Vorlesung "Computerlinguistische Techniken"

2. Übung (7.11.2023)

Wintersemester 2023/2024 – Prof. Dr. David Schlangen

Aufgabe 2 setzt voraus, dass Sie auf Ihrem Computer die NLTK-Bibliothek installiert haben. NLTK ist korrekt installiert, wenn man `import nltk` in Python ohne Fehlermeldung ausführen kann.

1 Perrin-Zahlen

Die Perrin-Zahlen $p_0, p_1, \ldots$ sind wie folgt definiert:

$$
\begin{aligned}
p_0 &= 3 \\
p_1 &= 0 \\
p_2 &= 2 \\
p_n &= p_{n-2} + p_{n-3} \text{ für alle } n > 2.
\end{aligned}
$$

a) Schreiben Sie eine rekursive Python-Funktion `perrin_r`, so dass `perrin_r(n)` die Zahl p_n zurückgibt.

b) Schreiben Sie eine nichtrekursive Python-Funktion `perrin_i`, so dass `perrin_i(n)` die Zahl p_n zurückgibt. Sie brauchen entweder eine `for`-Schleife mit `range` oder eine `while`-Schleife.

c) Berechnen Sie mit beiden Funktionen die Zahl p_{70}. Welche Funktion ist schneller?

d) Geben Sie die asymptotischen Laufzeiten der beiden Funktionen in Abhängigkeit von n an. Bei exponentiellen Laufzeiten können Sie einfach "exponentiell" sagen; Sie müssen nicht die genaue Basis angeben. Begründen Sie Ihre Antwort.

Anmerkung:
Beide Funktionen sollen die angegebenen Namen tragen und in einer gemeinsamen Datei `perrin.py` liegen. Für die Teilaufgaben c) und d) reichen Sie eine PDF-Datei, benannt nach dem Schema `ue02-NACHNAME-MATRIKELNR.pdf`, ein.

2 CKY mit NLTK

In der Vorlesung haben wir den Kern einer Python-Implementierung eines CKY-Erkenners und eines CKY-Parsers gesehen. Vervollständigen und integrieren Sie

diese beiden Implementierungen, so dass Ihr Programm sowohl das Wortproblem als auch das Parsingproblem lösen kann. Die Implementierung soll Grammatiken aus einer Datei und Sätze von der Konsole einlesen. Ihr Programm soll sich folgendermaßen verhalten:

```
$ python cky.py --file grammatik --sentence "Hans isst ein Käsebrot"
True
$ python cky.py --file grammatik --sentence "Hans isst"
False
$ python cky.py --file grammatik --sentence "Hans isst" --parser
[]
```

Dabei ist `grammatik` der Name einer Datei, in der eine kontextfreie Grammatik im NLTK-Format steht. Der zu parsende Satz wird als String in Anführungszeichen übergeben. Ohne weiteres Argument wird der Erkenner ausgeführt. Mit dem Flag `--parser` wird der Parser aufgerufen.[1]

Schließlich soll Ihr Programm, wenn es als Erkenner ausgeführt wird, True oder False ausgeben (je nachdem, ob der Satz in der Sprache ist) und als Parser eine Liste von Parsebäumen als `nltk.tree.Tree`-Objekte.

Um das Parsingproblem zu lösen, müssen Sie in der Chart neben den einzelnen Nichtterminalsymbolen auch *Backpointer* speichern, in denen Sie sich merken, wie man einen Chart-Eintrag aus kleineren Chart-Einträgen bauen kann. Verwenden Sie die Klasse `Tree` aus dem Modul `nltk.tree`, um die Parsebäume zu repräsentieren. Probieren Sie Ihren Parser mit der Grammatik aus Aufgabe 2 der 1. Übung aus.

Anmerkung: Wenn Sie die Bäume mit der `pretty_print`-Methode ausgeben möchten, muss der entsprechende Aufruf außerhalb der `cky`-Funktion passieren.

Eine Python-Datei mit dem Namen `cky.py`, die bereits ein Code-Gerüst enthält, steht auf Moodle bereit. Nutzen Sie alle vordefinierten Funktionsnamen und ändern Sie diese nicht. Fügen Sie auch keine anderen Funktionen hinzu.

Beachten Sie, dass um Ihr Programm auszuführen am Ende nur die Funktion `cky(grammar_file, sentence, parser)` aufgerufen werden soll, um True/False beziehungsweise die Liste der Bäume zu erhalten.

Reichen Sie Ihre Implementierung unter dem Namen `cky.py` ein.

Abgabe bis 21.11.2023, 13:00 Uhr, via Moodle

[1] Das Übungsmaterial enthält bereits eine Implementierung für das Flag und in der Übung besprechen wir, wie man es benutzt.