Übung zur Vorlesung "Computerlinguistik I"

Wintersemester 2016/17, Prof. Dr. Udo Hahn, Sven Büchel
Übungsblatt 5 vom 25.11.2016
Abgabe bis 01.12.16, 23.59 Uhr; per Email (PDF-Format) an
sven-eric.buechel@uni-jena.de

Aufgabe 1 : Algorithmen verstehen (4 Punkte)

a) Beschreibung (2 Punkte)

Beschreiben Sie **präzise** in ganzen Sätzen, was der folgende Algorithmus für die englische Sprache macht. Gehen Sie dabei vor allem auf die Gesamtaufgabe des Programms ein. Würden Sie einen präziseren Namen als `"umwandlung"` vorschlagen?

```
umwandlung(↓eingabewort, ↑ausgabewort)
    endung ← eingabewort[length(eingabewort)−3:length(eingabewort)]
    if endung = "ies"
        ausgabewort ← eingabewort[0:length(eingabewort)−3]
        ausgabewort ← ausgabewort + "y"
    else
        ausgabewort ← eingabewort
```

b) Identifikation von Fehlerquellen (2 Punkte)

Hat der Algorithmus aus der vorigen Teilaufgabe Unzulänglichkeiten? Welche? Beschreiben Sie informell, wie die Probleme das Algorithmus' behoben werden könnten. Benutzen Sie auch hier bitte ganze Sätze!

Aufgabe 2 : Algorithmus zur Satzsegmentierung (6 Punkte)

Eine der grundlegendsten Aufgabe in der Computerlinguistik ist es, in einem Fließtext Sätze zu erkennen. Ein Programm zur Satzerkennung bekommt als Eingabe einen Text, beispielsweise "Peter lief nach draußen. Seine Jacke wurde vom Regen durchweicht.". Die Ausgabe des Programms wären dann die Sätze "Peter lief nach draußen." und "Seine Jacke wurde vom Regen durchweicht."

a) (2 Punkt)

Eine naive Herangehensweise zur Satzerkennung ergibt sich, wenn jeder Punkt im Text als Satzgrenze aufgefasst wird. Welche Probleme sehen Sie bei diesem Vorgehen? Nennen Sie mindestens vier Beispiele.

b) (4 Punkte)

Geben Sie einen einfachen Algorithmus an, der einen Text entgegen nimmt und diesen satzweise ausgibt. Hinweise:

- Als Hilfestellung sei die Funktion `getWords(text)` gegeben. Diese Funktion bekommt einen Text, trennt ihn an Leerzeichen und gibt die so erhaltenen Wörter (ggf. mit Interpunktion!) in einer Liste zurück. Die Leerzeichen selbst gehen dabei verloren. Beispiel: `words ← getWords("Peter lief nach draußen.")` zerlegt den eingegeben Satz in die Elemente `"Peter"`, `"lief"`, `"nach"` und `"draußen."` (man beachte, dass das letzte Element auch den Punkt enthält, da er nicht durch ein Leerzeichen abgetrennt ist). Die Variable `words` enthält nun alle diese Wörter. Wie gehabt kann mit `words[0]` auf das `1`te Wort, mit `words[1]` auf das `2`te Wort etc. zugegriffen werden.
- Als weitere Hilfe sei eine Liste aller Titel `titleList` gegeben. Diese Liste enthalte beispielsweise "Dr.", "Prof.", "Dipl." etc. Diese Liste kann verwendet werden, um zu überprüfen, ob ein Wort eine Abkürzung oder ein Satzende darstellt. Bsp: `if wort in titleList: ...`
- Eine einzelne Variable kann beliebig viele Zeichen und Wörter enthalten. Betrachten Sie folgendes Beispiel:

```
wort1 ← "Peter"
wort2 ← "lief"
satz ← wort1 + " " + wort2
```

Die Variable `satz` enthält nun die Zeichenkette `"Peter lief"` (man beachte das Leerzeichen, das wieder hinzugefügt werden muss).

- Eine Variable kann "geleert" werden, indem man ihr die leere Zeichenkette zuweist: `satz` ← `""`.