6 Übung: Funktionen

Achtung: Diese Aufgabe besteht aus 2 Teilaufgaben. Die erste ist bereits mit Inhalten aus der Vorlesung am 13.06. lösbar, die zweite bezieht sich auf die Vorlesung am 20.06.
Beide Teilaufgaben werden zusammen abgegeben und zählen jeweils 50%.

6.1 Programmieraufgabe 5: Funktionen

6.1.1 P: Funktionen

Probabilistische kontextfreie Grammatiken (PCFG) werden häufig in der CL eingesetzt, um Sätze zu parsen, d.h. in ihre Konstituenten zu zerlegen. Eine PCFG besteht aus Phrasenstrukturregeln, die zusätzlich mit Wahrscheinlichkeiten versehen sind. Bsp:

```
NP --> DET N    0.5
NP --> DET A N  0.25
NP --> N        0.25
```

Eine NP expandiert also mit einer Wahrscheinlichkeit von 50% zu den beiden Präterminalen DET N. In der Grammatikdatei steht jeweils eine Regel pro Zeile.

In den ersten der folgenden Aufgaben geht es darum, Funktionen zu definieren, die die Verarbeitung und den Einsatz einer PCFG ermoeglichen. Zu jeder Funktion ist vorgegeben:

- Name der Funktion
- zu übergebende Argumente
- Rückgabewert

Es ist Teil der Aufgabe, geeignete Datenstrukturen für die zu übergebenden Daten zu definieren.

In dem zweiten Teil der Aufgabe geht es darum, eine PCFG aus einer Baumbank, d.h. einer Sammlung von Phrasenstrukturbäumen, abzuleiten. Die Baumbank ist in einem Klammerformat gegeben (s. Folien); pro Zeile steht eine Baum-Klammerstruktur.

Die Wahrscheinlichkeiten berechnen sich wie folgt: Die Wahrscheinlichkeit einer Regel ergibt sich aus ihrer Frequenz geteilt durch die Frequenz aller Regeln mit gleicher linker Seite (= Mutterknoten) Beispiel: Es kommen insgesamt 200 NX-Regeln vor:

```
NX --> NN      Freq: 140  -> 140/200 = 0.7
NX --> ART NN  Freq:  60  -> 60/200 = 0.3
```

Die Wahrscheinlichkeiten der verschiedenen NX-Regeln müssen zusammen 100% ergeben.

Orientieren Sie sich beim rekursiven Aufruf und der Analyse der Teilkonstituenten am Vorgehen, das auf den Folien zum Thema Rekursion gezeigt wird. Setzen Sie überall dort, wo sinnvoll möglich, Funktionen ein.

Zum Testen und Debuggen können Sie pprint und z.B. auch dieses Tool nutzen: http://lrv.bplaced.net/syntaxtree/

```
################################################################
# Aufgabenteil 1
################################################################

# Es sollen folgende Funktionen implementiert werden:

# 1. read_file
#    Einlesen einer Datei
#    Argument:      Name der einzulesenden Datei
#    Rueckgabewert: Inhalt der Datei in einem String

# 2. read_grammar
#    Einlesen einer Grammatik im obigen Format aus einem String
#    Argument:     String mit der Grammatik
#    Rückgabewert: Grammatikregeln in einer geeigneten Datenstruktur

# 3. write_grammar
#   Ausgabe einer Grammatik in eine Datei (gleiches Format wie oben;
#   2 Tabulatoren zwischen der Regel und ihrer Wahrscheinlichkeit)
#   Argumente: 1. Dateiname der Ausgabedatei
#              2. Grammatikregeln in einer geeigneten Datenstruktur
#   Rückgabewert: keiner

# 4. extract_rules
#   Extraktion aller Grammatikregeln mit einer bestimmten Mutterkategorie
#   aus einer Grammatik
#   Argumente: 1. Gewünschte Kategorie der Mutter
#              2. Grammatikregeln in einer geeigneten Datenstruktur
#   Rückgabewert: Grammatik mit nur denjenigen Regeln,
#                 die die gewünschte Mutterkategorie
#                 enthalten, in einer geeigneten Datenstruktur
################################################################
# Aufgabenteil 2
################################################################

# 5. read_treebank
#    Einlesen einer Baumbank im Klammerformat
#    Argument:     String mit der Baumbank
#    Rückgabewert: Baumrepraesentationen mit Frequenzen
#                  in einer geeigneten Datenstruktur

# 6. write_grammar_in_file
#   Ausgabe einer Grammatik in eine Datei (Format wie oben Z.17-19;
#   2 Tabulatoren zwischen Regel und ihre Wahrscheinlichkeit)
#   Argumente: 1. Dateiname der Ausgabedatei
#              2. Grammatikregeln in einer geeigneten Datenstruktur
#   Rückgabewert: keiner
```

```
##########################################################
# Hauptprogramm
##########################################################

# Am Ende sollte folgende Aufrufsequenz moeglich sein:

# Grammatik einlesen
grammar = read_file("grammatik.txt")

# Regeln parsen
rules = read_grammar(grammar)

# NP-Regeln extrahieren (beispielhafte Anfrage
# -- andere Kategorien sollten natuerlich ebenfalls moeglich sein)
subset = extract_rules("NP",rules)

# Subset ausgeben
write_grammar(outfile,subset)


# Baumbank einlesen
treebank = read_file("tueba5000.penn")

# Baeume parsen
rules = read_treebank(treebank)

# Grammatik ausgeben
write_grammar_in_file("grammatik.txt",rules)
```