Methoden zur intelligenten Informationsextraktion im Internet

Günter Neumann*
DFKI GmbH
Stuhlsatzenhausweg 3
66123 Saarbrücken, Germany
neumann@dfki.uni-sb.de

*Erscheint in den Proceedings ONLINE97, Hamburg, 1997.

Inhaltsangabe

1 Einführung

Mit der rasanten Verbreitung des Internet tritt das Problem der Informationsüberflutung immer stärker in den Vordergrund: Je mehr on-line Texte zur Verfügung stehen, desto schwieriger wird es, das Informationspotential gezielt zu nutzen, d.h. relevante Informationen zu finden und zu extrahieren.

Um der Informationsüberflutung adäquat entgegen treten zu können, wird bereits fieberhaft nach neuen Technologien für zukünftige intelligente Informationsmanagementsysteme geforscht. Eine sich neu etablierte Forschungsrichting ist die Erforschung und Realisierung von Systemen zur *Informationsextraktion* (IE). Das Ziel der IE ist die Konstruktion von Systemen, die gezielt aufgabenspezifische Information aus elektronischen Texten aufspüren und strukturieren können, bei gleichzeitigem "überlesen" irrelevanter Information [Cowie and Lehnert, 1996].

Die Kernfunktionalität eines IE-Systems läßt sich dabei kurz wie folgt charakterisieren: Auf der Basis einer vorgegebenen Beschreibung der erwünschten Information (in Form von Anfragen oder Suchmustern) und einer Menge von Texten, isoliert das IE-System die relevanten Textfragmente, extrahiert aus jedem Fragment die spezifisch erwünschte Information und konstruiert aus den verschiedenen Informationsteilen in systematischer Weise die komplexe Zielstruktur. [Cowie and Lehnert, 1996]. Die so extrahierten Daten können dann vielseitig eingesetzt werden, z.B. zur feinkörnigen Textfilterung oder -klassifikation, als Einträge für Datenbanken oder Expertensysteme, oder sie können zur inhaltsbasierten Indizierung in Textretrievalsystemen eingesetzt werden.

Die ersten aktuell (meist in Forschungslaboratorien) implementierten IE-Systeme verfügen bereits über sehr vielversprechende schnelle und robuste Technologien zur Verarbeitung großer Textmengen (cf. [Sundheim, 1995], [Chinchor *et al.*, 1993], und [Grishman and Sundheim, 1996]). Dies ist darin begründet, daß sie eine gezielte *partielle* Textanalyse mit einer ansprechenden Genauigkeit durchführen können, wobei sehr effiziente und robuste *flache* Sprachverarbeitungstechnologien eingesetzt werden. Im gewissen Sinne sind diese Systeme für bestimmte Arten von Information "sensibilisiert" worden und können dadurch in einfacher Weise irrelevante Textpassagen überspringen ohne eine "tiefe" Verstehensanalyse durchführen zu müssen.

Die Mehrzahl der existierenden IE-Systeme ist für die Verarbeitung von Texten in englischer Sprache entwickelt worden. Ein zentraler Nachteil der aktuellen IE-Technologien ist der sehr eingeschränkte Grad an Portabilität auf neue Anwendungsdomänen und Aufgaben, was auch in einer eingeschränkten Wiederverwendung der Wissensquellen begründet ist. Konsequenterweise wurden während der sechsten "Message Understanding Conference" (MUC-6) als wesentliche Ziele zukünftiger IE-Technologien folgende Schwerpunkte identifiziert (cf. [Grishman and Sundheim, 1996]):

- Entwicklung aufgabenunabhängiger Komponenten,

- Steigerung der Portabilität,
- Steigerung der "tiefen" Verstehensleistung.

In diesem Artikel stellen wir das IE-System SMES vor, ein an der DFKI GmbH entwickeltes IE-Kernsystem zur Verarbeitung speziell deutschsprachiger elektronischer Texte. Das zentrale Entwicklungsziel ist hierbei die einfache Portabilität und Adaptierung des Kernsystems für IE-Anwendungen unterschiedlicher Domänen und Komplexität. Wir konzentrieren uns in diesem Artikel auf die Beschreibung der zentralen Technologien und ihrer Implementation, die zur Erreichung der angestrebten Portabilität entwickelt wurden. Wir werden nur kurz auf einige auf der Basis des Kernsystems realisierten Anwendungssysteme eingehen (cf. Abschnitt 7).

2 Die Architektur des IE-Kernsystems SMES

Das zentrale Designkriterium des IE-Kernsystems SMES ist die Bereitstellung von leistungsstarken, robusten und effizienten natürlichsprachlichen Komponenten und linguistischen Wissensquellen, die in einfacher und dennoch flexibler Weise für unterschiedliche Aufgaben konfigurierbar sind. In diesem Sinne betrachten wir SMES als ein *IE-Kernsystem*. Die Konfigurierbarkeit wird erreicht durch:

- die Definition des Informationskontrollflusses (kaskadiert und/oder verzahnt),
- der Auswahl der notwendigen linguistischen Wissensquellen,
- die Spezifikation des Domainwissens,
- die Definition von zusätzlicher, aufgabenspezifischer Funktionalität.

Bild 1 zeigt eine Grobstruktur des IE-Kernsystems. Die zentralen Komponenten sind:

- Ein Textscanner auf der Basis endlicher Automaten: Seine Aufgabe ist die Erkennung von Textstruktur und speziellen Zeichenketten, wie z.B. Datums- und Zeitangaben, Abkürzungen, Wortgrenzen und Interpunktionszeichen.
- Eine sehr schnelle und robuste Morphologiekomponente zur Flexions- und Kompositaanalyse. Für jedes analysierte Wort bestimmt diese Komponente den zugehörigen Stamm, die Wortart und die Flexionsinformation (für mehrdeutige Wortformen werden natürlich entsprechend viele Lesarten geliefert). Im Falle von mehrdeutigen Ergebnissen wird anschließend eine Desambiguierung mittels wortartenspezifischen Regeln durchgeführt, die auch eine Erkennung unbekannter Wortformen leistet.

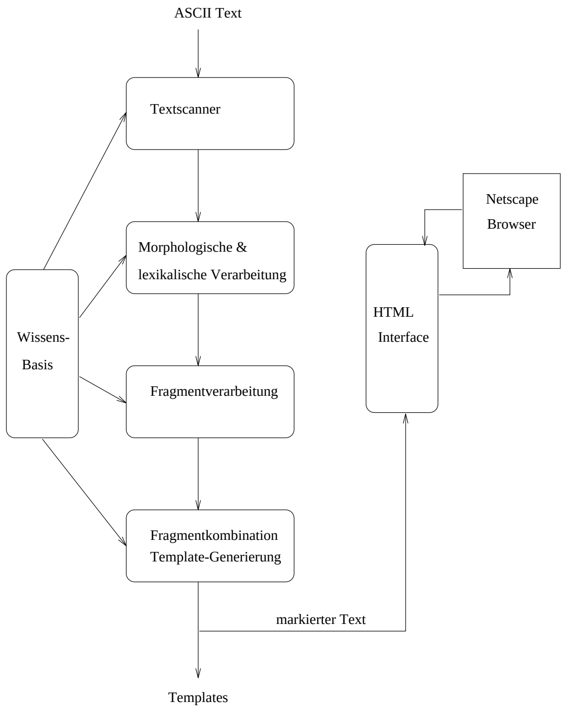

Abbildung 1: Grobstruktur der Architektur von SMES

- Eine Komponente zur deklarativen Spezifikation von Automatengrammatiken zur Beschreibung von Wortgruppen (z.B. Nominal-, Präpositional-, und Verbgruppen, komplexen Zeit- und Datumsausdrücken, Eigennamen). Eine Automatengrammatik besteht aus einer Menge von Fragment-Extraktions-Regeln, die als endlichen Transduktoren (finite state transducers FST) formuliert werden und die mittels einer generischen I/O-Schnittstelle über einen hohen Grad an Modularität bzw. Wiederverwendbarkeit verfügen. Ein Kompiler übersetzt die FST Spezifikationen in eine interne Form, so daß eine sehr effiziente Verarbeitung erreicht wird.

- Ein bidirektionaler, lexikongesteuerter flacher Parser führt eine Kombination von extrahierten Fragmenten durch. Das flache Parsing ist gesteuert durch sogenannte *Fragment- Kombinations-Regeln* von der Form $\langle FST_{links}, Anker, FST_{rechts} \rangle$, wobei *Anker* ein Lexikoneintrag ist (in der Regel ein Verb, wie z.B. “treffen”) oder eine Klasse von Lexikoneinträgen bezeichnet, wie z.B. “transitive-verb”. Diese Regeln sind direkt mit den entsprechenden Lexikoneinträgen assoziiert und werden unmittelbar nach der Erkennung entsprechender Einträge aktiviert. Angewendet werden sie auf den linken und rechten Strom von extrahierten Fragmenten.

- Eine Schnittstelle zu TDL, einer typisierten Beschreibungssprache für Constraint-basierte Grammatiken[1]. In SMES wird TDL sowohl zum konzeptbasierten Lexikonzugriff verwendet als auch zur Überprüfung grammatikalischer Regularitäten.

- Die Wissensbasis konstituiert die Ansammlung der verschiedenen Wissensquellen, u.a., das Lexikon, die verschiedenen Subgrammatiken, die Satzgrammatik und die Templatemuster. Gegenwärtig umfaßt das verwendete Lexikon ca. 120.000 Stammeinträge, umfangreiche Subgrammatiken für einfache und komplexe Datums-und Zeitausangaben, Eigennamen, Firmennamen, Währungsangaben, sowie Grammatiken für nominale und präpositionale Phrasen und Verb/Modifier-Ausdrücke.

Zusätzlich zu diesen Komponenten verfügt das System über einen graphischen Editor für Textitems und eine HTML-basierte Schnittstelle. Sie markiert alle erkannten Textteile durch getypte Klammern, die ebenfalls als Hyperlinks zu den internen Strukturen der extrahierten Information dienen. Damit kann der Benutzer nicht nur elegant durch die erkannten Textteile navigieren, sondern ebenfalls bei Bedarf die internen Ergebnisstrukturen inspizieren.

Eine hohe Portabilität des System wird durch folgende zentralen Eigenschaften erreicht:

- Die Ausgabestrukturen aller Komponenten werden uniform als Merkmal/Werte Paare (das sind verbundähnliche Strukturen) repräsentiert, zusammen mit einem Typbezeichner und den Start- und Endpunkten der relevanten Textstelle. Wir bezeichnen solche Strukturen als *Textitems*.

- Die Ergebnisstrukturen jeder Komponente werden in einer speziellen Datenstruktur gespeichert, so daß jede Komponente Zugriff auf die Ergebnisse der vorgelagerten Komponenten hat. Dies ermöglicht sowohl einen kaskadierten als auch eng verzahnten, inkrementellen Kontrollfluß. Der erste Fall ermöglicht die Spezifikation von kaskadierten endlichen Automaten [Appelt *et al.*, 1993]. Der zweite Fall erlaubt die Definition von Automaten, die eine enge Verzahnung von z.B. konzeptbasierter Schlüsselwortsuche, Fragmentextraktion und Templateinstanziierung leisten.[2]

Der zugrundeliegende Kontrollfluß in SMES ist wie folgt: Nach der Scanningphase, wird die Tokenkette morphologisch analysiert, wobei eine Flexions- und Kompositaanalyse durchgeführt wird, der eine Desambiguierungsphase folgt. Danach führen

[1] Ursprünglich wurde TDL zur Entwicklung von modernen Kompetenzgrammatiken entwickelt [Krieger and Schäfer, 1994].

[2] Natürlich können beide Methoden auch kombiniert werden.

die kompilierten endlichen Automaten eine Extraktion relevanter Fragmente durch (z.B. Erkennung von Eigennamen, Firmennamen, Nominalgruppen). An dieser Stelle sei betont, daß die Fragmenterkennung gleichzeitig die Ergebnisse des Scanners, der Morphologie, und sogar bereits erkannter Fragmente berücksichtigen kann. Die Menge der erkannten Fragmente zusammen mit einer Menge von Triggerausdrücken (lexikalischen Ankern, Interpunktionszeichen) bilden die Eingabe für den lexikalisch-gesteuerten flachen Parser. Dieser aktiviert die über die lexikalischen Anker ausgewählten Fragment-Kombinations-Regeln, die ihrerseits eine Auswahl zwischen den Fragmenten treffen. Als Ergebnis liefert der flache Parser instanziierte Templates, die eigentlichen Zielstrukturen. Zuletzt markiert die HTML-Schnittstelle die entsprechenden relevanten Textstellen im Ausgangsdokument und konstruiert entsprechende Hyperlinks zu den internen Ergebnisstrukturen.

SMES ist bereits erfolgreich für die Klassifikation von elektronisch versendeten Vortragsankündigungen eingesetzt, sowie zur automatischen Terminvereinbarung und zur Extraktion von Information aus Wirtschaftsmeldungen (vgl. Abschnitt 7 für weitere Details). Wir werden nun auf einige Details des Systems genauer eingehen.

3 Die Verarbeitung auf der Wortebene

Textscanning Jede Textdatei wird zuerst durch den *Textscanner* vorverarbeitet. Auf der Basis von regulären Ausdrücken (der Textscanner ist mit dem bekannten Unix-Werkzeug Lex implementiert) identifiziert diese Komponente die Textstruktur (z.B. Paragraphen, Einrückungen), Wörter, Zahlen, Datums- und Zeitangaben (z.B. "1.3.96", "12:00 h") und expandiert Abkürzungen. Die Ausgabe des Textscanners ist ein Tokenstrom, wobei jedes Wort als alphabetischer String repräsentiert ist (inklusive Trennzeichen, z.B. "Daimler-Benz"). Zahlen, sowie Datums- und Zeitangaben werden normalisiert und in Form von Attribut/Werte Paaren ausgegeben. Zum Beispiel wird die Zeichenkette "1.3.96" als (:date ((:day 1)(:mon 3)(:year 96)) und "13:15 h" als (:time ((:hour 13)(:min 15))) dargestellt.

Morphologische Verarbeitung Sie folgt dem Textscanning und führt eine Flexions- und Kompositaanalyse durch. Die Verfügbarkeit einer sehr effizienten Kompositaanalyse ist gerade für die deutsche Sprache von zentraler Bedeutung, da die Kompositabildung im Deutschen sehr produktiv ist und in der Regel durch simplifizierende Vollformenlexika nicht adäquat behandelt werden kann.

MONA, die morphologische Komponente des Systems SMES ist ein Nachfolger von MORPHIX, einer sehr schnellen klassifikationsbasierten Morphologiekomponente für das Deutsche [Finkler and Neumann, 1988]. MONA kombiniert den klassifikationsbasierten Ansatz mit dem bekannten zwei-Ebenen Modell der Morphologie [Koskenniemi, 1983]. Aktuell betreffen die Erweiterungen:

- Die Verwendung von TRIES (cf. [Aho *et al.*, 1983]) als uniformer Datentyp zur Speicherung der gesamten lexikalischen Information (z.B. Lexikoneinträge, Flexionsendungen, Präfixe).

- Die Kompositaanalyse ist als rekursive TRIE-Traversierung realisiert, wobei während der Traversierung zwei-Ebenen Regeln angewendet werden, die eine Erkennung der linguistisch wohlgeformten Zerlegung eines Kompositums leisten (z.B. "Staubecken" liefert "Stau Becken" und "Staub Ecken").

MONA liefert für jede Wortform, die Wortform zusammen mit all ihren Lesarten. Eine Lesart ist ein Tripel der Form $\langle Stamm, Flexion, POS \rangle$, wobei *Stamm* der Stamm der Wortform ist, *Flexion*, die Flexionsinformation kodiert (z.B. Numerus, Kasus) und *POS* die Wortart kodiert.

Gegenwärtig ist MONA für die deutsche und italienische Sprache implementiert. Die deutsche Version hat mit einem Lexikon von über 120.000 Stammeinträgen eine sehr große Abdeckung. Das Laufzeitverhalten von MONA ist exzellent (ca. 5000 Wortformen/Sek. ohne Kompositaanalyse, ca. 2800 Wortformen/Sek. inklusive der Berechnung aller lexikalisch möglichen Zerlegungen von zusammengesetzten Wortformen).[3]

Desambiguierung Morphologisch mehrdeutige Wortformen werden bezüglich der Wortarten mittels Groß/Kleinschreibregeln[4] und Filterregeln desambiguiert, die mittels einer Version des un-supervised Taggers von Brill bestimmt wurden [Brill, 1995]. Die Filterregeln erlauben auch die Prädiktion der Wortart unbekannter Wortformen.

Die Filterregeln werden auf der Basis nicht-annotierter Korpora bestimmt. Die Korpora werden zuerst mittels MONA analysiert, wobei unbekannte Wortformen als Nomen, Adjektiv und Verb markiert werden. Danach werden kontextsensitive Regeln der Form *Wähle Wortart Nomen von Nomen oder Verb, wenn das vorherige Wort als Artikel erkannt wurde* angewendet.

Erste Experimente mit einem Trainingskorpus von 100.000 Wortformen und einer Menge von ca. 280 gelernten Filterregeln ergaben eine Desambiguierungsrate von 91.8%.[5]

Es ist wichtig zu betonen, daß diese Art der Desambiguierung keine annotierten Testkorpora benötigt und unbekannte Wortformen behandelt. Wir erwarten durch einen vermehrten Einsatz der linguistischen Regularitäten von MONA eine erhebliche Verbesserung unserer aktuellen Werte.

---

[3] Die Messungen wurden auf einer Sun 20 und einem on-line Lexikon von 120.000 Einträgen durchgeführt.

[4] Im allgemeinen werden nur Nomen groß geschrieben (z.B. "der Wagen" vs. "wir wagen"). Da Schreibfehler in Pressenachrichten relativ selten vorkommen, ist eine Desambiguierung nach Groß/Kleinschreibung eine einfache, aber zweckmäßige Methode.

[5] Brill berichtet von einer 96% Genauigkeit, wobei er allerdings keine unbekannten Wortformen berücksichtigt. In [Aone and Hausman, 1996] wird eine Erweiterung von Brills Methode vorgestellt, die unbekannte Wortformen berücksichtigt. Für die spanische Sprache ergibt sich dort eine Genauigkeit von 92.1%.

4 Die Verarbeitung von Wortgruppen

Die Erkennung von Wortgruppen wird mittels Fragment-Extraktions-Regeln durchgeführt, die als endliche Transduktoren (finite state transducers FST) formuliert und in eine interne effiziente Form kompiliert werden, wobei der Kompiler eine Erweiterung von [Krieger, 1987] ist. Ein FST besteht aus einem Namen, dem Erkennungsteil, der Ausgabebeschreibung und einer Menge von Kompileroptionen.

Der Erkennungsteil Ein FST operiert auf einem Tokenstrom. Der Erkennungsteil wird zur Beschreibung von regulären Ausdrücken über Teilsequenzen des Tokenstroms verwendet. Um einen hohen Modularitätsgrad zu erreichen, werden die verschiedenen Tokenarten vermittels *Basiskanten* definiert, wobei eine Basiskante als Prädikat über Tokenklassen interpretiert wird. Eine Basiskante ist eine Tripel der Form $\langle name, test, variable\rangle$, wobei *name* der Name der Kante und *test* ein Prädikat ist. *variable* bindet das aktuell gelesene Token T_c, falls *test* angewendet auf T_c zutrifft.

Zum Beispiel testet die folgende Kante (:mona-cat "partikel" pre), ob T_c (von MONA berechnet) ein Partikel ist und bindet im positiven Fall T_c an *pre* (um genauer zu sein, jede Variable kodiert einen Stapel, so daß T_c auf den Stapel als neuestes oberstes Element gelegt wird).

Wir nehmen an, daß für jede Komponente eine Menge vordefinierter Basiskanten existiert. Weiterhin nehmen wir an, daß solch eine Basismenge ab einem bestimmten Zeitpunkt der Systementwicklung fix ist, so daß sie als Basisbausteine dem Grammatikentwickler zur Verfügung stehen.

Unter Verwendung solcher Basiskanten kann der Erkennungsteil dann einfach mittels regulären Ausdrücken formuliert werden, wobei wir eine funktionale Schreibweise verwenden. Zum Beispiel kann der Erkennungsteil für einfache nominale Gruppen wie folgt definiert werden:

```
(:conc
  (:star≤n (:mona-cat "det" det) 1)
  (:star (:mona-cat "adj" adj))
  (:mona-cat "n" noun))
```

In diesem Sinne ist eine Nominalphrase die Konkatenation eines optionalen Artikels (ausgedrückt durch den Wiederholungsoperator :star≤n, mit n von 0 bis 1), gefolgt von Null oder mehreren Adjektiven, gefolgt von einem Nomen.

Der Ausgabebeschreibungsteil Für die Konstruktion der Ausgabestruktur eines FST werden die Variablen des Erkennungsteiles gesammelt und anschließend mittels speziellen Funktionen die Zielstruktur aufgebaut. Um eine hohe Wiederverwendbarkeit der FST zu erreichen, ist es wichtig die Konstruktion der Ausgabe vom

Erkennungsteil zu trennen. Daher wird die Ausgabe mittels der generischen Funktion BUILD-ITEM realisiert, die als Eingabe die Variablen der Basiskanten und einen Klassenbezeichner des FST erhält. Wird zum Beispiel :np als Klassenname für Nominalphrasen (NP) verwendet, dann ist eine mögliche Ausgabebeschreibung für den obig definierten NP-Erkennungsteil

(build-item :type :np :out (list det adj noun)).

Die Funktion BUILD-ITEM diskriminiert dann gemäß des angegebenen Klassennamen zu speziell vordefinierten Ausgabekonstruktionsfunktionen[6]. Die Konstruktion der Ausgabe durch spezielle Operationen, die nicht unmittelbar im FST aufgerufen werden, sondern mittelbar in der generischen Funktion BUILD-ITEM, erlaubt die Änderung der Ausgabestruktur ohne den FST (insbesondere den Erkennungsteil) selbst ändern zu müssen.

Spezielle Kanten Die Basiskanten (:var var), (:current-pos pos) und (:seek name var) sind vordefiniert und haben eine besondere Bedeutung. Die Kante (:var var) wird zum Konsumieren eines beliebigen Tokens benutzt, d.h., daß das aktuelle Token wird auf jedenfall akzeptiert. Mithilfe der Kante :current-pos wird die aktuelle Position des Tokens in der Variable pos gespeichert, und die Kante :seek aktiviert den FST mit Namen name, wobei var die Ausgabebeschreibung des aufgerufenen FST bindet. Dies ist vergleichbar mit der :seek Kante in erweiterten Übergangsnetzwerken (Augmented Transition Networks) mit dem wesentlichen Unterschied, daß in unserem Formalismus rekursive Aufrufe von FST *verboten* sind. Daher kann :seek auch als Makroexpansionsoperator betrachtet werden. Der :seek Mechanismus ist sehr hilfreich für eine modulare und hierarchische Spezifikation von Subgrammatiken. Das folgende Beispiel demonstriert den Einsatz der verschiedenen speziellen Kanten:

```
(compile-regexp
 (:conc
   (:current-pos start)
   (:alt
     (:seek time-phase time)
     (:conc
       (:star≤n (:seek time-expr-vorfield vorfield) 1)
       (:seek mona-time time)))
   (:current-pos end))
  :name time-expr
  :output-desc
```

[6] Man Beachte, daß im obigen Beispiel die Variablen DET und ADJ eventuell keine Werte erhalten haben. In diesem Fall wird das spezielle Symbol NIL verwendet

```
(build-item :type time-expr :start start
            :end end :out (list vorfield time))))
```

Dieser FST erkennt Ausdrücke wie "spätestens um 14:00 h" mit der zugehörigen Ausgabestruktur ((:out (:time-rel . "spaet") (:time-prep . "um") (:minute . 0) (:hour . 14)) (:end . 4) (:start . 0) (:type . time-expr))

Schnittstelle zu TDL Die Schnittstelle zu TDL, einer getypten merkmalsbasierten Sprache und Typinferenzsystem (cf. [Krieger and Schäfer, 1994]) wird ebenfalls durch Basiskanten realisiert. TDL ermöglicht die Definition von hierarchischen Typverbänden bestehend aus Typ- und Merkmalsgleichungen. TDL wurde ursprünglich für die Spezifikation moderner Grammatikformalismen entwickelt. In SMES wird TDL eingesetzt zur

1. Definition von aufgabenspezifischen Konzepthierarchien und
2. zur Spezifikation von syntaktischen Kongruenzbedingungen, die zur Überprüfung von Kasus- und Numerusübereinstimmung benötigt werden.

Die Konzepthierarchien werden benutzt, um einen konzeptbasierten lexikalischen Zugriff zu erlauben (z.B. zur Extraktion von Wörtern, die kompatibel zu einer vorgegebenen Oberklasse sind). Die Kongruenzbedingungen werden zum Ausfiltern ungrammatikalischer Phrasen oder bei der Kombination von Phrasen nach syntaktischen Kriterien eingesetzt.

Die Integration von TDL in einen FST kann einfach und unmittelbar durch die Spezifikation von Basiskanten realisiert werden. Beispielsweise akzeptiert die Kante

```
(:mona-cat-type (:and "n" "device") var)
```

ein Wort genau dann, wenn es morphologisch als Nomen erkannt wurde und wenn sein lexikalischer Konzepteintrag von "device" subsumiert wird. Als ein Beispiel einer syntaktischen Kongruenzbedingung sei folgende Kante definiert

```
(:mona-cat-unify "det"
"[(num %1)(case %2 = gen-val) (gender %3)]"
agr det)
```

die überprüft, ob das aktuelle Token ein Artikel im Genitiv ist (z.B. "des Antrages"). Im positiven Fall wird die Kasusinformation an agr und das Token an det gebunden. Wenn im weiteren Erkennungsteil des FST eine vergleichbare Kante auch für ein eventuelles Nomen definiert ist, dann wird dieses Nomen (und damit eventuell der gesamte FST) nur akzeptiert, wenn auch das Nomen den Genitivkasus besitzt. In diesem Sinne werden also nur solche Wortgruppen erkannt, die im Kasus übereinstimmen (kongruent) sind.

Die Verwendung von TDL unterstützt die Spezifikation von sehr kompakten und modularen Subgrammatiken und erhöht dadurch ihre Portabilität für neue Aufgaben. Weiterhin bietet die Schnittstelle zu TDL eine sehr einfache und elegante Methode zur Integration "tiefer" Verarbeitungskomponenten.

Kontrollparameter Zur flexiblen Steuerung des Erkennungsteils kann spezifiziert werden, ob ein exakter Vergleich mit dem Tokenstrom durchzuführen ist, oder ob es bereits genügt, wenn der Erkennungsteil eines FST mit dem Präfix, Suffix oder Infix des Tokenstroms übereinstimmt. In Verbindung mit dem Kleene-Operator erlaubt der Präfixvergleich das Auffinden aller Teilsequenzen im Tokenstrom, die von dem Erkennungsteil eines oder mehrerer FST erkannt werden (z.B. zum Extrahieren aller Nominalphrasen und Zeitausdrücke).[7] Beispielsweise extrahiert der folgende FST alle genitiven NPs, die im Eingabestrom enthalten sind:

```
(compile-regexp
  (:star
   (:alt
    (:seek gen-phrase x)
    (:var dummy)))
  :output-desc (build-item :type list :out x )
  :prefix T
  :suffix NIL
  :name gen-star)
```

Zusätzlich kann mittels eines booleschen Parameters angegebenen werden, ob längste oder kürzeste übereinstimmende Tokensequenzen in der Eingabe priorisiert werden sollen ("längstmöglicher Vergleich" ist die Voreinstellung, vgl. auch [Appelt *et al.*, 1993] wo ebenfalls diese Voreinstellung präferiert wird).

5 Die Konstruktion komplexer Strukturen und Templategenerierung

Bidirektionales flaches Parsing Zur Kombination von extrahierten Fragmenten wird ein lexikalisch gesteuerter, bidirektionaler flacher Parser eingesetzt, der Fragment-Kombinations-Regeln (fragment combination patterns FCP) anwendet, die direkt mit Lexikoneinträgen (in der Hauptsache Verben) assoziiert sind. Wir nennen solche Lexikoneinträge *Anker*.

Der Eingabestrom für den flachen Parser besteht aus einer *doppelt verketteten* Liste aller extrahierten Fragmente, die in einem Eingabetext gefunden wurden, allen

[7] Das resultierende Verhalten ist vergleichbar mit dem, welches [Roche and Schabes, 1995] "die lokale Erweiterung eines FST" nennen.

Interpunktions- und Textzeichen und allen gefundenen Ankern (d.h. alle anderen Tokens des Eingabetextes werden ignoriert). Für jeden Anker wendet der flache Parser das assoziierte FCP an. Vom Gesichtspunkt eines Ankers ist der Eingabestrom in eine linke und rechte Teilsequenz aufgespalten. Die Anwendung eines FCP startet unmittelbar von der Position des Ankers und durchsucht abwechselnd die linke und rechte Teilsequenz nach Fragmenten, die mit der Beschreibung im FCP übereinstimmen (vgl. auch Bild 2). Die Suche endet entweder, wenn der Textanfang oder das Textende erreicht ist oder, wenn ein Interpunktions- oder Textzeichen oder ein anderer Anker, die explizit als Endemarkierung definiert sind, erkannt werden.

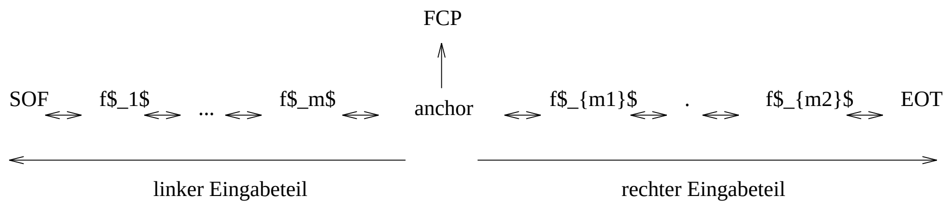

Abbildung 2: SOF steht für start-of-text, EOT für end-of-text, f_i ist das i-te Fragment. Ein FCP, das mit dem Anker assoziiert ist, wird auf den linken und rechten Eingabeteil angewendet. Wenn der Punkt als Endemarkierung definiert ist, dann wird das Fragment f_{m2} nicht mehr vom FCP betrachtet.

Die allgemeine Form einer Fragment-Kombinations-Regel Ein FCP besteht aus einem Namen, einem Erkennungsteil für den linken und einem für den rechten Teil, einer Ausgabebeschreibung und einer Menge von Bedingungen über den Typ und die Anzahl der als relevant betrachteten Fragmente. Das folgende Beispiel zeigt ein FCP für intransitive Verben, wie z.B. *kommen* oder *beginnen*:

```
(compile-anchored-regexp
 ((:set (cdr (assoc :start ?*)) anchor-pos)
  (:set ((:np (1 1) (nom-val (1 1)))) nec)
   (:set ((:tmp (0 2))) opt))
  ((:dl-list-left
    (:star
      (:alt
       (:ignore-token (“,” “;”))
       (:ignore-fragment :type (:time-phase :pp))
       (:add-nec (:np :name-np)
              :np nec lcompl)
       (:add-opt (:time-expr :date-expr)
              :tmp opt lcompl))))
```

```
(:dl-list-right
  (:star
    (:alt
      (:ignore-token ("," ";"))
      (:add-nec (:np) :np nec rcompl)
      (:add-opt (:time-expr :date-expr)
              :tmp opt rcompl)))))
:name intrans
:output-desc (build-item :type :intrans
              :out (list anchor-pos lcompl rcompl)))
```

Die erste Liste speichert die aktuelle Position des aktiven Ankers und führt eine Menge von Bedingungen ein, die zur Beschränkung der Typen und der Anzahl von notwendigen und optionalen Fragmenten verwendet werden. Beispielsweise besagt die erste Bedingung, daß genau ein NP-Fragment (angegeben durch das Intervall (1 1)) im Nominativ gesammelt, und die zweite Bedingung besagt, daß höchstens zwei optionale Fragmente vom Typ :tmp gesammelt werden dürfen.

Diese zwei Bedingungen werden in den Basiskanten :add-nec und :add-opt verwendet. Die Arbeitsweise von :add-nec ist wie folgt: Ist das aktuelle Token ein Fragment vom Typ :np oder :name-np, dann wähle die Bedingung mit Typ :np. Stimmt das aktuelle Token im Kasus mit den Angaben in der Bedingung überein (was mittels Typsubsumption getestet wird), füge es der Stapelvariablen nec hinzu und reduziere die obere Grenze des Intervalls (1 1) um 1. Da im nächsten Zyklus die obere Grenze nun 0 ist, kann kein weiteres Element der Variable nec hinzugefügt werden. In ähnlicher Weise wird die Basiskante :add-opt verarbeitet.

Die Kanten :ignore-token und :ignore-fragment werden benötigt, um explizit zu definieren, welche anderen Tokens (Interpunktionszeichen oder Fragmente) nicht von :add-nec oder :add-opt nicht betrachtet werden sollen, also einfach "überlesen" werden sollen. Mit anderen Worten bedeutet dies, daß jedes Token, welches nicht in einem FCP erwähnt ist, die Anwendung des linken oder rechten Erkennungsteiles stoppt und somit gegebenenfalls, das gesamte FCP.

Die Generierung von Templates Ein FCP definiert eine Beschränkung auf die Anzahl und den Typ der zu kombinierenden Fragmente. Nach erfolgreicher Anwendung eines FCP bilden die kombinierten Fragmente zusammen mit dem Anker ein instanziiertes Template. Ein Template ist eine verbundähnliche Struktur bestehend aus Merkmalen und ihren Werten, wobei jedes Fragment und der Anker ein Merkmal/Werte Paar konstituiert. Ein FCP definiert ebenfalls, welche Klasse von Fragmenten notwendig bzw. optional für die Konstruktion des gesamten Templates sind. Im allgemeinen werden FCPs zur Spezifikation von linguistisch orientierten Satzstrukturen und zur Definition von Datenbankeinträgen eingesetzt.

6 Abdeckung der linguistischen Wissensquellen

Die Zeit- und Datumsgrammatik deckt einen sehr weiten Bereich von komplexen Ausdrücken ab, u.a. nominale, präpositionale und koordinierte Angaben, sowie kombinierte Zeit-Datumsausdrücke (z.B. "vom 19. (8.00 h) bis einschl. 21. Oktober (18.00 h)" liefert: (:pp (from :np (day . 19) (hour . 8) (minute . 0)) (to :np (day . 21) (month . 10) (hour . 18) (minute . 0))))

Die Grammatik für Nominal-und Präpositionalphrasen deckt u.a. ab: koordinierte Phrasen, verschiedene Formen von Adjektiv- und Genitivkonstruktionen und Pronomen. Die Ausgabe spiegelt die Abhängigkeitsbeziehungen der beteiligten Wörter wider (z.B. " Die neuartige und vielfältige Gesellschaft " liefert: (((:sem (:head "gesellschaft") (:mods "neuartig" "vielfaeltig") (:quantifier "d-det")) (:agr nom-acc-val) (:end . 6) (:start . 1) (:type . :np)))

Mehr als dreißig syntaktische Subkategorisierungsrahmen für Verben sind als FCP definiert (z.B. verschiedene Transitivformen). Diese Verbrahmen definieren keine linearen Ordnungrelationen, was aber keine wesentliche Übergenerierung zur Folge hat, da die deutsche Sprache über eine sehr freie Wortstellung verfügt.

Die Spezifikation der linguistischen Wissenssourcen basiert auf einer Analyse von ungefähr 250 Texten aus den Bereichen Vortragsankündigung, Terminplanung und Wirtschaftsnachrichten, wobei ein bottom-up Grammatikentwicklungsansatz gewählt wurde.

7 Aktuell realisierte Anwendungen

Auf Basis von SMES sind bisher folgende Anwendungssysteme implementiert:

1. Klassifikation von elektronisch versendeten Vortragsankündigungen nach Sprecher, Titel, Ort und Zeit;
2. automatische Terminvereinbarung: Extraktion der Terminart, Dauer, Zeitraum, Sender, Empfänger, Thema;
3. Umsatz-/Gewinninformation von Firmen aus elektronischen Wirtschaftsmeldungen: Extraktion von Firmenname, Datum, Umsatz, Gewinn, Betrag, Differenz.

Bei der Realisierung dieser Anwendungssysteme konnte die Kernmaschinerie bestehend aus Textscanner, Morphologiekomponente, Basiskanten, Subgrammatiken und Fragment-Kombinations-Regeln in ihrem Kern unverändert übernommen werden.

Im Fall (1) wurden zusätzliche Subgrammatiken zur Erkennung bestimmter Textstrukturen definiert, da die Textstruktur hier eine wichtige Quelle zur Lokalisierung von relevanter Information ist. Da aber die Form von Vortragsankündigungen nicht standardisiert ist, mußten flache natürlichsprachliche Methoden eingesetzt werden. Dadurch ergibt sich als zentrale Strategie eine Mischung aus konzeptbasierter Schlüsselwortsuche und eingeschränkter flacher syntaktischer Analyse.

In (2) ist SMES in dem System COSMA eingebettet, ein deutschsprachiger Server für planbasierte Terminvereinbarungssysteme. Im Kontext von COSMA wird SMES als flaches natürlichsprachliches Kernsystem eingesetzt. Da COSMA über eigene Strategien zur semantisches Analyse und Diskursanalyse verfügt, konnte SMES ohne Änderung seiner Funktionalität eingebettet werden.

Die meisten Änderungen mußten für die Anwendung (3) durchgeführt werden. Dies betrifft vor allem die Definition neuer Subgrammatiken zur Behandlung von Firmennamen und Währungsangaben, sowie die Entwicklung spezieller Methoden zur Referenzauflösung. Die Kernmaschine konnte jedoch ohne wesentliche Änderung eingesetzt werden.

Die Verarbeitung mit SMES ist sehr robust und effizient (zwischen einer und zehn Sekunden in Abhängigkeit der Textgröße—von kurzen Pressemitteilungen bis zu kurzen Texten (etwa eine DinA4 Seite)). In allen drei Anwendungen erreichten wir eine hohe Abdeckung und gute Ergebnisse. Da es allerdings zum einen an vergleichbaren deutschsprachigen System mangelt und zum Anderen noch keine allgemeinen Evaluierungsstandards für Deutschsprachige IE-Systeme existieren, sind die aktuellen Meßergebnisse zu idiosynkratisch. Allerdings haben wir gerade die Implementation einer vierten Anwendung zusammen mit einem Industriepartner begonnen, in dem eine systematische Evaluierung durchgeführt werden wird. Wir erhoffen uns hierdurch aussagekräftiges Zahlenmaterial.

8 Verwandte Arbeiten

In Deutschland steckt die Entwicklung von IE-Systemen auf der Basis innovativer Sprachtechnologie noch in den Kinderschuhen. Zwar beschreiben [Hahn, 1992; Bayer *et al.*, 1994] ebenfalls natürlichsprachlich basierte IE-Systeme, allerdings verwenden sie weniger ausgereifte Technologien. Unsere Arbeit ist am meisten beeinflußt durch die Arbeiten von [Hobbs, 1992; Appelt *et al.*, 1993; Grishman, 1995], sowie in [Anderson *et al.*, 1992; Dowding *et al.*, 1993].

9 Schlußbemerkungen

In diesem Papier haben wir ein IE Kernsystem zur Extraktion relevanter Daten aus realen deutschsprachigen on-line Textdokumenten vorgestellt. Das zentrale Designkriterium des IE-Kernsystems SMES ist die Bereitstellung von leistungsstarken, robusten und effizienten natürlichsprachlichen Komponenten und linguistischen Wissensquellen, die in einfacher und dennoch flexibler Weise für unterschiedliche Aufgaben konfigurierbar sind. Das System ist vollständig in CommonLisp und C implementiert.

Unsere zukünftige Arbeit wird sich auf die Entwicklung und Integration von automatischen Methoden zur Adaption und Acquisition des Systems an neue Anwendungsdomainen konzentrieren, insbesondere zur automatischen Extraktion von Subgrammatiken aus einer gegebenen Kompetenzbasis und Lernverfahren für aufgabenspezifische Extraktionsregeln.

10 Danksagung

Die hier beschriebenen Forschungsarbeiten wurden unterstützt vom BMBF im Rahmen des Projektes Paradime (FKZ ITW 9704).

Mein besonderer Dank für die Zusammenarbeit bei der Entwicklung und Implementation gilt Rolf Backofen, Christian Braun, Markus Becker, Judith Baur, Thierry Declerck, Ludwig Dickmann und Judith Klein. Besonderer Dank auch an Hans Uszkoreit und Gregor Erbach für die vielen fruchtbaren Diskussionen zum Thema Informationsextraktion.

Literatur

[Aho *et al.*, 1983] A. Aho, J. Hopcraft, and J. Ullmann. *Data structures and and algorithms.* Addison Wesley, Reading, Mass., 1983.

[Anderson *et al.*, 1992] P. Anderson, P. Hays, A. Huettner, L. Schmandt, I. Nirenburg, and S. Weinstein. Automatic extraction of facts from press releases to generate news stories. In *3rd international conference of applied natural language*, pages 170–177, Trento, Italy, 1992.

[Aone and Hausman, 1996] C. Aone and K. Hausman. Unsupervised learning of a rule-based Spanish part of speech tagger. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, pages 53–58, Kopenhagen, Denmark, Europe, 1996.

[Appelt *et al.*, 1993] D. Appelt, J. Hobbs, J. Bear, D. Israel, and M. Tyson. Fastus: A finite state processor for information extraction from real world text. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, Chambery, France, August 1993.

[Bayer *et al.*, 1994] T. Bayer, U. Bohnacker, and H. Mogg-Schneider. Infoportlab – an experimental document understanding system. In *Proceedings of the 1st DAS*, 1994.

[Brill, 1995] E. Brill. Unsupervised learning of disambiguation rules for part of speech tagging. In *Very Large Corpora Workshop*, 1995.

[Busemann *et al.*, 1997] S. Busemann, T. Declereck, A. Diagne, L. Dini, J. Klein, and S. Schmeier. Natural language dialogue service for appointment scheduling agents, 1997. submitted to a conference.

[Chinchor *et al.*, 1993] N. Chinchor, L. Hirschman, and D. Lewis. Evaluating message understanding systems: An analysis of the third message understanding conference (muc-3). *Computational linguistics*, 19(3), 1993.

[Cowie and Lehnert, 1996] J. Cowie and W. Lehnert. Information extraction. *Communications of the ACM*, 39(1):51–87, 1996.

[Dowding *et al.*, 1993] J. Dowding, J. Gawron, D. Appelt, J. Bear, L. Cherny, R. Moore, and D. Moran. Gemini: A natural language system for spoken-language understanding. In *31th Annual Meeting of the Association for Computational Linguistics*, Ohio, 1993.

[Finkler and Neumann, 1988] W. Finkler and G. Neumann. Morphix: A fast realization of a classification–based approach to morphology. In H. Trost, editor, *Proceedings der 4. Österreichischen Artificial–Intelligence Tagung, Wiener Workshop Wissensbasierte Sprachverarbeitung*, Berlin, August 1988. Springer.

[Grishman and Sundheim, 1996] R. Grishman and B. Sundheim. Message Understanding Conference – 6: A Brief History. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, pages 466–471, Kopenhagen, Denmark, Europe, 1996.

[Grishman, 1995] R. Grishman. The NYU MUC-6 System or Where's the Syntax? In *Sixth Message Understanding Conference (MUC-6)*. Morgan Kaufmann, November 1995.

[Hahn, 1992] U. Hahn. On text coherence parsing. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING)*, pages 25–31, Nantes, France, Europe, 1992.

[Hobbs, 1992] J. Hobbs. The generic information extraction system. In B. Sundheim, editor, *Fourth Message Understanding Conference (MUC-4)*, McLean, Virginia, June 1992. Distributed by Morgan Kaufmann Publishers, Inc.,San Mateo, California.

[Koskenniemi, 1983] K. Koskenniemi. Two-level model for morphological analysis. In *Eighth IJCAI*, pages 683–685, Karlsruhe, 1983.

[Krieger and Schäfer, 1994] Hans-Ulrich Krieger and Ulrich Schäfer. $\mathcal{TDL}$—a type description language for constraint-based grammars. In *Proceedings of the 15th International Conference on Computational Linguistics, COLING-94*, pages 893–899, 1994.

[Krieger, 1987] Hans-Ulrich Krieger. Nil—eine lisp-basierte natürlichsprachliche schnittstelle zu ramses. unterstützung der nmr-diagnostik von hirn- und mammatumoren. Master's thesis, RWTH Aachen, 1987.

[Roche and Schabes, 1995] E. Roche and Y. Schabes. Deterministic part-of-speech tagging with finite state transducers. *Comnputational Linguistics*, 21(2):227–253, 1995.

[Sundheim, 1995] B. Sundheim, editor. *Sixth Message Understanding Conference (MUC-6)*, Washington, 1995. Distributed by Morgan Kaufmann Publishers, Inc.,San Mateo, California.