**Schriftliche Wiederholungsprüfung zur Übung**
**Statistische Methoden in der maschinellen Sprachverarbeitung**
**SS 2021**
**Dozent: Helmut Schmid**

**Sie haben für die Bearbeitung 90 Minuten Zeit. Bitte lesen Sie die ganze Aufgabe durch, bevor Sie mit der Lösung beginnen.**

**Aufgabe:** Sie sollen einen Cocke-Younger-Kasami-Parser (CYK-Parser) in Python implementieren. Der Parser schlägt zunächst die Eingabewörter im Lexikon nach und bildet dann mit den Grammatikregeln bottom-up immer größere Konstituenten, bis der ganze Satz geparst ist. CYK-Parser funktionieren nur mit Grammatiken in Chomsky-Normalform (CNF).

CNF-Grammatiken enthalten

- lexikalische Regeln der Form A → w, wobei A ein Nichtterminal (Grammatiksymbol) und w ein Terminalsymbol (Wort) ist,
- und Grammatikregeln der Form A → B C mit drei Nichtterminalen A, B, C.

Bitte verwenden Sie bei Ihrer Implementierung die folgenden **Datenstrukturen:**

- Das **Lexikon** *lexrules* ist ein Dictionary, welches Wörter auf eine Liste von Paaren abbildet. Jedes Paar umfasst eine Wortart und seine Wahrscheinlichkeit.

  Beispiel: lexrules[“store”] = [(“VB”, 0.01),(“NN”, 0.02)]

- Die **Grammatik** *gramrules* ist eine Liste von Paaren. Jedes Paar umfasst eine Regel (bestehend aus 3 Nichtterminalen A,B,C) und eine Wahrscheinlichkeit p.

  Beispiel: gramrules = [((S, NP, VP), 1.0), ((VP, V, NP), 0.3), ...]

- Die **Chart** *vitprob* ist ein Dictionary, welches Tripel (startpos, endpos, symbol) auf Wahrscheinlichkeiten abbildet.

  Beispiel: vitprob[1, 2, “NN”] = 0.02

**Lexikon-Lookup:** Der CYK-Parser schlägt die möglichen Wortarten für jedes Eingabe-Wort $w_i$ im Lexikon nach und trägt jede mögliche Wortart T und die entsprechende Wahrscheinlichkeit p mit dem Befehl vitprob[i, i+1, T] = p in die Chart ein.

**Parsing:** Nun iteriert der Parser in einer vierfach geschachtelten Schleife

- über alle Endpositionen k von 2 bis einschließlich n (n ist die Eingabelänge)
- über alle Startpositionen i von k-2 bis 0
- über alle Mittepositionen j von i+1 bis k-1
- über alle Grammatik-Regeln A $\rightarrow$ B C mit Wahrscheinlichkeit p

Er speichert mit dem Befehl vitprob[i, k, A] = $p * p_1 * p_2$ die Konstituente A in der Chart,

- falls die Keys vitprob[i, j, B] und vitprob[j, k, C] existieren und die Werte $p_1$ und $p_2$ besitzen, und
- falls vitprob[i, k, A] nicht existiert oder einen Wert kleiner als $p * p_1 * p_2$ hat.

**Parsebaum:** Der Parser soll außerdem einen Parsebaum in Klammer-Notation erstellen. Dazu wird ein zusätzliches Dictionary *bestparse* erzeugt. Nach jedem Befehl vitprob[i,k,A] = p soll die beste Analyse der Konstituente A als String in bestparse[i,k,A] gespeichert werden. Die Methode **parse** liefert den besten Parsebaum für den ganzen Satz als Ergebnis.

Beispiel: (S (NP (ART die) (N Erde)) (VP (V ist) (NP (ART ein) (N Planet))))

Implementieren Sie den Parser, indem Sie die Definition der Klasse Parser in der Datei **parser.py** vervollständigen. Mit den beiden Methoden **read_lexicon** und **read_grammar** lesen Sie das Lexikon und die Grammatik ein. Die Methode **parse** analysiert einen Eingabesatz (Initialisierung der Chart, Nachschlagen der Wörter im Lexikon, Parsing mit den Grammatikregeln) und gibt den wahrscheinlichsten Parsebaum als String zurück.

Die Dateien **lexrules.txt** und **gramrules.txt** zeigen Ihnen die Formate des Lexikons und der Grammatik und können auch zum Testen verwendet werden (falls Ihnen die Zeit dafür noch reicht).

Das **Startsymbol** ist das erste Symbol in der Grammatikdatei.

Viel Erfolg!