Wörter eines Satzes addiert und dieser Wert durch die Anzahl der Wörter im Satz geteilt, um unterschiedlich lange Sätze gleich zu behandeln. Nun kann durch Versuche ein Schwellenwert bestimmt werden, den ein Satz erreichen muß, um als Treffer zu gelten bzw. für eine weitere Kürzung interessant zu sein.

Satzkürzungen

Ist eine weitere Kürzung der Ergebnisse gewünscht, können die Sätze in mehreren Stufen gekürzt werden. In einer ersten Stufe werden "übliche Abkürzungen" (Euler 2001a) eingeführt sowie Grußformeln, Anreden und "meistens inhaltsleere Wörter wie *naja* und *überhaupt*" (ebd.) gestrichen. Auf der nächsten Stufe werden Artikel, Adverbien und Adjektive gestrichen, auf der Stufe mit der maximalen Kürzung schließlich ganze Phrasen, außer Verbalphrasen. Phrasen, die Wörter aus der Liste enthalten sowie Wörter, die in der Liste enthalten sind, werden nie gestrichen, wodurch die Wortliste auch hier eine Art semantische Richtschnur bildet. Die beschriebenen Satzkürzungen erfordern zumindest teilweise eine linguistische Analyse der zu kürzenden Sätze, etwa zur Phrasenstrukturermittlung und Wortartenbestimmung. Hierzu verwendete Euler das sprachverarbeitende System MESON des Deutschen Forschungszentrums für Künstliche Intelligenz (DFKI).

1.4.4. Vor- und Nachteile der Extraktion durch gezielte Zusammenfassung

Der wichtigste Unterschied zum Verfahren mit strukturierter Ausgabe in Form von Ausgabeschablonen ist die Tatsache, daß die Ergebnisse nicht strukturiert sind. Wenn die Ergebnisse sofort von einem Menschen verarbeitet werden sollen ist dies aber nicht notwendigerweise ein Nachteil, sondern kann den Anforderungen angemessen sein. Bei den Ergebnissen der gezielten Zusammenfassung wird die Verknüpfung der Elemente eines Vorganges – etwa von *Ariane-5, Kourou* und *verließ* zu dem Vorgang "Raketenstart" – vom Menschen vorgenommen. Der menschliche Rezipient erkennt hier aufgrund seines Hintergrundwissens, daß *Ariane-5* die Rakete ist – das System speichert lediglich, daß diese Wörter etwas mit einem Raketenstart zu tun haben. Die Vorteile, die durch diese Einschränkung gewonnen werden, sind der geringe Annotierungsaufwand beim Erstellen des Korpus (und daher zumindest theoretisch eine hohe Portabilität auf neue Szenarios) sowie die Möglichkeit der Anpassung von Vollständigkeit und Präzision an die konkreten Erfordernisse – auf Ebene der Satzauswahl durch den Schwellenwert, ab dem ein Satz als Treffer ausgezeichnet wird und auf Ebene der Satzkürzung durch die gewählte Kürzungsstufe.

1.4.5. Beispiel und Evaluation

Euler hat das beschriebene Verfahren anhand eines E-Mail-to-SMS-Service getestet. Der Dienst soll E-Mails, die Terminabsprachen (Ankündigungen, Verschiebungen, Ab- und Zusagen) enthalten, kürzen und sie als SMS an ein Mobiltelefon verschicken. Der verwendete Korpus umfaßte 560 deutschsprachige E-Mails mit insgesamt ca. 45000 Wörtern. Der Korpus bestand zur Hälfte aus E-Mails, die Informationen vom gesuchten Szenario 'Terminabsprache' enthielten und zur anderen Hälfte aus beliebigen E-Mails. "Etwa 13%" (Euler 2001a) der Sätze des Korpus wurden als terminbezogen gekennzeichnet. Für das Training wurden 90% des Korpus verwendet, zum Testen die übrigen 10%. Beim reinen Satzfiltern ohne weitere Kürzungen erreichte Euler F-Werte bis zu 81,2% mit der beschriebenen Markierung relevanter Sätze im Trainingskorpus und bis zu 76,5%, wenn

relevante Dokumente (d.h. E-Mails) gekennzeichnet wurden. Eine Evaluation der in Abschnitt 1.4.3 beschriebenen Satzkürzungen durch Befragung der Empfänger der SMS-Nachrichten ergab, daß Informationen wie der Zeitpunkt des Termins und sein Status, d.h. ob er "ausfällt oder verschoben wird etc." "am besten erhalten" (Euler 2001a) bleiben. Die Art des Termins gehe oft verloren, wäre aber für einen Empfänger mit Hintergrundwissen rekonstruierbar gewesen[12] (Euler 2001a). Euler kommt bezüglich der Kürzungsstufen zu folgendem Fazit: "Bei mittlerer Kürzung, die nicht mehr die Phrasenentfernung durchführt, sollte in den meisten Fällen Halt gemacht werden" (Euler 2001a). Später hat er die Stufen der Satzkürzung weiter ausdifferenziert (Euler 2002). Die guten Ergebnisse nach F-Maß selbst beim Kennzeichnen ganzer Dokumente im Trainingskorpus lassen einfach handhabbare und leistungsfähige Systeme denkbar erscheinen. So könnte im E-Mail-Client jede E-Mail auf Knopfdruck einer bestimmten Kategorie zugeordnet werden und so für verschiedenen Bereiche Wortlisten mit geringem Aufwand erstellt werden. Denkbar ist für Euler (2001a) sogar eine ganz allgemeine Kennzeichnung der Nachrichten in 'interessant' und 'uninteressant' und somit ein auf das Interesse des Benutzers abgestimmtes System. Allerdings merkt Euler an, daß das Verfahren "noch für andere Domänen und andere Sprachen getestet werden" (Euler 2001a) muß.

1.5. Schlußbetrachtung

In der Informationsextraktion bietet sich auf mehreren Ebenen ein vielfältiges Bild. Informationsextraktionssysteme können für verschiedene Aufgabenbereiche von der automatischen Analyse von Stellenanzeigen bis zur Vorbereitung einer allgemeinen Textzusammenfassung eingesetzt werden. Entsprechend diesen Anforderungen können die Systeme strukturierte oder unstrukturierte Ergebnisse liefern. Weiter können die Systeme völlig unterschiedliche linguistische Tiefe aufweisen, von der Extraktion durch gezielte Zusammenfassung mit reiner Satzfilterung, wo lediglich semantische Orientierung in Form der Wortliste gegeben ist, bis hin zu Systemen mit Analysemodulen für sämtliche Ebenen der Sprache. In einigen Bereichen führt unser mangelndes Verständnis für die Funktionsweise natürlicher Sprache zu einer Stagnation der Entwicklung, doch da Informationsextraktion eine eingeschränktere Aufgabe als ein komplettes Textverständnis darstellt, sind vielfach im Sinne eines "appropriate language engineering" (Grishman 2003) den Anforderungen angemessene Lösungen (vielleicht auch gerade in Verbindung mit den Nachbargebieten) möglich. Als Beispiel hierfür möge das von Euler (2001b,a, 2002) entworfene Verfahren dienen, das im Unterschied zu den die IE dominierenden Systemen lediglich unstrukturierte Ergebnisse liefert. Dafür erreicht es hohe Leistung nach F-Maß und verlangt lediglich einen geringen oder gar minimalen Annotierungsaufwand des Trainingskorpus, was eine hohe Portabilität auf neue Domänen und Szenarios bedeuten könnte, etwa in Form einer Erstellung von Wortlisten *en passant* bei einer Textklassifikation.

12 Zur Rolle des Hintergrundwissens vgl. Abschnitt 1.4.4.

Kapitel 2

Theorien und Modelle

Simon C. Dik: Functional Grammar, Allgemeine Sprachwissenschaft, Proseminar *Theorien und Modelle* bei Prof. Hans-Jürgen Sasse, Sommersemester 2005.

2.1. Functional Grammar

"Functional Grammar" (FG) wurde Ende der 1970er Jahre von Simon C. Dik in Amsterdam entwickelt – ausdrücklich als Gegenmodell zum Standard-Modell der Tranformationsgrammatik von Noam Chomsky. Es ist das einzige vollständige solche Gegenmodell, das ausserhalb des MIT (wo Chomsky das Standardmodell entwickelt hatte) entstanden ist. Nach dem Tod Diks 1995 wurde die Theorie vor allem durch seinen Mitarbeiter Kees Hengeveld weiterentwickelt und ist in ihrer heutigen Form der ursprünglichen Formulierung noch sehr nah.

2.2. Allgemeine Annahmen über Sprache

Die zentrale Annahme Diks über Sprache ist ihr zweckgebundener Charakter als Mittel zur Kommunikation. Dik rückt damit die Funktion der Sprache in den Mittelpunkt. In diesem Sinne ist die Bezeichnung "Functional Grammar" zu sehen: Ein sprachliches Modell, das von der Funktion der Sprache, statt von ihrer äusseren Form ausgeht. Mit dieser zentralen Annahme fordert Dik eine Abkehr von der früher häufig angewandten heuristischen Reduktion der Ausblendung der Pragmatik. Konkret ist jedoch in der Behandlung der Pragmatik bei Dik nicht die allgemeine Pragmatik im Sinne von Sprechakten und Sprache als Handlung gemeint, sondern der Bereich der Diskurspragmatik, im Wesentlichen also das Verhältnis der Informationsstruktur eines sprachlichen Ausdrucks zu seiner Realisierung, etwa bei der Behandlung von Topik und Fokus (Dik 1991, 267ff.).

Der von Dik beschriebene Grammatikformalismus ist in diesem Sinne pragmatikbasiert. Die nächst wichtigste sprachliche Ebene ist aus Diks Sicht die Semantik, die – selbst von der Pragmatik beeinflusst – ihrerseits Einfluss auf die Syntax hat. Ein Beispiel für eine solche Beeinflussung der Syntax wäre etwa eine Aktiv-Passiv-Alternation, die von den semantischen Rollen der Mitspieler in der Äusserung bestimmt wird, und in diesem Sinne semantisch motiviert ist.

Zugleich ist FG ein formales Modell, da es Methoden der formalen Semantik – etwa Prädikatenlogik

– verwendet und den Anspruch der Implementierbarkeit als Computerprogramm und damit der Testbarkeit erhebt.

2.3. Vorstellungen über Grammatik

2.3.1. Grundlegender Aufbau des Grammatikformalismus

Der Grammatik-Formalismus der FG besteht im wesentlichen aus der Beschreibung abstrakter Ausdrücke, der *Underlying Clause Structures* (UCS), die schrittweise aus Prädikaten und Termen gebildet werden und die durch Ausdrucksregeln zu konkreten sprachlichen Äusserungen in Bezug gesetzt werden oder diese erzeugen.

2.3.2. Aufbau der Underlying Clause Structure

Prädikate

Die UCS werden aus Prädikaten und Termen gebildet. Einige elementare Prädikate und Terme sind Teil des Lexikons, andere werden aus diesen elementaren Prädikaten und Termen erstellt (durch *predicate formation* und *term formation*, siehe Abbildung 4.1.3 auf Seite 55). So wäre das Prädikat für *throw back* ein aus den elementaren Prädikaten für *throw* und *back* abgeleitetes Prädikat. Alle Prädikate und Terme zusammen bilden den Fundus (*fund*) einer Sprache.

Prädikate sind Ausdrücke für Eigenschaften oder Relationen. Es handelt sich hier um Prädikate im Sinne der Prädikatenlogik, nicht um die grammatische Relation des Prädikates aus der lateinischen Schulgrammatik. In diesem Sinne sind nicht nur Verben Prädikate, sondern alle Inhaltswörter einer Sprache. So ist “haus(x)” etwa ebenso ein Prädikat wie “schlagen(x,y)”.

Ein Unterschied der Prädikate in der FG zur klassischen Prädikatenlogik ist die Verwendung sogenannter Restriktoren. Wenn in der FG Prädikate zusammengesetzt werden, geschieht dies durch die Verwendung dieser Restriktoren, geschrieben als “:”, etwa in der Form “japanisch(x):buddhistisch(x)”. Dies lässt sich paraphrasieren mit “Die Menge der x, für die gilt: x ist japanisch, eingeschränkt auf die Menge der x, für die gilt: x ist buddhistisch”. Die entsprechende prädikatenlogische Form wäre “japanisch(x) & buddhistisch(x)”, wobei das “&” ein prädikatenlogisches “UND” ist. Der entsprechende Sachverhalt ist in beiden Fällen gleich. Der Unterschied besteht darin, dass das prädikatenlogische “&” umkehrbar ist, dass also “japanisch(x) & buddhistisch(x)” aquivalent ist zu “buddhistisch(x) & japanisch(x)”. Bei den Restriktoren ist dies nicht der Fall und sie sind damit in der Lage, den Unterschied der sprachlichen Äusserungen *Der japanische Buddhist* und *Der buddhistische Japaner* zu erfassen (Dik 1997, Kap. 6.2).

Prädikate sind stets Teil eines Prädikatrahmens, der die Eigenschaften des Prädikats beschreibt. Ein Beispiel für den Prädikatrahmen eines transitiven Verbs wäre etwa:

$$\text{throw[V](x1:<animate>(x1))}_{Agent}\ \text{(x2:<concrete>(x2))}_{Goal}\ \text{(x3)}_{Direction}$$

Zunächst erscheint die Wortform (throw), anschließend die Wortart (V). Im Folgenden sind die Argumentpositionen des Verbs beschrieben. Das Argument in der Mitspielerposition mit der semantischen Rolle des Agens muss belebt (animate) sein, der vom Sachverhalt betroffene Mitspieler

(Goal[1]) – hier der geworfene Gegenstand – muss konkret (concrete) sein und das dritte Argument (mit der semantische Rolle Location) unterliegt keiner solchen Selektionsbeschränkung.

Zu solchen nuklearen Prädikaten können nun die fakultativen sogenannten Satelliten hinzukommen, die Positionen einnehmen, die nicht vom Prädikatrahmen spezifiziert werden, etwa zu einer zeitlichen Präzisierung des Prädikats mit Hilfe von Wörtern wie *gestern* oder *bald*. Einen solchen um Satelliten erweiterten Prädikatrahmen nennt Dik einen erweiterten Prädikatrahmen (*extended predicate frame*). Dies ist im Kasten "Predicate-Frames" in Abbildung 4.1.3 auf Seite 55 schematisch dargestellt.

Terme

Der zweite wesentliche Bestandteil einer UCS sind neben Prädikaten die Terme. Formal sind Terme die Argumente der Prädikate, semantisch sind es Ausdrücke, die Entitäten referenzieren[2]. Beispiele für Terme wären *Das Haus* oder *Die lila Plastiktüte*. Es existieren nur sehr wenige elementare Terme, so sind lediglich Eigennamen und Personalpronomina als elementare Terme vorhanden, andere Terme, wie *Die lila Plastiktüte* werden aus Prädikaten erstellt. Terme sind also die Entitäten, die durch ein Prädikat zueinander in Beziehung gesetzt werden.

Eine Prädikation, die zwei Terme (*the garden* und *the dog*) enthält wäre z.B.:

present: (definite singular x1: garden [N])$_{Location}$ (definite singular x2: dog [N])

Ebenen in der UCS

Innerhalb der UCS werden in Form von Funktionen drei verschiedenen Ebenen unterschieden:

1. Äusserungssituation: Ebene der pragmatischen Funktionen wie Topik und Fokus.
2. Mitspielerebene: Ebene der semantischen Funktionen wie Agent und Goal (Patiens).
3. Ebene der Perspektive: Ebene der syntaktischen Funktionen Subjekt und Objekt.

In diesem Sinne nehmen einzelne Elemente einer sprachlichen Äusserung auf den verschiedenen Ebenen verschiedene Kategorien an. In dem Satz *Peter kauft ein Eis* etwa ist *Peter* zugleich Agens, Topik und Subjekt, während *Eis* zugleich Goal (Patiens), Fokus und Objekt ist.

Der sprachliche Ausdruck, der durch die Ausdrucksregeln auf die UCS

present: (definite singular x1: garden [N])$_{Location}$ (definite singular x2: dog [N])

bezogen (oder in einer Implementierung des Formalismus auch aus der UCS erzeugt) werden kann ist so noch nicht eindeutig. Der UCS entspricht etwa die Äusserung *The dog is in the garden*. In einer bestimmten Äusserungssituation – etwa in einer Aufzählung der für einen Einbruch zu überwindenden Hindernisse – wäre aber folgende Äusserung denkbar, die ebenfalls mit der

1 Allgemein hat sich für die Rolle, die Dik *Goal* nennt die Bezeichnung *Patient* bzw. *Patiens* durchgesetzt.

2 Genau genommen schreibt (Dik 1991, 255), dass Terme den Adressaten instruieren, eine Entität zu identifizieren, die dem Profil des Terms entspricht.

UCS übereinstimmt: *There is the dog in the garden*. Dieses Beispiel verdeutlicht die Möglichkeiten, die eine Berücksichtigung der pragmatischen Ebene bietet, denn durch die Kennzeichnung des Aufzählungscharakters ist es möglich, die beiden sprachlichen Äusserungen in der zugrunde liegenden Struktur zu unterscheiden (Dik 1997, Kap. 8.7.2).

Die Unterscheidung der Ebene der semantischen Rollen, d.h. der Mitspieler und der syntaktischen (grammatischen) Relationen ermöglicht die Beschreibung syntaktischer Alternationen wie der Passivierung ohne dass dabei die eine Form aus der anderen abgeleitet werden müsste. So gibt es in einem Aktivsatz eine Übereinstimmung von Subjekt und Agens, während bei einer Übereinstimmung von Subjekt und Goal (Patiens) in der UCS dieser ein Passivsatz entspräche.

Operatoren auf Prädikationen

Wenn wie beschrieben die Terme in die Prädikatrahmen eingesetz wurden, haben wir eine Prädikation, die die vollständige Proposition oder den Sachverhalt (*State of Affair*, SoA) des Satzes enhält, jedoch noch nicht weiter spezifiziert ist. Dazu werden nun Operatoren auf die gesamte Prädikation angewandt, etwa in der UCS oben der Operator "present", der selbst wieder als ein Prädikat mit der vollen Prädikation als Argument gesehen werden kann. Ebenso werden in diesem Schritt Operatoren zum Modus (etwa Interrogativ oder Deklarativ) eingefügt.

Diese nun voll spezifizierte Prädikation wird schließlich mit Hilfe von Ausdrucksregeln zu Form, Reihenfolge und Intonation spezifiziert und damit zu einer konkreten sprachlichen Äusserung in Bezug gesetzt (bei der Beschreibung) bzw. in eine solche umgewandelt (bei der Generierung).

2.3.3. Zusammenfassung des Grammatikformalismus

Es handelt sich damit bei FG um ein monostratales Modell, da zwar zwischen der UCS und den sprachlichen Ausdrücken unterschieden wird und diese durch Ausdrucksregeln aufeinander bezogen werden, jedoch werden keine verschiedenen Ebenen angenommen, auf denen konkrete sprachliche Äusserungen stehen, so sind etwa keine syntaktischen Derivationsmechanismen vorhanden. In diesem Sinne findet die Bildung der sprachlichen Äusserungen schrittweise innerhalb einer Prozesskette, auf einer einzigen Ebene statt, wie in Abbildung 2.1 auf Seite 29 deutlich wird, die den Aufbau einer FG zeigt.

Die Pragmatikorientiertheit macht FG zu einem deszendenten Grammatikmodell, das von der Gesamtsituation ausgeht, in der eine Äusserung getätigt wird, im Gegensatz zu einem aszendenten Grammatikmodell, das von den kleinsten Teilen ausgeht, etwa von der Phonologie über die Morphologie zur Syntax.

2.4. Behandlung der Daten

Die Bedeutung von sprachlichen Daten setzt Dik im Allgemeinen sehr hoch an:

> "Whenever there is some overt difference between two constructions X and Y, start out on the assumption that this difference has some kind of functionality in the linguistic system" (Dik 1997a, Kap. 1.6).

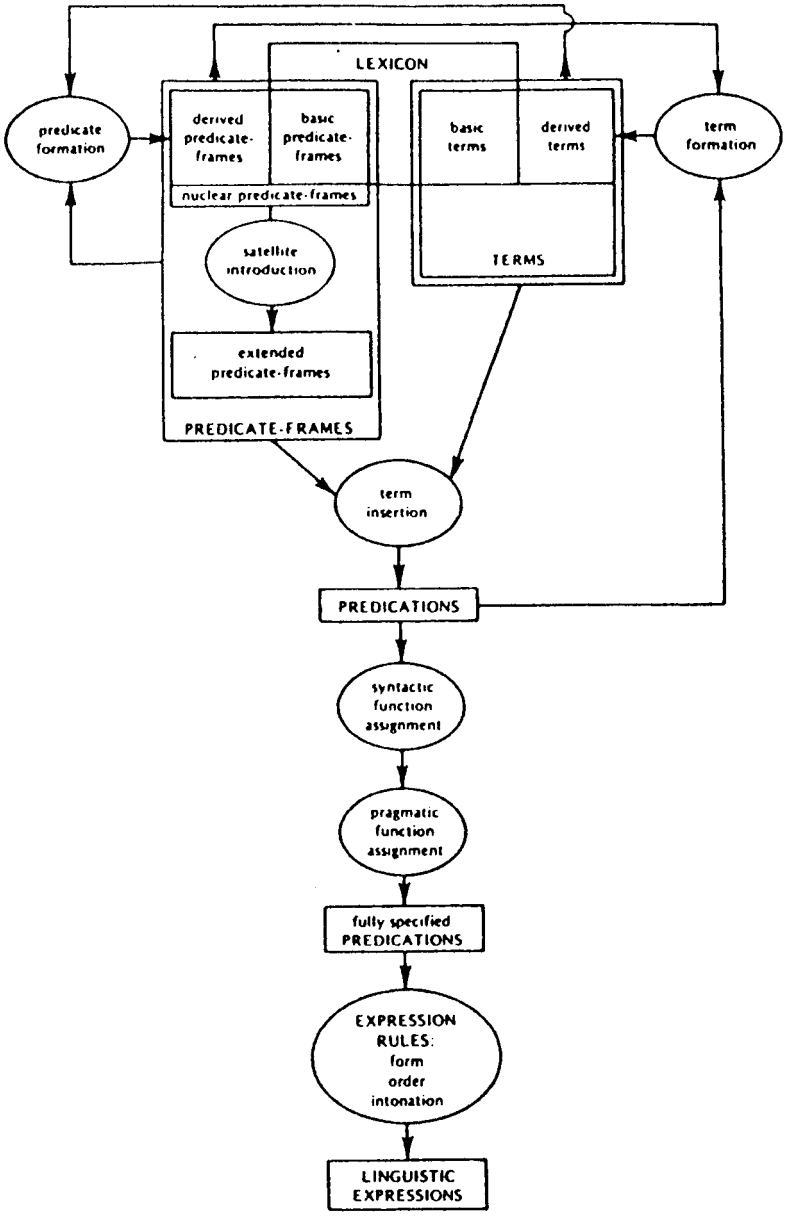

Abbildung 2.1.: Aufbau einer FG (aus Dik 1991, 250)

Damit hat FG einen induktiven Charakter, da sie ähnlich dem Bloomfieldschen Deskritptivismus von konkreten sprachlichen Daten ausgeht, im Gegensatz zu einem deduktiven Modell wie der Generativen Grammatik nach Chomsky, wo eine ideale, vom konkreten Sprachgebrauch abstrahierte Sprachkompentenz im Mittelpunkt der Theorie steht.

In den zentralen Bereichen Pragmatik und Semantik ist die FG vor allem auf die Befragung von Informanten (Elizitierung) sowie die Konsultation der eigenen muttersprachlichen Einsichten (Introspektion) angewiesen. Andere Quellen wie Experimente oder Korpora sind nicht ohne weiteres[3] zur Ermittlung semantischen Wissens (etwa für die Selektionsbeschränkungen in Prädikatrahmen) verwendbar.

Zur Evaluierung des Gesamtmodells können dagegen auch in der FG Korpora und damit spontansprachliche Daten verwendet werden, etwa zur Überprüfung, ob Äusserungen in Korpora durch den FG-Formalismus beschrieben werden können.

2.5. Anspruch des Modells

Die Zielsetzung der FG ist sehr umfassend, (Dik 1997a, Kap. 1) formuliert folgende zentrale Frage: "How does the natural language user (NLU) work?". Diese Fragestellung kennzeichnet FG klar als Modell mit mentalistischem Anspruch.

Dik identifiziert im Anschluss an die Formulierung dieser zentralen Fragestellung fünf Fähigkeiten des NLU, die essentielle Rollen für die menschliche Kommunikation spielen:

1. *linguistic capacity*: Fähigkeit zur Produktion und Interpretation sprachlicher Ausdrücke.
2. *epistemic capacity*: Fähigkeit zu Aufbau und Verwaltung einer Wissensbasis, die zur Sprachverarbeitung genutzt wird.
3. *logical capacity*: Die Fähigkeit, Schlussfolgerungen aus dem verfügbaren Wissen zu ziehen.
4. *perceptual capacity*: Fähigkeit, seine Umwelt wahrzunehmen und bei der Sprachverarbeitung zu berücksichtigen.
5. *social capacity*: Fähigkeit, die Situation bei der Sprachverarbeitung mit zu berücksichtigen.

Darüber hinaus formuliert Dik in Anspielung auf die von Chomsky geforderten drei Adäquatheitskriterien der Beschreibungs-, Erklärungs- und Beobachtungsadäquatheit drei eigene, völlig andere Adäquatheitskriterien:

1. Pragmatische Adäquatheit: Direkte Folge der Grundannahme, dass Sprache ein Mittel zur Kommunikation darstellt.
2. Psychologische Adäquatheit: Erkenntnisse aus der psycholinguistischen Forschung zu Spracherwerb, -verarbeitung und -interpretation müssen berücksichtigt werden.

3 Eine Generierung von semantischem Wissen wäre eventuell durch eine automatische Verarbeitung von Korpora, etwa zur Ermittlung paradigmatischer oder syntagmatischer Relationen möglich.

3. Typologische Adäquatheit: Die Theorie soll auf Sprachen von unterschiedlichem typologischem Status anwendbar sein.

Insbesondere durch den Anspruch der typologischen Offenheit erhält das Modell einen stark beschreibungsorientierten Charakter, da es eine solche Offenheit zu einem universellen Beschreibungswerkzeug machen würde, sowie einen universalistischen Anspruch, der es als Ziel sieht, allgemeingültige Aussagen über Sprache insgesamt, nicht über ein bestimmte Sprache oder Sprachfamilie zu machen.

Der Anspruch der psychologischen Adäquatheit kennzeichnet FG, wie schon im Zusammenhang mit der zentralen Fragestellung erwähnt, als ein mentalistisches Modell, das wie etwa die generative Syntaxtheorie ein Modell für die menschliche Sprachfähigkeit sein will, im Gegensatz zu rein anwendungs- bzw. beschreibungsorientierten Ansätzen wie HPSG.

FG geht im Gegensatz zur nativistischen Hypothese Chomskys davon aus, dass sprachliche Universale nicht angeborenen Eigenschaften entspringen sondern den Notwendigkeiten der menschlichen Kommunikation[4] sowie der physischen und psychologischen Konstitution des Menschen[5], und kann damit als nicht-nativistisches Modell charakterisiert werden.

2.6. Hauptaufgabe linguistischer Forschung

Ziel der Forschung im Rahmen der FG ist die Entwicklung eines sprachunabhängigen Formalismus zur Sprachbeschreibung. Dazu ist eine ausgiebige Anpassung der bestehenden Formalismen an viele verschiedene Sprachen nötig (Dik 1991, 248). In diesem Sinne ist die Sprachbeschreibung ein zentraler Gegenstand der FG-Forschung.

Aus dem Anspruch der Formalisierbarkeit ergibt sich ein weiteres Forschungsgebiet: Die Implementierung von FG auf einem Computer. Dik selbst hat seit den 1980er Jahren vor allem auf diesem Gebiet gearbeitet. Diks eigene und auch andere Implementierungen (etwa Samuelsdorff 1989) verwenden die logikorientierte Programmiersprache Prolog (*Programming in Logic*), die sich aufgrund ihrer starken Orientierung an der Prädikatenlogik besonders zu eignen schien, eine Umsetzung ist jedoch auch in jeder anderen Programmiersprache möglich. Die Arbeiten in diesem Bereich konzentrieren sich stark auf das Gebiet der Generierung und abstrakten Darstellung von sprachlichen Ausdrücken, nicht auf die Verarbeitung (Parsing), die ebenso wie die Generierung Teil der *linguistic capacity* ist (siehe Abschnitt 2.5 auf Seite 30).

Darüberhinaus legen die Forderungen nach pragmatischer und psychologischer Adäquatheit (siehe Abschnitt 2.5 auf Seite 30) eine gewisse Offenheit und interdisziplinäre Zusammenarbeit nahe, wenn Erkenntnisse relevanter Fächer wie Psychologie und Soziologie berücksichtigt werden sollen.

4 In diesem Sinne wäre etwa die Tatsache, dass alle Sprachen eine Unterscheidung zwischen Funktions- und Inhaltswörtern haben, in der Notwendigkeit begründet, die Inhalte einer sprachlichen Äusserung zueinander in Bezug zu setzen.

5 Etwa eine Einschränkung der Schachtelungstiefe von Nebensätzen durch die begrenzten Möglichkeiten des menschlichen Kurzzeitgedächnisses.

2.7. Anwendungsorientiertheit und Anwendbarkeit

2.7.1. Vorteile

Diks "Functional Grammar" scheint viele in anderen Modellen vernachlässigte, jedoch zur vollständigen Sprachbeschreibung wichtige Aspekte von Sprache zu berücksichtigen:

- Eine Berücksichtgung der Pragmatik – etwa wie oben beschrieben bei einer Aufzählung oder zur Beschreibung des Unterschiedes zwischen den Ausdrücken *Buddhist Japanese* und *Japanese Buddhist*.
- Die zentrale Rolle der Semantik – etwa bei der Zuordnung von *who* an belebte und *which* an unbelebte Mitspieler in einem Relativsatz oder bei den Selektionsbeschränkungen der Argumente in Prädikatrahmen.
- Die Unterscheidung von semantischen Rollen und grammatischen Relationen – etwa zur Beschreibung von Aktiv-Passiv-Alternation ohne diese gegenseitig auseinander abzuleiten.
- Die Berücksichtigung typologischer Eigenheiten vieler Sprachen – etwa in Form der *Semantic Function Hierachy* (SFH) zur Subjektivierbarkeit von Mitspielern mit bestimmten semantischen Rollen.

2.7.2. Mögliche Schwächen und Probleme bei der Anwendung

Die Verwendung merkmalsemantischer Primitive zur Selektionsbeschränkung bestimmter Argumentpositionen in den UCS könnte in der Praxis zu den mit diesem sematischen Modell bekannten Problemen führen, etwa bei relationalen Eigenschaften wie Verwandschaftsverhältnissen, sowie bei Verben, graduellen Unterschieden und Farben. Die Kodierung der Feinsemantik im Lexikon stellt jedoch allgemein ein ungelöstes Problem dar.

Das Vorgehen, Verletzungen der Selektionsbeschränkungen nicht als ungrammatisch zu bezeichenen, sondern z.B. als Metapher zu behandeln stellt eventuell eine Immunisierungstrategie dar (etwa wenn keine spezielle Interpretationsstrategie ausgearbeitet wurde), die in diesem Fall Testbarkeit, Anwendbarkeit und den wissenschaftlichen Wert des Modells verringern könnte.

Auch die aus der semantische Ausrichtung ergebende Konzentration auf Introspektion und Elizitierung zur Datengewinnung für die Bestimmung von Selektionsbeschränkungen von Argumentpositionen in Prädikatrahmen könnte Probleme verursachen und den wissenschaftstheoretischen Wert der gewonnenen Daten schmälern, da elizitierte und aus Introspektion gewonnene Daten durch die vorgegebene Fragestellung leicht missinterpretiert werden können, etwa wenn Einflussfaktoren, die über die Fragestellung hinaus gehen, nicht berücksichtigt werden.

Auch im Bereich der Operatoren zur zeitlichen Spezifizierung der Prädikation gehen der universelle Anspruch und die praktischen Erfordernisse auseinander, denn die auf dieser Ebene von Dik genannten Operatoren wie "present" und "progressive" sind keine universellen Kategorien, doch die UCS hat den Anspruch vor Anwendung der Ausdrucksregeln sprachunabhängig kodiert zu sein.

2.8. Zusammenfassende Charakterisierung des Modells

Zusammenfassend lässt sich Simon C. Diks "Functional Grammar" als ein monostratales, nicht-derivationelles (s. Abschnitt 2.3.3 auf S. 28), deszendentes (s. Abschnitt 2.3.3 auf S. 28), induktives (s. Abschnitt 2.4 auf S. 28), mentalistisches, nicht-nativistisches (s. Abschnitt 2.5 auf S. 30), universalistisches (s. Abschnitt 2.5 auf S. 30), pragmatikbasiertes, und in diesem Sinne funktionales (s. Abschnitt 2.2 auf S. 25 sowie Abschnitt 2.5 auf S. 30) sowie beschreibungsorientiertes und testbares, und in diesem Sinne formales (s. Abschnitt 2.2 auf S. 25 sowie Abschnitt 2.5 auf S. 30) linguistisches Modell charakterisieren.

Teil II.

Hauptstudium

Kapitel 3

Stringverarbeitung

Suffixbäume und Ähnlichkeitsmaße, Sprachliche Informationsverarbeitung, Hauptseminar *Stringverarbeitung* bei Prof. Dr. Jürgen Rolshoven, Wintersemester WS 2005–2006.

3.1. Suffixbäume in der maschinellen Sprachverarbeitung

Gegenstand dieser Arbeit ist die Untersuchung der Anwendbarkeit von Suffixbäumen zur Ermittlung von Ähnlichkeiten im Rahmen einer Strukturanalyse in der maschinellen Sprachverarbeitung.

3.1.1. Suffixbäume: Eine vielseitige Datenstruktur

Suffixbäume sind eine vielseitige Datenstruktur mit vielen effizienten Anwendungsmöglichkeiten in der Stringverarbeitung.

Ein Suffixbaum T für einen String S mit m Symbolen ist ein gerichteter Baum mit m Blättern. Jede Kante ist mit einem Teilstring von S beschriftet. Jeder innere Knoten von T hat mindestens zwei Kinder, deren Kantenbeschriftungen nie mit dem gleichen Symbol beginnen. Für jedes Blatt i in T ergeben die Beschriftungen der Kanten auf dem Pfad von der Wurzel zu i aneinander gehangen das Suffix von S, das an Index i beginnt.

Somit enthält T alle Suffixe von S, wobei mehrfach auftretende Teilstrings nur einmal in T enthalten sind (siehe Abbildung 3.1, in dieser und allen weiteren Abbildungen von Suffixbäumen in dieser Arbeit sind die Knoten der Bäume *depth-first* durchnummeriert). Dadurch ermöglicht ein Suffixbaum die Lösung zahlreicher Probleme im Bereich der Stringverarbeitung in linear wachsender oder sogar konstanter Laufzeit. Für eine ausführliche Darstellung von Suffixbäumen siehe Gusfield (1997).

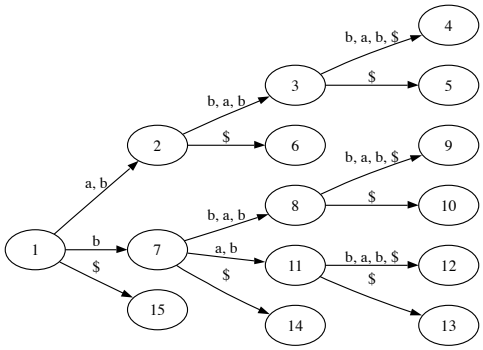

Abbildung 3.1.: Suffixbaum für *abbabbab*

Damit erscheinen Suffixbäume durch zwei Eigenschaften interessant:

1. Suffixbäume ermöglichen die effiziente Analyse großer Korpora durch verschiedene effiziente Algorithmen.
2. Suffixbäume kodieren Struktur, der Suffixbaum fasst wiederholte Strukturen zusammen und ist damit eine Generalisierung des Eingabetextes.[1]

Das Haupteinsatzgebiet für Suffixbäume liegt traditionellerweise in der Bioinformatik (vgl. Gusfield 1997; Böckenhauer & Bongartz 2003), wo große Datenmengen manipuliert und analysiert werden. Anwendungen im Bereich der Verarbeitung natürlicher Sprache sind bislang deutlich weniger zu finden. Ein Gegenbeispiel ist Geertzen (2003), siehe dazu Abschnitt 3.3 auf Seite 41.

3.1.2. Morphologie: Suffixbäume für Buchstaben und Wörter

Suffixbäume zur morphologischen Analyse enthalten als Symbole die Buchstaben des entsprechenden natürlichsprachlichen Alphabets und ähneln damit den Suffixbäumen aus der Bioinformatik. Jedoch sind bei der Darstellung mehrerer Wörter Suffixe, die über Wortgrenzen hinausgehen, nicht von Belang. Hier würde daher ein *generalisierter* Suffixbaum (vgl. Gusfield 1997, 116) für alle Types im Korpus erstellt (siehe Abbildung 3.2 auf Seite 39).

3.1.3. Syntax: Suffixbäume für Wörter und Sätze

Für die syntaktische Analyse mit Suffixbäumen müssen diese als Symbole nicht einzelne Buchstaben, sondern Wörter enthalten. Zum Vergleich von Sätzen kann dann ein wortbasierter, über mehrere Sätze generalisierter Suffixbaum erstellt werden. Ein generalisierter, wortbasierter Suffixbaum enthält Informationen über die Konstituentenstruktur der im Baum enthaltenen Sätze, denn

1 Ein Verfahren zum Lernen von Grammatiken, das auf diesem Zusammenhang zwischen Kompression und Generalisierung sowie bayesscher Schätzung basiert, ist das Prinzip der *minimum description length* (MDL). (Geertzen 2003, 2of.) umreißt das Verfahren und gibt weitergehende Literaturhinweise.

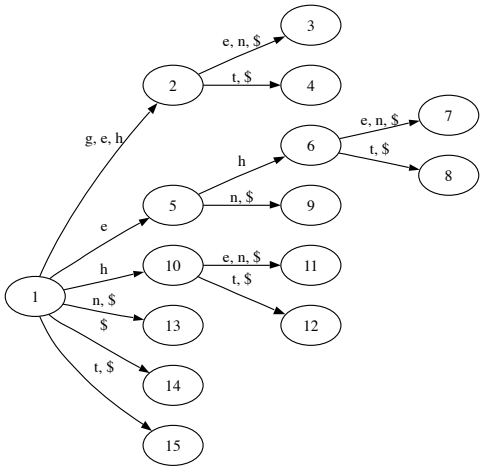

Abbildung 3.2.: Generalisierter, zeichenbasierter Suffixbaum für *gehen geht*

aufgrund der Struktur des Suffixbaumes bilden die Beschriftungen der ausgehenden Kanten jedes inneren Knotens Paradigmen (siehe Abbildung 3.3 auf Seite 40, mehr dazu in Abschnitt 3.3 auf Seite 41).

3.2. Laufzeit und Speicherplatzbedarf

Traditionelle Algorithmen zur Erstellung von Suffixbäumen basieren auf der Tatsache, dass *alle* Suffixe des Eingabestrings eingefügt werden und nutzen aus, dass das Eingabealphabet sehr klein ist (Andersson *et al.* 1999).

Das Einfügen aller Suffixe ist jedoch für eine Anwendung zur syntaktischen Analyse weder nötig noch erwünscht, da lediglich solche Suffixe eingefügt werden sollen, die an Wortgrenzen beginnen. Zudem ist hier das Alphabet viel größer, es umfasst alle unterschiedlichen Wörter *m'* (Types) in der Menge aller Wörter *m* (Token). Das bedeutet zum einen, dass die Effizienz der traditionellen Algorithmen nicht ohne weiteres übertragbar ist auf die Erstellung von wortbasierten Suffixbäumen, und zum anderen sollten wortbasierte Suffixbäume mit weniger Speicherplatzbedarf konstruierbar sein, da die Anzahl von Wörtern *m* deutlich kleiner ist als die Anzahl der Buchstaben *n* (siehe Tabelle 3.1 auf Seite 41).

Ziel ist also die Erstellung eines wortbasierten Suffixbaumes für einen String *S* mit *n* Buchstaben und *m* Wörtern mit einer Laufzeitkomplexität von *O(n)* und einer Speicherplatzkomplexität von *O(m)*.

3.2.1. Naiver Algorithmus

Wie ein traditioneller Suffixbaum kann auch ein wortbasierter Suffixbaum mit einem naiven Algorithmus (siehe etwa Gusfield 1997, 93ff. oder Böckenhauer & Bongartz 2003, 56ff.) erstellt werden.

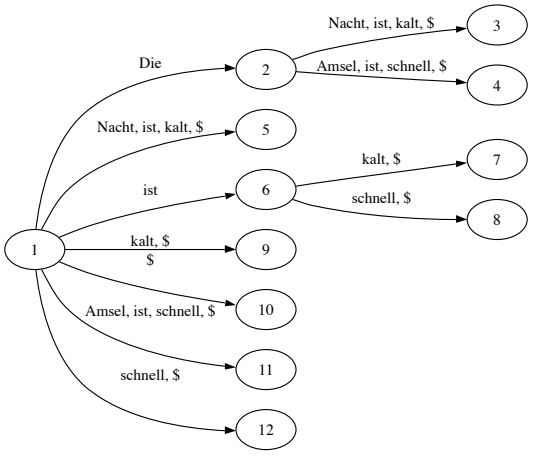

Abbildung 3.3.: Generalisierter, wortbasierter Suffixbaum für *Die Nacht ist kalt. Die Amsel ist schnell.*

Die Speicherplatzkomplexität beträgt dabei von Anfang an wie angestrebt *O(m)*, da der Eingabestring vor der Konstruktion des Baumes in Wörter zerlegt werden kann und so nur Wörter eingefügt werden. Die Laufzeitkomplexität beträgt jedoch beim naiven Algorithmus aufrund der mit Größe des Baumes immer weiter zunehmenden Zeit zum Auffinden der Einfügestelle $O(m^2)$.

3.2.2. Modifikation traditioneller Algorithmen

Eine Möglichkeit der Nutzung traditioneller Algorithmen zur Erstellung von wortbasierten Suffixbäumen ist in (Geertzen 2003, 39ff.) beschrieben. Dort wird im Wesentlichen der Algorithmus von Ukkonen (siehe Gusfield 1997, 94ff.) verwendet, wobei Problemen durch das wesentlich größere Alphabet durch effiziente Speicherverwaltung begegnet wird. Geertzen macht keine expliziten Angaben zu Laufzeit und Speicherplatzbedarf des angepassten Algorithmus.

Ein weiterer Ansatz, unter Nutzung traditioneller Algorithmen einen wortbasierten Suffixbaum zu erstellen, besteht in der Erstellung eines traditionellen Suffixbaumes mit einer Laufzeitkomplexität und einer Speicherplatzkomplexität von *O(n)*. Anschließend wird der Baum auf die Größe *m* komprimiert. Dieser Ansatz hat zwar wie angestrebt eine Laufzeitkomplexität von *O(n)*, jedoch auch eine Speicherplatzkomplexität von *O(n)* und nicht, wie angestrebt, *O(m)* (Andersson *et al.* 1999).

3.2.3. Effiziente Konstruktion von wortbasierten Suffixbäumen

In (Andersson *et al.* 1999, 5ff.) wird ein Algorithmus vorgestellt, der die Erstellung eines wortbasierten Suffixbaumes mit einer Laufzeitkomplexität von *O(n)* und einer Speicherplatzkomplexität von *O(m)* ermöglicht.[2] Der Algorithmus besteht aus den folgenden Schritten:

2 Für kleine Alphabete stellen Andersson *et al.* (1999) zudem ein sublineares Verfahren zur Erstellung von wortbasierten Suffixbäumen vor.

Autor	Titel	Zeichen (n)	Token (m)	Types (m')
J. W. v. Goethe	Faust, erster Teil	200.957	46.028	7.386
Mark Twain	Tom Sawyer	387.922	71.457	7.389
August Strindberg	Röda rummet	539.473	91.771	13.425
Leo Tolstoi	Krieg und Frieden	3.217.395	692.328	19.512

Tabelle 3.1.: Anzahl von Buchstaben, Token und Types in natürlichsprachlichem Text (aus: Andersson *et al.* 1999, ergänzt)

1. Konstruktion eines Trie für die Menge der unterschiedlichen Wörter, der Types *m'* im Eingabetext *S*. Laufzeitkomplexität: *O(n)*, Speicherplatzkomplexität: *O(m')*.
2. Wörter im Trie werden durchnummeriert. Laufzeit- und Speicherplatzkomplexität: *O(m')*.
3. Jedem Wort im Originalstring wird sein Zahlenwert aus dem Trie zugeordnet. So entsteht ein String aus Zahlen. Laufzeit- und Speicherplatzkomplexität: *O(m)*.
4. Erstellung eine lexikographischen Suffixbaumes mit den Zahlen als Symbole mithilfe eines traditionellen Algorithmus. Laufzeit- und Speicherplatzkomplexität: *O(m)*.
5. Mithilfe des Tries den lexikographischen Suffixbaum zu einen nicht-lexikographischen, wortbasierten Suffixbaum expandieren. Laufzeit- und Speicherplatzkomplexität: *O(m)*.

So hat das effiziente Verfahren einen deutlichen Vorteil in Bezug auf den Speicherplatzbedarf, da nicht zwischenzeitlich ein Suffixbaum für alle Buchstaben des Eingabetextes erstellt werden muss (vgl. Tabelle 3.1 auf Seite 41).

3.3. Syntaxanalyse

Geertzen (2003) beschreibt mit dem von van Zaanen (2002) entwickelten *Alignment-Based Learning* (ABL) ein Verfahren zur Ermittlung von Konstituenten. Konstituenten sind Teile von Sätzen, die gegen andere austauschbar sind. Konstituenten, die im gleichen Kontext stehen können (die also die gleiche Distribution aufweisen), bilden ein Paradigma. Zu paradigmatischen und syntagmatischen Relationen siehe etwa (Lyons 1968, 70)[3].

ABL verwendet ursprünglich die Ermittlung der gewichteten Levenshtein-Distanz aller Sätze (*all-against-all* Vergleich) über *dynamic programming* zur Erkennung der Konstituenten (siehe Abschnitt

3 Die Begrifflichkeit der paradigmatischen und syntagmatischen Relationen sowie die zugrunde liegende synchrone Sprachbeschreibung gehen zurück auf den Begründer der modernen Sprachwissenschaft, Ferdinand de Saussure, der damit zu Beginn des 20. Jahrhunderts an eine Tradition anknüpft, die bis zum altindischen Grammatiker Panini zurückgeführt werden kann, der bereits um 500 v. Chr. die Grammatik des Sanskrit in einer Form beschrieb, die in ihrer Mächtigkeit der in den späten 50er Jahren des 20. Jahrhunderts entwickelten Backus-Naur-Form oder Backus-Normalform (die daher auch Panini-Backus-Form genannt wird) entspricht. Eine aktuelle Betrachtung von Paninis Grammatik findet sich in Kiparsky (2002).

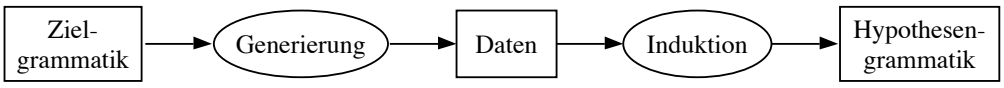

Abbildung 3.4.: Grundgedanke von ABL (aus: Geertzen 2003)

3.3.2 auf Seite 42), doch dieses Verfahren ist zu ineffizient, um größere Korpora zu analysieren. Geertzen nennt als Grenze 100.000 Sätze, er bricht bei der Evaluation des Verfahrens jedoch bereits bei einer Anzahl von 38.009 Sätzen ab (siehe Abschnitt 3.3.3 auf Seite 43). Zur Anwendung des Verfahrens auf größere Korpora untersucht Geertzen daher Suffixbäume.

3.3.1. Grundgedanke

Der Gedanke, der ABL zugrunde liegt, ist dass aus einem Korpus die Grammatik induziert werden kann, die das Korpus ursprünglich generiert hatte (siehe Abbildung 3.4 und Abbildung 3.8 auf Seite 45).

Zur Evaluation des Verfahrens wird die generierte Hypothesengrammatik mit der Zielgrammatik verglichen. Dies setzt die Kenntnis der Zielgrammatik voraus, etwa in Form einer Baumbank. Im besten Fall ist die Hypothesengrammatik vollständig, d.h. sie erzeugt alle Sätze der Zielgrammatik, sowie konsistent, d.h. sie erzeugt keine Sätze, die nicht in der Baumbank vorhanden sind.

Der Unterschied zwischen der Analyse beim ABL und einem Parsing-Vorgang besteht darin, dass im Gegensatz zum Parsing bei der ABL-Analyse die Struktur bei der Verarbeitung der Eingabe nicht bekannt ist. Dies wird auch als *structure bootrapping* bezeichnet und stellt eine Form von maschinellem Lernen syntaktischer Struktur dar.

Das Verfahren ist ein unüberwachtes Lernen, da dem Lerner, d.h. dem System, die korrekte Struktur nicht bekannt ist, die Baumbank wird lediglich zur Evaluierung genutzt, das Lernen des syntaktischen Wissens geschieht aus nicht annotierten Korpora und ohne Eingriff durch den Menschen.

3.3.2. Vergleich von Sätzen

Konstituenten werden beim ABL über paarweisen Vergleich aller im Korpus enthaltenen Sätze ermittelt. Dies kann über die Levenshtein-Distanz erfolgen, so wäre etwa die (wortbasierte) Levenshtein-Distanz zwischen *Ich kaufe ein Auto* und *Ich kaufe ein Boot* 1, das geänderte Wort (*Haus* oder *Boot*) erscheint im gleichen Kontext (nämlich nach *Ich kaufe ein*) und ist damit eine Konstituente des Satzes. Die beiden Wörter, die im gleichen Kontext auftauchen (*Auto* und *Boot*) bilden gemeinsam ein Paradigma. In ähnlicher Weise ist diese Information in einem Suffixbaum gespeichert (siehe Abbildung 3.3 auf Seite 40).

3.3.3. Alignment-Based Learning (ABL)

Ziel von ABL ist das Ermitteln einer Hypothesengrammatik durch das oben beschriebene Verfahren. Es werden verschiedene Hypothesen pro Satz aufgestellt und im Anschluss kontrolliert. Kriterium

bei dieser Kontrolle ist, dass geänderte Sätze valide sein müssen, d.h. im Korpus vorhanden sind. Wenn Sätze valide sind, werden sie in den Hypothesenraum aufgenommen. Für das Beispiel im vorigen Abschnitt wären die Hypothesen für beide Sätze etwa:

Ich kaufe ein [Auto]
Ich kaufe ein [Boot]

Über die Levenshtein-Distanz

Durch Erstellung eines *edit transcript* (siehe Gusfield 1997, 215ff.) für die zu vergleichenden Sätze können Gemeinsamkeiten ermittelt werden: Gleiche Wörter bilden Wortcluster, ungleiche Teile sind dann im gleichen Kontext austauschbar und damit Konstituenten-Hypothesen.

Über Suffixbäume

Im Suffixbaum sind alle Wortcluster bereits in der Struktur gespeichert: Alle Wortcluster beginnen am Wurzelknoten und enden in einem inneren Knoten. Von inneren Knoten ausgehende Verzweigungen markieren so stets den Beginn von Konstituenten und bilden Paradigmen. Jedoch ist im Suffixbaum an sich das Ende von Konstituenten nicht markiert. Geertzen evaluiert verschiedene Strategien zur Ermittlung des Endes von Konstituenten:

1. Konstituenten beginnen an inneren Knoten und enden am Satzende. So können Konstituenten am Ende von Sätzen erkannt werden (siehe Abbildung 3.5 auf Seite 44).
2. Wird der Suffixbaum zusätzlich andersherum aufgebaut können auch Konstituenten am Anfang von Sätzen erkannt werden (siehe Abbildung 3.6 auf Seite 44).
3. Durch Kombination der öffnenden und schließenden Klammern der ersten beiden Schritte können auch Konstituenten in der Satzmitte erkannt werden (siehe Abbildung 3.7 auf Seite 45)

Letzteres, kombiniertes Verfahren lieferte Geertzen die besten Ergebnisse, daher werden im Folgenden lediglich diese wiedergegeben.

Ergebnisse

Die Evaluation von ABL erfolgt über die im Information Retrieval üblichen Werte Precision und Recall, die im Kontext von ABL das Folgende ausdrücken:

- Recall: Anteil der Konstituenten in der ursprünglichen Baumbank, die auch in der gelernten Baumbank vorhanden sind.
- Precision: Anteil der korrekten Konstituenten an den ermittelten Konstituenten. Geertzen nimmt eine perfekte Selektion von korrekten Hypothesen an, daher beträgt die Precision immer 100% für diese Auswertung, im Folgenden werden daher nur die Recall-Werte wiedergegeben.

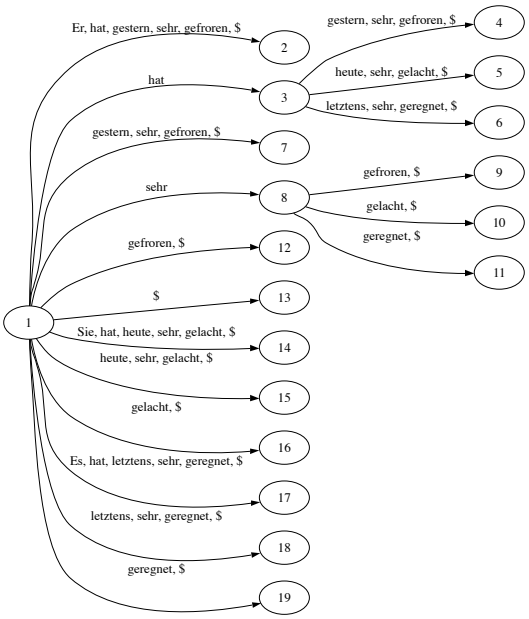

Abbildung 3.5.: Generalisierter, wortbasierter Suffixbaum für *Er hat gestern sehr gefroren. Sie hat heute sehr gelacht. Es hat letztens sehr geregnet.*

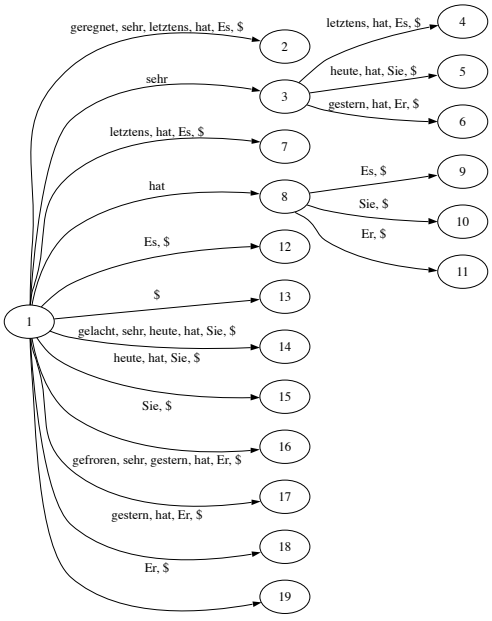

Abbildung 3.6.: Generalisierter, umgekehrter, wortbasierter Suffixbaum für *Er hat gestern sehr gefroren. Sie hat heute sehr gelacht. Es hat letztens sehr geregnet.*

$[_2[_2\text{Er}]_2$ hat $[_1\text{gestern}]_2$ sehr $[_1\text{gefroren}]_1]_1$

Abbildung 3.7.: Ermittelte Konstituenten beim kombinierten Verfahren (Abschnitt 3.3.3 auf Seite 43): Mit einer 1 gekennzeichnete Klammern wurden durch den ersten, vorwärts aufgebauten Baum ermittelt, mit einer 2 gekennzeichnete Klammern durch den zweiten, rückwärts aufgebauten Baum. Konstituenten in der Mitte von Sätzen können durch die Kombination der so ermittelten Klammerung bestimmt werden, hier *gestern*.

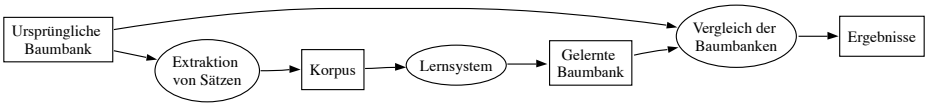

Abbildung 3.8.: Vorgehen bei der Evaluierung von ABL (nach Geertzen 2003, 58, vgl. Abb. 3.4 auf Seite 42)

Geertzen verwendet drei Baumbanken zur Evaluation des Verfahrens, dies sind im Einzelnen ATIS (*Air Traffic Information System*, aus der Penn-Baumbank[4]), OVIS (*Openbaar Vervoer Systeem*, eine niederländische Baumbank mit Texten aus einem Verkehrs-Informationssystem), sowie zwei Korpora verschiedener Größe aus dem *Wall Street Journal*. Die Korpora unterscheiden sich deutlich in Umfang und Beschaffenheit (siehe Tabelle 3.2).

Aus Tabelle 3.2 wird deutlich, dass die gewichtete Levenshtein-Distanz[5] für alle Korpora die besseren Ergebnisse liefert, außer bei einer Korpusgröße, die mit dem Verfahren nicht mehr zu bewältigen ist. Jedoch sind die Ergebnisse bei dem Korpus mit den längsten Sätzen nahezu identisch. Zudem ist zu bemerken, dass die Qualität der Ergebnisse des Suffixbaumes auf dem gleichen Korpus mit steigender Korpusgröße abnimmt. Möglicherweise ist der Grund hierfür jedoch, dass in dem von Geertzen untersuchten Korpora die Anzahl unterschiedlicher syntaktischer Konstruktionen mit einer Zunahme der Korpusgröße zunächst ebenso zunahm. Interessant könnte in diesem Zusammenhang eine Evaluation mit mehr als zwei Korpusgrößen gleicher Beschaffenheit und größeren Umfangs sein. Das größte von Geertzen untersuchte Korpus hatte in der Anzahl der Sätze etwa den Umfang von *Krieg und Frieden* von Leo Tolstoi. Denkbar und aufgrund der in Abschnitt 3.2 auf Seite 39 beschriebenen Eigenschaften von Suffixbäumen auch realisierbar wäre ein Analyse mit vielfach größeren Korpora.

Es ist nicht klar, wie die Qualität des Levenshtein-basierten Verfahrens bei größeren Korpora ausfallen würde. Für kleinere Korpora wäre jedoch eine Verbesserung der Ergebnisse des Suffixbaumbasierten Verfahrens und damit eine Annäherung an die Ergebnisse des Levenshtein-basierten Verfahrens wünschenswert. Dies könnte etwa durch eine Annäherung an die Vorgehensweise des Levenshtein-basierten Verfahrens geschehen, eventuell durch Verfahren wie Matching mit *k-mismatch*

4 The Penn Treebank Project: <http://www.cis.upenn.edu/~treebank/>

5 Die Gewichtung für das Hinzufügen, Löschen und Ersetzen von Symbolen beträgt 1,1,2.

Korpus	ATIS	OVIS	WSJ 1	WSJ 2
Beschaffenheit				
Sätze	568	10000	1904	38009
Satzlänge (Wörter pro Satz)	7,5	3,5	>20	>20
Recall				
Levenshtein	48.8	94.22	53.26	–
Suffixbaum	38.35	59.18	52.05	37.27

Tabelle 3.2.: Beschaffenheit der drei Korpora und Ergebnisse

oder Matching mit *wildcards* (siehe Abschnitte 3.4.3 und 3.4.2 auf Seite 48 und 48).

Ein anderer Ansatz bestünde darin, die Levenshtein-basierte Variante von ABL mithilfe von hybridem *dynamic programming* und Suffixbäumen (siehe Abschnitt 3.4.4 auf Seite 49) effizienter zu machen und so auch größere Korpora mit diesem Verfahren zu analysieren.

3.4. Ähnlichkeitsmaße

3.4.1. Suffixbäume zur Optimierung im Hintergrund

Lowest Common Ancestor

Der *lowest common ancestor* (LCA) von zwei Knoten in einem Baum ist der tiefste Knoten im Baum, der gleichzeitig Vorfahr beider Knoten ist (siehe Abbildung 3.9 auf Seite 47).

Der LCA kann in konstanter Zeit ermittelt werden. Das in (Gusfield 1997, 184ff.) beschriebene Vorgehen basiert auf der Darstellung der Baumstruktur als Zahlen in binärer Form. Binärzahlen an den Knoten des Baumes repräsentieren den Pfad zum entsprechenden Knoten: Eine 0 steht für *links*, eine 1 für *rechts*, daran angehangen wird eine 1, anschließend wird die Zahl bis zur Länge der Baumtiefe + 1 mit Nullen aufgefüllt (siehe Abbildung 3.10 auf Seite 47). Durch Vergleiche der einzelnen Pfadnummern ist eine Ermittlung des *LCA* nach Erweiterungen auch für nicht-binäre Bäume möglich (Gusfield 1997, 184ff.).

Longest Common Extension

Die Ermittlung der längsten gemeinsamen Erweiterung (*longest common extension*, LCE) von zwei Strings ab zwei Positionen i in String 1 und j in String 2 ist eine Teilaufgabe von vielen Algorithmen in der Stringverarbeitung, insbesondere im Bereich des weichen String-Matching, z.B. für Matching mit *wildcards* oder *k-mismatch* (siehe Gusfield 1997, 196ff.).

Die Ermittlung der LCE ist mit Suffixbäumen in konstanter Zeit möglich, über die Ermittlung des *lowest common ancestor* (LCA, siehe Abschnitt 3.4.1 auf Seite 46), denn die Stringtiefe des LCA der zwei Blätter, die den Positionen i und j in den beiden Strings entsprechen ist die Länge der LCE (siehe Abbildung 3.11 sowie Gusfield 1997, 196f., 181ff.).

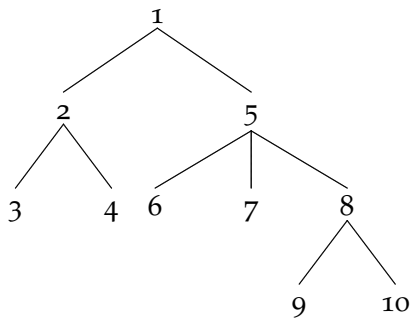

Abbildung 3.9.: Ein Baum mit *depth-first* Nummerierung. Der LCA der Knoten 10 und 7 etwa wäre Knoten 5, der LCA von Knoten 3 und 5 wäre Knoten 1.

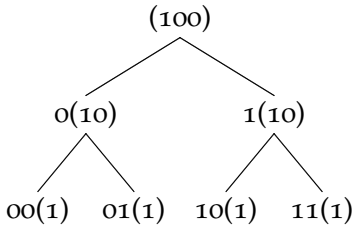

Abbildung 3.10.: Ein Binärbaum mit Pfadnummern im Binärformat. Eine 1 bedeutet 'rechts', eine 0 'links', die Zahlen in Klammern sind Ergänzungen (siehe Abschnitt 3.4.1 auf Seite 46).

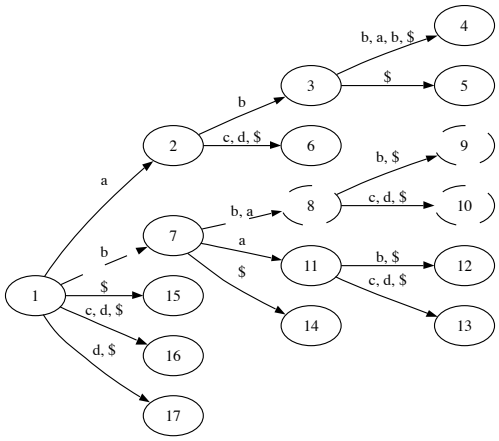

Abbildung 3.11.: Ermittlung der *longest common extension* (LCE) über den *lowest common ancestor* (LCA): Der LCA der Knoten 9 und 10 ist Knoten 8. Der Pfad von der Wurzel zu Knoten 8 ist beschriftet mit der gesuchten LCE der Strings *abbab* ab Position 2 sowie *bbacd* ab Position 1. Die Blätter 9 und 10 korrespondieren mit den Teilstrings beginnend an Position 2 respektive 1 (Indizes sind 1-basiert). Die gesuchte LCE ist hier *bba*.

3.4.2. Matching mit Slots: *wildcards*

Matching mit *wildcards*, d.h. mit Positionen, an denen ein beliebiges Symbol stehen kann (hier symbolisiert durch '*') könnte etwa eingesetzt werden zur Ermittlung aller Sätze, die folgendem Ausdruck entsprechen:

Er kauft ein *

Auf diese Weise könnte eine effiziente Ermittlung von ähnlichen Sätzen und damit von Konstituenten und Paradigmen möglich sein. Zur Ermittlung längerer Konstituenten könnte mit mehreren *wildcards* gesucht werden. Das Verfahren hat eine Laufzeitkomplexität von *O(k*m)* zur Ermittlung aller Treffer, bei einer Zahl *k* von wildcards in einem Text der Länge *m* (Gusfield 1997, 199).

3.4.3. Matching mit Fehlern: *k-mismatch*

Ein mögliches Vorgehen beim Matching mit einer bestimmten Anzahl von erlaubten Fehlern *k* (*k-mismatch*) wäre etwa die Suche nach allen Vorkommen von *Er kauft ein Haus* mit einem Fehler (k=1). Eine solche Suche könnte Ergebnisse wie *Er kauft ein Pferd, Sie kauft ein Haus, Er hat ein Haus* liefern. Diese Sätze könnten zur Ermittlung von Paradigmen miteinander verglichen werden. Das Verfahren hat eine Laufzeitkomplexität von *O(n+m)* zur Ermittlung aller Treffer eines Suchmusters der Länge *n* in einem Text der Länge *m* (Gusfield 1997, 200).

3.4.4. Levenshtein-Distanz über hybrides *dynamic programming* mit Suffixbäumen

Eine *dynamic programming* Lösung zur Ermittlung der gewichteten Levenshtein-Distanz beim *p-against-all* und *all-against-all* Matching mit quadratisch wachsender Laufzeit kann durch den Einsatz von Suffixbäumen in seiner Effizienz verbessert werden. Der Grundgedanke ist dabei, redundante Teilstrings nicht jedes mal neu, sondern nur einmal miteinander zu vergleichen. Dazu wird die im Rahmen des *dynamic programming* aufgebaute *edit table* beim Traversieren des Suffixbaumes statt beim Verarbeiten des Textes aufgebaut (Gusfield 1997, 279ff.).

Eine Effizienzsteigerung des Verfahrens ist dadurch davon anhängig, wie viele redundante Strukturen im Text enthalten sind. Ein Maß für den Umfang redundanter Strukturen ist die Länge des Baumes:[6] Je kürzer der Baum, desto höher die Redundanz. (Gusfield 1997, 286) berichtet von einer Effizienzsteigerung um den Faktor 100 bei menschlicher DNA als Text. Da auch in natürlicher Sprache redundante Strukturen vorhanden sind,[7] wäre mit diesem Verfahren eine Anwendung des Levenshtein-basierten ABL (Abschnitt 3.3.3 auf Seite 43) mithilfe von Suffixbäumen auch für größere Korpora möglich, als durch *dynamic programming* alleine.

3.5. Fazit

Suffixbäume sind eine effizient nutzbare, vielseitige Datenstruktur für die Stringverarbeitung. Sie können auch für die Verarbeitung natürlicher Sprache verwendet werden, sowohl auf Wort- als auch auf Satzebene, da auch für diese Bereiche effiziente Algorithmen zur Verfügung stehen (siehe Andersson *et al.* 1999 und Abschnitt 3.2.3 auf Seite 40). Es gibt erste Anwendungen im Bereich der Erkennung syntaktischer Strukturen mit auf Suffixbäumen basierenden Verfahren mit linear wachsender Laufzeit (siehe Geertzen 2003 und Abschnitt 3.3 auf Seite 41). Solche Ansätze sind möglicherweise ausbaubar, einerseits durch eine Beschleunigung bestehender Verfahren mit quadratisch wachsender Laufzeit (etwa *all-against-all* Matching) durch den Einsatz von Suffixbäumen (hybrides *dynamic programming*, siehe (Gusfield 1997, 279ff.). und Abschnitt 3.4.4 auf Seite 49), andererseits durch Anwendung von Verfahren mit linear wachsender Laufzeit wie Matching mit *wildcards* oder *k-mismatch* (siehe (Gusfield 1997, 196f.) und Abschnitt 3.4.1 auf Seite 46). Eine Übersicht der beschriebenen Verfahren findet sich in Abbildung 3.12.

3.6. Implementation

Die Darstellungen von Suffixbäumen in der vorliegenden Arbeit wurden mit der im Rahmen dieser Arbeit entstandenen Software erstellt. Die Software ermöglicht eine flexible Erstellung und Darstellung von Suffixbäumen aus unmittelbar eingegebenem oder in einer Datei gespeichertem Text. Darüber hinaus kann mit zwei Matching-Algorithmen auf dem eingegebenen Text experimentiert werden. Neben der graphische Oberfläche zur Erstellung von Abbildungen kann die Software als API genutzt werden. Die Software ist im WWW verfügbar[8] und enthält weitergehende Dokumentation.

6 Die Länge eines Baumes ist die Summe aller Symbole aller Kanten (siehe Gusfield 1997, 282)

7 Das Ausmaß an redundanten Strukturen in natürlichsprachlichem Text ist aufgrund des größeren Alphabets (insbesondere bei Wörtern als Symbole) vermutlich erheblich geringer als in der Bioinformatik, wo mit einem Alphabet aus nur 4 Symbolen gearbeitet wird.

8 Suffixbäume und Ähnlichkeitsmaße: <http://stnl.sourceforge.net/>

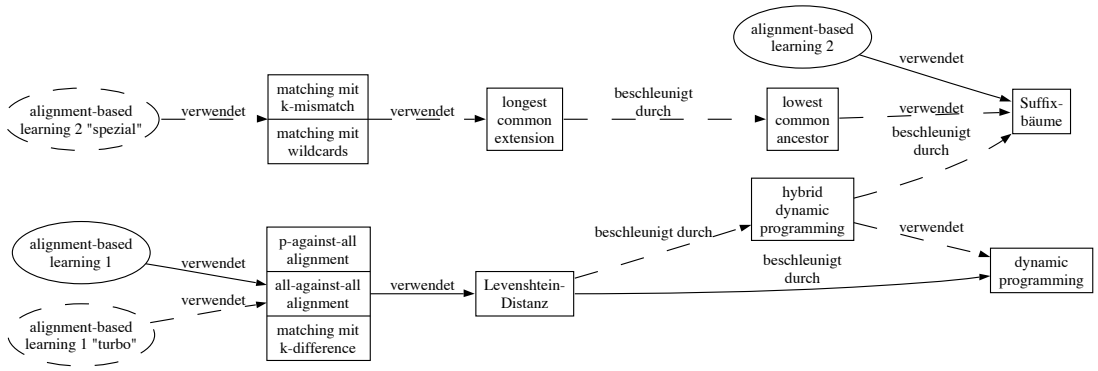

Abbildung 3.12.: Übersicht der beschriebenen Verfahren. Durchgezogene Linien beschreiben die vorgestellten Verfahren des *Alignment-Based Learning* aus Geertzen (2003), gestrichelte Linien beschreiben mögliche Erweiterungen durch die beschriebenen Verfahren aus (Gusfield 1997).

3.6.1. Programmierschnittstelle

Über die Programmierschnittstelle (*application programming interface*, API) können mit linear wachsender Laufzeit (basierend auf dem Algorithmus von Andersson *et al.* (1999) und und einer angepassten Version der Klasse *UkkonenSuffixTree* der BioJava API[9]) wortbasierte Suffixbäume konstruiert, traversiert und als *dot* (siehe folgender Abschnitt) exportiert werden. Für diese Bäume ist ein LCA-Retrieval in konstanter Zeit implementiert (basierend auf einer Implementation von A.G. McDowell[10]). Es bestehen Implementationen von Matching mit *k-mismatch* (Gusfield 1997, 200) und *wildcards* (Gusfield 1997, 199) sowie eine experimentelle Implementation der Ermittlung der LCE über den LCA (siehe Gusfield 1997, 196f.). Darüber hinaus können kleine Suffixbäume naiv in quadratischer Laufzeit sowohl buchstaben- sowie wortbasiert konstruiert und im *dot* Format exportiert werden.

3.6.2. Graphische Oberfläche

Die graphische Darstellung der Suffixbäume basiert auf einem in Java implementierten naiven Algorithmus zur Konstruktion von Suffixbäumen (siehe Böckenhauer & Bongartz 2003:56ff.), optional mit Wörtern als Symbole, generalisiert für mehrere Sätze und umkehrbar. Die graphische Darstellung erfolgt mithilfe der Open-Source Software GraphViz,[11] das generierte *dot* Format kann abgespeichert werden, um daraus andere Bildformate zu erstellen.

Größere Texte können aus einer Datei eingelesen und als wortbasierter Suffixbaum im *dot* Textformat gespeichert werden. Der hier verwendete Suffixbaum ist eine experimentelle Java-Implementation

9 BioJava API: <http://www.biojava.org>
10 Software von A. G. McDowell: <http://www.mcdowella.demon.co.uk/programs.html>
11 GraphViz Graph Visualization Software: <http://www.graphviz.org>

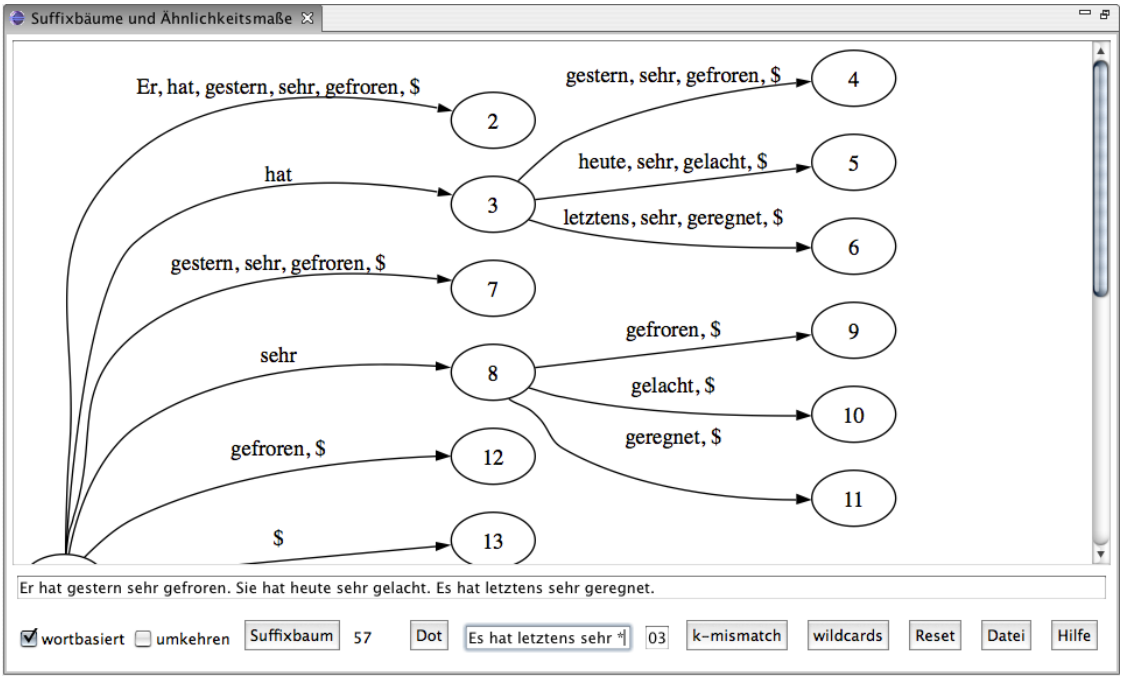

Abbildung 3.13.: Screenshot der im Rahmen dieser Arbeit entstandenen Software zur Visualisierung von Suffixbäumen und zwei der vorgestellten Matching-Algorithmen.

auf Grundlage des in Andersson *et al.* (1999) beschriebenen Verfahrens mit linear wachsender Laufzeit (siehe Abschnitt 3.2.3 auf Seite 40). Die graphische Oberfläche (*graphical user interface*, GUI) ist als Eclipse Plug-in[12] implementiert.

In der Software sind darüber hinaus Java-Implementationen und Visualisierungen der Algorithmen zum Matching mit *k-mismatch* (Gusfield 1997, 200) und *wildcards* (Gusfield 1997, 199) enthalten, sowie angepasste Versionen für wort- bzw. satzbasiertes Matching. Abbildung 3.13 zeigt einen Screenshot der Anwendung.

12 Eclipse Platform: <http://www.eclipse.org>

Kapitel 4

Komplexe Prädikate

Strukturell-Funktionale Analyse Komplexer Prädikate im Patwa, Allgemeine Sprachwissenschaft, Hauptseminar *Komplexe Prädikate* bei Dr. Dagmar Jung, Sommersemester 2006.

4.1. Überblick und Abgrenzung

Gegenstand dieser Arbeit ist die Untersuchung von komplexen Prädikaten im *Patwa* (Jamaican Creole, Patois) unter Anwendung des typologieorientierten, strukturell-funktionalen (Butler 2003) Modells *Functional Grammar* (FG) sowie dessen Nachfolger, *Functional Discourse Grammar* (FDG). Abschnitt 4.1.1 beschreibt kurz das Patwa, Abschnitt 4.1.2 beschreibt komplexe Prädikate und grenzt die im Patwa vorkommenden seriellen Verbkonstruktionen von anderen komplexen Prädikaten ab. Anschließed wird in den Abschnitten 4.1.3 und 4.1.3 kurz FG und FDG beschrieben. Nach einer Übersicht über serielle Verbkonstruktionen im Patwa in Abschnitt 4.2 findet sich in den folgenden Abschnitten dann die Analyse einer seriellen Verbkonstruktionen des Patwa in FG (Abschnitt 4.3) und FDG (Abschnitt 4.4).

4.1.1. Patwa (Jamaican Creole)

Patwa ist eine westatlantische, englischbasierte Kreolsprache mit einer Sprecherzahl von insgesamt 3,181,171, wovon 2,665,636 auf Jamaika leben (2001). Außer auf Jamaika wird Patwa in Teilen von Canada, Costa Rica, der Dominikanischen Republik, Panama, dem UK und den USA gesprochen.[1] Patwa weist SVO-Wortstellung und ein Nominativ-Akkusativ Kasussystem[2] auf.

4.1.2. Komplexe Prädikate

Unter dem Begriff "Komplexe Prädikate" werden mitunter sehr unterschiedliche Konstruktionen mit zwei oder mehr prädikativen Elementen zusammengefasst (Alsina *et al.* 1997). Der Schwerpunkt dieser Arbeit liegt auf seriellen Verbkonstruktionen im Patwa. Eine allgemeinere Darstellung von seriellen Verbkonstruktionen in den Sprachen der Welt findet sich in Durie (1997). Auf andere

1 Ethnologue, http://www.ethnologue.com/show_language.asp?code=jam
2 In Bezug auf die Wortstellung, es gibt keine morphologische Kasusmarkierung, Patwa ist eine isolierende Sprache

komplexe Prädikate, nämlich *light verb* Konstruktionen und Resultativ-Konstruktionen, wird zum Zwecke der Abgrenzung eingegangen.

Serielle Verbkonstruktionen

Serielle Verbkonstruktionen bestehen aus mehreren Verben (was serielle Verbkonstruktionen von *light verb* Konstruktionen abgrenzt, siehe auch Abschnitt 4.1.2) ohne Konjunktionen, mit nur einen strukturellen Subjekt sowie nur einer Markierung für Tempus und Aspekt (falls vorhanden). Darüber hinaus gelten folgende Einschränkungen (Durie 1997, 291, Lin 2004, 95):

- Der Verbkomplex drückt ein einzelnes Ereignis aus (siehe auch Abschnitt 4.4.2).
- Die Verben teilen sich mindestens ein Argument, etwa den Agens oder den Patiens (siehe auch Abschnitt 4.4.2).
- Kein Verb ist ein Argument eines anderen oder in dieses eingebettet (dies grenzt serielle Verbkonstruktionen etwa von Resultativ-Konstruktionen ab, siehe auch Abschnitt 4.1.2).

Light-Verb-Konstruktionen

Bei *light verb* Konstruktionen handelt es sich um Konstruktionen, die seriellen Verbkonstruktionen sehr ähnlich sind, bei denen jedoch ein *light verb* (welches wie die seriellen Verben aus einer geschlossenen Wortklasse stammt) mit Vertretern anderer Wortarten (Nomen, Adjektiv, Adverb oder Verb) ein komplexes Prädikat bildet, während in seriellen Verbkonstruktionen nur Verben das komplexe Prädikat bilden. Darüber hinaus wird in *light verb* Konstruktionen das *light verb* vom *heavy verb* (oder auch *main verb*) regiert. Mitunter wird auch bei morphologiereichen Sprachen von *light verbs* und bei isolierenden Sprachen von *serial verbs* gesprochen. Durie (1997) etwa behandelt jedoch serielle Verbkonstruktionen auch in morphologiereichen Sprachen.

Resultativ-Konstruktionen

Bei Resultativ-Konstruktionen handelt es sich um resultative Koprädikationen, wie in *Er aß das Essen auf*. Diese werden unterschieden von depiktiven Koprädikationen, wie in *Er aß das Essen hastig*.

Eine Konstruktion im Patwa, die als Resultativ-Konstruktion analysiert werden kann ist die mitunter auch als 'Resultativ-Perfektiv' bezeichnete Konstruktion in (4.1)[3] (Siegmund 2004). Es handelt sich nicht um eine serielle Verbkonstruktionen im engeren Sinn (siehe Abschnitt 4.1.2), da das *don* das *giv* näher spezifiziert und damit eine Prädikat-Argument-Struktur vorliegt.

(4.1) *Ai* ***don*** ***giv*** *im* *a* *dairekshan*
I done give him a direction
'I have already given him an adress'

Patrick (2004, 7) beschreibt das *don* als kompletive Aspektmarkierung. Er stellt dabei jedoch heraus, dass das *don* im Gegensatz zu anderen TAM-Markierungen auch nach dem Verb stehen kann. Dies entspricht der Struktur serieller Verben (siehe Abschnitte 4.2 und 4.4.3). Resultativ-Konstruktionen im Patwa ähneln damit in ihrer Struktur seriellen Verbkonstruktionen.

3 Bei der verwendeten phonemischen Orthographie handelt es sich um die sog. Cassidy-Orthographie von 1961.

4.1.3. Strukturell-Funktionale Analyse

Eine Analyse der seriellen Verbkonstruktionen soll im Rahmen der typologieorientierten, strukturell-funktionalen (Butler 2003) *Functional Grammar* (FG) (Dik 1997a,b), sowie dessen Nachfolger, *Functional Discourse Grammar* (FDG) (Hengeveld & Mackenzie 2006) erfolgen.

Functional Grammar (FG)[4]

Functional Grammar (FG) wurde Ende der 1970er Jahre von Simon C. Dik in Amsterdam entwickelt – ausdrücklich als Gegenmodell zum Standard-Modell der Tranformationsgrammatik von Noam Chomsky (1965). Es ist das einzige vollständige solche Gegenmodell, das außerhalb des MIT (wo Chomsky das Standardmodell entwickelt hatte) entstanden ist. Nach dem Tod Diks 1995 wurde die Theorie zunächst vor allem durch seinen Mitarbeiter Kees Hengeveld weiterentwickelt. Heute wird die Theorie unter der Bezeichnung *Functional Discourse Grammar* (FDG, siehe Abschnitt 4.1.3) vor allem durch Kees Hengeveld und Lachlan Mackenzie weiterentwickelt.

Die zentrale Annahme Diks über Sprache ist ihr zweckgebundener Charakter als Mittel zur Kommunikation. Dik rückt damit die Funktion der Sprache in den Mittelpunkt. In diesem Sinne ist die Bezeichnung *Functional Grammar* zu sehen: Ein sprachliches Modell, das von der Funktion der Sprache, statt von ihrer äußeren Form ausgeht. Mit dieser zentralen Annahme fordert Dik eine Abkehr von der früher häufig angewandten heuristischen Reduktion der Ausblendung der Pragmatik. Konkret ist jedoch in der Behandlung der Pragmatik bei Dik nicht die allgemeine Pragmatik im Sinne von Sprechakten und Sprache als Handlung gemeint, sondern der Bereich der Diskurspragmatik, im Wesentlichen also das Verhältnis der Informationsstruktur eines sprachlichen Ausdrucks zu seiner Realisierung, etwa bei der Behandlung von Topik und Fokus (Dik 1991, 267ff.).[5] Nach der Pragmatik ist die nächst wichtigste sprachliche Ebene aus Diks Sicht die Semantik, die – selbst von der Pragmatik beeinflusst – ihrerseits Einfluss auf die Syntax hat.

Neben dieser funktionalen Ausrichtung is FG zugleich ein formales Modell, da es zur Notation eine formale Sprache (Kalkül, lat. *calculus*) verwendet. Dadurch ist das Modell als Computerprogramm implementierbar und somit testbar (siehe auch Abschnitt 4.5).

Functional Discourse Grammar (FDG)

FDG ist der Nachfolger der in Abschnitt 4.1.3 beschriebenen FG. Ziel von FDG ist die Entwicklung eines Rahmens zur systematischen Beschreibung aller menschlichen Sprachen, insbesondere nicht nur unter Einbeziehung der syntaktischen und morphologischen, sondern auch der pragmatischen und semantischen Typologie. Den generellen Aufbau von FDG beschreibt Abbildung 4.1 auf Seite 56, wobei zu beachten ist, dass lediglich die *Grammatical Component* die FDG ist, eingebettet in eine allgemeinere Theorie verbaler Interaktion (Hengeveld & Mackenzie 2006).

Neuerungen gegenüber dem klassischen Model sind insbesondere eine top-down Organisation – die bei der Konzeptualisierung und nicht wie bei FG bei der Auswahl der lexikalischen Elemente beginnt – und eine stärkere Berücksichtigung der Pragmatik, sowie allgemein eine stärkere Umsetzung der schon von Dik aufgestellten Forderungen nach psychologischer, pragmatischer und

4 Dieser Abschnitt ist eine überarbeitete Version eines Ausschnitts aus Steeg (2006).

5 Die Pragmatik bekommt in FDG eine größere Bedeutung, siehe Abschnitte 4.1.3 und 4.4.1.

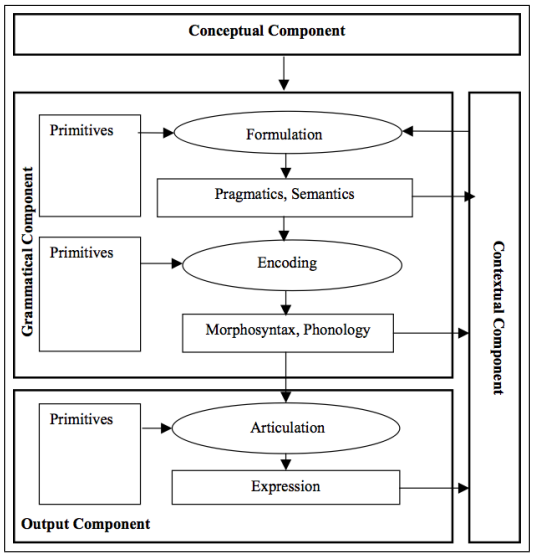

Abbildung 4.1.: FDG als Grammatikkomponente in einer allgemeineren Theorie verbaler Interaktion (Hengeveld & Mackenzie 2006, 669)

typologischer Adäquatheit. Mit letzterer einher geht etwa die Aufgabe der in FG angenommenen sprachunabhängigen Darstellungen von Strukturen (siehe Abschnitte 4.3.2 und 4.4.2).

4.2. Serielle Verbkonstruktionen im Patwa

Das Patwa verfügt über serielle Verbkonstruktionen (siehe Abschnitt 4.1.2), die vermutlich Substrat-Relikte[6] darstellen (Patrick 2004). Serielle Verbkonstruktionen haben im Allgemeinen den folgenden Aufbau (Lin 2004, 93):

$$(\mathrm{NP}_1)\ \mathbf{V}_1\ (\mathrm{NP}_2)\ \mathbf{V}_2\ (\mathrm{NP}_3)\ \ldots$$

Serielle Verbkonstruktionen können verschiedene Funktionen haben, die durch serielle Verben,[7] die der Funktion semantisch nahe stehen, ausgedrückt werden (siehe Tabelle 4.1).

Entsprechend der in einer Sprache möglichen Funktionen von seriellen Verbkonstruktionen können Kreolsprachen mit seriellen Verbkonstruktionen in zwei große Gruppen eingeteilt werden: Jene mit Instrumental mit *take* und jene ohne (Muysken & Veenstra 1995, 290). Im Patwa gibt es serielle Verbkonstruktionen mit *take* wie in (4.2) (Patrick 2004, 290), sowie mit den meisten anderen Funktionen (Patrick 2004, 21, Muysken *et al.* 1978, 131).

6 Das Substrat bilden die beim Sprachkontakt verdrängten oder ersetzen Sprache, hier also die Muttersprachen der verschleppten afrikanischen Sklaven, vor allem Sprachen der westafrikanischen Kwa-Familie (Patrick 2004).

7 Inwiefern für serielle Verbkonstruktionen nur bestimmte Verben möglich sind, ist nicht völlig geklärt (Muysken & Veenstra 1995, 291), es scheinen aber stets Verben zu sein, die Aktivität anzeigen (Lin 2004).

locational	*go*	direction away
	come	direction torwards
	surround	around
	be	locative
argument	*give*	benefactive
	take	intrumental, comitative, object
	say	finite complementizer
aspectual	*finish*	perfective
	return	iterative
	be	continuative
degree	*pass*	comparative, excessive
	suffice	enough

Tabelle 4.1.: Funktionen von seriellen Verbkonstruktionen in Kreolsprachen (Muysken & Veenstra 1995, 290f.)

(4.2) *Im* ***tek*** *naif* ***kot*** *mi.*
He took knife cut me
'He cut me with a knife'

Aufgrund der reichhaltig vorhandenen seriellen Verbkonstruktionen kann man Patwa als stark (inklusive Instrumental mit *take*, siehe Muysken & Veenstra 1995, 290) serialisierende Sprache charakterisieren, die sich damit in einer Gruppe mit anderen karibischen und westafrikanischen Kreolsprachen wie Sranan, Haitianisch oder Krio befindet.

Neben der beschriebenen instrumentalen Funktion werden serielle Verbkonstruktionen auch zur Einführung anderer Funktionen verwendet. So werden etwa Konstruktionen mit *se* ('say') statt Konjunktionen zum Einleiten von Nebensätzen verwendet, etwa in (4.3) (Muysken *et al.* 1978, 130). In Patrick (2004, 20) wird *se* als (sich aus einer seriellen Verbkonstruktionen entwickelten) Konjunktion und unabhängig von seriellen Verbkonstruktionen analysiert.[8]

(4.3) *Mi* ***miin*** ***se*** *yu* *fi* *go.*
I mean say you must go
'I mean that you must go'

Die Verwendung von *paas* ('surpass') in seriellen Verbkonstruktionen zur Bildung des Komparativ wie in (4.4) ist im Patwa möglich, aber heute selten (Patrick 2004, 22).

(4.4) *Manggo* ***de*** *a* *yaad* ***paas*** *plenti.*
Mango be there PROG yard pass plenty
'A great many mangoes are in the yard'

8 Als Grund für diese Analyse gibt Patrick seine synchrone, Mesolekt-orientierte (vgl. Patrick 1999) Beschreibung an (Patrick 2004, 5).

In ähnlicher Weise werden auch für andere Funktionen serielle Verben verwendet, die der Funktion semantisch nahe stehen, etwa Kausativität durch *make* in (4.5) (Muysken *et al.* 1978, 130) oder Benefaktiv durch *give* in (4.6) (Patrick 2004, 22).

(4.5) *Jan ena **fait** **mek** im kluoz tieraf.*
John PAST PROG fight make his clothes tear off
'John's clothes are torn because he was fighting'

Bei seriellen Verbkonstruktionen mit drei Verben ist eins der Verben immer direktional, wie etwa in (4.6) (Patrick 2004, 22).

(4.6) ***Kya** di buk **kom** **gi** mi.*
Carry the book come give me
'Bring the book for me'

4.3. Die Instrumental-Konstruktion in FG

Im folgenden Abschnitt soll die serielle Verbkonstruktion mit Instrumental-Funktion aus (4.2) im Rahmen von FG beschrieben werden.

4.3.1. Aufbau des Grammatikformalismus[9]

Der Grammatik-Formalismus von FG besteht im wesentlichen aus der Beschreibung abstrakter Ausdrücke, der *underlying clause structures* (UCS), die schrittweise aus Prädikaten und Termen (die Argumente der Prädikate) gebildet werden und die durch Ausdrucksregeln (*expression rules*) zu konkreten sprachlichen Äusserungen in Bezug gesetzt werden oder diese erzeugen.

Prädikate sind Ausdrücke für Eigenschaften oder Relationen. Ein Unterschied der Prädikate in FG zur klassischen Prädikatenlogik ist die Verwendung sogenannter Restriktoren. Wenn in FG Prädikate zusammengesetzt werden, geschieht dies durch die Verwendung dieser Restriktoren, geschrieben als ":", etwa in Form von (4.7).

(4.7) `japanisch(x) : buddhistisch(x)`

Dies lässt sich paraphrasieren mit "Die Menge der x, für die gilt: x ist japanisch, eingeschränkt auf die Menge der x, für die gilt: x ist buddhistisch". Die entsprechende prädikatenlogische Form wäre (4.8), wobei das "&" ein prädikatenlogisches "UND" ist. Der entsprechende Sachverhalt ist in beiden Fällen gleich. Der Unterschied besteht darin, dass das prädikatenlogische "&" umkehrbar ist. Bei den Restriktoren ist dies nicht der Fall und sie sind damit in der Lage, den Unterschied der sprachlichen Äusserungen *Der japanische Buddhist* und *Der buddhistische Japaner* zu erfassen (Dik 1997a, Kap. 6.2).

(4.8) `japanisch(x) & buddhistisch(x)`

9 Dieser Abschnitt ist eine überarbeitete Version eines Ausschnitts aus Steeg (2006).

(4.11) past: (*cut* ((x_1: *im*)$_{Agent}$ (x_2: *mi*)$_{Patient}$ (x_3: *knife*)$_{Intrument}$))
Im tek naif kot mi; *He cut me with a knife*

(4.12) past:(
cut (
(x_1: *im*)$_{Agent}$
(x_2: *mi*)$_{Patient}$
(x_3: *knife*)$_{Intrument}$
)
)

Abbildung 4.2.: Underlying Clause Structure: Semantische Struktur

Prädikate sind stets Teil eines Prädikatrahmens, der die Eigenschaften des Prädikats beschreibt. Ein Beispiel für den Prädikatrahmen eines transitiven Verbs wäre etwa (4.9).

(4.9) *throw*[V](x_1:<animate >)$_{Agent}$ (x_2:<concrete>)$_{Patient}$ (x_3)$_{Direction}$

Zunächst erscheint die Wortform (*throw*), anschließend die Wortart (V). Im Folgenden sind die Argumentpositionen des Verbs beschrieben. Das Argument in der Mitspielerposition mit der semantischen Rolle des Agens muss belebt (*animate*) sein, der vom Sachverhalt betroffene Mitspieler (Patient) – hier der geworfene Gegenstand – muss konkret (*concrete*) sein und das dritte Argument (Location) unterliegt keiner Selektionsbeschränkung dieser Art. Zu solchen nuklearen Prädikaten können sogenannte Satelliten hinzukommen, etwa zur zeitlichen Präzisierung des Prädikats.

Der zweite wesentliche Bestandteil einer UCS sind neben Prädikaten die Terme. Formal sind Terme die Argumente der Prädikate, semantisch sind es Ausdrücke, die Entitäten referenzieren[10]. Terme repräsentieren also die Entitäten, die durch ein Prädikat zueinander in Beziehung gesetzt werden. Eine Prädikation, die zwei Terme (*the garden* und *the dog*) enthält wäre z.B. (4.10).

(4.10) present: (def sing x_1: *garden* [N])$_{Location}$ (def sing x_2: *dog* [N])
The dog is in the garden

4.3.2. Semantik: Underlying Clause Structures

Die instrumentale Funktion in (4.2), die im Deutschen oder im Englischen mit einer Präposition realisiert wird, wird im Patwa mit einer seriellen Verbkonstruktionen realisiert. Die zugunde liegende UCS ist in allen Fällen jedoch (4.11) bzw. (4.12) (mit Einrückungen), da die UCS sprachunabhängig sein soll.

10 Genau genommen schreibt Dik (1991, 255), daß Terme den Adressaten instruieren, eine Entität zu identifizieren, die dem Profil des Terms entspricht.

4.3.3. Morphosyntax: Expression Rules

Die Realisierung als Präposition oder als serielle Verbkonstruktionen ist in den *expression rules* festgehalten. So würden die *expression rules* etwa für das Englische eine Regel wie (4.13) enthalten, während für das Patwa eine Regel wie (4.14) zur Realisierung von Äußerungen mit instrumentaler Funktion enthalten wäre.

(4.13) $Instrument \rightarrow N_{Agent} + V + N_{Patient} + with + N_{Instrument}$

(4.14) $Instrument \rightarrow N_{Agent} + tek + N_{Instrument} + Verb + N_{Patient}$

Der Anspruch, die UCS soll sprachunabhängig sein, ist jedoch schwer aufrechtzuerhalten, da etwa Konstruktionen wie in (4.15) in unterschiedlichen Sprachen unterschiedliche lexikalische Elemente verwenden (hier *kya* ('carry') im Patwa und *bring* im Englischen).

(4.15) $kya\ ((x_1)_{Agent}\ (x_2\text{: } buk)_{Patient}\ (x_3\text{: } mi)_{Beneficiary})$
Kya di buk kom gi mi, *Bring the book for me*

Die Sprachunabhängigkeit der Darstellungen wird in FDG – dem Nachfolgemodell von FG – zugunsten einer stärkeren typologischen Adäquatheit aufgegeben (siehe Abschnitte 4.1.3 und 4.4).

4.4. Die Instrumental-Konstruktion in FDG

Die einzelnen Ebenen von FDG sind hierarchisch organisiert und entsprechen dem generellen Schema in (4.16) (Hengeveld & Mackenzie 2006).

(4.16) $(\pi\alpha_1 : [\text{(complex) head}]\ (\alpha_1) : \sigma(\alpha_1))_\varphi$

Hierbei steht α_1 für Variablen auf den entsprechenden Schichten (*layers*). Diese werden restringiert durch ein – möglicherweise zusammengesetztes (*complex*) – Kopfelement (*head*), sowie durch optionale weitere Modifizierer (*modifier*) σ. Desweiteren kann α durch weitere Operatoren (*operators*) π sowie eine Funktion (*function*) φ weiter spezifiziert werden. Hierbei repräsentieren Modifizierer lexikalische Strategien, währen Operatoren (nicht-relational) und Funktionen (relational) grammatikalische Strategien repräsentieren (Hengeveld & Mackenzie 2006, 671).

4.4.1. Pragmatik: Interpersonal Level

Das *interpersonal level* gliedert sich in *moves* (M), die aus einem oder mehreren *acts* (A) bestehen. Diese sind charakterisiert durch ihren *illocutionary value* (F), die *speech participants* (P) und den kommunizierten *content* (C), welcher aus mehreren *subacts* der *ascription* (T) und *reference* bestehen (R).

(4.17)

Im	***tek***	*naif*	***kot***	*mi.*
R_1	T_1	R_2	T_2	R_3

Nimmt man eine Zuordnung der *subacts* wie in (4.17) vor, ergibt sich, eingesetzt in das Schema in (4.16) auf dem *interpersonal level* die Gesamtstruktur in (4.18) bzw. (4.19) (mit Einrückungen) (vgl. Van Staden 2006).

(4.18) (M_1:[(A_1: [(F_1:DECL) (P_1) (P_2)
(C_1:[[T_1:*tek*((R_1:*im*)(R_2:*naif*))][T_2:*kot*((R_1)(R_3:*mi*))]])])])

(4.19)
```
(M1:[ (A1: [ (F1:DECL) (P1) (P2)
        (C1:[
            [T1:tek(
                    (R1:im)
                    (R2:naif)
            )]
            [T2:kot(
                    (R1)
                    (R3:mi)
            )]
        ])
    ]) ])
```

Abbildung 4.3.: Interpersonal Level: Pragmatische Struktur

4.4.2. Semantik

Ein wichtiges Merkmal von seriellen Verbkonstruktionen – das *argument sharing* – wird in Rahmen von FDG auf dem *representational level* behandelt.

Argument Sharing

Ein entscheidendes Kriterium für serielle Verbkonstruktionen im engeren Sinn ist das *argument sharing*, worunter zu verstehen ist, daß die Verben in der Konstruktion ein gemeinsames Argument haben. So teilen sich etwa in (4.20) V_1 und V_2 den Agens der Konstruktion (*im*).

(4.20) *Im* ***tek*** *naif* ***kot*** *mi.*
He took knife cut me
'He cut me with a knife'

Representational Level

Der *representational level* enthält Propositionen (p), bestehend aus einem oder mehreren *events* (e), die wiederum aus einem oder mehreren *predicate frames* bestehen (f). Die einzelnen Schichten können modifiziert werden, etwa in Bezug auf Modalität, Negation und Tempus (bei *events*) oder Aspekt (bei *frames*).

(4.21) *Im* ***tek*** *naif* ***kot*** *mi.*
x_1 f_1 x_2 f_2 x_3

(4.22) p_1:(e_1:past[(f_1:*tek*((x_1:*im*)$_{Ag}$(x_2:*naif*)$_{Inst}$)) (f_2:*kot*((x_1)$_{Ag}$(x_3:*mi*)$_{Pat}$))])

(4.23) p_1:(
e_1: past[
(f_1:*tek*(
(x_1:*im*)$_{Ag}$
(x_2:*naif*)$_{Inst}$
))
(f_2:*kot*(
(x_1)$_{Ag}$
(x_3:*mi*)$_{Pat}$
))
]
)

Abbildung 4.4.: Representational Level: Semantische Struktur

Bei einer Zuweisung von Entitäten (x) und Relationen (f) wie in (4.21) und einer Einsetzung in das generelle Schema in (4.16) erhalten wir auf dem *representational level* die Gesamtstruktur in (4.22) bzw. (4.23) (mit Einrückungen). Die Darstellung beruht auf der Analyse der seriellen Verbkonstruktion als ein einzelnes Ereignis, wofür mutterprachliche Intuition sowie semantische Analysen sprechen (Durie 1997, 291). Eine Analyse mit zwei separaten Ereignissen wäre (4.24) (vgl. Van Staden 2006).

(4.24) p_1:(e_1:past[f_1:*tek*((x_1:*im*)$_{Ag}$(x_2:*naif*)$_{Inst}$)])(e_2:past[f_2:*kot*((x_1)$_{Ag}$(x_3:*mi*)$_{Pat}$)])

Das *argument sharing* der beiden prädikativen Elemente V_1 und V_2 ist hier ausgedrückt durch die Variable x_1, die in beiden Prädikatrahmen (f_1 unf f_2) als Argument enthalten ist.

Die Analyse auf semantischer Ebene in FDG ist auf diese Weise im Gegensatz zur beschriebenen Analyse in FG typologisch adäquat. Dem geopfert wurde die sprachunabhängige Darstellung. Es ist hier vielleicht zu überlegen, ob auf einer anderen Ebene als der Beschreibung der grammatikalischen Struktur die Gemeinsamkeit von strukturell unterschiedlichen, aber in der Welt identischen Ereignissen (hier etwa Person A schneidet Person B mit einem Messer) ausgedrückt werden kann. Möglicherweise könnten solche Strukturen in einer konzeptuellen Komponente einer allgemeineren Theorie sprachlicher Interaktion dargestellt werden (vgl. Abbildung 4.1 auf Seite 56).

4.4.3. Syntax

Die Beschreibung des syntaktischen Phänomens der Köpfigkeit in seriellen Verbkonstruktionen kann in FDG auf dem *morphosyntactic level* erfolgen.

(4.27) $[[[[im]_{N_1}]_{shared}$ **[**$[tek]_{V_1}$ $[naif]_{N_2}]_{serial}$ $[[kot]_{V_2}$ $[mi]_{N_3}]_{head}]_{SVC}]_{Cl}$

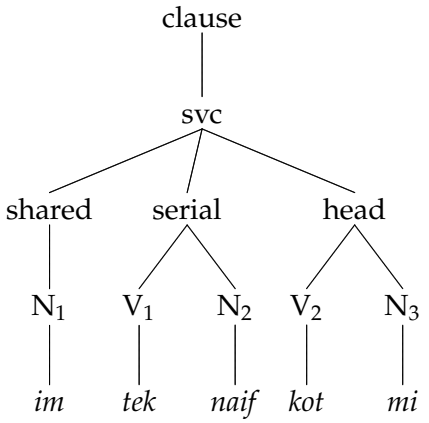

Abbildung 4.5.: Morphosyntactic Level: Konstituentenstruktur

Köpfigkeit

Im Patwa ist wie in den anderen karibischen Kreolsprachen und in westafrikanischen Sprachen mit seriellen Verbkonstruktionen der Kopf[11] der seriellen Verbkonstruktionen in der Regel V_1 (Lin 2004, 99), außer bei *take* (Muysken & Veenstra 1995, 290). Allerdings sind eventuell Ausnahmen möglich, siehe (4.25) (Patrick 2004, 22), wo V_1 das serielle Verb und V_2 der Kopf ist.

(4.25) *Dis naga man* ***kom collar*** *me de same like a say me da him sexis.*
This black man come collar me the same like PROG say me is his sex
'This black man comes and collars me just as if I were the same sex as he'

Morphosyntactic Level

Bei einer Zuordnung von Wortarten und einer Strukturierung von Kopfelement (*head*) und seriellem Element (*serial*) wie in (4.26) (vgl. Abschnitte 4.1.2 und 4.4.3) ergibt sich eine mögliche Gesamtdarstellung der Konstituentenstruktur[12] des Satzes (*clause*) mit einer seriellen Verbkonstruktion (*svc*) wie in (4.27) (in indexierter Klammerung) bzw. (4.28) (als Baum).

(4.26) *Im* ***tek*** *naif* ***kot*** *mi.*
$[N_1]_{shared}$ $[V_1$ $N_2]_{serial}$ $[V_2$ $N_3]_{head}$

11 Der Kopf einer seriellen Verbkonstruktionen enhält (falls vorhanden) Negation und TAM-Markierung, im Patwa in Form von Adpositionen an V_1 (meist Präpositionen, aber in (4.4) offenbar als Postposition). Das serielle Verb hingegen kann nicht negiert werden und hat keine Markierung, es ist ein nacktes (*bare*) Verb (Lin 2004, 99).

12 Konstituentenstrukturen wurden erstmals durch Bloomfield (1933) beschrieben (Lyons 1968, 209ff.).

Die Darstellungen in (4.27) bzw. (4.28) enthalten keine weitergehenden Informationen über die Beschaffenheit des *argument sharing*. Diese finden sich in der Darstellung auf dem *representational level* (Abschnitt 4.4.2).

4.4.4. Phonologie

Auf der phonologischen Ebene schließlich finden sich Darstellungen in IPA-Notation wie in (4.29) sowie Silbenstrukturanalysen. So könnte für den Fall von seriellen Verbkonstruktionen auf dieser Ebene etwa die Tatsache festgehalten werden, dass die Artikulation eines Satzes mit einer seriellen Verbkonstruktion wie die eines Satzes mit nur einem Verb erfolgt (Durie 1997, 291).

(4.29) *Im* ***tek*** *naif* ***kot*** *mi.*
[im] [tɛk] [naif] [kot] [mi]

In einer komplett ausgestalteten FDG kommen zu der phonologischen noch eine graphologische bzw. für Zeichensprachen eine gestische Ebene (Hengeveld & Mackenzie 2006).

4.4.5. Die Ebenen in der Übersicht

In Tabelle 4.2 findet sich eine Zusammenfassung der Darstellungen auf den verschiedenen Ebenen, in Tabelle 4.3 die Zuordnung der Symbole der einzelnen Ebenen zueinander. Wie Tabelle 4.3 zeigt, liegt hier eine eins-zu-eins Beziehung zwischen den Elementen auf den verschiedenen Ebenen vor. In nicht-isolierenden Sprachen könnten sich Elemente des *interpersonal level* und des *representational level* etwa auf Morpheme beziehen statt auf freie Formen.

IL	$(M_1[(A_1:[(F_1:\text{DECL})(P_1)(P_2)(C_1:[[T_1((R_1)(R_2))][T_2((R_1)(R_3))]])])])$
RL	$p_1:(e_1:\text{past}[(f_1:tek((x_1:im)_{Ag}(x_2:naif)_{Inst}))(f_2:kot((x_1)_{Ag}(x_3:mi)_{Pat}))])$
ML	$[[[[im]_{N_1}]_{shared}\ [[tek]_{V_1}\ [naif]_{N_2}]_{serial}\ [[kot]_{V_2}\ [mi]_{N_3}]_{head}]_{SVC}]_{Cl}$
PL	[im] [tɛk] [naif] [kot] [mi] ...
GL	*im tek naif kot mi*

Tabelle 4.2.: Zusammenfassung der Darstellungen auf den verschiedenen Ebenen

IL	RL	ML	PL	GL
T_1	f_1	V_1	[tɛk]	*tek*
R_1	x_1	N_1	[im]	*im*
R_2	x_2	N_2	[naif]	*naif*
T_2	f_2	V_2	[kot]	*kot*
R_3	x_3	N_3	[mi]	*mi*

Tabelle 4.3.: Zuordnung der Symbole der verschiedenen Ebenen zueinander

4.5. Rechnergestützte Implementierung

Die Arbeit in einem Modell mit formaler Notation erhöhnt nicht nur die Konsistenz von linguistischen Beschreibungen sondern bietet auch die Möglichkeit einer rechnergestützten Implementierung. Die Idee einer rechnergestützten Implementierung ist ein zentraler Aspekt nicht nur von FG (Dik 1997a, 1) und anderen linguistischen Modellen, sondern ein wertvolles Hilfsmittel für die Linguistik im Allgemeinen ("Linguistics may learn from being applied", Bakker 1994, 4). Eine rechnergestützte Implementierung bietet potentiell unterschiedlichste Möglichkeiten auf verschiedenen Gebieten – von der rechnergestützten, linguistisch fundierten Datenerfassung bis zum automatischen Ermitteln von Strukturen (Parsing).

Zur Implementierung eines modularen Modells wie FDG bietet sich eine modulare Systemarchitektur an. Ein Beispiel für eine modulare Implementierung von FG stellt etwa das in Steeg *et al.* (2006) beschriebene System dar, dessen Architektur sich auch zur Implementierung von FDG eignet.

4.6. Fazit

Patwa ist eine Kreolsprache mit seriellen Verbkonstruktionen – einer Form von komplexen Prädikaten. Zur Beschreibung solcher Konstruktionen bietet FDG einen typologisch, semantisch und pragmatisch adäquaten Rahmen für Strukturbeschreibungen in formaler Notation auf allen sprachlichen Ebenen.

Die Analyse im Rahmen einer linguistischen Theorie mit formaler Notation[13] ermöglicht dabei nicht nur konsistente Analysen, sondern erhöhnt auch die Anwendbarkeit linguistischer Erkenntnisse sowie die rechnergestützte Umsetzbarkeit. Dies trägt nicht nur zur Verfügbarkeit von Werkzeugen zur linguistischen Analyse sondern ebenso zur Überprüfbarkeit linguistischer Modelle und damit zum Fortschritt in der Linguistik selbst bei.

13 Als Vertreter der generativen Tradition beschreibt Jackendoff (2002, 6) sprachliche Phänomene auf Ebenen mit anderer Notation, jedoch den Ebenen der FDG entsprechenden Inhalten, insbesondere in den Bereichen Syntax und Semantik. Dies nähert die entgegengesetzten strukturalistischen und funktionalistischen Traditionen einander an. Moderne Grammatikformalismen berücksichtigen die Notwendigkeit von Beschreibungen auf verschiedenen Ebenen (mindestens für Syntax und Semantik). Die Unterschiede in der Notation verhindern jedoch trotzdem bislang eine gemeinsame formal-deskriptive Sprache der Linguistik (vergleichbar etwa mit Algebra, Geometrie oder Analysis).

Kapitel 5

Linguistische Evidenz

Eine kurze Einführung in die Korpuslinguistik, Allgemeine Sprachwissenschaft, Referat im Hauptseminar *Linguistische Evidenz* bei Prof. Dr. Hans-Jürgen Sasse, Wintersemester 2006–2007.

5.1. Korpuslinguistik

Korpuslinguistik ist eine sprachwissenschaftliche Arbeitsweise und umfasst verschiedene Tätigkeiten, die mit der Erstellung und der Auswertung von Textkorpora (Köhler 2005) sowie mit der Erstellung von Werkzeugen für die beiden ersten Tätigkeiten zu tun haben. McEnery (2003) (Hauptquelle) definiert ein Textkorpus als "a large body of linguistic evidence". Damit sind Korpora eine sehr wertvolle Quelle für die linguistische Arbeit. Korpuslinguistische Methoden werden in verschiedenen Bereichen der Sprachwissenschaft verwendet: etwa in der allgemeinen Sprachwissenschaft, der Computerlinguistik und dem *Natural Language Processing* (NLP). Somit ist Korpuslinguistik kein eigener Teilbereich der Linguistik sondern eine Arbeitsweise, die innerhalb der genannten Teilbereiche verwendet wird und auf verschiedene Ebenen der Sprache (wie Morphologie, Syntax, Semantik oder Pragmatik) angewendet werden kann (McEnery & Wilson 1996, 2). Korpuslinguistik ist eine Form von Evidenz- oder datenbasierter (Köhler 2005) Linguistik. Im Grunde ist alle Linguistik vor Chomsky datenbasiert. McEnery & Wilson (1996) bezeichnen diese dementsprechend als frühe Korpuslinguistik. Eine Besonderheit der Arbeit mit Korpora im Gegesatz zu anderen Formen linguistischer Evidenz ist, dass nicht nur empirisch untersucht werden kann, ob eine Konstruktion möglich ist, sondern auch, wie häufig diese auftritt (Kennedy 1998, 8).

5.2. Korpora

Korpora können aus verschiedenen Quellen stammen, etwa aus aufgenommener Konversation (gesprochener Teil des British National Corpus BNC), Radionachrichten (IBM/Lancaster Spoken English Corpus) oder schriftlichen Veröffentlichungen (schriftlicher Teil des BNC). Korpora sind durch verschiedene Eigenschaften gekennzeichnet. Zunächst sind Korpora heute typischerweise maschinenlesbar. Da Korpora nicht nur in den oben genannten Bereichen der Linguistik sowie etwa zur literarischen Stilanalyse eingesetzt werden, handelt es sich bei Korpora um eine multifunktionale

Ressource (McEnery 2003). Ein Korpus ist darüber hinaus im Hinblick auf eine bestimmte Fragestellung organisiert. Diese Ausrichtung auf eine bestimmte Fragestellung bezeichnet McEnery (2003) als *sampling frame*, innerhalb dessen das Korpus ausgewogen und repräsentativ sein soll.

Als Beispiel nennt McEnery (2003) die Entwicklung eines Dialogsystems zum Verkauf von Eintritts- und Fahrkarten. Wenn hierfür ein Korpus zusammengestellt werden soll, sollten zum einen nur relevante Texte verwendet werden, d.h. Dialoge von Kartenverkäufen (dies entspricht dem gewählten *sampling frame*). Es sollten jedoch verschiedene Arten von Verkaufsgesprächen verwendet werden, etwa von Bus- und Flugzeugtickets sowie von Telefon- und Schalterverkäufen. Ausserdem sollten in jedem Bereich Gespräche verschiedener Personen verwendet werden, um Idiosynkrasien einzelner Sprecher zu vermeiden. So erhält man ein ausgewogenes und repräsentatives Korpus.

5.3. Korpusarten

McEnery (2003) unterscheidet verschiedene Arten von Korpora: Monolinguale Korpora sind Korpora mit Texten in einer einzigen Sprache. Vergleichbare Korpora sind Korpora aus Texten in verschiedenen Sprachen, die in Bezug auf *sampling frame*, Ausgewogenheit und Repräsentativität vergleichbar sind. Diese Korpora eignen sich etwa für den kontrastiven Sprachvergleich. Parallele Korpora sind Korpora, die zunächst aus Texten einer einzigen Sprache zusammengestellt und dann in andere Sprachen übersetzt werden. Solche Korpora eignen sich etwa für Systeme zur maschinellen Übersetzung, die aus Beispielübersetzungen lernen. Monitorkorpora werden laufend aktualisiert und dienen etwa zur Beobachtung von Sprachwandel (z.B. *Bank of English*).

Neben diesen Unterscheidungen können gesprochene von reinen Schriftkorpora unterschieden werden. Gesprochene Korpora (die monolingual, vergleichbar oder parallel sowie Monitorkorpora sein können) enthalten Informationen über die gesprochene Form des Textes. Dies kann bedeuten, dass es sich um akustisches Material handelt (solche Korpora sind schwerer zu verwenden, etwa zur Suche nach bestimmten Wörtern), oder auch um transkribierte Sprachdaten, wie im Fall des gesprochenen Teils des BNC (was u.a. zum Verlust prosodischer Feinheiten führt). Da diese beiden Formen Vor- und Nachteile haben, gibt es zunehmend Korpora, die aus akustischem und aus transkribiertem Material bestehen, deren Inhalte aufeinander abgestimmt sind, wodurch es möglich ist, auf die jeweils andere Form zuzugreifen (McEnery 2003). Solche multimodale Korpora (Evert & Fitschen 2001) können darüber hinaus auch Informationen über Mimik oder Gestik enthalten, etwa in Form von Videomaterial.

5.4. Geschichte

Eine frühe Form der Korpuslinguistik sind etwa Bibelkonkordanzen, die alle Wörter der Bibel alphabetisch sortiert in ihrem Kontext auflisten und ab dem 13. Jahrhundert entstanden. Käding erstellte und beschrieb 1897 ein großes, von Hand zusammengestelltes Korpus mit einem Umfang von 11 Mio. Wörtern (McEnery & Wilson 1996, 3). Seit den 1920ern gab es insbesondere in den USA und im UK eine Tradition des Wortzählens, um die häufigsten Wörter zu finden; die Anwendung lag hier vor allem im Bereich des Sprachlernens. In den 1930ern wurden von Vertretern der Prager Schule quantitative Untersuchungen etwa zur Häufigkeit von bestimmten grammatikalischen Konstruktionen unternommen (Kennedy 1998, 10).

In der heutigen, computerisierten Form gibt es die Korpuslinguistik seit den späten 1940er Jahren (McEnery 2003). Das Beispiel der Bibelkonkordanzen zeigt den radikalen Wandel durch die Entwicklung des Computers: Während das Erstellen der ersten Bibelkonkordanz 14 Jahre dauerte,[1] ist es durch die Entwicklung des Computers möglich geworden, innerhalb von Sekundenbruchteilen Konkordanzen beliebiger Texte zu erstellen; das Erstellen eines solchen Programms selbst ist innerhalb von Tagen oder Stunden möglich. Aufgrund dieser Unterschiede hat die Erfindung des Computers die Korpuslinguistik im heutigen Sinn erst ermöglicht.

In den 1950ern wurde das erste *vergleichbare* Korpus zusammengestellt, das auch schon den Grundlagen der *sampling frames*, der Ausgewogenheit und der Repräsentativität entsprach. In den 1980ern wuchsen Anzahl und Größe von Korpora, was sich in den 1990ern fortsetzte, als das BNC und die *Bank of English* Größen von 100 Mio. und 300 Mio. Wörtern erreichten. Zudem wuchs in den 1990ern die Anzahl paralleler Korpora. Köhler (2005) erwähnt eine "lange, vor allem europäische Tradition" der quantitativen Analyse empirischer Daten in der zweiten Hälfte des 20. Jahrhunderts, die jedoch aufgrund der dominierenden "formalen, Kompetenz-orientierten" Linguistik in den USA kaum rezipiert wurde. Erst seit einigen Jahren sei die Korpuslinguistik auch in den USA auf dem Vormarsch und wirke von dort auch wieder auf Europa zurück. McEnery (2003) ist der Ansicht, dass mit der fortschreitenden Entwicklung in der Korpuslinguistik zukünftig auch Probleme lösbar werden, die heute mit Mitteln der Korpuslinguistik nicht lösbar sind.

5.5. Korpusannotation

Annotierte Korpora sind Korpora, die mit verschiedenen Arten linguistischer Information angereichert sind (McEnery & Wilson 1996, 24). Ein Beispiel für Korpusannotationen ist etwa die Kennzeichnung von Wortarten (Part-of-Speech-Tags, POS-Tags). Andere mögliche Annotationen enthalten etwa Informationen über Stammformen (Stemming oder Lemmatisierung), syntaktische Struktur (solche Korpora werden auch Baumbanken genannt) sowie semantische, stilistische oder Informationen zur Diskursstruktur[2] (in abnehmender Häufigkeit und Verbreitung). Diese Anreicherung basiert immer auf einer bestimmten Interpretation der Daten. Diese stellt kein Hinzufügen neuer informationen dar, sondern macht lediglich implizit vorhandene Informationen explizit (McEnery 2003).

McEnery (2003) nennt vier Vorteile von Korpusannotationen: Zunächst eine verbesserte Nutzbarkeit der Korpora für eine größere Anzahl von Benutzern, seien dies Menschen, die einer bestimmten Fremdsprache unkundig sind oder Computerprogramme, die Sprache generell nicht verstehen können. Hier werden die Annotationen benötigt, können jedoch von dem, der sie benötigt nicht selbst erstellt werden. Darüber hinaus ist ein annotiertes Korpus auch für Benutzer, die die Analysen selbst vornehmen könnten viel schneller zu verwenden als ein nicht-annotiertes Korpus. Ein zweiter Vorteil ist die Wiederverwertbarkeit der bei der Analyse gewonnen Daten, etwa der ermittelten POS-Tags, der Stammformen, syntaktischen Strukturen etc. Als weiteren Vorteil nennt McEnery (2003) die Multifunktionalität der Daten, so kann bei einer Wiederverwertung ein ganz anderer Zweck verfolgt werden. Schließlich ist die explizite Formulierung und objektive Erfassung von Ergebnissen einer Analyse ein weiterer Vorteil.

1 http://de.wikipedia.org/wiki/Bibelkonkordanz

2 Beispiele verschiedener Korpusannotationen und weiteres Material gibt es unter: http://bowland-files.lancs.ac.uk/monkey/ihe/linguistics/corpus2/2fra1.htm

Korpusannotationen können automatisch, von Hand oder in einer kombinierten Form erstellt werden: Für Aufgaben wie das POS-Tagging oder Lemmatisieren für Sprachen wie Englisch, Französisch und Spanisch können Annotationen automatisch erstellt werden. Häufig aber wird das Ergebnis der maschinellen Annotation von Hand kontrolliert und erfolgt damit semi-automatisch, was immer noch ein schnelleres Annotieren als allein von Hand ermöglicht. Einige Bereiche erfordern aber auch rein manuelle Erstellung der Annotationen, etwa zur Darstellung anaphorischer und cataphorischer Relationen (McEnery 2003).

Als Argument gegen die manuelle oder semi-automatische Annotation ist vorgebracht worden, dass diese nur eine geringe Konsistenz aufweisen könne, da menschliche Annotatoren vergleichbare Stellen im Korpus nicht immer übereinstimmend annotieren. Es wurden jedoch Untersuchungen unternommen, die gezeigt haben, dass bei geschulten Annotatoren der leichte Rückgang der Konsistenz mehr als kompensiert wurde durch eine gesteigerte Genauigkeit der Annotationen (McEnery 2003, 457).

5.6. Sprachverarbeitung

McEnery (2003) bezeichnet Textkorpora als "the raw fuel of NLP" und damit als Grundlage und Voraussetzung des NLP. Der Grund hierfür besteht darin, dass annotierte Korpora es Coputerprogrammen ermöglichen, die Intuitionen von Experten, die die Annotationen erstellt oder verbessert haben, in Bezug zu den Texten zu setzen und so menschliche Intuitionen zu reproduzieren. Ein Beispiel wäre etwa ein POS-Tagger, der aus annotierten Korpora lernt und dadurch in die Lage versetzt wird, neue Texte zu annotieren. So bilden annotierte Korpora die Grundlage für maschinelles Lernen im NLP-Bereich (McEnery 2003, 459).

Ein weiterer wichtiger Einsatzbereich für Korpora bildet die gemeinsame Evaluation von NLP-Systemen durch die Verwendung eines gemeinsamen Korpus, wie etwa bei den *Message Understandig Conferences*. Im Bereich des NLP und der Computerlinguistik gibt es zahlreiche Bereiche, die Verbindungen zur Korpuslinguistik aufweisen oder als Teilbereich der Korpuslinguistik verstanden werden können. Dies umfasst etwa die großen Teilbereiche *Information Retrieval*, *Text Data Mining* bzw. *Text Mining* (Mustersuche in Texten) und verwandte Gebiete wie die Informationsextraktion. Hier werden Korpora nicht als linguistische Evidenz zur Theoriebildung verwendet, sondern pragmatisch als Grundlage für verschiedene Problemlösungen in der Sprachverarbeitung.

Architekturen zur Erstellung und Verwendung von korpuslinguistischen Werkzeugen, auch SALE genannt (Software Architecture for Language Engineering, siehe auch Cunningham & Bontcheva 2006 und Köhler 2005), sind etwa UIMA oder GATE. An der Abteilung für Sprachliche Informationsverarbeitung[3] am Institut für Linguistik der universität zu Köln wird unter dem Namen Tesla (Text Engineering Software Laboratory) ein vergleichbares System entwickelt. Schwerpunkt der Arbeit bilden hier Wiederverwertbarkeit von Analysekomponenten (z.B. Tokenisierung) und Ergebnissen sowie die Verteilbarkeit der Analysearbeit zur Durchführung rechenaufwändiger Verfahren.

3 http://www.spinfo.uni-koeln.de

5.7. Quantitative Linguistik

Da ein großer Teil der Auswertung von Korpora mithilfe von statistischen Verfahren erfolgt, kann diese Art der Arbeit mit Korpora auch als Quantitative Linguistik charakterisiert werden (Köhler 2005), einem eigenständigen Bereich des NLP (siehe etwa Manning & Schütze 1999). Köhler (2005) betont dabei die Notwendigkeit, linguistische Fragestellungen in die Sprache der Statistik zu überführen und Ergebnisse in die Sprache der Linguistik zurückzuübersetzen und beklagt ein bislang zu wenig ausgeprägtes Methodenbewusstsein im Bereich statistischer korpuslinguistischer Arbeit.

5.8. Standards

Zentrale technische Aspekte bei der Korpuserstellung sind Zeichenkodierung und Dateiformat der Daten. In den vergangenen Jahre haben sich Unicode als Standard zur Zeichenkodierung (hiermit können alle Zeichen aller bekannten Schrift- und Zeichensysteme eindeutig dargestellt werden) sowie SGML oder dessen Nachfolger XML als Standard-Dateiformat etabliert, z.B. im *Corpus Encoding Standard* (CES) und der *Text Encoding Initiative* (TEI). Intern werden auch binäre Formate verwendet, etwa zur Speicherung eines erstellten Index zum effizienteren Zugriff auf die Daten. Detailliertere Informationen zu technischen Aspekten der Korpuserstellung finden sich etwa in Evert & Fitschen (2001).

5.9. Korpusabfrage

Man kann drei Arten der Abfrage von Korpora unterscheiden: Die Konkordanzsuche, die Wörter in ihrem Kontext zeigt (*key word in context*, KWIC), musterbasierte Suche, etwa mit regulären Ausdrücken, die eine Suche nach bestimmten Mustern ermöglichen, in einem Korpus mit POS-Tags etwa einfache Nominalphrasen des Musters

(DET)? ((ADV)? ADJ)* NN

sowie statistische Suche, die etwa die Bestimmung von Häufigkeiten bestimmter Folgen von Wortformen (Kollokationen, etwa zum Vergleich der Häufigkeiten von *different from*, *different to* und *different than*) oder Wortarten (Kolligationen) ermöglicht (Evert & Fitschen 2001, 376 und Kennedy 1998, 11).

Einen Überblick über verschiedene Korpora gibt es bei der *European Language Resources Association* oder dem *Linguistic Data Consortium*. Einige relevante englischsprachige Korpora sind etwa das *British National Corpus*, das *IBM/Lancaster Spoken English Corpus*, das *Leverhulme Corpus of Children's Writing*, das *Survey of English Dialects* (McEnery 2003) sowie das *Brown Corpus*. Abfragewerkzeuge für das BNC sind etwa *Sara* zur Konkordanzsuche oder *BNCweb* zur statistischen Suche (Evert & Fitschen 2001).

Kapitel 6

Intelligente Systeme

Paradigmen als Merkmale zur Textklassifikation: Entwicklung eines korpusbasierten Lernverfahrens, Sprachliche Informationsverarbeitiung, Hauptseminar *Intelligente Systeme* bei Prof. Dr. Jürgen Rolshoven, WS 2006–2007.

6.1. Überblick

Gegenstand dieser Arbeit ist die Beschreibung von Entwurf und Implementierung eines Verfahrens zur Textklassifikation, basierend auf einem unüberwachten, korpuslinguistischen Lernverfahren. Als Merkmale zur Klassifikation dienen Paradigmen,[1] die mithilfe von Suffixbäumen effizient ermittelt werden. Die Paradigmen erlauben dabei eine Form von lexikalischer Disambiguierung durch Berücksichtigung von Kontexten. Erste Evaluierungen mit kleinen Korpora ergeben einen Recall von 80-90% und eine Precision von 40-50%.

Die Arbeit ist wie folgt gegliedert: Abschnitt 6.2 führt in die Texklassifikation ein; Abschnitt 6.3 beschreibt die enge Beziehung zwischen maschineller Sprachverarbeitung und Korpuslinguistik, insbesondere im maschinellen Lernen. In Abschnitt 6.4 wird das verwendete, Web-basierte Korpus beschrieben; in Abschnitt 6.5 Konzept, Implementierung und Evaluierung des Verfahrens. Abschnitt 6.6 schließlich argumentiert aufbauend auf eine Diskussion möglicher Verbesserungen für den Einsatz einer *Software Architecture for Language Engineering* in der maschinellen Sprachverarbeitung.

6.2. Textklassifikation

Textklassifikation ist eine Form maschineller Sprachverarbeitung, bei der Texte vorgegebenen Klassen zugeordnet werden. Sie unterscheidet sich von der Schlüsselwort-Extraktion (*keyword extraction*), bei der die für den Text relevantesten Wörter im Text ermittelt werden. Hier sind die resultierenden Kategorien selbst immer als Wort im Text enthalten, während bei der Textklassifikation die Kategorien unabhängig vom Text sein können – so muss etwa ein Text der Kategorie 'Politik' nicht das

1 *Paradigma* meint im Kontext dieser Arbeit eine Menge von Wörtern, die zueinander in paradigmatischer Relation stehen, d.h. die in gemeinsamen Kontexten vorkommen und damit gegeneinander austauschbar sind (siehe etwa Lyons 1968, 70)

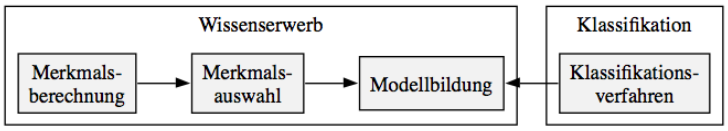

Abbildung 6.1.: Generische Architektur eines Systems zur Textklassifikation (nach Brückner 2001).

Wort *Politik* enthalten. Das Ergebnis beider Ansätze ist dabei gleich: sie klassifizieren automatisch Texte, einer Zielsetzung mit großem praktischen Potential, insbesondere in Anbetracht der heute verfügbaren Menge an maschinenlesbaren Dokumenten.

Textklassifikationssysteme bestehen konzeptuell aus zwei Hauptkomponenten: dem Wissenserwerb und der eigentlichen Klassifikation. Der Wissenserwerb wiederum gliedert sich in drei weitere Teile: erstens der Merkmalsberechnung, bei der die Merkmale, anhand derer klassifiziert werden soll, ermittelt werden; zweitens der Merkmalsauswahl, bei der die relevanten Merkmale ausgewählt werden und schließlich drittens der eigentlichen Modellbildung. Das so im Wissenserwerb gebildete Modell wird dann bei der eigentlichen Klassifikation verwendet (siehe Abb. 6.1). Weitere Informationenen zu verschiedenen Textklassifikationsverfahren finden sich etwa in Goller *et al.* (2000).

6.3. Korpuslinguistik und maschinelle Sprachverarbeitung

Das nachfolgend beschriebene Verfahren beruht im Wissenserwerb auf einem korpuslinguistischen, examplarbasierten Lernverfahren. McEnery (2003) bezeichnet Textkorpora als "the raw fuel of NLP" und damit als Grundlage und Voraussetzung der maschinellen Sprachverarbeitung. Der Grund hierfür besteht darin, dass annotierte Korpora eine maschinelle Reproduktion menschlicher Intuition ermöglichen (siehe Abschnitt 6.3.4). So bilden annotierte Korpora die Grundlage für maschinelles Lernen in der Sprachverarbeitung (McEnery 2003, 459).

6.3.1. Korpuslinguistik

Korpuslinguistik ist eine sprachwissenschaftliche Arbeitsweise und umfasst verschiedene Tätigkeiten, die mit der Erstellung und der Auswertung von Textkorpora (Köhler 2005) sowie mit der Erstellung von Werkzeugen für die beiden ersten Tätigkeiten zu tun haben. McEnery (2003) definiert ein Textkorpus als "a large body of linguistic evidence". Damit sind Korpora eine sehr wertvolle Quelle für die linguistische Arbeit. Korpuslinguistische Methoden werden nicht nur in der maschinellen Sprachverarbeitung und der Computerlinguistik[2] angewendet, sondern auch in anderen Bereichen der Sprachwissenschaft, etwa in der allgemeinen Sprachwissenschaft oder den Philologien. Somit ist Korpuslinguistik kein Teilbereich der Linguistik, sondern eine Arbeitsweise, die innerhalb der genannten Teilbereiche verwendet wird und auf verschiedene Ebenen der Sprache (wie Morphologie, Syntax, Semantik oder Pragmatik) angewendet werden kann (McEnery & Wilson 1996, 2). Da

2 Eine scharfe Trennung der Bereiche *Computerlinguistik* und *maschinelle Sprachverarbeitung* (Natural Language Processing, NLP) ist schwierig; mitunter wird maschinelle Sprachverarbeitung als angewandte Computerlinguistik charakterisiert, mitunter als eine von der Computerlinguistik zu unterscheidende Ingenieursdisziplin.

ein großer Teil der Auswertung von Korpora mithilfe von statistischen Verfahren erfolgt, wird diese Art der Arbeit mit Korpora auch als *Quantitative Linguistik* bezeichnet (siehe Köhler 2005, Manning & Schütze 1999). Köhler (2005) betont dabei die Notwendigkeit, linguistische Fragestellungen in die Sprache der Statistik zu überführen und Ergebnisse in die Sprache der Linguistik zurückzuübersetzen und beklagt ein bislang zu wenig ausgeprägtes Methodenbewusstsein im Bereich statistischer, korpuslinguistischer Arbeit.

In der heutigen, computerisierten Form gibt es Korpuslinguistik seit den späten 1940er Jahren (McEnery 2003). Ein Beispiel für frühe, nicht computerisierte Korpuslinguistik stellen Bibelkonkordanzen dar, die alle Wörter der Bibel alphabetisch sortiert in ihrem Kontext auflisten und ab dem 13. Jahrhundert entstanden; diese zeigen den radikalen Wandel durch die Entwicklung des Computers: Während das Erstellen der ersten Bibelkonkordanz 14 Jahre dauerte,[3] ist es durch die Entwicklung des Computers möglich geworden, innerhalb von Sekundenbruchteilen Konkordanzen beliebiger Texte automatisch zu erstellen; das Erstellen eines solchen Programms selbst ist innerhalb von Tagen oder Stunden möglich. So hat die Erfindung des Computers die Korpuslinguistik im heutigen Sinn erst ermöglicht. Schon durch ihre Enstehung ist die Korpuslinguistik also auf das Engste mit der maschinellen Sprachverarbeitung verbunden.

6.3.2. Korpora

Korpora sind durch verschiedene Eigenschaften gekennzeichnet. Zunächst sind Korpora wie oben beschrieben heute typischerweise maschinenlesbar. Ein Korpus ist darüber hinaus im Hinblick auf eine bestimmte Fragestellung organisiert. Dies bezeichnet McEnery (2003) als *sampling frame* des Korpus, innerhalb dessen es ausgewogen und repräsentativ sein soll. Als Beispiel nennt McEnery (2003) die Entwicklung eines Dialogsystems zum Verkauf von Eintritts- und Fahrkarten. Wenn hierfür ein Korpus zusammengestellt werden soll, sollten zum einen nur relevante Texte verwendet werden, d.h. Dialoge von Kartenverkäufen (dies entspricht dem gewählten *sampling frame*). Es sollten dabei jedoch verschiedene Arten von Verkaufsgesprächen verwendet werden, etwa von Bus- und Flugzeugtickets sowie von Telefon- und Schalterverkäufen. Außerdem sollten in jedem Bereich Gespräche verschiedener Personen verwendet werden, um Idiosynkrasien einzelner Sprecher zu vermeiden. So erhält man ein ausgewogenes und repräsentatives Korpus. Diesen Grundgedanken folgend wurden die Korpora für das implementierte Verfahren zusammengestellt (siehe Abschnitt 6.4).

McEnery (2003) unterscheidet verschiedene Arten von Korpora: *Monolinguale Korpora* sind Korpora mit Texten in einer einzigen Sprache. *Vergleichbare Korpora* sind Korpora aus Texten in verschiedenen Sprachen, die in Bezug auf *sampling frame*, Ausgewogenheit und Repräsentativität vergleichbar sind. Diese Korpora eignen sich etwa für den kontrastiven Sprachvergleich. *Parallele Korpora* sind Korpora, die zunächst aus Texten einer einzigen Sprache zusammengestellt und dann in andere Sprachen übersetzt werden. Solche Korpora eignen sich etwa für Systeme zur maschinellen Übersetzung, die aus Beispielübersetzungen lernen. *Monitorkorpora* werden laufend aktualisiert und dienen etwa zur Beobachtung von Sprachwandel. Neben diesen Unterscheidungen können gesprochene von reinen Schriftkorpora unterschieden werden. Gesprochene Korpora (die monolingual, vergleichbar oder parallel sowie Monitorkorpora sein können) enthalten Informationen über die gesprochene Form des Textes. Dies kann bedeuten, dass es sich um akustisches Material handelt, oder auch um tran-

3 http://de.wikipedia.org/wiki/Bibelkonkordanz

skribierte Sprachdaten. Da diese beiden Formen Vor- und Nachteile haben,[4] gibt es zunehmend Korpora, die aus akustischem und aus transkribiertem Material bestehen und deren Inhalte aufeinander abgestimmt sind (*time alignment*), wodurch es möglich ist, auf die jeweils andere Form zuzugreifen (McEnery 2003, 451). Solche multimodale Korpora (Evert & Fitschen 2001) können darüber hinaus auch Informationen über Mimik oder Gestik enthalten, etwa in Form von Videomaterial. Die Multimodalität der Korpora kann auch als eine Form von Korpusannotation gesehen werden (siehe Abschnitt 6.3.3).

Das implementierte Verfahren verwendet monolinguale Schriftkorpora. Eine Zusammenstellung eines *vergleichbaren* (s.o.) Korpus wäre im beschriebenen Vorgehen durch die Nutzung von Texten entsprechender Ressorts der englischsprachigen Ausgabe von Spiegel-Online relativ einfach realisierbar (siehe Abschnitt 6.4).

6.3.3. Korpusannotation

Annotierte Korpora sind Korpora, die mit verschiedenen Arten linguistischer Information angereichert sind (McEnery & Wilson 1996, 24). Ein Beispiel für Korpusannotationen ist etwa die Kennzeichnung von Wortarten durch Part-of-Speech-Tags (POS-Tags). Andere Annotationen[5] enthalten etwa Informationen über Stammformen (als Ergebnis einer Stammformenreduktion, auch *Stemming* oder *Lemmatisierung* genannt), über die syntaktische Struktur (solche Korpora werden auch *Baumbanken* genannt) sowie semantische, stilistische oder Informationen zur Diskursstruktur.[6] Diese Anreicherung basiert immer auf einer bestimmten Interpretation der Daten (McEnery 2003). Beim Annotieren ist daher zu beachten, dass der ursprüngliche Text auch nach einer Annotation noch in seiner Rohform verfügbar sein sollte, etwa durch *dynamische Annotation* (siehe Benden & Hermes 2004, Hermes & Benden 2005). McEnery (2003) nennt vier Vorteile von Korpusannotationen: Zunächst eine verbesserte Nutzbarkeit der Korpora für eine größere Anzahl von Benutzern, seien dies Menschen, die einer bestimmten Fremdsprache unkundig sind oder Computerprogramme, die Sprache verarbeiten sollen. In beiden Fällen werden die Annotationen benötigt, können jedoch von dem, der sie benötigt, nicht selbst erstellt werden. Darüber hinaus ist ein annotiertes Korpus aber auch für Benutzer, die die Analysen selbst vornehmen könnten viel schneller zu verwenden als ein nicht-annotiertes Korpus. Ein zweiter Vorteil ist die Wiederverwertbarkeit der bei der Analyse gewonnen Daten, etwa der ermittelten POS-Tags, der Stammformen, syntaktischen Strukturen etc. Als weitere Vorteile nennt McEnery (2003) die Multifunktionalität der Daten sowie die explizite Formulierung und objektive Erfassung von Ergebnissen einer Analyse.

Korpusannotationen können automatisch, von Hand oder in einer kombinierten Form erstellt werden. Für Aufgaben wie das POS-Tagging oder Lemmatisieren für große Sprachen wie Englisch, Französisch, Deutsch oder Spanisch können Annotationen automatisch erstellt werden. Häufig aber wird das Ergebnis der maschinellen Annotation von Hand kontrolliert und erfolgt damit halbautomatisch, was immer noch ein schnelleres Annotieren als komplett von Hand ermöglicht. Einige Bereiche erfordern aber auch rein manuelle Erstellung der Annotationen, etwa zur Darstellung

4 Korpora mit akustischem Material sind schwerer zu erschließen, etwa zur Suche nach bestimmten Wörtern, während es bei transkribierten Daten etwa zum Verlust prosodischer Feinheiten kommt.

5 Beispiele verschiedener Korpusannotationen und weiteres Material findet sich unter: `http://bowland-files.lancs.ac.uk/monkey/ihe/linguistics/corpus2/2fra1.htm`

6 In abnehmender Häufigkeit und Verbreitung, d.h. morphologische sind verbreiteter als syntaktische Annotationen, diese wiederum verbreiteter als semantische oder pragmatische Annotationen.

anaphorischer und kataphorischer Relationen (McEnery 2003). Als Argument gegen manuelle und halbautomatische Annotation ist vorgebracht worden, dass diese nur eine geringe Konsistenz aufweisen könne, da menschliche Annotatoren vergleichbare Stellen im Korpus nicht immer übereinstimmend annotieren. Es wurden jedoch Untersuchungen unternommen, die gezeigt haben, dass bei geschulten Annotatoren der leichte Rückgang der Konsistenz mehr als kompensiert wurde durch eine gesteigerte Genauigkeit der Annotationen (McEnery 2003, 457).

6.3.4. Lernen und Evaluieren

Annotierte Korpora ermöglichen es Computerprogrammen, die Intuitionen von Experten, die die Annotationen erstellt oder verbessert haben, in Bezug zu den Texten zu setzen und so menschliche Intuitionen zu reproduzieren. Ein Beispiel hierfür wäre etwa ein auf Hidden-Markov-Modellen basierender POS-Tagger, der aus annotierten Korpora lernt und dadurch in die Lage versetzt wird, neue Texte zu annotieren (siehe etwa Manning & Schütze 1999). Analog ist das nachfolgend beschriebene Verfahren: aus klassifizierten Texten wird induktiv gelernt und anhand des erworbenen Wissens werden neue Texte klassifiziert. In diesem Sinn ist das beschriebene Verfahren ein korpusbasiertes, induktives Lernverfahren (siehe Abschnitt 6.5.1). Ein solches maschinelles Lernen von Strukturen aus Korpora wird auch als *Grammatik-Bootstrapping* bezeichnet. Die obigen Beispiele (POS-Tagging und Textklassifikation) deuten an, dass es in der maschinellen Sprachverarbeitung und der Computerlinguistik zahlreiche Bereiche gibt, die als korpuslinguistische Verfahren charakterisiert werden können. Dies umfasst neben den oben genannten Beispielen etwa die Volltextsuche (*Information Retrieval*), die Mustersuche in Texten (*Text Data Mining* oder *Text Mining*) sowie verwandte Gebiete wie die Informationsextraktion.

Ein weiterer, wichtiger Einsatzbereich für Korpora bildet die gemeinsame Evaluierung von Systemen zur maschinellen Sprachverarbeitung durch die Verwendung eines gemeinsamen Korpus, wie etwa bei den *Message Understandig Conferences* (siehe etwa Grishman & Sundheim 1996). Im nachfolgend beschriebenen Verfahren wird ein dem Lernkorpus vergleichbares, kleineres Korpus zur Evaluierung eingesetzt (siehe Abschnitt 6.5.4). Software, die die Arbeit mit annotierten Korpora zur Entwicklung und Evaluierung von sprachverarbeitenden Komponenten unterstützt wird als *Software Architecture for Language Engineering* bezeichnet (siehe Abschnitt 6.6).

6.4. Das Web als Basis für Korpora

Neben linguistisch aufbereiteten Korpora, auf die der Zugriff häufig beschränkt ist (etwa rechtlich oder technisch), steht mit dem Web eine große, öffentliche, maschinenlesbare Textsammlung zur Verfügung, die allerdings weder linguistisch organisiert (siehe Abschnitt 6.3.2) noch aufbereitet ist (siehe Abschnitt 6.3.3). Das Web ist dennoch eine attraktive Quelle für die Zusammenstellung von Korpora, dessen Erschließung aber abhängig von Werkzeugen ist, etwa zur Suche (z.B. Google) oder zur Kategorisierung (z.B. Delicious, s.u.); insbesondere Ressourcen mit Formen von sozialer Intelligenz – wie etwa die Kategorien in Delicious oder die kollaborative Arbeit an der Wikipedia – haben großes Potential als Quelle für das maschinelle Lernen in der Sprachverarbeitung. Auf dieser Grundlage wird im vorgestellten System Delicious zur Zusammenstellung von Korpora verwendet. Mit Delicious lassen sich Web-Bookmarks verwalten; diese können klassifiziert werden, wobei mehrere Klassen frei vergeben werden können (*tagging*); die Klassen lassen sich wiederum zu *bundles*

Abbildung 6.2.: Screennshot der in Delicious zusammengestellten Korpora für Training und Evaluierung des Systems.

zusammenfassen, d.h. ein *bundle* umfasst dann mehrere Klassen, denen je mehrere Webseiten zugeordnet sind (siehe auch Abb. 6.2). Konzeptuell entsprechen die *bundles* einem Korpus mit einem gewünschten *sampling frame*; die Klassen können als Korpusannotation betrachtet werden und die Grundlage des zu lernenden Wissens darstellen, und sind damit die menschliche Intuition, die von der Maschine reproduziert werden soll (vgl. Abschnitt 6.3.3).

Delicious stellt eine öffentliche, Web-basierte API für den Dienst bereit, für die es eine Java-API gibt, welche im vorgestellten System verwendet wird.[7] Daneben besteht die Möglichkeit des Exports in das Netscape-Bookmark-HTML-Format, das vom implementierten System als alternative Quelle der Links eingelesen werden kann. Zum Parsen der verlinkten Seiten und damit zum Einlesen der eigentlichen Texte wird mit NekoHTML[8] ein korrigierender HTML-Parser verwendet. Das Korpus wird also aus den Inhalten der in Delicious angegebenen Webseiten zusammengestellt. Ein solches Programm, das mehrere Dateien abruft und verarbeitet wird allgemein als *Crawler* bezeichnet. Ein Crawler, der aus dem Internet Informationen abruft und zusammenführt, wird auch als *Bot* oder spezieller als *Aggregator* bezeichnet (vgl. Heaton 2002). Im implementierten Verfahren wird zur Evaluierung ein Testkorpus aus Online-Nachrichten von Spiegel-Online verwendet. Zum Training des Systems wird dementsprechend ein Korpus verwendet, das aus ebensolchen Artikeln besteht (vgl. Abschnitt 6.3.2). Die Ausgewogenheit wird in diesem Fall durch die Tatsache sichergestellt, dass Artikel verschiedenener Ressorts im Lernkorpus vorhanden sind (siehe auch Abb. 6.2). Es handelt sich hierbei um eine experimentelle Umsetzung der geschilderten Gedanken in kleinem Maßstab;

7 `http://sourceforge.net/projects/delicious-java/`
8 `http://people.apache.org/~andyc/neko/doc/`

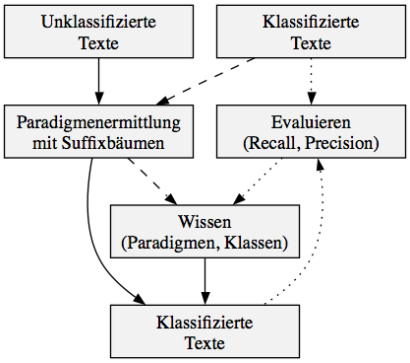

Abbildung 6.3.: Wissenserwerb, Klassifikation und Evaluierung im implementierten Verfahren; die gestrichelte Linie beschreibt den Wissenserwerb, die durchgezogene Linie die Klassifikation und die gepunktete Line die Evaluierung.

bei einer praxisnahen Umsetzung würden etwa Artikel verschiedener Zeitungen verwendet um Ausgewogenheit und Repräsentativität des Korpus zu gewährleisten.

6.5. Klassifikation mit Paradigmen

Grundgedanke des implementierten Verfahrens ist es, Paradigmen als Merkmale zur Klassifikation zu verwenden. So könnte etwa das Paradigma [Eis, Chips] als Merkmal für die Klasse 'Essen' verwendet werden. Würde nun etwa Text (6.1) klassifiziert, würde das Paradigma [Eis, Chips] gefunden und der Text entsprechend klassifiziert.

(6.1) *Hans mag Eis. Anna mag Chips nicht. Ich mag Chips gern.*

Text (6.2) dagegen würde nicht der Klasse 'Essen' zugeordnet, da hier zwar die Wörter *Chips* und *Eis* vorkommen, jedoch nicht im gleichen Kontext. Dies ermöglicht eine Form von lexikalischer Disambiguierung, hier etwa für *Chips* als 'Kartoffelchips' im Gegensatz zu 'Mikrochips'. Das Verfahren kann aufgrund der symbolischen Natur des verwendeten Wissens – nämlich der Paradigmen – als symbolisches Verfahren bezeichnet werden. Eine Übersicht des Systems und der einzelnen Schritte findet sich in Abb. 6.3.

(6.2) *In Dresden produziert Infineon Chips. Die Produktion liegt zur Zeit auf Eis.*

6.5.1. Maschinelles Lernen

Der Wissenserwerb des Systems stellt damit eine Form von maschinellem Lernen dar. Genaugenommen handelt es sich um ein unüberwachtes, induktives, exemplarbasiertes Lernverfahren mit einer

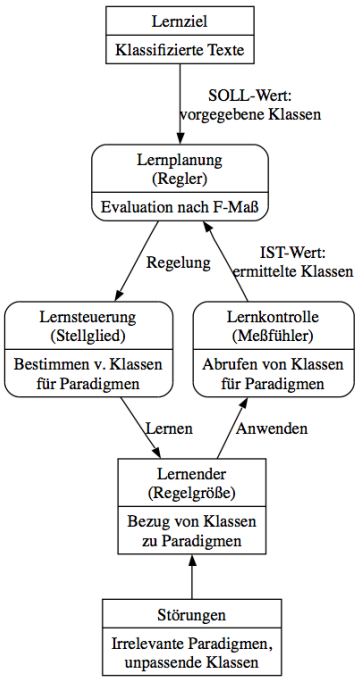

Abbildung 6.4.: Kybernetischer Regelkreis des Lernens (von Cube 1965), im unteren Teil der Kästen ergänzt für das beschriebene Verfahren.

symbolischen Wissensrepräsentation. Abb. 6.4 stellt das Verfahren in den Kontexts des Lernbegriffs in der Kybernetik. Verschiedene maschinelle Lernverfahren zur Textklassifikation (siehe Brückner 2001) verwenden numerische Wissenrepräsentationen. Für die Verwendung des beschriebenen Verfahrens in einem solchen Kontext wären dies Werte, die ausdrücken, ob und wie sehr ein Paradigma für eine Klasse relevant ist, repräsentiert in einen Merkmalsvektor pro Klasse, mit Werten, die die Relevanz der Klasse für jedes Paradigma kennzeichnen.

6.5.2. Paradigmen und Suffixbäume

Zur Ermittlung der Paradigmen im Lernkorpus und in den zu klassifizierenden Texten kommen Suffixbäume zum Einsatz,[9] die eine effiziente Ermittlung der Paradigmen ermöglichen. Ein Suffixbaum ist mit linearer Laufzeit- und Speicherplatzkomplexität konstruier- und nutzbar,[10] während etwa eine Ermittlung von Paradigmen über einen paarweisen Vergleich aller Sätze über die Levenshtein-Distanz eine quadratische Laufzeitkomplexität aufweist. Ein Suffixbaum mit Wörtern als Symbole enthält Informationen über Paradigmen im repräsentierten Text in seiner Struktur: Da mehrfach auftretende Teilstrings im Baum nur einmal vorkommen und er an den Stellen verzweigt, an de-

9 `http://stnl.sourceforge.net/`
10 Weitere Informationenen zu Konstruktion und Nutzung von Suffixbäumen findet sich etwa in Gusfield (1997).

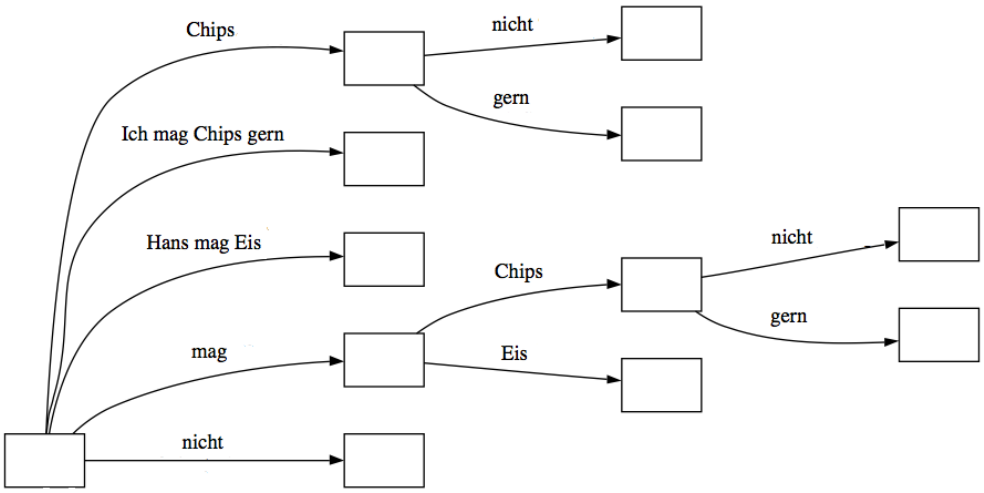

Abbildung 6.5.: Ausschnitt des Suffixbaums für Text (6.1)

nen sich die Sätze unterscheiden, stehen alle Beschriftungen von Kanten, die von inneren Knoten ausgehen, zueinander in paradigmatischer Relation. In einem Suffixbaum können Paradigmen mit gemeinsamen Kontexten *vor* den Paradigmen (siehe Abb. 6.5), in einem Suffixbaum für den umgekehrten Text – einem Präfixbaum – können Paradigmen mit gemeinsamem Kontext *nach* den Paradigmen ermittelt werden (siehe Abb. 6.6).

6.5.3. Programmstruktur und Vorgehen

Die Struktur der Implementierung basiert auf dem grundlegenden Aufbau aus Wissenserwerb und Klassifikation; mit letzterer ist die Evaluierung assoziiert. Hinzu kommen Klassen für das eigentliche Programm, das Crawling, die Vorverarbeitung in Form von HTML-Parsing und Stopwortlisten-Filtern sowie eine Klasse zur Repräsentation von kategorisierten Texten. Ein UML-Klassendiagramm der beschriebenen Implementierung findet sich in Abb. 6.7.

Das Vorgehen besteht im Wissenserwerb aus den drei in Abschnitt 6.2 beschriebenen Schritten: dies sind erstens die Merkmalsberechnung, hier in Form der Ermittlung von Paradigmen mit einem Suffixbaum, zweitens die Merkmalsauswahl, hier durch Filtern der Paradigmen mithilfe von Stopwortlisten und schließlich drittens die Modellbildung, hier das Ablegen der Paradigmen und der für diese relevanten Klassen in einer *Map*; hierbei bilden die Paradigmen die Schlüssel und die Klassen die Werte, etwa $[Chips, Eis] \rightarrow [Essen, Kino]$ oder $[Chips, Monitore] \rightarrow [Computer]$. Die Klassifikation besteht ebenfalls aus drei Schritten: erstens der Ermittlung der Paradigmen im zu klassifizierenden Text, zweitens wird für jede mögliche Klasse die beste Übereinstimmung eines der für diese Klasse relevanten Paradigmen mit einem Paradigma im zu klassifizierenden Text ermittelt. Bei einem Vergleich etwa von $[Chips, Eis]$ mit $[Chips, Cola]$ wäre die Übereinstimmung 50%. Drittens schließlich

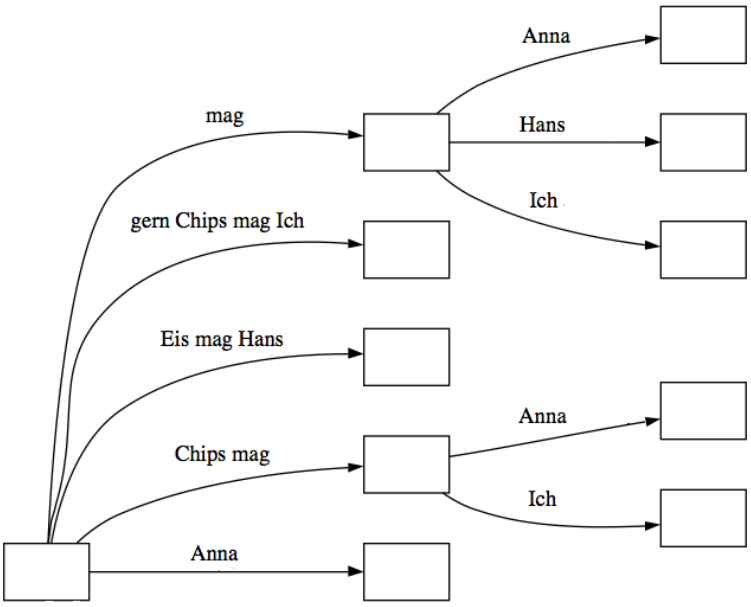

Abbildung 6.6.: Ausschnitt des Präfixbaums für Text (6.1)

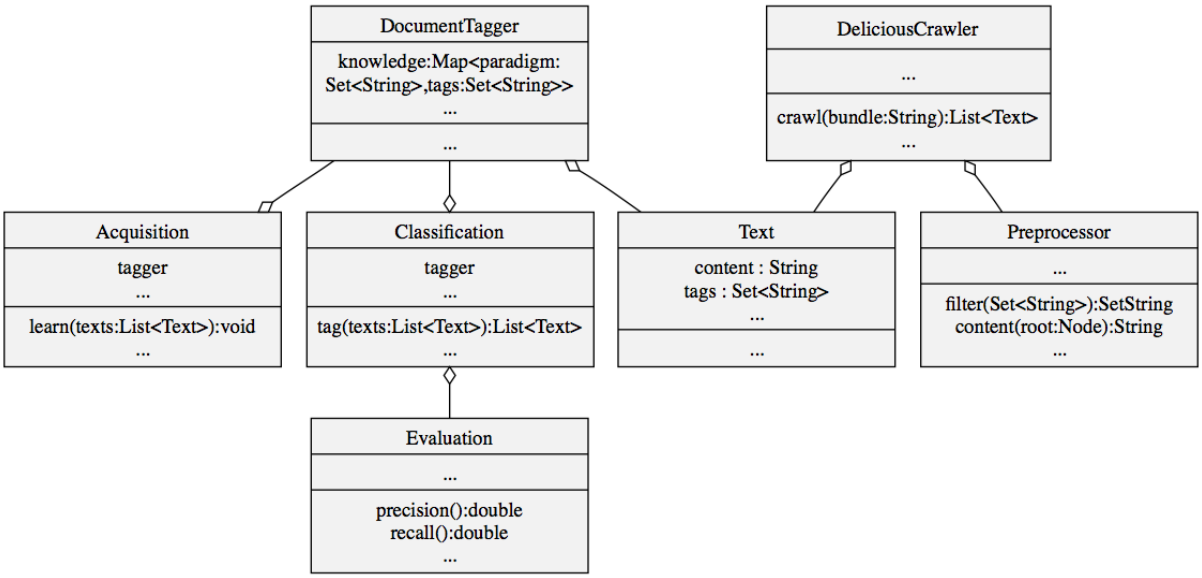

Abbildung 6.7.: UML-Klassendiagramm der Implementierung

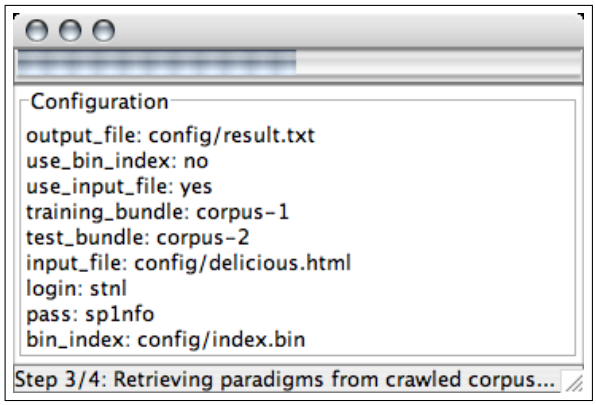

Abbildung 6.8.: Statusanzeige

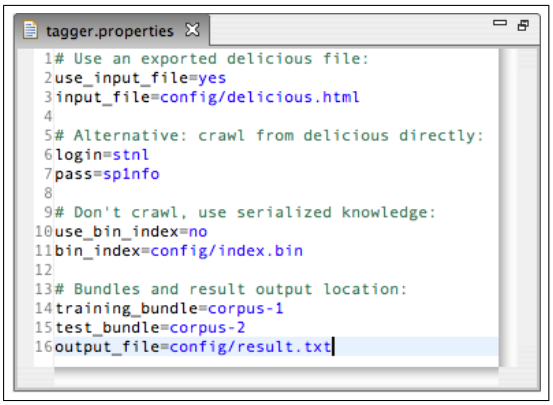

Abbildung 6.9.: Properties-Datei

werden alle Klassen, deren beste Übereinstimmung einen bestimmten Schwellenwert überschreitet, als Ergebnis der Klassifikation ausgewählt; im Versuch wurde als Schwellenwert 50% verwendet.

6.5.4. Evaluierung

Zur Evaluierung wurden zwei Korpora in einem eigenen Delicious-Account eingerichtet.[11] Die Software ist online verfügbar,[12] in Form einer ausführbaren Jar-Datei und als Quelltext. Nach Doppelklick der Jar-Datei öffnet sich ein Fenster, das die verwendeten Konfigurationswerte und den Fortgang der Analyse anzeigt (siehe Abb. 6.8). Dabei werden die angegebenen Texte zum Lernen und zum Klassifizieren verwendet, die Ergebnisse der Klassifikation werden in eine Textdatei geschrieben. Die Konfigurierung erfolgt über eine Java Properties-Datei (siehe Abb. 6.9). Eine solche *dynamische Konfigurierung* hat softwaretechnische Vorteile gegenüber einem Einbau der Details im Programmcode und kann als eine Form von Metaprogrammierung gesehen werden (siehe auch Hunt & Thomas 2003, 135ff.). Dies erleichtert etwa eine Verwendung in anderen Zusammenhängen, etwa in einer SALE (siehe Abschnitt 6.6).

Die Evaluierung erfolgt nach den Standardmaßen Precision, Recall und F-Maß. Die Precision drückt dabei hier aus, wie viele der ermittelten Klassen korrekt sind; der Recall drückt aus, wie viele der zu ermittelnden Klassen auch ermittelt wurden; das F-Maß erlaubt eine einheitliche Betrachtung von Recall und Precision. Über die Bildung des Mittels der Ergebnisse lässt sich das Verfahren selbst evaluieren. Versuche mit kleinen Korpora ergaben einen Recall von 80-90% und eine Precision von 40-50%. Ergebnisse der ersten Evaluierung mit 16 Spiegel-Online-Artikeln, ca. 10.000 Paradigmen als Merkmale und einer Klasse pro Text, ermittelt in einem Lernkorpus aus 120 Spiegel-Online-Artikeln finden sich in Abb. 6.10; weitere Experimente mit mehreren Klassen pro Dokument und einem anderen Lernkorpus von etwa doppelter Größe (211 Artikel), resultierend in etwa der doppelten Anzahl von Merkmalen (18.452) sowie einem Testkorpus mit ebenfalls verdoppelter Größe (31 Artikel) lieferten vergleichbare, leicht verbesserte Ergebnisse mit einem Recall von 91,7% und

11 http://del.icio.us/stnl
12 http://stnl.sourceforge.net/applications/

Text	Recall	Precision	F-Maß
1	0	0	0
2	1	0,25	0,4
3	0	0	0
4	1	0,25	0,4
5	1	0,5	0,67
6	1	0,5	0,67
7	1	1	1
8	1	0,5	0,67
9	1	0,5	0,67
10	1	0,5	0,67
11	1	0,35	0,5
12	1	0,5	0,67
13	1	0,5	0,67
14	1	0,25	0,4
15	1	0,25	0,4
16	1	1	1
ϕ	0,88	0,43	0,55

Abbildung 6.10.: Ergebnisse der ersten Evaluierung mit einer Klasse pro Dokument

einer Precision von 52,1%. Bei heterogenen Korpora ist die Qualität geringer, was aber angesichts der grundsätzlichen Forderungen an Korpora (siehe Abschnitt 6.3.2) zu erwarten ist.

Es sind verschiedenen Ansätze für eine Verbesserung des Verfahrens denkbar: Als Verbesserungen, die die Sprachunanhängigkeit des Verfahrens erhalten sind eine Verbesserung der Qualität der Paradigmen, eine Anpassung von Größe und Beschaffenheit der Korpora sowie eine Anpassung des Algorithmus denkbar. Die Qualität der Paradigmen könnte etwa durch verbessertes Filtern oder eine Berücksichtigung von Mehrwort-Paradigmen geschehen. Die Korpora könnten aus anderen Quellen stammen, größer sein und anders vorverarbeitet[13] werden. Der Algorithmus könnte etwa in der Form angepasst werden, dass nicht ein fester Schwellenwert verwendet wird, sondern z.B. die besten 5% ausgewählt werden. Der Wert könnte darüber hinaus abhängig von Ergebnissen der Evaluierung angepasst werden (vgl. *Regelung* in Abb. 6.4). Als weitere, meist sprachspezifisch implementierte Verbesserungen wären etwa eine Stammformenreduktion und POS-Tagging denkbar. Eine Stammformenreduktion würde eine Berücksichtigung aller Wortformen für die Paradigmenermittlung ermöglichen und sollte damit die Qualität der Paradigmen verbessern, während POS-Tagging eine Unterscheidung verschiedener Wortarten mit derselben Wortform ermöglicht, was zu einer Verbesserung der Precision beitragen könnte. Für eine Stammformenreduktion wäre ein korpusbasiertes Verfahren mit Suffixbäumen denkbar, das ähnlich wie das hier beschriebene Verfahren prinzipiell sprachunabhängig wäre.[14] Ein weiterer möglicher Ansatz bestünde in der Nutzung semantischer Relationen, etwa zur Berücksichtigung von Hyperonymen beim Vergleich von Paradigmen.

13 Im implementierten Verfahren werden etwa lediglich die Inhalte aller Paragraph-Elemente der HTML-Dateien ausgelesen.

14 Unter `http://stnl.sourceforge.net/applications/` findet sich eine Umsetzung eines solchen Verfahrens für das Spanische.

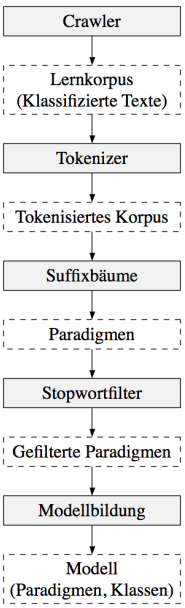

Abbildung 6.11.: Wissenserwerb

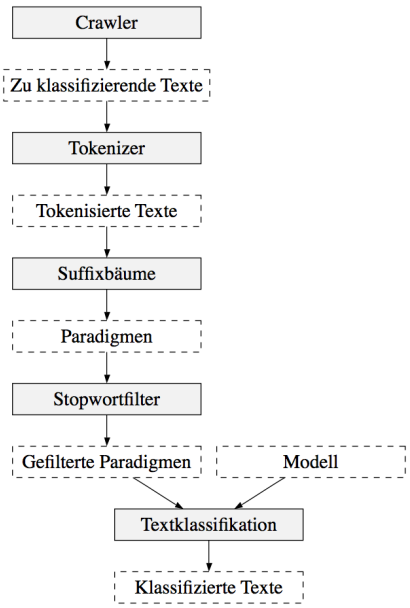

Abbildung 6.12.: Klassifikation

6.6. Software Architecture for Language Engineering

Der größte Teil der genannten Verbesserungen erfordert eine Integration mit anderen sprachverarbeitenden Komponenten. Zu diesem Zweck bietet sich der Einsatz einer *Software Architecture for Language Engineering* (SALE) an, einer Infrastruktur für die maschinelle Sprachverarbeitung (siehe Cunningham & Bontcheva 2006 sowie Köhler 2005). Eine solche Infrastruktur besteht aus Frameworks, Referenzarchitekturen und einer Entwicklungsumgebung und bildet damit eine Art Werkzeugkasten für die computerlinguistische Arbeit. Beispiele für SALEs sind etwa UIMA oder GATE. An der Abteilung für Sprachliche Informationsverarbeitung[15] am Institut für Linguistik der Universität zu Köln wird unter dem Namen *Tesla* (Text Engineering Software Laboratory) ein vergleichbares System entwickelt. Schwerpunkt der Arbeit bilden hier Wiederverwertbarkeit von Analysekomponenten und Ergebnissen durch dynamische Annotation (siehe Benden & Hermes 2004), die IDE-Integration in Eclipse sowie die Verteilbarkeit der Analysearbeit in einer Java-EE-Architektur zur Durchführung rechenaufwändiger Verfahren. Es fällt auf, dass bereits in einer solch kleinen, experimentellen Implementierung viele individuelle Komponenten identifizierbar sind, die verbessert und zu Evaluierungszwecken gegen andere ausgetauscht werden könnten. Im vorgestellten Verfahren etwa sind beim Wissenserwerb (siehe Abb. 6.11) alle der Modellbildung und bei der Klassifikation (siehe Abb. 6.12) alle der eigentlichen Textklassifikation vorgelagerte Schritte prinzipiell austauschbar und nicht spezifisch für das Verfahren. Denkbar wären hier etwa im Rahmen von Tesla der Einsatz von SPre zur Vorverarbeitung (Benden & Hermes 2004, Hermes & Benden 2005) und SOG zur Paradigmenbildung (Schwiebert 2005, Schwiebert & Rolshoven 2006).

15 `http://www.spinfo.uni-koeln.de`

6.7. Fazit

In der vorliegenden Arbeit wurde beschrieben, wie sich das Web als Grundlage für den Aufbau spezifischer Korpora für maschinelle Lernverfahren in der Sprachverarbeitung einsetzen lässt, implementiert in einem System zur Textklassifikation. Paradigmen als Klassifikationsmerkmale erlauben dabei eine Form von lexikalischer Disambiguierung durch Berücksichtigung von Kontexten. Zur effizienten Ermittlung der Paradigmen wird mit Suffixbäumen eine bisher in der Computerlinguistik wenig verwendete, vielseitige Datanstruktur eingesetzt. Eine experimentelle Evaluierung ergibt einen Recall von 80-90% und eine Precision von 40-50%. Verbesserte Vorverarbeitung, etwa in Form einer Stammformenreduktion oder POS-Tagging, könnte ohne großen Aufwand zu Verbesserungen führen; im Kontext möglicher Verbesserungen wurden Motivation und Mehrwert einer Integration verschiedener sprachverarbeitender Komponenten in einer *Software Architecture for Language Engineering* (SALE) beschrieben.

Literaturverzeichnis

Alsina, A., J. Bresnan & P. Sells (eds.): 1997, *Complex Predicates*, Center for the Study of Language and Information, Stanford, CA, USA.

Andersson, A., J. Larsson & K. Swanson: 1999, 'Suffix Trees on Words', *Algorithmica* **23**.

Appelt, D., J. Bear, J. Hobbs, D. Israel, M. Kameyama, M. Stickel & M. Tyson: 1993, 'FASTUS: A Cascaded Finite-State Tranducer for Extracting Information from Natural- Language Text, Sri International', Sri International, 8. September 2003.

Appelt, D. & D. Israel: 1999, 'Introduction to Information Extraction Technology: A Tutorial Prepared for IJCAI-99', SRI International.

Bakker, D.: 1994, *Formal and Computational Aspects of Functional Grammar and Language Typology*, Ph.D. thesis, Amsterdam.

Benden, C. & J. Hermes: 2004, 'Präprozessierung mit Nebenwirkungen: Dynamische Annotation', in E. Buchberger (ed.), *Beiträge zur 7. Konferenz zur Verarbeitung natürlicher Sprache (KONVENS)*, Riegelnik, Wien, pp. 25–28.

Bloomfield, L.: 1933, *Language*, Holt, New York.

Böckenhauer, H.-J. & D. Bongartz: 2003, *Algorithmische Grundlagen der Bioinformatik: Modelle, Methoden und Komplexität*, Teubner.

Brückner, T.: 2001, 'Textklassifikation', in K. U. Carstensen, C. Ebert, E. Endriss, S. Jekat, R. Klabunde & H. Langer (eds.), *Computerlinguistik und Sprachtechnologie*, Spektrum, Heidelberg, Berlin, pp. 442–447.

Butler, C. S.: 2003, *Structure and Function: A Guide to Three Major Structural-Functional Theories* (*Studies in Language* 64), John Benjamins Publishing Company, Amsterdam.

Cardie, C.: 1997, 'Empirical Methods in Information Extraction', *AI Magazine* **18**(4), 65–68.

Chomsky, N.: 1965, *Aspects of the Theory of Syntax*, The MIT Press, Cambridge.

von Cube, F.: 1965, *Kybernetische Grundlagen des Lernens und Lehrens*, Klett.

Cunningham, H. & K. Bontcheva: 2006, 'Computational Language Systems, Architectures', in K. Brown, A. H. Anderson, L. Bauer, M. Berns, G. Hirst & J. Miller (eds.), *The Encyclopedia of Language and Linguistics*, second edn., Elsevier, München.

Cunningham, H., D. Maynard, K. Bontcheva, V. Tablan, C. Ursu & M. Dimitrov: 2003, 'Developing Language Processing Components with GATE (a User Guide)', .

Dik, S. C.: 1991, 'Functional Grammar', in F. G. Droste & J. E. Joseph (eds.), *Linguistic Theory and Grammatical Description*, John Benjamins Publishing Co., Amsterdam & Philadelphia, pp. 246–274.

Dik, S. C.: 1997a, *The Theory of Functional Grammar. Part 1: The Structure of the Clause*, 2nd edn., Mouton de Gruyter, Berlin.

Dik, S. C.: 1997b, *The Theory of Functional Grammar. Part 2: Complex and Derived Constructions*, 2nd edn., Mouton de Gruyter, Berlin.

Durie, M.: 1997, 'Grammatical Structures in Verb Serialization', in A. Alsina, J. Bresnan & P. Sells (eds.), *Complex Predicates*, Center for the Study of Language and Information, Stanford, CA, USA.

Euler, T.: 2001a, 'Informationsextraktion durch gezielte Zusammenfassung von Texten', Universität Dortmund.

Euler, T.: 2001b, 'Informationsextraktion durch Zusammenfassung maschinell selektierter Textsegmente', Universität Dortmund.

Euler, T.: 2002, 'Tailoring Text using Topic Words: Selection and Compression', in *Proceedings of the 13th International Workshop on Database and Expert Systems Applications (DEXA)*, IEEE Computer Society Press.

Evert, S. & A. Fitschen: 2001, 'Textkorpora', in K. U. Carstensen, C. Ebert, E. Endriss, S. Jekat, R. Klabunde & H. Langer (eds.), *Computerlinguistik und Sprachtechnologie*, Spektrum, Heidelberg, Berlin, pp. 369–376.

Geertzen, J.: 2003, *String Alignment in Grammatical Inference: What Suffix Trees can do*, Master's thesis, Universität Tilburg.

Goller, C., J. Löning, T. Will & W. Wolff: 2000, 'Automatic document classification: A thorough evaluation of various methods', *7. Internationales Symposium für Informationswissenschaft* .

Grishman, R.: 2003, 'Information Extraction', in R. Mitkov (ed.), *The Oxford Handbook of Computational Linguistics*, Oxford Handbooks in Linguistics, Oxford University Press, Oxford, pp. 545–559.

Grishman, R., S. Huttunen, P. Tapanainen & R. Yangarber: 2000, 'Unsupervised Discovery of Scenario-Level Patterns for Information Extraction', in *Proceedings of the Conference on Applied Natural Language Processing*, pp. 282–289.

Grishman, R. & B. Sundheim: 1996, 'Message Understanding Conference - 6: A Brief History', in *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, Kopenhagen, pp. 466–471.

Gusfield, D.: 1997, *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*, Cambridge University Press.

Heaton, J.: 2002, *Programming Spiders, Bots and Aggregators in Java*, Sybex.

Hengeveld, K. & L. J. Mackenzie: 2006, 'Functional Discourse Grammar', in K. Brown (ed.), *Encyclopedia of Language and Linguistics*, second edn., Elsevier, Oxford.

Hermes, J. & C. Benden: 2005, 'Fusion von Annotation und Präprozessierung als Vorschlag zur Behandlung des Rohtextproblems', in B. Fisseni, H.-C. Schmitz, B. Schröder & P. Wagner (eds.), *Sprachtechnologie, mobile Kommunikation und linguistische Ressourcen. Beiträge zur GLDV-Tagung 2005 in Bonn* (*Sprache, Sprechen und Computer* 8), Lang, Frankfurt a.M., pp. 78–90.

Hunt, A. & D. Thomas: 2003, *Der Pragmatische Programmierer*, Hanser, München, Wien.

Jackendoff, R.: 2002, *Foundations of Language: Brain, Meaning, Grammar, Evolution*, University Press, Oxford.

Kennedy, G.: 1998, *An Introduction to Corpus Linguistics*, Longman, London, New York.

Kiparsky, P.: 2002, 'On the Architecture of Panini's Grammar', Three lectures delivered at the Hyderabad Conference on the Architecture of Grammar.

Köhler, R.: 2005, 'Korpuslinguistik - zu wissenschaftstheoretischen Grundlagen und methodologischen Perspektiven', *GLDV-Journal for Computational Linguistics and Language Technology* **20**(2), 1–16.

Lin, H.-L.: 2004, 'Serial Verb Constructions vs. Secondary Predication', *Concentric: Studies in Linguistics* **30**(2), 93–122.

Lyons, J.: 1968, *Introduction to Theoretical Linguistics*, University Press, Cambridge.

Manning, C. D. & H. Schütze: 1999, *Foundations of statistical natural language processing*, MIT Press, Cambridge, MA, USA.

McEnery, T.: 2003, 'Corpus Linguistics', in R. Mitkov (ed.), *The Oxford Handbook of Computational Linguistics*, Oxford Handbooks in Linguistics, Oxford University Press, Oxford, pp. 448–463.

McEnery, T. & A. Wilson: 1996, *Corpus Linguistics*, Edinburgh University Press, Edinburgh.

Mitkov, R.: 2003, *Anaphora Resolution*, Oxford University Press, pp. 267–283.

Muysken, P. C., B. Jansen & H. Koopman: 1978, 'Serial Verbs in the Creole Languages', *Amsterdam Creole Studies* **2**, 125–159.

Muysken, P. C. & T. Veenstra: 1995, 'Serial Verbs', in J. Arends, P. Muysken & N. Smith (eds.), *Pidgins and Creoles: An Introduction* (*Creole Language Library* 15), John Benjamins Publishing Co., Amsterdam & Phildadelphia, pp. 289–301.

Neumann, G.: 2001, 'Informationsextraktion', in K. U. Carstensen, C. Ebert, E. Endriss, S. Jekat, R. Klabunde & H. Langer (eds.), *Computerlinguistik und Sprachtechnologie*, Spektrum, Heidelberg, Berlin, pp. 448–456.

Patrick, P.: 1999, *Urban Jamaican Creole. Variation in the Mesolect.*, no. G17 in Varieties of English Around the World, John Benjamins Publishing Co., Amsterdam & Phildadelphia.

Patrick, P.: 2004, 'Jamaican Creole: Morphology and syntax.', in B. Kortmann, E. W. Schneider, C. Upton, R. Mesthrie & K. Burridge (eds.), *A Handbook of Varieties of English. Vol 2: Morphology and Syntax*, Topics in English Linguistics, Mouton de Gruyter, Berlin & New York, pp. 407–438.

Samuelsdorff, P.-O.: 1989, 'Simulation of a Functional Grammar in Prolog', in J. H. Connolly & S. C. Dik (eds.), *Functional Grammar and the Computer*, pp. 29–44.

Schwiebert, S.: 2005, 'Entwicklung eines agentengestützten Systems zur Paradigmenbildung', in B. Fisseni, C, B. Schröder & P. Wagner (eds.), *Sprachtechnologie, mobile Kommunikation und linguistische Ressourcen. Beiträge zur GLDV-Tagung 2005 in Bonn*, no. 8 in Sprache, Sprechen und Computer, Lang, Frankfurt a.M., pp. 633–646.

Schwiebert, S. & J. Rolshoven: 2006, 'SOG: Ein selbstorganisierender Graph zur Bildung von Paradigmen', in Rapp, Reinhard, Sedlmeier & Zunker-Rapp (eds.), *Perspectives on Cognition. A Festschrift for*, Pabst Science Publishers, Lengerich.

Siegmund, P.: 2004, 'English', in T. Roelke (ed.), *Variationstypologie*, De Gruyter, Berlin.

Steeg, F.: 2006, 'Functional Grammar', in *Wikipedia, Die freie Enzyklopädie. Bearbeitungsstand: 11. Feb 2006, 6:56 UTC.*

Steeg, F., C. Benden & P.-O. Samuelsdorff: 2006, 'Generating Linguistic Expressions from Underlying Clause Structures', Poster presented at the Twelfth International Conference on Functional Grammar. Sao Jose do Rio Preto, Brazil: Universidade Estadual Paulista.

Strube, G. (ed.): 2001, *Digitales Wörterbuch der Kognitionswissenschaft*, Klett-Cotta.

Van Staden, M.: 2006, 'Papuan narratives in Functional Discourse Grammar', Poster presented at the Eleventh Biennial Symposium: Intertheoretical Approaches to Complex Verb Constructions. Houston: Rice University.

Witten, I. & E. Frank: 2000, *Data Mining: Praktische Werkzeuge und Techniken für das maschinelle Lernen*, Hanser.

van Zaanen, M.: 2002, *Bootstrapping Structure into Language: Alignment-Based Learning*, Ph.D. thesis, University of Leeds.