Vorlesung “Computerlinguistische Techniken”

1. Übung (16.10.2015)

Wintersemester 2015/16 – Prof. Dr. Alexander Koller

1 Mehrdeutigkeiten

Der folgende Beispielsatz (von Hans Uszkoreit) ist massiv mehrdeutig.

> Früher stellten die Frauen der Inseln am Wochenende Kopftücher mit Blumenmotiven her, die ihre Männer an den folgenden Montagen auf dem Markt im Zentrum der Hauptinsel verkauften.

Finden Sie möglichst viele Mehrdeutigkeiten, zeigen Sie, wie sie sich auf die Bedeutung des Satzes auswirken, und berechnen Sie, wie viele Lesarten der Satz insgesamt hat. Es ist dabei nicht wichtig, dass die einzelnen Lesarten in der wirklichen Welt möglich sind; diese Unterscheidung kann eine Grammatik auch nicht ohne weiteres treffen. Denken Sie besonders an syntaktische, lexikalische und referentielle Ambiguitäten.

Ich komme auf etwas über 250.000 Lesarten mit verschiedenen Strukturen und/oder Bedeutungen.

2 Kontextfreies Parsing

Die folgenden Regeln definieren eine kontextfreie Grammatik G:

S	→	NP VP	VP	→	V NP	VP	→	VP PP
NP	→	PN	NP	→	Det N	NP	→	NP PP
PP	→	P NP	PN	→	Hans	V	→	beobachtet
Det	→	den	Det	→	dem	N	→	Mann
N	→	Fernrohr	P	→	mit			

Bestimmen Sie alle Parsebäume, mit denen G den Satz “Hans beobachtet den Mann mit dem Fernrohr” erzeugen kann. Sie müssen nicht einen der Parsing-Algorithmen verwenden, die wir in der Vorlesung besprochen haben. Beschreiben Sie aber auf jeden Fall, wie Sie auf diese Parsebäume gekommen sind und warum Sie überzeugt sind, dass es keine weiteren gibt.

3 Der Recursive-Descent-Algorithmus

Implementieren Sie einen Recursive-Descent-Erkenner in Python. Parsen Sie die Sätze “Hans isst ein Kaesebrot” und “Kaesebrot isst Hans” mit der Käsebrot-Grammatik aus der Vorlesung. Ihr Erkenner soll den ersten Satz als grammatisch und den zweiten als ungrammatisch erkennen.

Ihre Implementierung darf auf die konkrete Grammatik angepasst sein. Sie müssen also keine Grammatiken einlesen oder im Speicher darstellen, sondern Ihr Programm darf aus einer Reihe von Funktionen mit den Namen S, NP, VP usw. bestehen, die sich gegenseitig aufrufen.

Ihr Programm soll ausgeben, welche möglichen Zerlegungen der RD-Erkenner ausprobiert und welche von ihnen erfolgreich sind, z.B. so:

```
>>> ww = ["Hans", "isst"]
>>> S(ww, 0, len(ww))
Call: S von 0 bis 2?
Call: NP von 0 bis 1?
Wort 'Hans' von 0 bis 1? -> ja
NP von 0 bis 1? -> ja
Call: VP von 1 bis 2?
VP von 1 bis 2? -> nein
S von 0 bis 2? -> nein
False
```

Verfolgen Sie damit nach, wie der RD-Erkenner die Beispielsätze akzeptiert oder ablehnt.

* Wenn Sie am Ende der Übung noch Lust und Zeit haben, können Sie für Bonuspunkte versuchen, eine allgemeine Implementierung des RD-Erkenners zu schreiben, die eine beliebige kontextfreie Grammatik in Chomsky-Normalform als Argument nimmt. Dafür müssen Sie sich eine geeignete Datenstruktur für kontextfreie Regeln ausdenken und eine allgemeine RD-Funktion schreiben, die diese Regeln anwendet.

Abgabe bis 23.10.2015, 12:00 Uhr per Mail an `mgerdes@uni-potsdam.de`.